Security Threat Modelling

For

Enterprises

Enterprise Security Threat modelling

Version 0.01

Version and Amendment Register

Ver.	Date	Auth	Amendment	Details
0.1	2023-06-08	GC	Initial Draft	Initial Draft

Approval Register

Name	Title	Signature	Date

Contents

Document Location

1 Introduction

Although there are not explicit requirements for this stream of work, this document has been produced as result of the need of the organisation to show compliancy and increasingly advance in the handling security for itself and its customers. For the proposed design has been conceived based on the best understanding of the requirements gathered from stakeholders in the context of ISO-27001 certification.

1.1 References

These documents are related to other documents.

Documents

#	Document Name and Version	Author
1		

Reference Material

#	Reference Name	Link
1		

1.2 Terminology

#	Acronym/Term	Description
1		

2 Context

Introduction

This document conforms to the standards of the organisation for the for Solution Architecture. In particular to the Security Architecture viewpoint.

3 Scope

Introduction

This document is one of the several artefact deliverables that will be delivered for the business owner of the capability. In particular, this document focuses on the Security Threat modelling that affects all systems and projects delivered in the organisation.

Thid document must be read along with the architecture and technology design best practices and baselined architecture of the organisation. It includes: business, application, technology, security, information, infrastructure, software and operations considerations.

This document will focus on the different aspects to be considered for assessing the Security model proposed in the baselined architecture used in the organisation.

4 Goals

Introduction

The Security model architected and used in projects need to be assess and benchmarked. This document provides a solution for the postulation of the security threats, risk assessment and mitigation.

The goals is to articulate the Threat assessment consist in:

- Threat Postulation
- Theat assessment
- Threat assessment rationale
- Risk mitigation

In addition, the document will pursue the following goals:

- Specify in an orderly manner a documentation that is clear and objective, so that the solution proposed can be followed by business owners and stakeholders.
- Provide design material for professionals to be informed about the proposed options for specific capabilities.
- Provide guidance and design for more technical team members to consider this document when developing any system or product that uses the architecture postulated in this document.

5 Approach

Introduction

The organisation address security using the most modern method and techniques. Security threat modelling and the addressing of the security concerns is not considered an extra task in the project or a layer afterthought in the architecture or development process. The security is embedded all over the conception of the project and the lifecycle development. In particular the conception of requirements, the architecture design, the processes follow, the tools selected, and if required, manual operation processes. The organisation and architecture embrace agile process, therefore all aspects of architecture must be adapted to the agile process principles, including the security design.

5.1 Governance

This domain focuses on establishing and maintaining a software security governance framework within the organization. It includes activities such as policy management, risk management, compliance management, awareness, and training.

In particular when implementing Security by Governance in an agile process, the role of the architecture capability plays a crucial role in ensuring that security is integrated into the architecture design, processes, and tools. The most important is that security is not treated as an afterthought but is embedded throughout the development lifecycle. This approach helps organizations build secure systems by design, mitigating security risks and ensuring the confidentiality, integrity, and availability of their software applications.

The architecture takes a security-first approach when designing the architecture considering security requirements, threats, and risks as essential design considerations. It ensures that the architecture provides the necessary controls and safeguards to protect the system and its data.

The architect leverages secure design patterns and architectural best practices to address common security challenges. Proven security measures are incorporated into the architecture, such as secure communication protocols, access controls, authentication and authorization mechanisms, and data encryption techniques.

The architecture is presented and adopted at early stages of the project by the development team, which collaborates along with the delivery manager in conducting regular security reviews and assessments of the architecture. This includes threat modelling exercises, security code reviews, and architectural risk assessments to identify and mitigate potential security vulnerabilities.

The architecture must align with relevant compliance and regulatory requirements in projects where it is important the industry-specific security standards and guidelines. Also in incorporating necessary controls and audit mechanisms that facilitate the incorporation of those standards.

The architecture selects and integrates appropriate security tools and technologies into the architecture for vulnerability scanning, security testing, log monitoring, and intrusion detection.

The architecture also considers technologies that enhance security, such as encryption libraries, secure communication frameworks, and identity management systems when appropriate.

The architecture introduces secure development practices and guidance, which are transformed into secure coding techniques, secure configuration management, and secure deployment strategies.

The architecture proposed must be in the context of a software security governance framework within the organization, where stakeholders define security policies, risk management processes, compliance management procedures, and security awareness and training initiatives.
Any architecture produced must align with this governance framework and helps enforce security controls and processes.
The architecture must also work in agile environments, where it must embrace the agile principles of adaptability and responsiveness. For this, architecture must address evaluating the security posture of the project or the organisation, monitor emerging threats, and adapt the architecture as needed to address new security challenges. The architecture consider these data points as security-related information and interprete them as feedback and lessons learned, so it uses each iteration to improve the overall security.

5.2 Design

The architecture capability must ensure security by design whilst performing it is embedded in an agile process.
The architecture embraces iterative and incremental development, therefore security designs must be address continuously and refined throughout the development process.
The architect must address the involvement of teams starting from stakeholders and ensuring that the right type of requirements is included into the design, then work with developers, testers and control that security that is embedded from the beginning is considered and followed whilst the development process.
The architecture must be adapted and refined by embracing continuous improvement when security threats are identified. It must consider feedback and changing requirements.
The architecture is based on the adopted baselined architecture the organisation has adopted and baselined. The architecture is based on accumulated experiences and knowledge transfer, which has been iterated multiple times until it becomes solid enough to have it defined to be a toolkit that can be used in combinations to conform the backbone of the solution architecture, comprising technologies of preferences, platforms and design patterns in which the security is embedded by design at different levels on the architecture stack.
The architecture reflects the prioritisation from stakeholders about the categorisation of security requirements. These requirements are integrated into the architecture design and treated as part of the integral solution rather than an afterthought.
It must adhere to existing security principles and at the same time help to prioritise the identified threats and risks with the business. It can conduct threat modeling exercises and use a probabilistic risk-based approach to security. to help with the prioritisation if not clear.
The architecture is continuously tailored considering contextual factors because not all the industries and the systems are the same, and implementing security elements and hardening measures must provide a measurable cost benefit.
The architecture is well-documented, capturing security considerations, the options evaluated, the design decisions taken and rationale behind. This helps to achieve an effective knowledge transfer and helps maintain a solid foundation for future projects.

The architecture addresses compliance requirements because follows governance mechanisms that are in place to ensure that security requirements are adhered in the project.
The architecture is adapted because the agile process permits continuous evaluation and improvement, that include the evaluation of the security measures.

5.3 Implementation

The architecture use automation to enhance security implementations. Automation tools and techniques, such as Infrastructure as Code (IaC) and configuration management tools, are utilized to standardize and automate security configurations, provisioning, deployment, and testing processes.
This allow integrating the security measures effectively throughout the development lifecycle, leveraging automation, continuous integration and deployment, collaboration, continuous improvement, and a tailored approach to achieve a robust security posture.
The approach integrate security practices into the CI/CD pipeline. This consist in the inclusion of several type of quality assurance and also security to improve overall quality and reliability, compliance. For example, the implementation process is embedded with testing, vulnerability scanning, and code analysis tools to identify and address security issues early in the development lifecycle.
In addition, Architecture acts as a bridge between teams, development, operations, and security teams, to ensure that security requirements, controls, and guidelines are effectively communicated, fostering a shared understanding and joint responsibility for security throughout the development process.
Architecture adapts to continuous improvement. Regular reviews and refinements of security practices and processes are conducted to address emerging security challenges, adopt new technologies and techniques, and align with evolving security best practices.

5.4 Verification

The verification domain focuses on validating the effectiveness of security controls and measures in software systems. It includes activities such as security testing, code review, security regression testing, security architecture review, and threat modelling.
In particular when implementing Security by Verification in an agile process, the role of the architecture capability plays a crucial role, ensuing that the security principles are embed security as an inherent part of the architecture propose and also the processes and tools, verifying that security are seamlessly integrated into the agile process and contribute to the overall security posture of the software system.
The architecture must ensure that the security is embedded in the design of the software system from the outset. This includes verifying and identifying security requirements are appropriate and defining the security controls.
The architecture must promotes security verification activities, such as security testing, code review, security architecture review, threat modelling, and security regression testing.
Architecture implements several types of verifications mechanisms into the CI-CD process, this will ensure that the security standards are met. For example, the implementation process is embedded with

testing, vulnerability scanning, and code analysis tools to identify and address security issues early in the development lifecycle.
The architecture collaborates with other teams, including security experts and testers. These work together to identify security risks, review code, perform security testing, and conduct architectural reviews.
The security activities are timeboxed and performed iteratively within each development iteration to fit into the agile process cadence.
The architecture is adapted with a just-in-time approach to make necessary adjustments when needed.
The architecture capability must promote the continuous improvement and adaptation to address security concerns when needed. For this, the architecture must be re-worked or adapted and collaborate with the development team to implement remediation measures about any security concern.
The architecture ensures the security verification activities and align the finding with the overall security governance framework that include security policies, risk management processes, compliance requirements, and security awareness.

5.5 Operations

The Architecture capability ensures that security is a continuous and integrated part of the agile process including the Operationalisation of new systems and the security related with these considerations.
Overall and not exclusively to Operations, the Security tasks considered are addressed iteratively, timeboxed, and aligned with the principles of agility.
These security considerations could include Standard Operating Procedures (SOPs) that are used as a method when other security measures are insufficient.
The integration of SOPs are considered a last resort, the architecture capability can integrate them within the agile process when security concerns have not been adequately addressed through design and automation. SOPs should be documented, communicated, and regularly reviewed to provide clear instructions for manual security tasks. They should be integrated with automation processes wherever possible to maintain consistency and reduce reliance on manual interventions.
The architecture capability ensures that relevant stakeholders, including operations teams, are trained on security practices, including the use of SOPs. This helps improve awareness, knowledge, and competency in executing security tasks, ensuring consistency and effectiveness in security operations.

6 Disclaimer

This document uses the preferred solution architecture adopted by the organisation. This architecture is constructed based on an opinionated foundation of architecture principles, architecture patterns, technologies, platforms, software providers, etc., that are assembled to provide a working solution. These architecture principles, the software have been chosen carefully with the idea that can be assembled with minimal risks of finding interoperability issues and because they are considered among the best in class for its purpose. It is understood that similar architecture could be achieved using

different software and platforms that provide similar capabilities, in particular the interoperability and the integration profile with the rest of the stack.
This document and the architecture solution are not meant to be prescriptive about the technologies; therefore, the design artifacts that include named software and technologies must be used as references. The products and software mentioned in the technical architecture are one of the many proposed architectures and one of the possible instantiations.
In any case, it is expected that any software suites presented in this document can be interchanged in the development phase by other substantial equivalents.

7 Threat Modeling Process

7.1 Introduction

Threat modeling is a systematic approach to identifying and evaluating potential threats and vulnerabilities in a software system, network, or application. It is a proactive activity conducted during the early stages of the software development lifecycle to understand and mitigate potential security risks. The primary goal of threat modeling is to identify, prioritize, and address potential threats before they can be exploited by attackers.
By performing threat modeling, organizations can proactively identify and mitigate security vulnerabilities, design more secure systems, allocate resources effectively, and reduce the likelihood of successful attacks. It helps guide the implementation of security controls and informs security testing efforts, resulting in more resilient and secure software systems.
There are several Threat modelling techniques and processes. Most of the techniques consist in defining a lifecycle management of the threats and the mitigation.
The cycle in general terms consist in identifying what type of risks the system could have for the business. For example, dealing with Personal Identifiable Information (Individual personal data), Financial Data, Tampering of data (fraud), etc, and what would be the consequences of these threats for the business. Consequently, it focuses in the system architecture and flows, where the risks can be assessed in terms of the components involved and what options are available for mitigating each of the threats. This is defined as a cycling process, where the organisation can perform multiple iterations and revisions periodically.
Threat modeling typically involves the following steps as details below.

7.2 Process

#	Name	Description
1	Identify Objective and main Assets	Identify Security and Compliance Requirements. Identify and define the assets that need protection, such as sensitive data, intellectual property, user information, infrastructure components, or critical functionalities. Identify and agree what would be the impact of the business and prioritise the objectives.

#	Name	Description
2	Create a System Overview	Identify and Capture the boundaries of the technical environment. Develop a detailed understanding of the system architecture, including components, interfaces, data flows, and external dependencies. This overview helps identify potential entry points and paths that could be exploited by attackers.
3	Create System break down	Decompose the system in application and technical components, platofforms, network compartments, etc. Identify use cases, Entry points, Trust Levels. Identify Actors, assets, Services, Roles, Data sources. Identify Information flows and trust boundaries.
4	Identify Threats	Identify potential threats and vulnerabilities that could compromise the security of the system. Consider the potential consequences, such as data breaches, unauthorized access, or service disruptions. This includes both technical threats (e.g., SQL injection, cross-site scripting) and non-technical threats (e.g., physical access, social engineering).
5	Assess Risks	Quantify threat likelihood Analyse probabilistic attack scenarios. Evaluate the potential impact and likelihood of each identified threat. Regression Analysis on Security Events. Threat intelligence correlation and analytics. Prioritize the risks based on their severity and likelihood.
6	Vulnerability and Weakness Analysis	Design flay analysis using use and abuse cases Build Scorings and Enumerations. Consider using Vulnerability Trees Consider using Common Weakness Scoring System (CWSS) Consider using Common Vulnerability Scoring System (CVSS)
7	Attack Modelling	Attack surface analysis Attach Tree Development Attack to Vulnerability and Exploit analysis using Attack Trees
8	Define Mitigation Measures	Qualify and quantify business impact. Risk mitigation strategy Define and prioritize countermeasures to address the identified risks. These can include security controls, architectural changes, secure coding practices, encryption, access controls, or monitoring and logging mechanisms.
9	Validate and Iterate	Review the threat model with relevant stakeholders, including developers, architects, security experts, and business representatives. Validate the assumptions, identify potential gaps, and refine the threat model iteratively.
10	Communicate Findings	Document and communicate the threat model findings to the development team, management, and other relevant stakeholders. This helps create awareness and guides decision-making regarding security measures and risk mitigation strategies.

Resources

https://owasp.org/www-community/Threat_Modeling

https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html

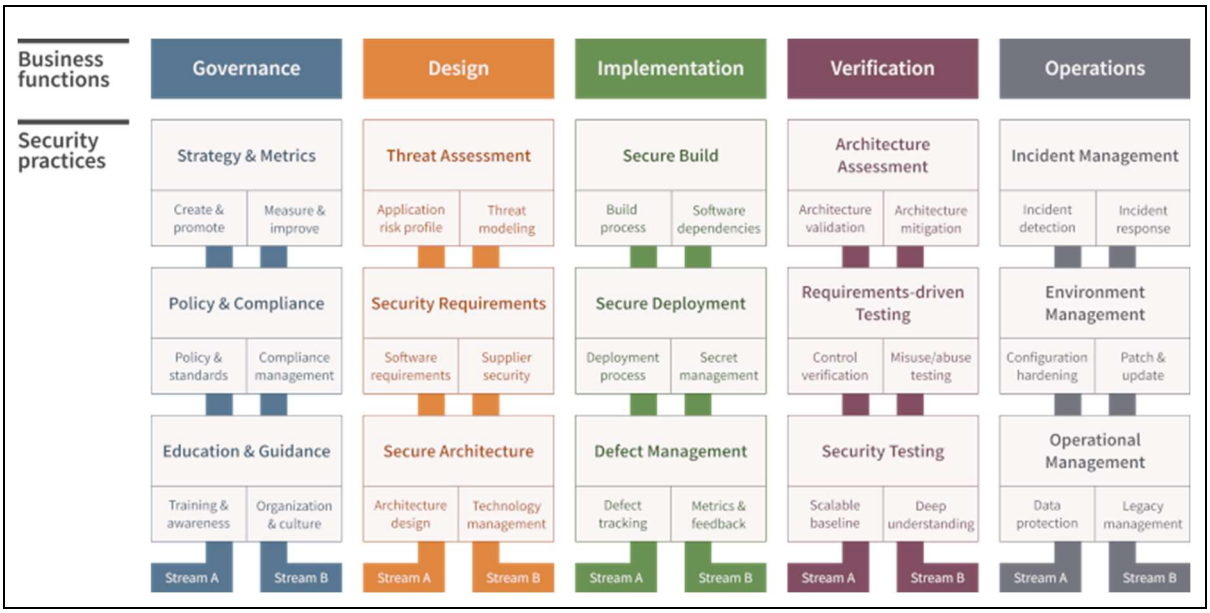
https://owasp.org/www-community/Threat_Modeling_Process
https://en.wikipedia.org/wiki/Threat_model
https://www.threatmodelingmanifesto.org/
https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

8 Software Assurance Maturity Model (SAMM)

8.1 Introduction

There is no specific maturity model associated with the OWASP Top 10.
The OWASP Top 10 primarily serves as a list of the top web application security risks, providing guidance on identifying and mitigating these vulnerabilities. It is a widely recognized industry standard that helps organizations prioritize their security efforts and improve the security posture of their web applications.
However, OWASP does provide other resources and initiatives that focus on maturity models and continuous improvement in application security.
The OWASP SAMM (Software Assurance Maturity Model) is a framework designed to help organizations assess and improve their software security practices. SAMM provides a maturity model that organizations can use to evaluate their current security practices across various domains and identify areas for improvement.
SAMM consists of four security practices domains: Governance, Design, Implementation, Verification, and Operations.

8.2 SAMM Security domains



SAMM security domains

Description

#	Name	Description
1	Governance	This domain focuses on establishing and maintaining a software security governance framework within the organization. It includes activities such as policy management, risk management, compliance management, awareness, and training.
2	Design	The design domain emphasizes incorporating secure design principles and practices into the software development process. It covers areas like secure architecture, security requirements, secure design review, secure coding standards, and security testing and review.
3	Implementation	This domain deals with implementing secure development and deployment practices. It includes activities such as secure build, secure deployment, secure configuration management, secure software lifecycle, and security training.
4	Verification	The verification domain focuses on validating the effectiveness of security controls and measures in software systems. It includes activities such as security testing, code review, security regression testing, security architecture review, and threat modeling.
5	Operations	The operations domain deals with ongoing security practices for software systems in production. It covers areas such as incident management, patch management, vulnerability management, security monitoring, and security assurance.

Each domain has several maturity levels, and organizations can assess their current maturity level and define a roadmap to advance to higher levels. SAMM helps organizations establish a holistic approach to software security by considering both technical and non-technical aspects.
While SAMM is not directly tied to the OWASP Top 10, it aligns with the overall goals of improving application security and can be used alongside the OWASP Top 10 as part of a comprehensive security program.

8.3 SAMM Maturity Model

Level	SAMM Governance	SAMM Design	SAMM Implementation	SAMM Verification	SAMM Operations
Level 1	Policy Management - Establishing and documenting software security policies and objectives.	Secure Architecture - Implementing secure design principles and practices in software architecture.	Secure Build - Ensuring that the software is built and deployed securely.	Security Testing - Conducting security testing to identify vulnerabilities and weaknesses.	Incident Management - Establishing processes for handling security incidents and breaches.
Level 2	Risk Management - Identifying and assessing software security risks and integrating them into the organization's risk management processes.	Security Requirements - Defining and incorporating security requirements into the software development lifecycle.	Secure Deployment - Implementing secure deployment configurations and practices.	code Review - Performing secure code reviews to identify security issues.	Patch Management - Implementing effective patch management processes to address software vulnerabilities.
Level 3	Compliance Management - Implementing processes to ensure compliance with relevant software security standards and regulations.	Secure Design Review - Conducting reviews and evaluations of the software design for security vulnerabilities.	Secure Configuration Management - Managing and maintaining secure configurations for software systems.	Security Regression Testing - Implementing security regression testing to ensure that security controls are not inadvertently weakened.	Vulnerability Management - Implementing vulnerability management practices to identify, assess, and remediate vulnerabilities.
Level 4	Awareness and Training - Raising awareness about software security and providing appropriate training to employees.	Secure Coding Standards - Implementing and enforcing secure coding standards and guidelines.	Secure Software Lifecycle - Integrating security into the entire software development lifecycle.	Security Architecture Review - Reviewing the security architecture and design of software systems.	Security Monitoring - Implementing security monitoring and logging capabilities to detect and respond to security events.
Level 5	Organizational Security Culture - Developing a security-conscious culture within the organization and integrating software security into the overall organizational culture	Security Testing and Review - Performing security testing and code reviews to identify and address security issues.	Security Training - Providing ongoing security training and awareness programs for development teams.	Threat Modeling - Incorporating threat modeling techniques to identify and mitigate security risks.	Security Assurance - Ensuring ongoing security assurance through continuous monitoring, assessment, and improvement.

Resources

https://owasp.org/www-project-samm/

9 Threats (OWASP Top 10)

9.1 Introduction

OWASP (Open Web Application Security Project) is a nonprofit organization focused on improving the security of software and web applications. The OWASP Top 10 is a list of the most critical security risks for web applications, widely recognized as a benchmark for application security. The list is regularly updated to reflect the evolving threat landscape and provide guidance for developers, security professionals, and organizations.

In summary, the OWASP Top 10 has played a crucial role in shaping the industry's approach to web application security. It has provided a common language and framework for discussing security risks, promoting awareness, and guiding organizations in improving the security posture of their applications.

The OWASP Top 10 was first published in 2003 as a result of a collaborative effort by security experts and developers worldwide. It aimed to raise awareness about the most prevalent security risks in web applications and provide developers with a starting point for improving their application security practices.

Over the years, the OWASP Top 10 has become an industry-standard reference and a valuable resource for organizations to prioritize their security efforts. It has gained significant recognition and adoption within the software development and security communities, serving as a guideline for identifying and addressing common vulnerabilities.

9.2 Approach

The approach for assessing the security when adopting the OWASP Top 10 threats is as follow:

- Identify Objective and main Assets.
- Create a System Overview
- Create a System break down.
- Identify Threats
- Assess Risks
- Vulnerability and Weakness Analysis
- Attack Modelling
- Define Mitigation Measures
- Validate

Identifying threats as part of the threat modeling process, it's essential to consider a wide range of potential threats and vulnerabilities that could compromise the security of the system. These threats can be categorized into technical and non-technical threats.
It's important to note that this list is not exhaustive, and the specific threats will depend on the nature of the system, its components, and the potential attackers it may face.
The Threat modeling encourages a comprehensive and systematic approach to identify and understand potential threats, considering both technical vulnerabilities and the broader context in which the system operates. By considering a wide range of threats, organizations can develop appropriate countermeasures to mitigate risks and enhance the security of their systems.
In this Threat Identification and profiling we will be adopting the OWASP Top 10. The reason for this is that since 2001, OWASP has been compiling research from over volunteers world-wide to educate the IT industry on the most dangerous risks facing in modern systems. OWASP edits the top 10 frequently and change the order and sometimes introduces new categories.
As per this document is generic and not focused in any particular industry, by answering how the baselined architecture addresses these threats is a good starting point.

9.3 Security Standardisation

The OWASP Top 10 is not an official standard or specification set by any formal standards organization. However, it has become a de facto industry standard and widely recognized guideline for web application security. It is widely adopted and referenced by organizations, security professionals, and regulatory bodies as a basis for assessing and improving the security of web applications.
The OWASP Top 10 is developed and maintained by the Open Web Application Security Project (OWASP), a community-driven organization. OWASP follows an open and collaborative process to gather input and feedback from security experts, developers, and industry professionals worldwide. The list is periodically updated to reflect the evolving threat landscape and emerging vulnerabilities.
While the OWASP Top 10 is not a formal standard, it has influenced the development of security standards and frameworks. Many security standards and regulatory bodies reference the OWASP Top 10 as a valuable resource for application security best practices. For example, the Payment Card Industry Data Security Standard (PCI DSS) includes requirements related to the OWASP Top 10 vulnerabilities. Additionally, various security frameworks and guidelines incorporate the OWASP Top 10 as a benchmark for assessing and addressing web application security risks.
The OWASP Top 10 has gained widespread acceptance and recognition within the industry due to its practicality, relevance, and community-driven nature. Its focus on common web application vulnerabilities and mitigation strategies has helped organizations prioritize their security efforts and improve their application security posture.
While it is not a formal standard, the OWASP Top 10 has had a significant impact on the industry by promoting secure coding practices, raising awareness about common vulnerabilities, and providing a shared vocabulary for discussing application security risks. Its influence extends beyond a specific standardization process, making it a valuable resource for the industry as a whole.

9.4 Industry Reelevance

The OWASP Top 10 are not specific to any industry in particular and they are universally recognised and adopted as the benchmark that any organisation would want to conform when building a system. The OWASP Top 10 are relevant to any all industries because they address they implicitly address following security categories:

#	Name	Description
1	Risk Prioritization	The list helps organizations prioritize their security efforts by highlighting the most critical risks that can have a significant impact on application security. It provides a starting point for risk assessment and mitigation.
2	Awareness and Education	The OWASP Top 10 raises awareness about common security vulnerabilities, ensuring that developers and security professionals have a foundational understanding of these risks. It promotes education and knowledge sharing to drive improvements in secure coding practices.
3	Security by Design	The OWASP Top 10 encourages organizations to adopt a proactive and security-focused mindset from the early stages of application development. By considering security during the design phase, organizations can build more resilient and secure applications.
4	Compliance and Auditing	The OWASP Top 10 is often referenced in security standards, regulations, and compliance frameworks. It helps organizations meet industry requirements and provides a basis for security audits and assessments.
5	Industry Collaboration	The OWASP Top 10 is a collaborative effort that involves security professionals, developers, and organizations worldwide. This collaboration fosters the sharing of best practices, knowledge, and solutions to address the common security challenges faced by the industry.
6	Continuous Improvement	The OWASP Top 10 is regularly updated to reflect emerging threats and evolving attack techniques. It encourages continuous improvement in application security practices and provides guidance on new vulnerabilities and mitigation strategies.

Resources

https://owasp.org/www-project-top-ten/

https://sucuri.net/guides/owasp_top_10_2021_edition/

9.5 Threats

9.5.1 Broken Access Control (A01-2021)

Introduction

OWASP Broken Access Control is a category of security vulnerabilities that refers to flaws in the enforcement of access controls within a web application. Access controls are mechanisms designed to restrict user access to certain resources or functionalities based on their privileges and permissions. When these access controls are not properly implemented or enforced, it can lead to unauthorized access, privilege escalation, and exposure of sensitive data.

Addressing Broken Access Control vulnerabilities is crucial for protecting web applications from unauthorized access and maintaining the confidentiality and integrity of sensitive information. By implementing proper access controls and regularly assessing the application for potential weaknesses, organizations can reduce the risk of exploitation and enhance the overall security posture.

Definition

Broken Access Control vulnerabilities occur when an application fails to verify and enforce proper access controls, allowing attackers to bypass authorization mechanisms or gain unauthorized access to restricted resources.

9.5.2 Cryptographic Failures (A02-2021)

Introduction

OWASP (Open Web Application Security Project) Cryptographic Failures is a category of security vulnerabilities that refers to weaknesses and mistakes in the implementation of cryptographic algorithms and mechanisms within a web application. Cryptography plays a crucial role in ensuring the confidentiality, integrity, and authenticity of data in applications. However, when cryptographic practices are not implemented correctly, it can lead to vulnerabilities and compromise the security of the system.

Addressing Cryptographic Failures is crucial for maintaining the security and integrity of web applications that rely on cryptography for data protection. By following recommended cryptographic practices, organizations can minimize the risk of vulnerabilities and strengthen the overall security posture of their applications.

Here are some key points about OWASP Cryptographic Failures:

Definition

Cryptographic Failures encompass a wide range of vulnerabilities related to cryptography, including weak algorithms, improper usage, key management issues, and insecure configurations.

9.5.3 Injection (A03-2021)

Introduction

OWASP (Open Web Application Security Project) Injection is a category of security vulnerabilities that refers to the improper handling of untrusted data, allowing attackers to inject malicious code or unintended commands into a web application. Injection attacks can occur when user-supplied input is not properly

validated, sanitized, or parameterized, leading to the execution of unintended actions or unauthorized access to data.
Addressing Injection vulnerabilities is crucial for protecting web applications from unauthorized access, data breaches, and other security compromises. By implementing proper input validation, using secure coding practices, and regularly testing for vulnerabilities, organizations can significantly reduce the risk of Injection attacks and enhance the overall security posture of their applications.
Here are some key points about OWASP Injection vulnerabilities:

Definition
Injection vulnerabilities occur when untrusted data, such as user input or external data sources, is not properly handled and is used to manipulate the application's execution flow or interact with data stores.

9.5.4 Insecure Design (A04-2021)

Introduction
OWASP (Open Web Application Security Project) Insecure Design is a category of security vulnerabilities that refers to flaws and weaknesses in the overall design and architecture of a web application. Insecure design decisions can create vulnerabilities that attackers can exploit to compromise the security of the system.
Addressing Insecure Design vulnerabilities is essential for building robust and secure web applications. By adopting secure design principles, implementing appropriate security controls, and ensuring security is considered from the early stages of development, organizations can mitigate the risk of Insecure Design vulnerabilities and enhance the overall security of their applications.
Here are some key points about OWASP Insecure Design vulnerabilities:

Definition
Insecure Design vulnerabilities arise from poor architectural decisions, lack of security considerations during the design phase, or inadequate understanding of potential threats and risks.

9.5.5 Security Misconfiguration (A05-2021)

Introduction
OWASP (Open Web Application Security Project) Security Misconfiguration is a category of security vulnerabilities that refers to insecure configurations of web applications and associated components. Security misconfigurations occur when default or insecure settings are used, sensitive information is exposed, or unnecessary functionality is enabled, creating potential entry points for attackers.
Addressing Security Misconfiguration vulnerabilities is crucial for maintaining the security of web applications and associated components. By adhering to secure configuration practices, regularly updating software, and conducting security assessments, organizations can significantly reduce the risk of security misconfigurations and enhance the overall security posture of their applications
Here are some key points about OWASP Security Misconfiguration vulnerabilities:

Definition

Security misconfiguration vulnerabilities arise when systems, frameworks, servers, or web applications are not properly configured to adhere to secure practices and industry standards.

9.5.6 Vulnerable and Outdated Components (A06-2021)

Introduction

OWASP (Open Web Application Security Project) Vulnerable and Outdated Components is a category of security vulnerabilities that refers to the use of outdated or known vulnerable software components within a web application. This category focuses on the risks associated with using third-party libraries, frameworks, and other software dependencies that may contain known security vulnerabilities.

Addressing Vulnerable and Outdated Components vulnerabilities is crucial for maintaining the security of web applications. By actively managing software dependencies, staying updated with security patches, and following secure coding practices, organizations can reduce the risk of exploiting known vulnerabilities and enhance the overall security posture of their applications.

Here are some key points about OWASP Vulnerable and Outdated Components vulnerabilities:

Definition

Vulnerable and Outdated Components vulnerabilities occur when web applications use outdated or insecure versions of third-party software components, including libraries, frameworks, plugins, or modules.

9.5.7 Identification and Authentication Failures (A07-2021)

Introduction

OWASP (Open Web Application Security Project) Identification and Authentication Failures is a category of security vulnerabilities that relates to weaknesses and flaws in the identification and authentication mechanisms of web applications. These vulnerabilities can allow attackers to bypass or manipulate the authentication process, gain unauthorized access, or impersonate legitimate users.

Addressing Identification and Authentication Failures vulnerabilities is crucial for ensuring the security of web applications. By implementing secure authentication mechanisms, enforcing strong password policies, and regularly testing for vulnerabilities, organizations can reduce the risk of unauthorized access and enhance the overall security of their applications.

Here are some key points about OWASP Identification and Authentication Failures:

Definition

Identification and Authentication Failures refer to vulnerabilities that arise from weaknesses in the mechanisms used to identify and authenticate users of a web application.

9.5.8 Software and Data Integrity Failures (A08-2021)

Introduction

OWASP (Open Web Application Security Project) Software and Data Integrity Failures is a category of security vulnerabilities that relates to the integrity of software and data within web applications. These vulnerabilities involve the unauthorized modification, manipulation, or destruction of software components, configurations, or data, leading to potential security breaches and data integrity issues. Addressing Software and Data Integrity Failures vulnerabilities is crucial for maintaining the integrity and security of web applications. By following secure practices, implementing secure coding techniques, and regularly testing for vulnerabilities, organizations can mitigate the risk of unauthorized modifications, data tampering, and other integrity-related issues, thereby enhancing the overall security posture of their applications.
Here are some key points about OWASP Software and Data Integrity Failures:

Definition

Software and Data Integrity Failures refer to vulnerabilities that arise from weaknesses in ensuring the integrity and protection of software components, configurations, and data within web applications.

9.5.9 Security Logging and Monitoring Failures (A09-2021)

Introduction

OWASP Security Logging and Monitoring Failures is a category of security vulnerabilities that refers to the lack of effective logging and monitoring practices within web applications. These vulnerabilities involve deficiencies in the collection, analysis, and retention of security-related logs, as well as the absence of proper monitoring mechanisms to detect and respond to security incidents.
Addressing Security Logging and Monitoring Failures vulnerabilities is crucial for the timely detection, response, and mitigation of security incidents within web applications. By implementing comprehensive logging, effective log analysis, and real-time monitoring mechanisms, organizations
Here are some key points about OWASP Security Logging and Monitoring Failures:

Definition

Security Logging and Monitoring Failures refer to vulnerabilities that arise from weaknesses in the logging and monitoring capabilities of web applications, including the inadequate collection, analysis, and retention of security-related logs, and the lack of proper monitoring mechanisms.

9.5.10 Server-Side Request Forgery (A10-2021)

Introduction

OWASP Server-Side Request Forgery (SSRF) is a type of security vulnerability that occurs when an attacker can manipulate a web application to make unintended requests to internal or external resources

on the server-side. SSRF can lead to unauthorized access to sensitive data, service disruption, or even remote code execution.
By implementing robust input validation, secure configurations, and regular security testing, organizations can mitigate the risk of SSRF vulnerabilities and protect their web applications from unauthorized access and potential exploitation.
Here are some key points about OWASP Server-Side Request Forgery:

Definition
Server-Side Request Forgery (SSRF) refers to a vulnerability where an attacker can trick a vulnerable web application into making requests on behalf of the server to internal or external resources, often bypassing security controls.

10 Threat Mitigation analysis using SAMM

10.1 Introduction

The SAMM framework provides a structured approach to guide organizations in improving their software assurance practices, including addressing the OWASP Top 10 threats.
By aligning the SAMM domains with the specific risks outlined in the OWASP Top 10, organizations can focus their efforts on enhancing security practices in those areas.
Implementing the SAMM model is an ongoing process that requires continuous improvement. It involves regular assessments of the organization's maturity level, identification of gaps and risks, and the development and execution of action plans to address those gaps. By consistently monitoring progress and iterating on the analysis and improvement efforts, organizations can enhance their software assurance maturity and strengthen their security posture.
It's important to note that the OWASP Top 10 threats are regularly updated by the OWASP community to reflect emerging security risks. Therefore, organizations should stay up to date with the latest version of the OWASP Top 10 and adapt their analysis and improvement efforts accordingly.
By combining the SAMM framework with the knowledge and guidance provided by OWASP, organizations can effectively analyze and mitigate the most critical web application security risks, ultimately enhancing their overall software assurance maturity and reducing the likelihood of security incidents.
By using the SAMM framework to guide your analysis and improvement efforts, you can align your organization's security practices with the OWASP Top 10 threats and enhance the overall software assurance maturity.
This process should be an ongoing effort, continuously improving your organization's security posture.

10.2 Security by Governance

10.2.1 Introduction

SAMM (Software Assurance Maturity Model) recognizes the importance of security governance in ensuring effective software assurance practices within an organization. The Security by Governance domain in SAMM focuses on establishing and maintaining a robust security governance framework that enables effective decision-making, risk management, and oversight of software security activities.
By addressing the Security by Governance domain, organizations can establish a solid foundation for software security. It helps ensure that security is integrated into the organization's overall governance structure and that appropriate policies, processes, and controls are in place to support secure software development and operations
Here are some key aspects of the Security by Governance domain in SAMM:

#	Name	Description
1	Policies and Standards	This involves developing and implementing policies and standards that define the organization's expectations and requirements for software security. It includes areas such as secure coding practices, vulnerability management, incident response, and data protection.
2	Risk Management	This focuses on identifying and assessing risks associated with software development and deployment. It involves establishing risk management processes, conducting risk assessments, and implementing risk mitigation strategies to address identified risks.
3	Compliance Management	This involves ensuring compliance with relevant regulatory requirements, industry standards, and best practices related to software security. It includes activities such as compliance assessments, regulatory reporting, and alignment with security frameworks and guidelines.
4	Security Training and Awareness	This emphasizes the importance of educating and raising awareness among employees and stakeholders about software security. It involves providing training programs, awareness campaigns, and resources to promote a culture of security throughout the organization.
5	Security Metrics and Reporting	This involves defining and measuring key security metrics to assess the effectiveness of software security practices. It includes establishing reporting mechanisms to provide visibility into the status of software security and support decision-making at various levels of the organization.
6	Security Roles and Responsibilities	This focuses on clearly defining and assigning security roles and responsibilities within the organization. It includes establishing accountability for software security and ensuring that individuals have the necessary knowledge and skills to fulfill their roles effectively.

10.2.2 Policies and Standards

The Governance domain of the Software Assurance Maturity Model (SAMM) focuses on establishing policies, standards, and procedures to guide software security practices within an organization. In this

domain, SAMM provides guidance on developing and implementing effective policies and standards to support software security governance.

It's important to note that SAMM provides a framework for guiding the development of policies and standards within the Governance domain, but the actual policies and standards will vary depending on the organization's specific needs, industry, regulatory requirements, and risk appetite. Organizations should customize the policies and standards to align with their goals, objectives, and the unique aspects of their software development and security environment.

Here are some key considerations related to policies and standards within the SAMM Governance domain:

#	Name	Description
1	Policy Development	SAMM emphasizes the need for organizations to establish clear and comprehensive policies that define the organization's commitment to software security. Policies should outline the objectives, principles, and expectations for software security, and provide a foundation for establishing standards and procedures.
2	Standards Definition	SAMM encourages organizations to define standards that specify the requirements and best practices for software security. Standards help ensure consistency and provide guidance for implementing security controls and measures across the organization. These standards may cover areas such as secure coding practices, vulnerability management, access controls, and secure deployment processes.
3	Regulatory Compliance	SAMM recognizes the importance of complying with relevant laws, regulations, and industry standards related to software security. Policies and standards should address compliance requirements and ensure that software development and security practices align with applicable legal and regulatory obligations.
4	Security Roles and Responsibilities	SAMM highlights the importance of defining clear roles and responsibilities for software security within the organization. Policies and standards should outline the responsibilities of different stakeholders, including executives, developers, testers, security personnel, and other relevant parties. This helps ensure accountability and promotes a shared understanding of each individual's role in software security.
5	Secure Development Lifecycle	SAMM promotes the adoption of a secure development lifecycle (SDL) approach to software development. Policies and standards should define the specific phases, activities, and security controls to be integrated into the development process. They should also address considerations such as secure coding practices, code reviews, security testing, and secure deployment.
6	Risk Management	SAMM encourages organizations to integrate risk management practices into software security governance. Policies and standards should outline the processes for identifying, assessing, and managing software security risks. They should also define the risk tolerance levels and establish guidelines for risk treatment and mitigation strategies.

#	Name	Description
7	Security Incident Response	SAMM emphasizes the need for organizations to have robust incident response processes in place. Policies and standards should define the procedures for detecting, reporting, and responding to security incidents related to software. They should also outline the roles and responsibilities of incident response teams, escalation procedures, and post-incident analysis and remediation activities.
8	Security Metrics and Reporting	SAMM advocates for the use of metrics and reporting to measure and monitor software security performance. Policies and standards should outline the key metrics to be tracked, the frequency of reporting, and the stakeholders who should receive the reports. This helps organizations assess their security posture, identify areas for improvement, and communicate security-related information effectively.

10.2.2.1 Assessment – OWASP Top 10

SAMM's Governance domain, specifically through the establishment of policies and standards, can contribute to addressing several OWASP threats.

While SAMM's Governance domain provides guidance on policies and standards that can help address these OWASP threats, the effective implementation of these practices requires additional measures such as secure coding, regular security testing, and ongoing monitoring.

Here's how each contribution aligns with the respective OWASP threat:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's policies and standards can address this threat by defining access control requirements, specifying authentication and authorization mechanisms, and outlining proper access control configurations.
2	Cryptographic Failures (A02-2021)	SAMM's policies and standards can guide organizations in implementing secure cryptographic practices, such as using strong algorithms, proper key management, and secure encryption protocols.
3	Injection (A03-2021)	SAMM's policies and standards can emphasize secure coding practices, input validation, and parameterized queries to mitigate injection vulnerabilities and prevent untrusted data from being executed as code.
4	Insecure Design (A04-2021)	SAMM's policies and standards can promote secure design principles, threat modeling, and security requirements analysis to address insecure design decisions and prevent vulnerabilities in the overall architecture of web applications.
5	Security Misconfiguration (A05-2021)	SAMM's policies and standards can provide guidelines for secure configuration practices, including server, application, and framework configurations, to prevent misconfigurations that could expose sensitive information or introduce vulnerabilities.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	SAMM's policies and standards can emphasize the importance of managing software dependencies, keeping components up to date with security patches, and performing regular vulnerability assessments to address the risks associated with using outdated or vulnerable components.
7	Identification and Authentication Failures (A07-2021)	SAMM's policies and standards can guide organizations in implementing secure authentication mechanisms, enforcing strong password policies, and conducting proper user identity verification to prevent unauthorized access and mitigate identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	SAMM's policies and standards can promote secure coding practices, integrity checks, secure storage mechanisms, and encryption to ensure the integrity and protection of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's policies and standards can guide organizations in implementing comprehensive logging practices, establishing log analysis procedures, and setting up real-time monitoring mechanisms to detect and respond to security incidents effectively.
10	Server-Side Request Forgery (A10-2021)	SAMM's policies and standards can emphasize secure input validation, proper URL whitelisting, and server-side protection mechanisms to prevent server-side request forgery vulnerabilities and unauthorized requests to internal or external resources.

10.2.3 Risk Management

The Risk Management domain within the Software Assurance Maturity Model (SAMM) focuses on integrating risk management practices into software security governance. This domain helps organizations identify, assess, and manage software security risks effectively.

By addressing risk management within the SAMM Governance domain, organizations can effectively identify, assess, and manage software security risks, thereby reducing the likelihood and impact of security incidents and ensuring the overall security and integrity of their software assets.

Here are key considerations related to Risk Management within the SAMM Governance domain:

#	Name	Description
1	Risk Identification	SAMM emphasizes the importance of identifying and understanding the risks associated with software development and security. This involves conducting a comprehensive analysis of potential threats, vulnerabilities, and their potential impact on the organization's software assets.
2	Risk Assessment	SAMM encourages organizations to assess software security risks by evaluating the likelihood of occurrence and potential impact. This assessment helps prioritize risks based on their significance and determine appropriate mitigation strategies.

#	Name	Description
3	Risk Treatment	SAMM provides guidance on selecting and implementing risk treatment strategies to address identified risks. This includes measures such as risk mitigation, risk avoidance, risk transfer (e.g., insurance), or acceptance of residual risk.
4	Risk Monitoring	SAMM highlights the need for ongoing monitoring and reassessment of software security risks. This ensures that risk treatment measures remain effective and that new risks are identified and addressed in a timely manner.
5	Risk Communication	SAMM emphasizes the importance of effective communication of software security risks within the organization. This involves sharing risk-related information with relevant stakeholders, such as executives, developers, testers, and business owners, to foster a shared understanding of the risks and facilitate informed decision-making.
6	Risk Documentation	SAMM encourages organizations to maintain documentation of identified risks, risk assessments, risk treatment strategies, and their associated controls. This documentation serves as a reference for ongoing risk management activities and can support compliance requirements.
7	Risk Metrics	SAMM promotes the use of risk metrics to measure, track, and communicate software security risks. By defining appropriate risk metrics, organizations can gain insights into the effectiveness of risk management efforts and make data-driven decisions.
8	Integration with Other Domains	SAMM emphasizes the integration of risk management practices across all other domains within the model. This ensures that risk considerations are taken into account during policy development, secure development lifecycle, incident response, and other software security activities.

10.2.3.1 Assessment – OWASP Top 10

SAMM's Risk Management domain indirectly addresses the OWASP threats by providing a framework for identifying, assessing, treating, monitoring, and communicating risks associated with software development and security. While SAMM does not specifically mention the OWASP threats, its risk management practices can help organizations mitigate the vulnerabilities and risks that align with the OWASP Top 10. While SAMM provides a comprehensive framework for risk management, it is essential for organizations to apply it in conjunction with specific OWASP guidelines, best practices, and technical controls to address the OWASP threats effectively.

Here's how SAMM's Risk Management domain can contribute to addressing some of the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's risk identification and assessment processes can help identify and assess the risks associated with access control vulnerabilities.

#	Name	Description
2	Cryptographic Failures (A02-2021)	SAMM's risk assessment and treatment guidance can help organizations identify and address cryptographic vulnerabilities. By implementing proper cryptographic practices, organizations can reduce the risk of cryptographic failures.
3	Injection (A03-2021)	SAMM's risk identification and assessment practices can help organizations identify injection vulnerabilities. By implementing proper input validation and secure coding practices, organizations can mitigate the risk of injection attacks.
4	Insecure Design (A04-2021)	SAMM's emphasis on risk identification, assessment, and treatment can help organizations identify and address insecure design vulnerabilities. By adopting secure design principles and considering security from the early stages of development, organizations can mitigate the risk of insecure design.
5	Security Misconfiguration (A05-2021)	SAMM's risk identification and treatment practices can help organizations identify and address security misconfigurations. By following secure configuration practices and conducting regular security assessments, organizations can reduce the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's risk identification and treatment guidance can help organizations identify and address vulnerabilities arising from the use of outdated or vulnerable software components. By actively managing software dependencies and staying updated with security patches, organizations can reduce the risk of using vulnerable and outdated components
7	Identification and Authentication Failures (A07-2021)	SAMM's risk assessment and treatment practices can help organizations identify and address identification and authentication vulnerabilities. By implementing secure authentication mechanisms and regularly testing for vulnerabilities, organizations can mitigate the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	SAMM's risk identification and treatment practices can help organizations identify and address software and data integrity vulnerabilities. By following secure practices and implementing secure coding techniques, organizations can reduce the risk of software and data integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's emphasis on risk monitoring and communication can indirectly help organizations address security logging and monitoring failures. By establishing effective logging and monitoring practices and integrating them into the risk management process, organizations can enhance their ability to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	SAMM's risk identification and assessment practices can help organizations identify the risks associated with server-side request forgery vulnerabilities. By implementing robust input validation and secure configurations, organizations can mitigate the risk of server-side request forgery.

10.2.4 Compliance Management

SAMM's Governance domain addresses Compliance Management, which focuses on ensuring that software development and security practices align with applicable laws, regulations, standards, and

organizational policies. Compliance management involves establishing processes, controls, and documentation to demonstrate adherence to these requirements.

By incorporating Compliance Management practices from SAMM's Governance domain, organizations can ensure that their software development processes and security controls align with applicable compliance requirements. This helps mitigate legal and regulatory risks, protects sensitive data, and maintains trust with customers and stakeholders.

Here are some key aspects of SAMM's Compliance Management domain:

#	Name	Description
1	Compliance Controls	SAMM encourages organizations to implement specific controls to address compliance requirements. These controls may include access controls, data protection measures, secure coding practices, vulnerability management, incident response, and other relevant security controls. The selection and implementation of controls depend on the specific compliance requirements applicable to the organization.
2	Compliance Assessments	SAMM promotes regular assessments to evaluate compliance with applicable requirements. These assessments can include audits, reviews, and self-assessments to ensure that the software development process aligns with the defined compliance program. The assessment results help identify gaps and areas for improvement.
3	Compliance Documentation	SAMM emphasizes the importance of maintaining documentation related to compliance activities. This includes documenting compliance obligations, policies, controls, assessments, and any remediation actions taken. Documentation serves as evidence of compliance and supports internal and external audits or assessments.
4	Compliance Training and Awareness	SAMM recognizes the significance of training and raising awareness among software development teams and stakeholders about compliance requirements. Training programs can help individuals understand their roles and responsibilities regarding compliance, promote good practices, and foster a culture of compliance within the organization.
5	Compliance Monitoring and Reporting	SAMM highlights the need for ongoing monitoring of compliance with established controls and requirements. This involves tracking compliance metrics, conducting regular reviews, and generating reports to provide visibility into the organization's compliance posture. Monitoring and reporting enable timely identification of compliance issues and facilitate proactive remediation.
6	Compliance Governance	SAMM promotes the establishment of a governance framework that oversees compliance management activities. This includes defining roles and responsibilities, establishing accountability, and ensuring that compliance is integrated into the overall software development lifecycle and organizational processes.

10.2.4.1 Assessment – OWASP Top 10

The use of SAMM's Governance domain - Compliance Management does not directly address the specific OWASP threats you mentioned. SAMM focuses on software assurance and aligning development practices with compliance requirements, while the OWASP Top 10 list highlights specific web application security vulnerabilities.

However, it's important to note that SAMM's Compliance Management domain can indirectly contribute to addressing these threats by promoting secure development practices and risk management. By establishing a compliance program, implementing appropriate controls, conducting assessments, and maintaining documentation, organizations can create a robust software development lifecycle that inherently reduces the likelihood of OWASP threats.

It's important to note that addressing these OWASP threats requires a comprehensive approach that goes beyond compliance management. Organizations should consider additional security measures, such as secure coding practices, secure architecture design, continuous vulnerability testing, and regular security assessments, to effectively mitigate these threats.

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's emphasis on compliance controls and assessments can help identify and rectify access control weaknesses, ensuring that proper authorization mechanisms are in place to prevent unauthorized access.
2	Cryptographic Failures (A02-2021)	SAMM's focus on compliance controls and assessments can help organizations implement and validate proper cryptographic practices, reducing the likelihood of cryptographic vulnerabilities.
3	Injection (A03-2021)	SAMM's emphasis on secure coding practices, compliance controls, and assessments can help organizations prevent and identify injection vulnerabilities by ensuring proper input validation and handling.
4	Insecure Design (A04-2021)	SAMM's compliance program can promote secure design principles and risk management, leading to more robust and secure web application architectures.
5	Security Misconfiguration (A05-2021)	SAMM's compliance controls, assessments, and monitoring can help organizations ensure that configurations align with secure practices, reducing the risk of misconfigurations and associated vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's compliance program and controls can promote the management of software dependencies, including the regular update of components to mitigate the risk of known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	SAMM's focus on compliance controls, training, and assessments can help organizations improve authentication mechanisms, enforce strong password policies, and address identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	SAMM's compliance controls, secure coding practices, and assessments can contribute to ensuring the integrity and protection of software components, configurations, and data.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's compliance program emphasizes monitoring and reporting, which can help organizations enhance their logging and monitoring capabilities to detect and respond to security incidents more effectively.
10	Server-Side Request Forgery (A10-2021)	While SAMM's focus is not specifically on SSRF, organizations following SAMM's compliance management practices can adopt secure coding practices and implement proper input validation to mitigate the risk of SSRF vulnerabilities.

10.2.5 Security Training and Awareness

SAMM's Governance domain addresses Security Training and Awareness, which focuses on educating and raising awareness among software development teams and stakeholders about security best practices, threats, and mitigation strategies. By emphasizing security training and awareness, SAMM aims to cultivate a culture of security within an organization and empower individuals to make informed decisions that enhance the overall security posture.

By incorporating security training and awareness practices from SAMM's Governance domain, organizations can significantly improve their security posture. Security-aware individuals are better equipped to identify and mitigate security risks, make informed decisions, and contribute to the overall security of the software development process.

Here are some key aspects of SAMM's Security Training and Awareness domain:

#	Name	Description
1	Training Program	SAMM advocates for the development and implementation of a comprehensive security training program. This program should cover various topics related to secure coding practices, secure software development lifecycle, threat modeling, secure configuration management, vulnerability management, and other relevant security concepts. The training program should cater to different roles and responsibilities within the organization, ensuring that individuals have the necessary knowledge to perform their tasks securely.
2	Awareness Campaigns	SAMM encourages organizations to conduct awareness campaigns to promote security awareness and best practices. These campaigns can include regular communications, newsletters, posters, and workshops to educate employees about current security threats, emerging trends, and preventive measures. The goal is to keep security at the forefront of individuals' minds and foster a security-conscious culture.
3	Role-based Training	SAMM emphasizes the importance of providing role-based training that is tailored to the specific security responsibilities of each individual within the software development lifecycle. Different roles, such as developers, testers, architects, and project managers, require specific knowledge and skills related to security. By providing targeted training, organizations can ensure that individuals have the necessary expertise to perform their security-related tasks effectively.

#	Name	Description
4	Security Champions	SAMM recommends the identification and training of security champions within the organization. These individuals act as advocates for security, providing guidance, mentoring, and support to others. Security champions play a crucial role in promoting security awareness, assisting with security-related activities, and driving the adoption of secure practices across the organization.
5	Continuous Learning	SAMM emphasizes the need for continuous learning and professional development in the field of software security. It encourages organizations to provide opportunities for individuals to enhance their security knowledge and skills through training, certifications, conferences, and participation in security communities. By promoting continuous learning, organizations can keep up with evolving security threats and technologies.
6	Metrics and Evaluation	SAMM suggests establishing metrics to evaluate the effectiveness of security training and awareness programs. These metrics can include factors such as the number of individuals trained, training completion rates, feedback from participants, and improvements in security-related practices. Regular evaluation helps identify areas for improvement and ensures that the training and awareness efforts are achieving the desired outcomes.

10.2.5.1 Assessment – OWASP Top 10

Using SAMM's Governance domain - Security Training and Awareness can indirectly address several OWASP threats by promoting security best practices, raising awareness about potential vulnerabilities, and providing individuals with the necessary knowledge and skills to mitigate risks.

While SAMM's Security Training and Awareness domain does not directly address all OWASP threats, it plays a crucial role in promoting security knowledge, awareness, and best practices within organizations. By instilling a culture of security and providing individuals with the necessary skills, organizations can significantly reduce the likelihood of many OWASP threats.

Here is how SAMM's Security Training and Awareness domain contributes to addressing some of the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Broken Access Control (A01-2021): By providing training on secure coding practices, threat modeling, and secure configuration management, SAMM helps developers understand the importance of proper access controls and how to implement them correctly, reducing the risk of broken access control vulnerabilities.
2	Injection (A03-2021)	SAMM's role-based training emphasizes secure coding practices, input validation, and parameterization. By educating developers on how to handle user input securely, SAMM contributes to mitigating injection vulnerabilities.

#	Name	Description
3	Cryptographic Failures (A02-2021)	SAMM's training program covers relevant security concepts, including cryptographic algorithms, key management, and secure usage. By educating developers and architects about cryptographic best practices, SAMM helps prevent common cryptographic failures in web applications.
4	Insecure Design (A04-2021)	SAMM's emphasis on secure design principles and early consideration of security helps address insecure design vulnerabilities. By providing guidance on architectural decisions and security controls, SAMM promotes secure design practices in web application development.
5	Security Misconfiguration (A05-2021)	SAMM's training program covers secure configuration management, ensuring that developers and system administrators understand the risks of insecure configurations. By raising awareness about security misconfigurations, SAMM helps organizations avoid common pitfalls in web application deployments.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's continuous learning aspect encourages staying updated with security patches and best practices. By promoting the management of software dependencies and awareness of vulnerable components, SAMM contributes to reducing the risk associated with outdated or insecure third-party software.
7	Identification and Authentication Failures (A07-2021)	SAMM's role-based training focuses on secure authentication mechanisms and enforcing strong password policies. By educating developers and architects about secure identification and authentication practices, SAMM helps address vulnerabilities in this area.
8	Software and Data Integrity Failures (A08-2021)	SAMM's training program covers secure coding techniques and emphasizes the importance of data integrity. By promoting secure practices and regular testing for vulnerabilities, SAMM contributes to mitigating software and data integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Although SAMM's Security Training and Awareness domain does not directly address this threat, it can indirectly contribute by emphasizing the importance of logging and monitoring in secure software development. Organizations implementing SAMM may be more likely to incorporate effective logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	SAMM's training program covers secure coding practices and input validation. By educating developers about the risks of server-side request forgery and how to prevent it, SAMM helps mitigate vulnerabilities in this area.

10.2.6 Security Metrics and Reporting

SAMM's Governance domain includes the aspect of Security Metrics and Reporting, which focuses on establishing metrics and reporting mechanisms to track and measure the effectiveness of an organization's software security efforts. This domain recognizes the importance of collecting relevant data, analyzing it, and generating reports that provide valuable insights into the security posture of the software development process. By implementing security metrics and reporting practices, organizations can better understand

their security performance, identify areas for improvement, and make informed decisions to enhance their overall software security.

Implementing SAMM's Security Metrics and Reporting practices enables organizations to measure, monitor, and report on the effectiveness of their software security efforts. By having clear metrics and meaningful reports, organizations can identify strengths and weaknesses, allocate resources effectively, and make informed decisions to enhance their overall software security.

Here are some key aspects of SAMM's Security Metrics and Reporting domain:

#	Name	Description
1	Metric Identification	SAMM encourages organizations to identify and define appropriate security metrics that align with their software development goals and objectives. These metrics should be measurable, meaningful, and provide insights into the effectiveness of security practices and controls.
2	Reporting Framework	SAMM emphasizes the importance of collecting accurate and reliable data related to security activities, incidents, vulnerabilities, and other relevant aspects. This data can be obtained from various sources, such as security tools, incident reports, code reviews, and security assessments. Organizations should analyze this data to identify trends, patterns, and areas of concern.
3	Reporting Framework	SAMM advocates for establishing a reporting framework that outlines the frequency, format, and stakeholders of security reports. These reports should provide meaningful information about the current state of software security, progress towards security goals, identified risks, and recommended actions. The reports should be tailored to different audiences, such as management, development teams, and security personnel.
4	Key Performance Indicators (KPIs)	SAMM encourages the use of KPIs to measure the effectiveness of security practices and controls. KPIs should be aligned with organizational goals and objectives, and they should provide actionable insights. Examples of security KPIs may include the number of vulnerabilities discovered and remediated, the time taken to resolve security incidents, or the percentage of developers trained in secure coding practices.
5	Benchmarking and Comparison	SAMM promotes benchmarking against industry best practices and standards to assess the organization's security performance. By comparing their security metrics and practices against peers or industry benchmarks, organizations can identify areas where they may be lagging behind or excelling, enabling them to prioritize their improvement efforts.
6	Continuous Improvement	SAMM emphasizes the need for continuous improvement in security metrics and reporting. Organizations should regularly review and update their metrics based on changing business needs, evolving threats, and lessons learned. By continually refining their metrics and reporting practices, organizations can drive ongoing improvements in their software security posture.

10.2.6.1 Assessment – OWASP Top 10

SAMM's Governance domain, specifically the Security Metrics and Reporting aspect, indirectly addresses several OWASP threats by promoting a proactive and data-driven approach to software security. While SAMM doesn't explicitly mention individual OWASP threats, it provides a framework for organizations to assess, measure, and improve their software security practices, which can help mitigate various vulnerabilities and risks.

While SAMM's Security Metrics and Reporting domain doesn't directly address each OWASP threat individually, it provides a comprehensive framework for organizations to establish metrics, collect data, and generate reports that can contribute to addressing various vulnerabilities and risks associated with software security. By implementing SAMM's practices, organizations can enhance their overall software security posture and indirectly mitigate OWASP threats.

Here's a breakdown of how SAMM's Security Metrics and Reporting can contribute to addressing OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	By establishing security metrics related to access controls, organizations can track and measure the effectiveness of their access control mechanisms, identify access control weaknesses, and take corrective actions to address broken access control vulnerabilities
2	Cryptographic Failures (A02-2021)	Implementing security metrics around cryptographic practices can help organizations assess the correctness and effectiveness of their cryptographic implementation, identify weaknesses, and improve their cryptographic controls to address cryptographic failures.
3	Injection (A03-2021)	Security metrics and reporting can enable organizations to track the prevalence of injection vulnerabilities, identify the root causes, and prioritize remediation efforts to address injection vulnerabilities in a timely manner.
4	Insecure Design (A04-2021)	By incorporating security metrics related to secure design principles, organizations can assess the maturity of their design practices, identify insecure design decisions, and implement improvements to mitigate insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Implementing security metrics for configuration management can help organizations identify security misconfigurations, track their prevalence, and establish processes to ensure secure configurations, thus reducing security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Security metrics and reporting can help organizations track the usage of third-party components, monitor their vulnerability status, and establish processes for managing and updating these components, thereby addressing vulnerabilities associated with vulnerable and outdated components
7	Identification and Authentication Failures (A07-2021)	By establishing metrics related to identification and authentication mechanisms, organizations can assess the effectiveness of these controls, identify weaknesses, and take actions to address identification and authentication failures.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	Security metrics can help organizations track and measure the integrity of software components and data, identify integrity-related vulnerabilities, and implement measures to protect against software and data integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Implementing security metrics related to logging and monitoring can help organizations assess the effectiveness of their logging and monitoring practices, identify gaps or failures in these areas, and improve their incident detection and response capabilities.
10	Server-Side Request Forgery (A10-2021)	Security metrics and reporting can contribute to identifying and measuring the prevalence of server-side request forgery vulnerabilities, allowing organizations to implement measures to prevent and detect such attacks.

10.2.7 Security Roles and Responsibilities

SAMM's Governance domain addresses Security Roles and Responsibilities, which focuses on defining and assigning security-related roles and responsibilities within an organization's software development process. This domain emphasizes the importance of clear accountability and a well-defined division of security duties among individuals involved in software development.

By incorporating SAMM's Security Roles and Responsibilities practices, organizations can effectively assign and define security-related roles, clarify responsibilities, and establish a framework for collaboration and accountability. This helps ensure that security considerations are properly integrated into the software development process, enhancing the overall security posture of the organization's software products and services.

Here are some key aspects of SAMM's Security Roles and Responsibilities domain:

#	Name	Description
1	Role Definitions	SAMM encourages organizations to define specific security-related roles within their software development teams. These roles may include security architects, security analysts, security testers, security champions, and other relevant positions. By clearly defining these roles, organizations ensure that individuals have dedicated responsibilities and expertise in addressing security concerns.
2	Role Descriptions and Responsibilities	SAMM advocates for creating detailed role descriptions that outline the responsibilities and expectations of each security role. These descriptions should clearly define the knowledge, skills, and tasks associated with each role. By providing clarity on responsibilities, organizations ensure that individuals understand their security-related duties and can fulfill them effectively

#	Name	Description
3	Security Training and Awareness	SAMM emphasizes the need for security training and awareness among individuals in security roles. By providing relevant training programs and resources, organizations enable individuals to develop the necessary knowledge and skills to perform their security-related responsibilities effectively. Training should cover topics such as secure coding practices, secure software development lifecycle, threat modeling, and vulnerability management.
4	Collaboration and Communication	SAMM promotes effective collaboration and communication among individuals in security roles and other stakeholders. It encourages regular interactions, sharing of security information, and coordination of efforts to address security challenges. Collaboration ensures that security considerations are integrated into the software development process and that potential security issues are identified and addressed proactively.
5	Incident Response and Reporting	SAMM emphasizes the importance of incident response procedures and reporting mechanisms. It defines the roles and responsibilities of individuals involved in handling security incidents, including incident response teams and incident coordinators. By clearly defining these roles, organizations ensure a swift and coordinated response to security incidents, minimizing the impact and facilitating effective incident resolution.
6	Security Governance	SAMM's Security Roles and Responsibilities domain aligns with overall security governance practices within an organization. It promotes the establishment of governance frameworks, policies, and procedures that guide the activities of individuals in security roles. This ensures that security responsibilities are aligned with organizational goals, regulatory requirements, and industry best practices.

10.2.7.1 Assessment – OWASP Top 10

SAMM's Governance domain, specifically the Security Roles and Responsibilities aspect, indirectly addresses several OWASP threats by promoting security practices, collaboration, and accountability. While SAMM's Security Roles and Responsibilities domain does not directly address each OWASP threat, its principles and practices can have a positive impact on mitigating these threats by fostering a security-conscious culture, promoting collaboration, and integrating security considerations into the software development process.

Although SAMM does not provide direct mitigation techniques for each OWASP threat, the implementation of SAMM's Security Roles and Responsibilities can contribute to addressing the following threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's emphasis on role definitions, role descriptions, and responsibilities helps ensure that access control mechanisms are properly implemented and enforced, reducing the risk of unauthorized access to resources.

#	Name	Description
2	Cryptographic Failures (A02-2021)	SAMM's emphasis on security training and awareness helps individuals in security roles understand proper cryptographic practices and implement them correctly, reducing the likelihood of cryptographic vulnerabilities.
3	Injection (A03-2021)	SAMM's focus on collaboration and communication encourages secure coding practices and regular testing for vulnerabilities, which can help prevent injection attacks by promoting proper input validation and parameterization.
4	Insecure Design (A04-2021)	SAMM's emphasis on security governance and secure design principles helps organizations address insecure design vulnerabilities by considering security from the early stages of development and making informed architectural decisions.
5	Security Misconfiguration (A05-2021)	SAMM's focus on security governance, policies, and procedures helps organizations establish secure configuration practices and avoid insecure defaults, reducing the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's emphasis on security training and awareness promotes the adoption of secure coding practices and the active management of software dependencies, reducing the risk of using outdated or vulnerable components.
7	Identification and Authentication Failures (A07-2021)	SAMM's emphasis on role definitions and responsibilities can contribute to the implementation of secure authentication mechanisms and strong password policies, mitigating the risk of identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	SAMM's focus on security training and awareness helps individuals understand the importance of data integrity and the need for secure coding techniques, reducing the risk of unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's emphasis on incident response and reporting promotes the establishment of effective logging and monitoring practices, enhancing the ability to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	SAMM's emphasis on secure configurations and regular security testing can contribute to mitigating Server-Side Request Forgery vulnerabilities by promoting input validation and preventing unintended requests to internal or external resources.

10.3 Security by Design

10.3.1 Introduction

The Software Assurance Maturity Model (SAMM) Design domain focuses on ensuring that software applications are designed with security in mind. It provides guidelines and best practices for incorporating security controls and considerations into the design phase of software development. While the SAMM model does not explicitly address each individual OWASP threat, it provides a framework to improve an organization's security practices, which indirectly helps address the OWASP threats.

While the SAMM Design domain provides a comprehensive approach to designing secure software, it is important to note that addressing the specific details of each OWASP threat requires organizations to refer to the OWASP documentation and guidelines directly. SAMM provides a framework and guidance for improving software assurance practices, and when combined with OWASP resources, organizations can effectively address the OWASP threats throughout the design phase of software development.
Here are some key aspects of the SAMM Design domain that contribute to mitigating the OWASP threats:

#	Name	Description
1	Security Requirements	SAMM emphasizes the importance of defining and incorporating security requirements into the design phase. By identifying and addressing specific security concerns, organizations can proactively mitigate vulnerabilities associated with OWASP threats such as Broken Access Control, Injection, Insecure Design, and others.
2	Secure Architecture	SAMM encourages the adoption of secure architectural patterns and principles. Designing a robust and secure architecture helps address various OWASP threats such as Insecure Design, Cryptographic Failures, Security Misconfiguration, and others by minimizing potential vulnerabilities and creating a strong foundation for security controls.
3	Secure Coding Practices:	SAMM promotes the use of secure coding practices during the design phase. By adhering to secure coding guidelines, organizations can reduce the likelihood of OWASP threats such as Injection, Insecure Design, Security Misconfiguration, and others by preventing common coding mistakes and vulnerabilities.
4	Threat Modeling	SAMM suggests conducting threat modeling exercises during the design phase to identify and mitigate potential security risks. Threat modeling helps organizations assess the impact of OWASP threats and enables them to implement appropriate security controls and countermeasures.
5	Security Testing	SAMM encourages the inclusion of security testing activities in the design phase. By incorporating security testing techniques, such as static code analysis and security code reviews, organizations can detect and address vulnerabilities related to OWASP threats, ensuring that the design is resilient to potential attacks.

10.3.2 Security Requirements

In the SAMM (Software Assurance Maturity Model) Design domain, one of the key aspects is Security Requirements. This aspect focuses on ensuring that security requirements are identified, documented, and incorporated into the design phase of software development. By addressing security requirements, organizations can proactively mitigate vulnerabilities and address OWASP threats.
By focusing on these aspects, the Security Requirements aspect of the SAMM Design domain contributes to designing software applications that are resilient against OWASP threats. It helps organizations identify and incorporate security controls and measures from the early stages of design, reducing the risk of vulnerabilities and ensuring a more secure software development process.
Here are some key contributions of the Security Requirements aspect:

are some key aspects of the SAMM Design domain that contribute to mitigating the OWASP threats:

#	Name	Description
1	Identify Security Objectives	The Security Requirements aspect helps organizations identify and define specific security objectives for their software application. This includes determining the confidentiality, integrity, and availability requirements, as well as identifying specific security controls and measures needed to address OWASP threats effectively.
2	Risk Assessment	It involves conducting a risk assessment to identify potential security risks and threats that the software application may face. By understanding the potential risks, organizations can prioritize and define security requirements that address specific OWASP threats, such as Injection, Broken Access Control, Cryptographic Failures, and others.
3	Compliance and Regulatory Requirements	The Security Requirements aspect ensures that the software application meets the necessary compliance and regulatory requirements related to security. This includes identifying and incorporating security requirements specified by industry standards, legal regulations, and contractual obligations.
4	Secure Data Handling	It focuses on defining security requirements for handling sensitive data within the software application. This includes specifying encryption requirements, data storage and transmission requirements, and access control requirements to mitigate OWASP threats related to data security.
5	Authentication and Authorization	The Security Requirements aspect addresses the design considerations for authentication and authorization mechanisms within the software application. This includes specifying requirements for strong and secure authentication, proper authorization controls, and mitigation of privilege escalation risks.
6	Secure Configuration Management	It involves defining requirements for secure configuration management of the software application and its supporting infrastructure. This includes specifying secure baseline configurations, secure storage and management of secrets and credentials, and requirements for regular updates and patching.
7	Secure Communication Protocols	The Security Requirements aspect addresses the selection and implementation of secure communication protocols for the software application. This includes specifying requirements for secure network protocols, encryption, and secure communication channels to mitigate OWASP threats related to data transmission.

10.3.2.1 Assessment – OWASP Top 10

By addressing these aspects, the SAMM Design domain - Security Requirements contributes to designing software applications that are resilient against OWASP threats and helps organizations identify and incorporate security controls and measures from the early stages of design.

The Software Assurance Maturity Model (SAMM) Design domain - Security Requirements addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's Security Requirements aspect helps organizations identify and define specific security objectives, including access controls, to address broken access control vulnerabilities
2	Cryptographic Failures (A02-2021)	SAMM's Security Requirements aspect emphasizes proper cryptographic practices and the implementation of secure algorithms to mitigate cryptographic failures.
3	Injection (A03-2021)	SAMM's Security Requirements aspect promotes proper input validation and secure coding practices to prevent injection vulnerabilities
4	Insecure Design (A04-2021)	SAMM's Security Requirements aspect encourages the adoption of secure design principles to address insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	SAMM's Security Requirements aspect emphasizes the importance of secure configurations and adherence to industry standards to mitigate security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's Security Requirements aspect focuses on actively managing software dependencies and staying updated with security patches to address vulnerabilities associated with vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	SAMM's Security Requirements aspect addresses the design considerations for secure identification and authentication mechanisms to mitigate identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	SAMM's Security Requirements aspect promotes secure practices, secure coding techniques, and regular testing to ensure the integrity and protection of software and data.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's Security Requirements aspect emphasizes the need for effective logging and monitoring practices to detect and respond to security incidents and prevent security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	SAMM's Security Requirements aspect encourages robust input validation, secure configurations, and regular security testing to mitigate server-side request forgery vulnerabilities.

10.3.3 Secure Architecture

In the SAMM (Software Assurance Maturity Model) Design domain, another key aspect is Secure Architecture. This aspect focuses on designing software applications with a strong and robust architecture that incorporates security principles and mitigates potential vulnerabilities. By addressing secure architecture, organizations can enhance the overall security posture of their software applications.

By incorporating these considerations and contributions into the design phase, the Secure Architecture aspect of the SAMM Design domain helps organizations build software applications with a strong foundation of security. It enables the identification and implementation of security controls, patterns, and best practices that mitigate potential vulnerabilities and enhance the overall security of the application.

Here are some key considerations and contributions of the Secure Architecture aspect:

#	Name	Description
1	Threat Modeling	Secure Architecture involves conducting a threat modeling exercise to identify potential security threats and vulnerabilities specific to the software application. This includes analyzing the application's components, interfaces, and interactions to determine potential attack vectors and risks.
2	Secure Design Patterns	The Secure Architecture aspect promotes the use of secure design patterns and best practices in the development of software applications. Secure design patterns provide proven solutions to common security challenges and help architects make informed decisions during the design phase.
3	Defense-in-Depth	Secure Architecture emphasizes the implementation of a defense-in-depth strategy, which involves layering multiple security measures throughout the application's architecture. This approach ensures that even if one security control is bypassed, there are additional layers of protection to mitigate the risk.
4	Secure Communication Channels	The Secure Architecture aspect includes designing and implementing secure communication channels for data exchange between different components or systems. This involves using encryption, secure protocols, and secure transmission mechanisms to protect the confidentiality and integrity of data in transit.
5	Secure Data Storage	It focuses on designing secure data storage mechanisms within the software application. This includes considering encryption, access controls, secure key management, and secure storage practices to protect sensitive data from unauthorized access or compromise.
6	Secure Integration	Secure Architecture considers the secure integration of various components, modules, or external systems within the software application. This involves validating and sanitizing input, implementing secure interfaces, and ensuring that integrated components adhere to security requirements.
7	Scalability and Performance	The Secure Architecture aspect also takes into account the scalability and performance requirements of the software application while maintaining security. It ensures that security controls and measures do not hinder the application's performance and can scale effectively as the application grows.
8	Secure Deployment	Secure Architecture includes considerations for secure deployment practices, such as secure configuration of servers, hardening of the infrastructure, and secure installation and update processes. This helps ensure that the deployed application maintains its security posture in production environments.

10.3.3.1 Assessment – OWASP Top 10

While the SAMM Design Domain - Secure Architecture does not directly address each OWASP threat individually, it provides a framework and set of practices that contribute to mitigating these threats by incorporating security principles and best practices into the software architecture design process.
The SAMM Design Domain - Secure Architecture addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	By conducting threat modeling and implementing proper access controls, secure architecture helps address and mitigate the risks associated with broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	Secure architecture promotes the use of secure design patterns and best practices, including proper implementation and usage of cryptographic algorithms and mechanisms, to address cryptographic failures.
3	Injection (A03-2021)	Through secure design patterns, input validation, and secure coding practices, secure architecture helps mitigate the risks associated with injection vulnerabilities.
4	Insecure Design (A04-2021)	Secure architecture promotes the adoption of secure design principles and practices, reducing the risk of insecure design decisions that can lead to vulnerabilities.
5	Security Misconfiguration (A05-2021)	Secure architecture includes considerations for secure configuration practices, ensuring that web applications and associated components are properly configured to avoid security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Secure architecture emphasizes actively managing software dependencies, including third-party libraries and frameworks, to address vulnerabilities related to using outdated or known vulnerable components.
7	Identification and Authentication Failures (A07-2021)	By implementing secure authentication mechanisms and enforcing strong password policies, secure architecture helps mitigate identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	Secure architecture promotes secure coding techniques and practices, ensuring the integrity and protection of software components, configurations, and data.
9	Security Logging and Monitoring Failures (A09-2021)	Secure architecture includes considerations for comprehensive logging and effective monitoring mechanisms, addressing vulnerabilities related to security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	Through robust input validation, secure configurations, and regular security testing, secure architecture helps mitigate the risks associated with server-side request forgery vulnerabilities.

10.3.4 Secure Coding Practices

In the SAMM (Software Assurance Maturity Model) Design domain, another key aspect is Secure Coding Practices. This aspect focuses on promoting secure coding techniques and practices to prevent and mitigate potential vulnerabilities in software applications. By addressing secure coding practices, organizations can enhance the security of their codebase and reduce the risk of security breaches.
By incorporating these considerations and contributions into the software development process, the Secure Coding Practices aspect of the SAMM Design domain helps organizations build software applications with a strong foundation of security. It enables the identification and mitigation of vulnerabilities through secure

coding techniques, adherence to coding standards, and continuous evaluation of the codebase for potential security risks.

Here are some key considerations and contributions of the Secure Coding Practices aspect:

#	Name	Description
1	Secure Coding Standards	Secure Coding Practices emphasize the adoption and adherence to secure coding standards. These standards provide guidelines and best practices for writing secure code, including input validation, output encoding, proper error handling, secure authentication, and secure data storage.
2	Input Validation	Secure Coding Practices promote the proper validation of all user input to prevent common vulnerabilities such as injection attacks (e.g., SQL injection, cross-site scripting). Input validation includes checking data types, length limits, and using whitelisting or regular expressions to ensure the input is valid and safe.
3	Output Encoding	Secure Coding Practices emphasize the proper encoding of output data to prevent cross-site scripting (XSS) attacks. Output encoding involves converting special characters into their corresponding HTML entities or using context-specific encoding functions to ensure that user-supplied data is displayed as intended without introducing security risks.
4	Error Handling	Secure Coding Practices address the proper handling of errors and exceptions in software applications. This includes avoiding the display of sensitive error messages to users, logging relevant error information securely, and implementing appropriate error handling mechanisms to prevent information leakage or system vulnerabilities.
5	Secure Authentication	Secure Coding Practices focus on implementing secure authentication mechanisms to prevent unauthorized access. This includes using strong and properly hashed passwords, implementing multi-factor authentication (MFA), preventing brute force attacks, and protecting authentication credentials during transmission and storage.
6	Secure Data Storage	Secure Coding Practices address the secure storage of sensitive data within the application. This includes properly encrypting sensitive data, securely managing encryption keys, implementing access controls to limit data access, and protecting against common vulnerabilities such as insecure direct object references or insecure file storage.
7	Secure Session Management	Secure Coding Practices include considerations for secure session management to prevent session hijacking or session fixation attacks. This involves using secure session identifiers, properly handling session expiration, and implementing mechanisms to protect session data from unauthorized access or tampering.
8	Secure File Handling	Secure Coding Practices promote secure file handling techniques to prevent common vulnerabilities such as arbitrary file upload or directory traversal attacks. This includes validating file types, using secure file storage locations, and implementing access controls to restrict file access.

#	Name	Description
9	Secure Communication	Secure Coding Practices address secure communication between components or systems. This includes using secure protocols (e.g., HTTPS), encrypting sensitive data during transmission, properly configuring SSL/TLS settings, and implementing secure APIs or web services.
10	Code Reviews and Static Analysis	Secure Coding Practices encourage regular code reviews and the use of static analysis tools to identify and address potential security vulnerabilities in the codebase. This helps catch security issues early in the development process and ensures that secure coding practices are followed consistently.

10.3.4.1 Assessment – OWASP Top 10

The SAMM Design Domain - Secure Coding Practices contributes directly to addressing various OWASP threats by promoting secure coding techniques, adherence to coding standards, and continuous evaluation of the codebase for potential security risks.

The SAMM Design Domain - Secure Coding Practices addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	By emphasizing secure coding practices and adherence to secure coding standards, the SAMM Design Domain promotes the implementation and enforcement of proper access controls to prevent unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Secure Coding Practices in SAMM emphasize the correct implementation of cryptographic algorithms and mechanisms, including the use of strong encryption, proper key management, and secure configurations, to address vulnerabilities related to cryptographic failures.
3	Injection (A03-2021)	SAMM's Secure Coding Practices focus on proper input validation and handling of untrusted data, mitigating the risk of injection attacks by preventing the injection of malicious code or unintended commands into the application.
4	Insecure Design (A04-2021)	The Secure Coding Practices aspect promotes the adoption of secure design principles and consideration of security throughout the development lifecycle, helping organizations address flaws and weaknesses in the overall design and architecture of web applications.
5	Security Misconfiguration (A05-2021)	SAMM's Secure Coding Practices encourage secure configuration practices, regular updates, and security assessments to mitigate security misconfigurations and prevent unnecessary exposure of sensitive information or entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	By advocating for secure coding practices and regular code reviews, SAMM helps address the use of outdated or vulnerable software components, reducing the risk associated with relying on such components in web applications.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	The Secure Coding Practices aspect of SAMM emphasizes the implementation of secure authentication mechanisms, strong password policies, and regular vulnerability testing to address weaknesses in the identification and authentication mechanisms of web applications.
8	Software and Data Integrity Failures (A08-2021)	SAMM's Secure Coding Practices contribute to addressing integrity vulnerabilities by promoting secure coding techniques, implementing proper data validation, and conducting regular vulnerability testing to protect against unauthorized modifications and data tampering
9	Security Logging and Monitoring Failures (A09-2021)	The Secure Coding Practices aspect of SAMM encourages the implementation of comprehensive logging and effective log analysis, as well as real-time monitoring mechanisms, to address vulnerabilities related to insufficient logging and monitoring capabilities in web applications.
10	Server-Side Request Forgery (A10-2021)	Secure Coding Practices in SAMM, such as robust input validation and secure configurations, help mitigate the risk of SSRF vulnerabilities by preventing attackers from tricking the application into making unintended requests to internal or external resources.

10.3.5 Threat Modeling

In the SAMM (Software Assurance Maturity Model) Design domain, Threat Modeling is an essential aspect that helps organizations identify and analyze potential threats and vulnerabilities in their software applications.

By incorporating threat modeling into the software development process, organizations can proactively address security risks and make informed decisions to enhance the overall security posture of their applications.

By incorporating these considerations and contributions into the software development process, the Threat Modeling aspect of the SAMM Design domain helps organizations proactively address security risks, prioritize security efforts, and make informed decisions to enhance the overall security of their software applications.

Here are some key considerations and contributions of the Threat Modeling aspect:

#	Name	Description
1	Identification of Threats	Threat Modeling in SAMM involves systematically identifying potential threats to the application. This includes considering various threat sources, such as malicious actors, system vulnerabilities, or environmental factors, and understanding how they could exploit vulnerabilities within the application.

#	Name	Description
2	Asset Identification	SAMM's Threat Modeling aspect emphasizes identifying and prioritizing critical assets within the application. This includes identifying sensitive data, system resources, or functionality that could be attractive to attackers and ensuring that appropriate security measures are in place to protect these assets.
3	Vulnerability Analysis	Threat Modeling involves analyzing the application's design, architecture, and implementation to identify potential vulnerabilities that could be exploited by attackers. This analysis helps prioritize security efforts and determine where security controls should be applied to mitigate the identified vulnerabilities.
4	Attack Surface Analysis	SAMM's Threat Modeling aspect includes assessing the application's attack surface, which refers to the entry points or paths that attackers could exploit to gain unauthorized access or compromise the application's security. By understanding the attack surface, organizations can focus their efforts on securing critical entry points and reducing the overall risk exposure.
5	Mitigation Strategies	Threat Modeling helps organizations develop effective mitigation strategies to address identified threats and vulnerabilities. This may involve implementing security controls, applying secure design principles, or making architectural changes to reduce the impact and likelihood of successful attacks.
6	Risk Assessment	SAMM's Threat Modeling aspect encourages conducting risk assessments to evaluate the potential impact and likelihood of identified threats. By assessing risks, organizations can prioritize their security efforts, allocate resources effectively, and make informed decisions about risk acceptance, mitigation, or transfer.
7	Security Requirements	Threat Modeling contributes to defining security requirements for the application. By understanding potential threats and vulnerabilities, organizations can specify security controls, secure coding practices, and other measures to be implemented during the software development lifecycle to ensure the desired level of security.
8	Security Awareness	SAMM's Threat Modeling aspect promotes security awareness within the development team. By involving developers and other stakeholders in the threat modeling process, organizations can enhance their understanding of security risks, promote a security mindset, and foster collaboration in implementing effective security controls.

10.3.5.1 Assessment – OWASP Top 10

The SAMM Design Domain - Threat Modeling aspect indirectly addresses the OWASP threats by providing a systematic approach to identifying and analyzing potential threats and vulnerabilities in software applications. While the SAMM framework does not explicitly list each OWASP threat name and code, it encompasses broader principles and practices that can help address these threats effectively.

While the SAMM framework does not explicitly mention each OWASP threat name and code, the principles and practices within the Threat Modeling aspect provide a comprehensive approach to address security threats and vulnerabilities, including those highlighted by OWASP. Organizations can leverage SAMM's

Threat Modeling guidelines to identify, prioritize, and mitigate the specific OWASP threats relevant to their software applications.

Here's how the Threat Modeling aspect of SAMM contributes to addressing OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's Threat Modeling aspect helps identify and analyze access control vulnerabilities, ensuring proper enforcement of access controls and preventing unauthorized access or privilege escalation.
2	Cryptographic Failures (A02-2021)	SAMM's Threat Modeling aspect emphasizes analyzing the implementation of cryptographic algorithms and mechanisms to identify weaknesses and ensure secure cryptographic practices are followed.
3	Injection (A03-2021)	Threat Modeling in SAMM involves considering input validation and handling to prevent injection vulnerabilities, ensuring untrusted data is properly handled and cannot be used to manipulate the application's execution or interact with data stores.
4	Insecure Design (A04-2021)	SAMM's Threat Modeling aspect promotes considering security during the design phase to identify and address insecure design decisions, ensuring proper architectural decisions and security considerations are made to mitigate vulnerabilities.
5	Security Misconfiguration (A05-2021)	Threat Modeling in SAMM emphasizes the importance of secure configuration practices, helping organizations identify potential security misconfigurations that could create entry points for attackers and addressing them proactively.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's Threat Modeling aspect emphasizes the need to manage software dependencies effectively, identifying and addressing the use of outdated or vulnerable components that could introduce security risks to the application.
7	Identification and Authentication Failures (A07-2021)	Threat Modeling in SAMM focuses on the identification and authentication mechanisms of web applications, helping organizations identify vulnerabilities and ensure secure authentication mechanisms are implemented to prevent unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	SAMM's Threat Modeling aspect helps organizations identify vulnerabilities related to the integrity of software and data, ensuring appropriate measures are in place to protect against unauthorized modification, manipulation, or destruction of components and data.
9	Security Logging and Monitoring Failures (A09-2021)	Threat Modeling in SAMM emphasizes the importance of effective logging and monitoring practices, helping organizations identify vulnerabilities and ensure comprehensive logging, analysis, and real-time monitoring mechanisms are implemented.
10	Server-Side Request Forgery (A10-2021)	SAMM's Threat Modeling aspect helps organizations identify and address vulnerabilities related to server-side request forgery, ensuring robust input validation and secure configurations to prevent unintended requests to internal or external resources.

10.3.6 Security Testing

In the SAMM (Software Assurance Maturity Model) Design domain, Security Testing is a crucial aspect that focuses on evaluating the security of software applications through systematic testing and validation. By incorporating security testing into the software development lifecycle, organizations can identify and mitigate vulnerabilities, validate the effectiveness of security controls, and ensure the overall security of their applications.

By incorporating these considerations and contributions into the software development process, the Security Testing aspect of the SAMM Design domain helps organizations identify and mitigate security vulnerabilities, validate security controls, and ensure the overall security of their software applications.

Here are some key considerations and contributions of the Security Testing aspect:

#	Name	Description
1	Testing Methodologies	SAMM's Security Testing aspect emphasizes the adoption of appropriate testing methodologies to assess the security of applications. This includes techniques such as penetration testing, vulnerability scanning, code review, and security-focused testing frameworks.
2	Security Test Planning	SAMM promotes the development of comprehensive security test plans. These plans outline the scope, objectives, and techniques to be used during security testing. They also consider the identification of appropriate testing tools and resources required for effective security testing.
3	Security Test Coverage	SAMM's Security Testing aspect encourages organizations to ensure comprehensive test coverage by addressing various security dimensions. This includes testing for vulnerabilities related to authentication, authorization, input validation, output encoding, cryptography, error handling, session management, and other critical security aspects.
4	Security Test Automation	SAMM emphasizes the use of automation tools and techniques to streamline security testing processes. This includes leveraging automated scanning tools, security testing frameworks, and test case generation to enhance efficiency, accuracy, and coverage of security tests.
5	Vulnerability Assessment	SAMM's Security Testing aspect includes conducting vulnerability assessments to identify and prioritize potential vulnerabilities within the application. This involves using scanning tools and manual techniques to identify common security weaknesses and misconfigurations.
6	Penetration Testing	SAMM promotes the use of penetration testing to assess the security of applications by simulating real-world attacks. This involves attempting to exploit vulnerabilities to gain unauthorized access, escalate privileges, or manipulate sensitive data. Penetration testing helps identify critical vulnerabilities and provides insights into potential attack vectors.

#	Name	Description
7	Security Code Review	SAMM's Security Testing aspect encourages organizations to conduct security-focused code reviews. This involves analyzing the application's source code to identify potential security vulnerabilities, insecure coding practices, and adherence to secure coding standards.
8	Security Test Reporting	SAMM emphasizes the importance of comprehensive and well-documented security test reporting. This includes documenting test results, identified vulnerabilities, recommendations for remediation, and the overall security posture of the application. The reports facilitate effective communication between the development team, management, and stakeholders.
9	Security Test Integration:	SAMM's Security Testing aspect promotes integrating security testing into the software development lifecycle. This involves considering security testing at various stages, including requirements gathering, design, development, and deployment. Integration ensures that security is considered throughout the development process and vulnerabilities are identified and addressed early.
10	Continuous Improvement	SAMM encourages organizations to continuously improve their security testing practices. This includes learning from identified vulnerabilities, incorporating feedback into future testing cycles, and staying updated with emerging security threats and testing techniques.

10.3.6.1 Assessment – OWASP Top 10

The SAMM Design Domain - Security Testing contributes to addressing multiple OWASP threats by incorporating security testing methodologies, promoting secure coding practices, and ensuring the identification and mitigation of vulnerabilities throughout the software development lifecycle.

The SAMM Design Domain - Security Testing contributes to addressing several OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's Security Testing aspect helps identify and mitigate access control vulnerabilities by testing the enforcement of proper access controls within a web application.
2	Cryptographic Failures (A02-2021)	SAMM promotes the use of security testing methodologies, including code review and vulnerability scanning, to identify weaknesses and mistakes in the implementation of cryptographic algorithms and mechanisms.
3	Injection (A03-2021)	SAMM's Security Testing aspect emphasizes the importance of input validation and secure coding practices to address injection vulnerabilities by testing the proper handling of untrusted data.
4	Insecure Design (A04-2021)	SAMM encourages organizations to adopt secure design principles and consider security from the early stages of development, helping mitigate insecure design vulnerabilities.

#	Name	Description
5	Security Misconfiguration (A05-2021)	SAMM's Security Testing aspect promotes secure configuration practices and regular security assessments to address security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	SAMM emphasizes actively managing software dependencies, staying updated with security patches, and following secure coding practices to mitigate the risks associated with vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	SAMM's Security Testing aspect focuses on testing the identification and authentication mechanisms of web applications to address vulnerabilities related to weak or flawed authentication.
8	Software and Data Integrity Failures (A08-2021)	SAMM promotes secure coding techniques and regular security testing to ensure the integrity and protection of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM encourages organizations to implement comprehensive logging, effective log analysis, and real-time monitoring mechanisms to address vulnerabilities related to inadequate logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	SAMM's Security Testing aspect emphasizes robust input validation and secure configurations to mitigate server-side request forgery vulnerabilities by testing for unauthorized requests to internal or external resources.

10.4 Security by Implementation

10.4.1 Introduction

SAMM's Implementation domain focuses on the activities and practices related to implementing software security measures during the development and deployment of software systems. This domain encompasses various aspects that contribute to integrating security into the software development life cycle. By implementing the practices outlined in SAMM's Implementation domain, organizations can establish a strong foundation for building secure software systems. This domain helps organizations integrate security measures into the software development life cycle, ensuring that security is considered from the early stages of development and throughout the software's deployment and operation.
Here are the key components of SAMM's Implementation domain:

#	Name	Description
1	Secure Development Training	SAMM emphasizes the importance of providing training and awareness programs to developers and development teams on secure coding practices, secure design principles, and common software vulnerabilities. This helps ensure that developers have the necessary knowledge and skills to build secure software from the outset.

#	Name	Description
2	Secure Architecture	SAMM promotes the use of secure architecture practices, such as threat modeling, secure design patterns, and secure component selection. These practices help ensure that the software system is designed with security in mind and that potential vulnerabilities and risks are identified and addressed early in the development process.
3	Secure Coding Guidelines	SAMM advocates for the establishment and adoption of secure coding guidelines and standards. These guidelines provide developers with specific recommendations and best practices for writing secure code, covering areas such as input validation, output encoding, access controls, and secure error handling.
4	Security Requirements	SAMM emphasizes the need to define and document security requirements as part of the software development process. These requirements outline the security objectives and expectations for the software system and serve as a basis for designing and implementing security controls.
5	Security Testing Integration	SAMM encourages the integration of security testing activities throughout the software development life cycle. This includes activities such as static code analysis, dynamic application security testing (DAST), and security code reviews. By incorporating security testing at various stages of development, organizations can identify and address security vulnerabilities early on.
6	Security Verification	SAMM promotes the use of security verification techniques to validate the effectiveness of security controls implemented in the software system. This includes activities such as security code reviews, vulnerability assessments, and security testing against established security requirements. Verification helps ensure that the software system meets the desired security objectives.
7	Security Architecture Review	SAMM recommends conducting security architecture reviews to evaluate the security of the software system's design and architecture. This review process involves assessing the security controls, identifying potential vulnerabilities or weaknesses, and providing recommendations for improvement.
8	Security Operations Integration	SAMM emphasizes the importance of integrating security considerations into operational processes, such as incident management, vulnerability management, and secure deployment practices. This integration ensures that security measures are not only implemented during development but also supported and maintained throughout the operational life of the software system.

10.4.2 Secure Development Training

In the SAMM Implementation domain, Secure Development Training is a key component that focuses on providing education and training to development teams to enhance their understanding of secure coding practices and principles. By investing in secure development training, organizations can equip their developers with the necessary knowledge and skills to build more secure software applications.
Secure Development Training in the SAMM Implementation domain plays a crucial role in building the knowledge and skills of developers, empowering them to develop software applications with a strong focus

on security. By investing in this training, organizations can improve the overall security posture of their software and reduce the risk of security vulnerabilities and breaches.

Here are some key contributions of Secure Development Training in the SAMM Implementation domain:

#	Name	Description
1	Awareness of Secure Coding Practices	Secure Development Training raises awareness among developers about common security vulnerabilities and the best practices to prevent them. It covers topics such as input validation, secure authentication, secure data handling, and secure configuration, ensuring that developers have a solid foundation in secure coding.
2	Understanding of Security Requirements	Secure Development Training helps developers understand security requirements and how to incorporate them into the development process. It educates them about the importance of threat modeling, risk assessment, and secure design principles, enabling them to build security into the application architecture from the early stages.
3	Knowledge of Secure Coding Techniques	Secure Development Training provides developers with practical knowledge of secure coding techniques and patterns. It covers topics such as input validation and sanitization, output encoding, parameterized queries, secure session management, and secure error handling. This knowledge enables developers to write code that is more resistant to common vulnerabilities like injection attacks, cross-site scripting (XSS), and cross-site request forgery (CSRF).
4	Familiarity with Security Tools	Secure Development Training introduces developers to security testing and analysis tools. It educates them on how to leverage static code analysis, dynamic application security testing (DAST), and other security tools to identify potential vulnerabilities in their code. This knowledge helps developers identify and address security issues early in the development process.
5	Integration with Secure Development Lifecycle	Secure Development Training emphasizes the integration of security practices into the software development lifecycle. It helps developers understand the importance of conducting security reviews, performing secure code reviews, and incorporating security testing at different stages of the development process. This integration ensures that security is considered throughout the entire development lifecycle, from requirements gathering to deployment.
6	Collaboration and Communication	Secure Development Training promotes collaboration and communication between developers, security teams, and other stakeholders. It emphasizes the importance of ongoing communication about security requirements, vulnerabilities, and mitigation strategies. This collaboration fosters a culture of security awareness and enables better coordination in addressing security-related issues.

10.4.2.1 Assessment – OWASP Top 10

The SAMM Implementation domain - Secure Development Training contributes to addressing several OWASP threats directly.

By addressing these OWASP threats through Secure Development Training, organizations can enhance the security of their software applications and reduce the risk of security vulnerabilities and breaches.

Here are the contributions of Secure Development Training in addressing each OWASP threat:

#	Name	Description
1	Broken Access Control (A01-2021)	Secure Development Training raises awareness among developers about the importance of access controls and provides knowledge on implementing proper access control mechanisms, such as role-based access control (RBAC) and attribute-based access control (ABAC).
2	Cryptographic Failures (A02-2021)	Secure Development Training educates developers on secure cryptographic practices, including the proper implementation of encryption algorithms, key management, secure random number generation, and protection against cryptographic attacks.
3	Injection (A03-2021)	Secure Development Training teaches developers about secure coding practices, input validation, and parameterized queries to prevent injection attacks like SQL injection, LDAP injection, or OS command injection.
4	Insecure Design (A04-2021)	Secure Development Training emphasizes the importance of secure design principles, threat modeling, and risk assessment to identify and address security vulnerabilities during the design phase.
5	Security Misconfiguration (A05-2021)	Secure Development Training educates developers about secure configuration practices, including properly configuring servers, frameworks, and other components to minimize the risk of exposing sensitive information or enabling unnecessary functionality.
6	Vulnerable and Outdated Components (A06-2021)	Secure Development Training emphasizes the importance of managing software dependencies, staying updated with security patches, and conducting regular vulnerability assessments to address the risk of using outdated or vulnerable third-party components.
7	Identification and Authentication Failures (A07-2021)	Secure Development Training covers secure authentication mechanisms, strong password policies, and protection against common authentication vulnerabilities like credential stuffing, brute force attacks, or session hijacking.
8	Software and Data Integrity Failures (A08-2021)	Secure Development Training provides knowledge on secure coding practices, input validation, and integrity checks to prevent unauthorized modification or tampering of software components, configurations, and data.
9	Security Logging and Monitoring Failures (A09-2021)	Secure Development Training emphasizes the importance of logging and monitoring capabilities in web applications, including proper logging practices, log analysis, and real-time monitoring mechanisms to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	Secure Development Training educates developers on input validation and secure configurations to prevent server-side request forgery (SSRF) attacks and ensure that requests are made to intended and authorized resources.

10.4.3 Secure Architecture

In the SAMM Implementation domain, Secure Architecture is a crucial component that focuses on designing and implementing secure software architectures. It aims to incorporate security into the overall system architecture to build robust and resilient software applications.

Incorporating these aspects into the Secure Architecture, organizations can build software systems that are more resilient to security threats, reduce the attack surface, and provide a strong foundation for secure software development and deployment.

Here are some key aspects of Secure Architecture in the SAMM Implementation domain:

#	Name	Description
1	Security Design Principles	Secure Architecture emphasizes the use of security design principles to guide the architectural decisions. These principles include the least privilege, defense-in-depth, separation of concerns, and fail-safe defaults. By adhering to these principles, organizations can ensure that security is considered throughout the architectural design process.
2	Threat Modeling	Secure Architecture promotes the practice of threat modeling during the design phase. Threat modeling involves identifying potential threats and vulnerabilities, analyzing their impact, and devising appropriate countermeasures. By conducting threat modeling, organizations can proactively identify and address security risks at the architectural level.
3	Secure Communication	Secure Architecture addresses the secure communication requirements of the system. It includes the use of secure protocols such as TLS/SSL for data transmission, encryption of sensitive data in transit, and secure network configurations to protect against eavesdropping, tampering, and unauthorized access.
4	Access Control and Authentication	Secure Architecture incorporates strong access control mechanisms and authentication protocols to ensure that only authorized users can access system resources. It includes techniques such as role-based access control (RBAC), multi-factor authentication (MFA), and secure session management to prevent unauthorized access and protect sensitive data.
5	Secure Data Storage:	Secure Architecture considers the secure storage of sensitive data. It includes the use of encryption, secure key management, and appropriate data protection mechanisms to safeguard data at rest. This helps protect against data breaches and unauthorized access to stored information.
6	Resilience and Fault Tolerance	Secure Architecture addresses the resilience and fault tolerance aspects of the system. It includes designing for high availability, implementing redundant systems, and incorporating fault tolerance mechanisms to ensure that the system can withstand attacks, recover from failures, and maintain its critical functionalities.
7	Secure Integration and APIs	Secure Architecture considers the secure integration of different system components and the use of secure APIs (Application Programming Interfaces). It includes secure data exchange mechanisms, proper input validation, and protection against common integration vulnerabilities such as injection attacks or insecure direct object references.

#	Name	Description
8	Security Testing and Verification	Secure Architecture emphasizes the need for security testing and verification activities. It includes conducting security code reviews, architecture reviews, and security testing (such as penetration testing) to identify and address security issues at the architectural level before deployment.
9	Compliance and Regulatory Requirements	Secure Architecture takes into account applicable compliance and regulatory requirements. It ensures that the architectural design aligns with industry standards, legal regulations, and privacy requirements, enabling organizations to meet their obligations and protect user data

10.4.3.1 Assessment – OWASP Top 10

While the SAMM Implementation domain, specifically Secure Architecture, addresses several OWASP threats directly, it is important to note that the implementation of Secure Architecture alone may not fully address all potential vulnerabilities. Organizations should adopt a holistic approach to application security, including other practices such as secure coding, secure deployment configurations, and regular security testing, to comprehensively mitigate OWASP threats and ensure the overall security of their software applications.

The SAMM Implementation domain, specifically the Secure Architecture component, addresses several OWASP threats.

Here's how each OWASP threat is addressed:

#	Name	Description
1	Broken Access Control (A01-2021)	Secure Architecture emphasizes the use of proper access control mechanisms, such as role-based access control (RBAC) and secure session management, to prevent unauthorized access and ensure that access controls are properly enforced.
2	Cryptographic Failures (A02-2021)	Secure Architecture incorporates the use of secure cryptographic algorithms and proper key management practices to protect sensitive data and prevent cryptographic vulnerabilities.
3	Injection (A03-2021)	Secure Architecture promotes proper input validation and secure coding practices to mitigate the risk of injection attacks and ensure that untrusted data cannot be exploited to manipulate the application's execution or interact with data stores.
4	Insecure Design (A04-2021)	Secure Architecture focuses on secure design principles and considerations, including the use of secure architectures and appropriate security controls, to address potential flaws and weaknesses in the overall design of web applications.
5	Security Misconfiguration (A05-2021)	Secure Architecture emphasizes secure configuration practices for web applications and associated components to prevent security misconfigurations that could expose sensitive information or create entry points for attackers.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	Secure Architecture promotes the active management of software dependencies, including third-party libraries and frameworks, to ensure that only updated and secure components are used, reducing the risk of exploiting known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	Secure Architecture focuses on implementing secure authentication mechanisms, enforcing strong password policies, and conducting regular vulnerability assessments to mitigate vulnerabilities related to identification and authentication.
8	Software and Data Integrity Failures (A08-2021)	Secure Architecture includes secure coding practices and integrity protection mechanisms to prevent unauthorized modification, manipulation, or destruction of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	Secure Architecture emphasizes the importance of effective logging and monitoring practices, including comprehensive logging, log analysis, and real-time monitoring mechanisms, to detect and respond to security incidents in a timely manner.
10	Server-Side Request Forgery (A10-2021)	Secure Architecture promotes input validation and secure configurations to mitigate the risk of server-side request forgery (SSRF) vulnerabilities and prevent unauthorized requests to internal or external resources.

10.4.4 Secure Coding Guidelines

In the SAMM Implementation domain, Secure Coding Guidelines play a crucial role in ensuring that software development follows secure coding practices. Secure coding aims to minimize vulnerabilities and reduce the risk of security breaches in software applications. Here are some key aspects of Secure Coding Guidelines in the SAMM Implementation domain:

#	Name	Description
1	Secure Coding Standards	Secure Coding Guidelines provide developers with a set of predefined standards and best practices for writing secure code. These standards cover areas such as input validation, output encoding, secure communication, error handling, and data protection. By adhering to these standards, developers can minimize common coding errors that can lead to security vulnerabilities.
2	Input Validation	Secure Coding Guidelines emphasize the importance of validating and sanitizing all user input to prevent common attacks like injection and cross-site scripting (XSS). Developers are encouraged to use proper input validation techniques, such as white-listing, to ensure that only valid and expected input is processed by the application.

#	Name	Description
3	Secure Coding Guidelines	Following Secure Coding Guidelines, organizations can significantly reduce the likelihood of introducing security vulnerabilities during the software development process. These guidelines promote the adoption of secure coding practices and help developers build robust and secure software applications.
4	Output Encoding	Secure Coding Guidelines promote the use of output encoding techniques to protect against XSS attacks. By properly encoding user-generated or dynamic content before displaying it to users, developers can prevent malicious code from being executed in a user's browser.
5	Secure Communication	Secure Coding Guidelines address the secure communication requirements of software applications. They encourage the use of secure protocols, such as HTTPS, for transmitting sensitive data over networks. Additionally, developers are guided on how to properly implement encryption and decryption mechanisms to protect data in transit.
6	Error Handling and Logging	Secure Coding Guidelines provide recommendations for secure error handling and logging practices. Developers are encouraged to handle errors securely, avoiding the disclosure of sensitive information that could be useful to attackers. Proper logging techniques are also emphasized to aid in the detection and investigation of security incidents.
7	Authentication and Authorization	Secure Coding Guidelines include best practices for implementing secure authentication and authorization mechanisms. Developers are guided on the proper use of secure password storage techniques, such as hashing and salting, and the implementation of strong authentication protocols to protect user credentials.
8	Secure Session Management	Secure Coding Guidelines address the secure management of user sessions. Developers are advised on techniques such as using unique session identifiers, implementing session timeouts, and securely storing session data to prevent session hijacking and session fixation attacks.
9	Secure Error Handling	Secure Coding Guidelines promote secure error handling practices to prevent the leakage of sensitive information that could aid attackers in exploiting vulnerabilities. Developers are guided on how to handle errors without exposing details that could be used to compromise the application's security.
10	Secure File and Resource Handling	Secure Coding Guidelines provide recommendations for securely handling files and resources in software applications. This includes techniques for proper input validation and sanitization when interacting with files, secure file permissions, and preventing directory traversal attacks.
11	Code Review and Testing	Secure Coding Guidelines stress the importance of conducting code reviews and security testing to identify and address security vulnerabilities in the codebase. Developers are encouraged to adopt secure coding practices during code reviews and use automated security testing tools to identify common coding errors and vulnerabilities.

10.4.4.1 Assessment – OWASP Top 10

While Secure Coding Guidelines address many OWASP threats, the implementation of these guidelines alone may not fully mitigate all risks. It's recommended to adopt a comprehensive approach to application security that includes other security practices such as secure configuration, secure infrastructure, and regular security testing.

Secure Coding Guidelines in the SAMM Implementation domain address several OWASP threats directly. Here is a list of contributions from Secure Coding Guidelines to each OWASP threat:

#	Name	Description
1	Broken Access Control (A01-2021)	Secure Coding Guidelines emphasize the implementation and enforcement of proper access controls, such as role-based access control (RBAC) and attribute-based access control (ABAC), to prevent unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Secure Coding Guidelines provide recommendations for implementing secure cryptographic algorithms and practices, including proper key management, encryption/decryption, and secure storage of cryptographic keys.
3	Injection (A03-2021)	Secure Coding Guidelines promote input validation and parameterization techniques to prevent injection attacks. They guide developers on sanitizing user input and using prepared statements or parameterized queries to prevent SQL injection and other injection vulnerabilities.
4	Insecure Design (A04-2021)	Secure Coding Guidelines encourage the adoption of secure design principles from the early stages of development. They guide developers on considering security requirements, threat modeling, and secure architecture to minimize insecure design decisions.
5	Security Misconfiguration (A05-2021)	Secure Coding Guidelines promote secure configuration practices, such as disabling unnecessary features, removing default credentials, and keeping software and frameworks up to date. They guide developers on avoiding common misconfigurations that can lead to security vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Secure Coding Guidelines stress the importance of managing software dependencies and using up-to-date versions of third-party libraries and frameworks. They encourage developers to regularly update and patch software components to mitigate the risk of known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	Secure Coding Guidelines provide recommendations for implementing secure authentication mechanisms, including strong password storage techniques, multi-factor authentication (MFA), and session management techniques to prevent authentication bypass and identity-related vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Secure Coding Guidelines promote the use of secure coding practices to ensure the integrity of software components and data. They guide developers on implementing input validation, secure data storage, and tamper-proof mechanisms to prevent unauthorized modifications and data integrity issues.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Secure Coding Guidelines emphasize the importance of logging and monitoring in detecting and responding to security incidents. They provide recommendations for implementing proper logging techniques, such as log sanitization, and using security information and event management (SIEM) systems for real-time monitoring and analysis.
10	Server-Side Request Forgery (A10-2021)	Secure Coding Guidelines recommend input validation and sanitization techniques to prevent Server-Side Request Forgery (SSRF) vulnerabilities. They guide developers on validating and whitelisting user-supplied URLs to restrict requests to trusted resources and prevent SSRF attacks.

10.4.5 Security Requirements

In the SAMM Implementation domain, addressing security requirements is a crucial aspect of software development to ensure that applications meet the necessary security standards. Security requirements define the necessary security controls, features, and behaviors that the software application should exhibit to mitigate risks and protect against potential threats.

By addressing security requirements in the SAMM Implementation domain, organizations can ensure that their software applications are built with security in mind. This helps in reducing the likelihood of security breaches, protecting sensitive data, and maintaining the overall integrity and trustworthiness of the software.

Here are some key aspects of addressing security requirements in the SAMM Implementation domain:

#	Name	Description
1	Security Assessment	Before defining security requirements, it is important to conduct a comprehensive security assessment to identify potential threats, vulnerabilities, and risks associated with the software application. This assessment helps in understanding the specific security needs and potential areas of improvement.
2	Threat Modeling	Threat modeling is a technique used to identify and prioritize potential threats to the software application. It involves analyzing the application's architecture, design, and components to identify potential vulnerabilities and define appropriate security requirements to mitigate these risks.
3	Regulatory and Compliance Requirements	Security requirements should align with relevant regulatory and compliance standards that apply to the software application. This includes ensuring compliance with industry-specific regulations such as GDPR, HIPAA, PCI-DSS, etc., and following any specific security guidelines or frameworks mandated by regulatory bodies.

#	Name	Description
4	Secure Architecture	Security requirements should encompass the design and architecture of the software application. This involves defining security controls, such as access controls, authentication mechanisms, encryption protocols, and secure communication channels, to protect sensitive data and prevent unauthorized access.
5	Secure Coding Practices	Security requirements should address secure coding practices that developers should adhere to during the software development lifecycle. This includes guidelines for input validation, output encoding, secure error handling, secure session management, and secure file handling to prevent common vulnerabilities such as injection attacks, cross-site scripting, and security misconfigurations.
6	Secure Data Management	Security requirements should define how sensitive data should be handled and protected throughout its lifecycle within the application. This includes requirements for data encryption, secure storage, proper data sanitization, and access controls to ensure confidentiality, integrity, and availability of the data.
7	Secure External Integrations	If the software application interacts with external systems or APIs, security requirements should address the secure integration and communication between these entities. This involves defining authentication and authorization mechanisms, secure data exchange protocols, and ensuring the proper validation and sanitization of input received from external sources.
8	Security Testing and Validation	Security requirements should include provisions for security testing and validation to ensure that the implemented security controls and features are effective and functioning as intended. This may involve conducting regular security assessments, penetration testing, vulnerability scanning, and code reviews to identify and address any security gaps.

10.4.5.1 Assessment – OWASP Top 10

While SAMM does not directly specify or list the OWASP threats, the implementation of security requirements, as outlined in the SAMM Implementation domain, aligns with various OWASP threats and contributes to mitigating those vulnerabilities.

The SAMM Implementation domain, specifically addressing security requirements, can contribute to mitigating several OWASP threats.

Here's how each contribution aligns with specific OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	By conducting a security assessment and implementing proper access controls as part of security requirements, organizations can address vulnerabilities related to improper enforcement of access controls.

#	Name	Description
2	Cryptographic Failures (A02-2021)	Addressing cryptographic failures involves implementing secure cryptographic practices as part of security requirements, such as using strong algorithms, proper key management, and secure configurations.
3	Injection (A03-2021)	Security requirements should address secure coding practices, input validation, and parameterization to prevent injection vulnerabilities.
4	Insecure Design (A04-2021)	Secure architecture and design practices, including security requirements, help address vulnerabilities related to insecure design decisions and inadequate security considerations during the design phase.
5	Security Misconfiguration (A05-2021)	Security requirements should encompass secure configuration practices to mitigate vulnerabilities arising from insecure configurations of web applications and associated components.
6	Vulnerable and Outdated Components (A06-2021)	Security requirements should include provisions for managing software dependencies and following secure coding practices, which help address vulnerabilities related to using outdated or known vulnerable components.
7	Identification and Authentication Failures (A07-2021)	Security requirements should focus on implementing secure identification and authentication mechanisms, strong password policies, and regular vulnerability testing to mitigate authentication-related vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Security requirements should encompass secure coding techniques, data integrity checks, and protection mechanisms to address vulnerabilities related to unauthorized modifications, data tampering, and integrity issues.
9	Security Logging and Monitoring Failures (A09-2021)	Security requirements should include provisions for effective logging and monitoring practices, ensuring proper collection, analysis, and retention of security-related logs to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	Security requirements should address input validation and secure configurations to prevent server-side request forgery vulnerabilities.

10.4.6 Security Testing Integration

In the SAMM Implementation domain, Security Testing Integration is a crucial aspect of software development to ensure that applications are thoroughly tested for security vulnerabilities. Security testing helps identify weaknesses and vulnerabilities in the application's design, code, and configurations, allowing organizations to address them before deployment.

By integrating security testing into the software development lifecycle, organizations can enhance the overall security posture of their applications.

Here are some key aspects of Security Testing Integration in the SAMM Implementation domain:

#	Name	Description
1	Security Testing Strategy	Organizations should define a comprehensive security testing strategy that outlines the types of security tests to be performed, such as penetration testing, vulnerability scanning, code review, and security assessments. The strategy should also specify the frequency and coverage of these tests based on the application's risk profile.
2	Test Environment Setup	A dedicated test environment should be set up to conduct security tests effectively. This environment should closely resemble the production environment and include necessary security tools, configurations, and simulated attack scenarios.
3	Security Testing Tools	Organizations should identify and integrate appropriate security testing tools into their development and testing environments. These tools can automate various security testing activities, including vulnerability scanning, code analysis, and penetration testing.
4	Security Testing Activities	Security testing should be conducted throughout the software development lifecycle, including during the design, development, and testing phases. This includes conducting secure code reviews, static analysis, dynamic testing, and security assessments to identify vulnerabilities and weaknesses.
5	Continuous Integration and Testing	Security testing should be integrated into the continuous integration and continuous testing processes. Automated security tests should be included in the build and deployment pipelines to ensure that security vulnerabilities are identified early in the development process.
6	Collaboration with Testing Teams	Security teams should collaborate closely with the testing teams to ensure that security testing is properly integrated into the overall testing process. This collaboration helps in identifying security requirements, defining test cases, and coordinating efforts to address identified vulnerabilities.
7	Test Reporting and Remediation	Test results should be documented and shared with relevant stakeholders, including developers, testers, and project managers. Identified vulnerabilities should be prioritized based on their severity, and appropriate remediation actions should be taken to address them.
8	Retesting and Validation	After applying remediation actions, it is important to conduct retesting to validate that the identified vulnerabilities have been effectively addressed. This helps ensure that the application's security has been improved and that no new vulnerabilities have been introduced during the remediation process.

10.4.6.1 Assessment – OWASP Top 10

The SAMM Implementation domain, specifically Security Testing Integration, addresses several OWASP threats directly. Here is the mapping of the SAMM Implementation domain activities to the corresponding OWASP threat names and codes:

#	Name	Description
1	Broken Access Control (A01-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. This strategy contributes to addressing multiple OWASP threats. Test Environment Setup: Establishing a dedicated test environment resembling the production environment with necessary security tools and configurations helps address several OWASP threats. This strategy contributes to addressing multiple OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address OWASP threats.
2	Cryptographic Failures (A02-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Tools: Integrating appropriate security testing tools into the development and testing environments helps address several OWASP threats. Test Environment Setup: Establishing a dedicated test environment resembling the production environment with necessary security tools and configurations helps address several OWASP threats. It contributes to addressing several OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.

#	Name	Description
3	Injection (A03-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Tools: Integrating appropriate security testing tools into the development and testing environments helps address several OWASP threats. Test Environment Setup: Establishing a dedicated test environment resembling the production environment with necessary security tools and configurations helps address several OWASP threats. It contributes to addressing several OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.
4	Insecure Design (A04-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Tools: Integrating appropriate security testing tools into the development and testing environments helps address several OWASP threats. Test Environment Setup: Establishing a dedicated test environment resembling the production environment with necessary security tools and configurations helps address several OWASP threats. It contributes to addressing several OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.

#	Name	Description
5	Security Misconfiguration (A05-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Tools: Integrating appropriate security testing tools into the development and testing environments helps address several OWASP threats. Test Environment Setup: Establishing a dedicated test environment resembling the production environment with necessary security tools and configurations helps address several OWASP threats. It contributes to addressing several OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.
6	Vulnerable and Outdated Components (A06-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Tools: Integrating appropriate security testing tools into the development and testing environments helps address several OWASP threats. Test Environment Setup: Establishing a dedicated test environment resembling the production environment with necessary security tools and configurations helps address several OWASP threats. It contributes to addressing several OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.
8	Software and Data Integrity Failures (A08-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.
9	Security Logging and Monitoring Failures (A09-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Collaboration with Testing Teams: Close collaboration between security teams and testing teams ensures that security testing is properly integrated into the overall testing process. This collaboration helps address several OWASP threats.
10	Server-Side Request Forgery (A10-2021)	Security Testing Strategy: This activity helps address various OWASP threats by defining a comprehensive security testing strategy, including penetration testing, vulnerability scanning, code review, and security assessments. Security Testing Tools: Integrating appropriate security testing tools into the development and testing environments helps address several OWASP threats. Security Testing Activities: Conducting secure code reviews, static analysis, dynamic testing, and security assessments throughout the software development lifecycle. helps address various OWASP threats.

10.4.7 Security Verification

The SAMM (Software Assurance Maturity Model) Implementation domain - Security Verification covers several key aspects to ensure the effective implementation and verification of security controls throughout the software development lifecycle.

By covering these aspects, the Security Verification activity in the SAMM Implementation domain helps organizations ensure that security controls are implemented, verified, and continuously improved throughout the software development lifecycle, reducing the risk of security vulnerabilities and enhancing the overall security of the application.

Here are the aspects covered in the Security Verification activity:

#	Name	Description
1	Security Requirements	The activity emphasizes the establishment of security requirements during the software development process. This includes identifying and documenting specific security objectives, standards, and regulations that need to be met.
2	Secure Coding Practices	It promotes the adoption of secure coding practices by providing guidelines, standards, and training to developers. This includes addressing common coding vulnerabilities, such as input validation, output encoding, secure error handling, and secure session management.
3	Code Review	Security Verification involves conducting code reviews to identify security vulnerabilities and weaknesses in the application's source code. It helps identify coding flaws, insecure practices, and potential vulnerabilities that can be exploited by attackers.
4	Security Testing	The activity emphasizes the inclusion of security testing throughout the software development lifecycle. This includes various testing techniques such as penetration testing, vulnerability scanning, security assessments, and fuzz testing to identify security weaknesses and vulnerabilities.
5	Vulnerability Assessment	It involves performing regular vulnerability assessments to identify and prioritize potential vulnerabilities in the application. This includes scanning for known vulnerabilities in third-party components, libraries, and frameworks used in the software.
6	Secure Configuration Management	The activity focuses on verifying that secure configurations are applied to software components, servers, and other infrastructure elements. It ensures that default configurations are changed, unnecessary services are disabled, and proper security settings are implemented.
7	Security Training and Awareness	The activity promotes security training and awareness programs for developers, testers, and other stakeholders involved in the software development process. This helps improve their understanding of security best practices and the importance of incorporating security throughout the lifecycle.

#	Name	Description
8	Security Metrics and Measurement	It emphasizes the establishment of security metrics and measurements to assess the effectiveness of security verification activities. This includes tracking the number of vulnerabilities identified, remediation progress, and overall improvement in the security posture of the application.
9	Incident Response Planning	The activity addresses the importance of having an incident response plan in place. It includes defining roles and responsibilities, establishing incident handling procedures, and conducting exercises to test the effectiveness of the response plan.

10.4.7.1 Assessment – OWASP Top 10

In the SAMM Implementation domain, the Security Verification activity focuses on ensuring that security requirements and controls are effectively implemented and verified throughout the software development lifecycle. This activity contributes to addressing various OWASP threats by performing security verification activities such as code reviews, security testing, and vulnerability assessments.

By incorporating Security Verification activities into the software development lifecycle, organizations can proactively identify and address OWASP threats, reducing the risk of security vulnerabilities and enhancing the overall security posture of their applications.

Here are some examples of how Security Verification in the SAMM Implementation domain addresses specific OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Security Verification helps identify and verify the correct implementation of access control mechanisms, ensuring that user access is properly restricted and enforced.
2	Cryptographic Failures (A02-2021)	Security Verification involves verifying the correct implementation and usage of cryptographic algorithms and mechanisms, ensuring that they are applied securely and without vulnerabilities.
3	Injection (A03-2021)	Security Verification includes validating input validation and output encoding practices to prevent injection vulnerabilities, ensuring that untrusted data is properly handled and sanitized.
4	Insecure Design (A04-2021)	Security Verification helps identify and address insecure design decisions through code reviews and security testing, ensuring that the overall design and architecture of the software are robust and secure.
5	Security Misconfiguration (A05-2021)	Security Verification involves reviewing and verifying the configuration of the software and associated components to ensure that secure configurations are applied and potential misconfigurations are identified and resolved.
6	Vulnerable and Outdated Components (A06-2021)	Security Verification includes identifying and addressing vulnerable or outdated software components by conducting code reviews, vulnerability scans, and dependency analysis to ensure that secure and up-to-date components are used.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Security Verification focuses on verifying the proper implementation of identification and authentication mechanisms, including strong password policies, secure session management, and secure authentication protocols.
8	Software and Data Integrity Failures (A08-2021)	Security Verification ensures the integrity of software components and data by validating secure coding practices, input validation, and output encoding to prevent unauthorized modifications or tampering.
9	Security Logging and Monitoring Failures (A09-2021)	Security Verification includes verifying the implementation of comprehensive logging and monitoring mechanisms to detect and respond to security incidents, ensuring that logs are collected and analyzed effectively.
10	Server-Side Request Forgery (A10-2021)	Security Verification involves validating the proper handling of user input and verifying the prevention of server-side request forgery vulnerabilities, ensuring that requests are only made to intended and trusted resources.

10.4.8 Security Architecture Review

The SAMM (Software Assurance Maturity Model) Implementation domain - Security Architecture Review focuses on evaluating and improving the security architecture of software systems. It involves analyzing the overall design and structure of the software to identify potential security weaknesses and ensure that appropriate security controls are in place.

The Security Architecture Review domain in SAMM aims to ensure that security considerations are embedded into the software architecture, providing a strong foundation for building secure and resilient software systems.

Here are some aspects covered by the Security Architecture Review domain in SAMM:

#	Name	Description
1	Security Architecture Design	Reviewing the architecture design to ensure that security considerations are incorporated from the beginning. Assessing the overall security posture of the software architecture, including the identification of potential vulnerabilities and risks.
2	Threat Modeling	Conducting threat modeling exercises to identify potential threats and vulnerabilities specific to the software architecture. Analyzing the system's attack surface and identifying potential entry points for attackers.
3	Secure Design Patterns	Evaluating the use of secure design patterns and principles within the software architecture. Assessing the effectiveness of security controls and mechanisms in mitigating known security risks.

#	Name	Description
4	Secure Communication	Reviewing the communication channels and protocols used within the software architecture to ensure confidentiality, integrity, and authenticity of data. Assessing the implementation of encryption, secure protocols, and secure configuration for network communication.
5	Access Control	Evaluating the access control mechanisms and enforcement within the software architecture. Reviewing role-based access control, privilege escalation prevention, and least privilege principles.
6	Data Protection	Assessing how sensitive data is handled and protected within the software architecture. Reviewing data encryption, data anonymization, and secure data storage mechanisms.
7	Authentication and Authorization	Evaluating the authentication and authorization mechanisms used within the software architecture. Assessing the implementation of secure authentication protocols, multi-factor authentication, and secure session management.
8	Secure APIs and Interfaces	Reviewing the security of APIs and interfaces exposed by the software architecture. Assessing input validation, output encoding, and secure handling of data passed through APIs.
9	Security Configuration	Reviewing the security configuration settings and parameters within the software architecture. Assessing the proper configuration of security controls, such as firewalls, intrusion detection systems, and logging mechanisms.
10	Security Testing and Validation	Assessing the effectiveness of security testing activities performed on the software architecture. Reviewing the use of security testing techniques, such as penetration testing, vulnerability scanning, and code analysis.
11	Secure Development Lifecycle Integration	Evaluating the integration of security activities into the software development lifecycle. Assessing the inclusion of security checkpoints, code review processes, and security training for developers.

10.4.8.1 Assessment – OWASP Top 10

The Software Assurance Maturity Model (SAMM) Implementation domain - Security Architecture Review indirectly addresses several OWASP (Open Web Application Security Project) threats by focusing on evaluating and improving the security architecture of software systems. While SAMM does not specifically mention the OWASP threats by name, the practices and activities covered in the Security Architecture Review domain contribute to addressing these threats.

It's important to note that SAMM provides a framework and guidance for organizations to establish and improve their software assurance practices. The specific implementation of SAMM activities may vary based on an organization's context and requirements.

Here's how the SAMM Security Architecture Review domain aligns with some of the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM addresses this threat by reviewing the access control mechanisms and enforcement within the software architecture, evaluating role-based access control, privilege escalation prevention, and least privilege principles.
2	Cryptographic Failures (A02-2021)	SAMM evaluates the implementation of cryptographic mechanisms within the software architecture, reviewing encryption, secure protocols, and secure configuration for network communication.
3	Injection (A03-2021)	SAMM conducts threat modeling exercises and reviews input validation practices, which help identify and address vulnerabilities related to injection attacks.
4	Insecure Design (A04-2021)	SAMM emphasizes the importance of security architecture design and secure design patterns, aiming to prevent insecure design decisions and vulnerabilities in the overall software architecture.
5	Security Misconfiguration (A05-2021)	SAMM includes security configuration reviews, ensuring that security controls, such as firewalls, intrusion detection systems, and logging mechanisms, are properly configured within the software architecture.
6	Vulnerable and Outdated Components (A06-2021)	SAMM emphasizes the importance of secure development practices, including managing software dependencies, staying updated with security patches, and following secure coding practices to address vulnerabilities related to vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	SAMM evaluates the authentication and authorization mechanisms used within the software architecture, assessing the implementation of secure authentication protocols, multi-factor authentication, and secure session management.
8	Software and Data Integrity Failures (A08-2021)	SAMM reviews the implementation of secure coding techniques and practices, which contribute to addressing vulnerabilities related to the integrity and protection of software components, configurations, and data.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM emphasizes the importance of security testing and validation, which includes assessing the effectiveness of security testing activities and reviewing the use of logging and monitoring mechanisms to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	While SAMM does not explicitly mention SSRF, the practices covered in threat modeling, secure communication, and secure APIs and interfaces contribute to addressing vulnerabilities related to server-side request forgery.

10.4.9 Security Operations Integration

The SAMM (Software Assurance Maturity Model) Implementation domain - Security Operations Integration focuses on integrating security operations into the software development lifecycle to enhance the overall security of software systems. It aims to ensure that security activities, such as monitoring, incident

response, and vulnerability management, are effectively incorporated throughout the software development process.

It's important to note that the specific implementation of security operations integration may vary based on an organization's size, industry, and specific security requirements. SAMM provides a framework and guidance to organizations for implementing and improving their software assurance practices, including the integration of security operations.

Here are some aspects covered by the Security Operations Integration domain in SAMM:

#	Name	Description
1	Security Incident Management	Establishing incident response processes and procedures to handle security incidents related to software systems. Defining roles and responsibilities for incident response team members and ensuring effective communication and coordination during security incidents.
2	Security Monitoring	Implementing monitoring mechanisms and tools to detect and respond to security events and anomalies within the software system. Configuring security monitoring systems to collect relevant logs and events for analysis and investigation.
3	Vulnerability Management	Conducting regular vulnerability assessments and penetration testing on the software system to identify and address potential vulnerabilities. Establishing processes for tracking, prioritizing, and remediating vulnerabilities discovered during the software development lifecycle.
4	Secure Configuration Management	Defining and enforcing secure configuration standards for software components and infrastructure. Implementing change management processes to ensure that configurations are reviewed and approved before deployment.
5	Security Testing and Assurance	Incorporating security testing activities, such as penetration testing, code reviews, and security scanning, into the software development process. Establishing testing criteria and guidelines to evaluate the security of software components and identify potential vulnerabilities.
6	Security Training and Awareness	Providing security training and awareness programs to software development teams to enhance their understanding of security principles and best practices. Promoting a culture of security awareness and responsibility among software developers and stakeholders.
7	Security Metrics and Reporting	Establishing metrics and measurement processes to assess the effectiveness of security operations and identify areas for improvement. Generating regular reports on security incidents, vulnerabilities, and remediation efforts to inform decision-making and demonstrate the organization's security posture.

#	Name	Description
8	Integration with Incident Response	Integrating security operations activities with incident response processes to ensure a coordinated and effective response to security incidents related to software systems. Defining workflows and communication channels between security operations teams and incident response teams.
9	Continuous Monitoring and Improvement	Implementing continuous monitoring mechanisms to identify and respond to emerging security threats and vulnerabilities in software systems. Establishing feedback loops and processes for continuous improvement of security operations practices based on lessons learned and evolving threats.

10.4.9.1 Assessment – OWASP Top 10

The SAMM Implementation domain - Security Operations Integration indirectly addresses several OWASP threats by promoting security practices and activities that help mitigate vulnerabilities and enhance the overall security posture of software systems.

While SAMM provides a framework for software assurance and security operations integration, it's important to note that addressing OWASP threats requires a holistic approach that encompasses various security practices, including secure coding, secure configurations, vulnerability management, and continuous monitoring. Organizations should tailor their implementation based on their specific context and the OWASP threats they are most susceptible to.

While SAMM does not explicitly list OWASP threats, the following aspects of Security Operations Integration can contribute to addressing OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Implementing proper access controls and enforcing them through incident response processes and security monitoring mechanisms can help mitigate unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Establishing secure configuration management practices, including proper key management and encryption protocols, can help address vulnerabilities related to cryptographic failures.
3	Injection (A03-2021)	Incorporating security testing activities, such as code reviews and security scanning, into the software development process can help identify and mitigate injection vulnerabilities.
4	Insecure Design (A04-2021)	Following secure design principles and considering security from the early stages of development can help mitigate vulnerabilities arising from insecure design decisions.
5	Security Misconfiguration (A05-2021)	Implementing secure configuration management practices and regularly updating software components can help address security misconfiguration vulnerabilities.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	Conducting regular vulnerability assessments and tracking vulnerabilities discovered during the software development lifecycle can help mitigate risks associated with vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	Implementing secure authentication mechanisms, enforcing strong password policies, and integrating security monitoring and incident response can help address identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Implementing secure coding techniques, regular vulnerability assessments, and effective logging and monitoring practices can help mitigate software and data integrity vulnerabilities.
9	Security Logging and Monitoring Failures (A09-2021)	Implementing comprehensive logging, log analysis, and real-time monitoring mechanisms can help address vulnerabilities related to security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	Implementing robust input validation and secure configurations can help mitigate vulnerabilities related to server-side request forgery.

10.5 Security by Verification

10.5.1 Introduction

SAMM (Software Assurance Maturity Model) recognizes the importance of security verification in ensuring the effectiveness of software assurance practices. The Security by Verification domain in SAMM focuses on implementing processes and activities to validate and verify the security of software throughout its lifecycle. It aims to detect and address security vulnerabilities and weaknesses in software applications.
By addressing the Security by Verification domain, organizations can gain confidence in the security of their software applications. It helps identify and mitigate security vulnerabilities early in the development lifecycle, reducing the risk of security breaches and ensuring the delivery of secure and resilient software products.
Here are some key aspects of the Security by Verification domain in SAMM:

#	Name	Description
1	Security Testing	This involves conducting various types of security testing to identify vulnerabilities and weaknesses in software applications. It includes activities such as penetration testing, vulnerability scanning, code review, and security testing automation.
2	Code Review	This focuses on reviewing the source code of software applications to identify security flaws and adherence to secure coding practices. It includes manual code reviews as well as the use of automated code analysis tools.

#	Name	Description
3	Security Architecture Review	This involves assessing the security architecture of software applications to ensure that proper security controls and mechanisms are in place. It includes reviewing the design and implementation of security features and ensuring compliance with security standards and best practices.
4	Security Requirements Verification	This focuses on verifying that security requirements are properly defined, documented, and implemented in software applications. It involves reviewing security requirements against industry standards, regulatory requirements, and best practices.
5	Threat Modeling	This involves conducting threat modeling exercises to identify potential threats and risks to software applications. It includes analyzing the application's architecture, data flows, and potential attack vectors to identify and prioritize security threats.
6	Secure Deployment Verification	This focuses on ensuring that software applications are securely deployed and configured in the production environment. It includes verifying that security controls, encryption mechanisms, access controls, and other security measures are properly implemented during deployment.

10.5.2 Security Testing

SAMM's Verification domain focuses on security testing activities to ensure the effectiveness of security controls and identify vulnerabilities in software systems. This domain recognizes the importance of conducting thorough and systematic security testing throughout the software development life cycle.
By implementing the practices outlined in SAMM's Verification domain, organizations can enhance the security of their software systems. Thorough security testing helps identify vulnerabilities and weaknesses in the system, allowing organizations to address them proactively and improve the overall security posture of their software.
Here are the key aspects of SAMM's Verification domain, specifically related to security testing:

#	Name	Description
1	Security Testing Strategy	SAMM emphasizes the need for organizations to define a comprehensive security testing strategy. This strategy outlines the types of security testing to be performed, the frequency of testing, the testing methodologies and techniques to be used, and the roles and responsibilities of the testing team.

#	Name	Description
2	Security Testing Techniques	SAMM promotes the use of various security testing techniques to assess the security of software systems. These techniques include: a. Static Application Security Testing (SAST): SAST involves analyzing the source code, byte code, or binary code of an application to identify potential security vulnerabilities. b. Dynamic Application Security Testing (DAST): DAST involves testing the running application to identify vulnerabilities by sending crafted requests and analyzing the responses. c. Manual Code Review: Manual code review involves a manual examination of the source code to identify security flaws and vulnerabilities that automated tools may not detect. d. Penetration Testing: Penetration testing simulates real-world attacks to identify vulnerabilities and assess the effectiveness of security controls. e. Fuzz Testing: Fuzz testing involves sending unexpected and malformed data as inputs to the application to identify potential vulnerabilities and crashes.
3	Security Test Plan	SAMM encourages organizations to develop a comprehensive security test plan. This plan defines the scope, objectives, and resources required for security testing activities. It also outlines the test scenarios, test cases, and success criteria for evaluating the security of the software system.
4	Security Test Environment	SAMM highlights the importance of having a dedicated security test environment that closely resembles the production environment. This environment allows for realistic testing scenarios and reduces the risk of impacting the production system during testing.
5	Security Test Automation	SAMM promotes the use of automation tools and frameworks to streamline security testing activities. Automation can help in efficiently conducting repetitive tests, managing test data, and generating test reports.
6	Vulnerability Management	SAMM emphasizes the need for effective vulnerability management processes. This involves tracking and prioritizing identified vulnerabilities, remediating them in a timely manner, and verifying that the fixes are effective.
7	Security Testing Integration	SAMM encourages the integration of security testing activities throughout the software development life cycle. This includes testing during development, integration testing, system testing, and acceptance testing. Integrating security testing at different stages helps identify vulnerabilities early on and reduces the cost and impact of addressing them.
8	Security Testing Tools and Techniques	SAMM suggests leveraging a variety of security testing tools and techniques available in the industry. These tools can assist in automating security testing activities, identifying vulnerabilities, and generating reports.

10.5.2.1 Assessment – OWASP Top 10

While SAMM does not directly address each individual OWASP threat, it provides guidance and practices in the Verification domain that contribute to addressing various OWASP threats through comprehensive security testing and vulnerability management.
Here is a breakdown of how the SAMM's Verification domain addresses the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's emphasis on defining a comprehensive security testing strategy and conducting thorough security testing helps identify and address access control vulnerabilities that can lead to unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	SAMM promotes the use of security testing techniques such as penetration testing and manual code review, which can help identify weaknesses and mistakes in the implementation of cryptographic algorithms and mechanisms within a web application
3	Injection (A03-2021)	SAMM's security testing techniques, such as dynamic application security testing (DAST) and fuzz testing, can help identify and address injection vulnerabilities by testing for improper handling of untrusted data and ensuring proper input validation.
4	Injection (A03-2021)	SAMM's security testing techniques, such as dynamic application security testing (DAST) and fuzz testing, can help identify and address injection vulnerabilities by testing for improper handling of untrusted data and ensuring proper input validation.
5	Security Misconfiguration (A05-2021)	SAMM promotes the development of a comprehensive security test plan and the use of security test automation tools. By conducting security testing in a dedicated test environment and following secure configuration practices, organizations can address security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	SAMM encourages the use of security testing tools and techniques to identify vulnerabilities in third-party software components. By actively managing software dependencies and staying updated with security patches, organizations can reduce the risk of using vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	SAMM's focus on security testing and vulnerability management helps organizations identify and address weaknesses in identification and authentication mechanisms, reducing the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	SAMM's security testing activities help identify vulnerabilities related to the integrity of software components and data. By following secure practices, organizations can mitigate the risk of unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM highlights the importance of effective logging and monitoring practices. By implementing comprehensive logging, log analysis, and real-time monitoring mechanisms, organizations can detect and respond to security incidents, addressing the vulnerabilities associated with logging and monitoring failures.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	SAMM's security testing activities, such as penetration testing, can help identify server-side request forgery vulnerabilities. By implementing robust input validation and secure configurations, organizations can mitigate the risk of SSRF attacks.

10.5.3 Code Review

In the SAMM (Software Assurance Maturity Model) Verification domain, code review plays a crucial role in assessing the security of software applications. Code review involves manually examining the source code of an application to identify security vulnerabilities, coding errors, and other weaknesses that may impact the software's security and overall quality.

By incorporating code review practices into the SAMM Verification domain, organizations can effectively identify and address security vulnerabilities, improve the overall quality of their software, and ensure that security controls are appropriately implemented. Code review serves as an essential component of the software assurance process, enabling organizations to build more secure and reliable applications

Here's how code review contributes to the SAMM Verification domain:

#	Name	Description
1	Identification of Security Vulnerabilities	Code review allows security professionals to analyze the application's source code for potential security vulnerabilities such as injection flaws, insecure authentication, access control issues, and more. By examining the code line by line, reviewers can identify vulnerabilities that may lead to unauthorized access, data breaches, or other security incidents.
2	Detection of Coding Errors and Best Practices Violations	Code review helps identify common coding errors and violations of secure coding best practices. Reviewers can identify issues such as buffer overflows, error handling weaknesses, insecure cryptographic implementations, and other coding mistakes that could introduce security risks. By detecting these errors early on, developers can address them before they become potential security vulnerabilities.
3	Validation of Security Controls Implementation	Code review allows security professionals to validate whether security controls, such as input validation, output encoding, and access control mechanisms, are correctly implemented within the codebase. Reviewers can assess whether the controls are applied consistently, properly configured, and effectively mitigate potential security risks.
4	Assessment of Code Quality and Maintainability	Code review also helps evaluate the overall quality and maintainability of the codebase. Reviewers can identify code smells, redundant code, poor code organization, and other issues that may impact the software's long-term maintainability. By addressing these issues, organizations can improve the codebase's robustness and reduce the likelihood of introducing security vulnerabilities through code changes.

#	Name	Description
5	Feedback for Developers and Education	Code review provides an opportunity to offer feedback and guidance to developers. Reviewers can provide recommendations, suggest improvements, and educate developers about secure coding practices. This feedback loop helps developers enhance their understanding of secure coding principles and build secure software from the early stages of the development process.
6	Compliance Verification	Code review can assist in verifying compliance with security standards, regulatory requirements, and industry best practices. By examining the codebase against specific security requirements, reviewers can assess whether the application aligns with the necessary security controls and practices mandated by relevant standards or regulations.
7	Integration with Automated Testing	Code review can be integrated with automated testing tools and techniques to improve efficiency and coverage. Automated code analysis tools can assist in identifying common vulnerabilities and enforcing coding standards. The combination of manual code review and automated testing helps organizations achieve a more comprehensive assessment of the application's security posture.

10.5.3.1 Assessment – OWASP Top 10

By performing code reviews within the SAMM Verification domain, organizations can effectively address these OWASP threats, identify vulnerabilities, and apply appropriate security measures to mitigate risks and enhance the overall security of their software applications.

Here's how code review in the SAMM Verification domain addresses the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	During code review, access control mechanisms are examined to ensure that proper checks and validations are implemented to prevent unauthorized access and enforce proper authorization.
2	Cryptographic Failures (A02-2021)	Code review helps identify weaknesses and mistakes in cryptographic implementations, ensuring that secure cryptographic algorithms are used correctly and key management practices are followed.
3	Injection (A03-2021)	Code review examines input validation and sanitization techniques to identify potential injection vulnerabilities and ensure that user input is properly validated and sanitized before being processed.
4	Insecure Design (A04-2021)	Code review helps identify insecure design decisions and architectural flaws that can lead to security vulnerabilities. By reviewing the code, reviewers can identify design weaknesses and provide recommendations for secure design practices.
5	Security Misconfiguration (A05-2021)	Code review can identify misconfigurations in the application and associated components. Reviewers can check for default or insecure settings, exposed sensitive information, or unnecessary functionality that can create entry points for attackers.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021):	Code review includes examining third-party libraries and components for known vulnerabilities. Reviewers can identify outdated or insecure versions and recommend updating to secure versions or alternative components.
7	Identification and Authentication Failures (A07-2021)	Code review ensures that proper identification and authentication mechanisms are implemented. Reviewers can identify weaknesses in authentication logic, session management, and password handling to prevent unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Code review helps ensure the integrity of software components and data. Reviewers can identify potential vulnerabilities that may allow unauthorized modification, manipulation, or destruction of software or data.
9	Security Logging and Monitoring Failures (A09-2021)	Code review includes examining the logging and monitoring capabilities of the application. Reviewers can identify weaknesses in log collection, analysis, retention, and real-time monitoring, ensuring proper detection and response to security incidents.
10	Server-Side Request Forgery (A10-2021)	Code review can identify potential server-side request forgery vulnerabilities by examining how requests are handled and validating that proper input validation and security controls are in place to prevent unauthorized requests.

10.5.4 Security Architecture Review

In the SAMM (Software Assurance Maturity Model) Verification domain, security architecture review plays a significant role in assessing the security of software applications. Security architecture review involves evaluating the design and architecture of an application from a security perspective to identify potential vulnerabilities and weaknesses.

By conducting security architecture reviews within the SAMM Verification domain, organizations can proactively assess the security posture of their software applications, identify architectural weaknesses, and implement appropriate security measures. This process helps enhance the overall security and resilience of the application against potential threats and vulnerabilities

Here's how security architecture review contributes to the SAMM Verification domain:

#	Name	Description
1	Identification of Security Risks	Security architecture review helps identify potential security risks and threats that may exist in the application's design and architecture. By analyzing the security controls, data flows, trust boundaries, and communication channels, reviewers can identify areas where security vulnerabilities may be introduced or exploited.
2	Evaluation of Security Controls	Security architecture review assesses the implementation and effectiveness of security controls within the application's architecture. Reviewers examine whether appropriate security controls, such as authentication, access control, encryption, logging, and auditing mechanisms, are properly designed and implemented to mitigate identified risks.

#	Name	Description
3	Alignment with Security Best Practices	Security architecture review ensures that the application's architecture aligns with established security best practices and industry standards. Reviewers evaluate whether the application follows secure design principles, adheres to relevant security frameworks (e.g., OWASP Application Security Verification Standard), and incorporates industry-specific security requirements.
4	Assessment of Defense in Depth Strategies	Security architecture review evaluates the application's defense in depth strategies, which involve the use of multiple layers of security controls to protect against different types of threats. Reviewers assess whether the architecture includes measures such as network segmentation, secure coding practices, secure configuration management, and incident response plans.
5	Identification of Security Design Flaws	Security architecture review helps identify design flaws that may introduce vulnerabilities or weaken the overall security of the application. Reviewers analyze the architecture for potential weaknesses in data validation, session management, error handling, input/output encoding, and other security-sensitive areas.
6	Validation of Compliance Requirements	Security architecture review ensures that the application's architecture aligns with relevant compliance requirements, industry regulations, and privacy standards. Reviewers assess whether the architecture incorporates necessary controls to meet specific compliance obligations, such as GDPR, HIPAA, PCI DSS, or ISO 27001.
7	Recommendations for Security Enhancements	Security architecture review provides recommendations and guidance for improving the security of the application's architecture. Reviewers offer suggestions to address identified vulnerabilities, strengthen security controls, and mitigate risks through architectural changes or additional security measures.

10.5.4.1 Assessment – OWASP Top 10

The SAMM Verification domain, specifically the Security Architecture Review, addresses several OWASP threats.

Overall, security architecture review within the SAMM Verification domain addresses a wide range of OWASP threats by evaluating the design, implementation, and adherence to secure practices, thereby reducing the risk of vulnerabilities and enhancing the overall security posture of software applications.

Here are the contributions of Security Architecture Review to address specific OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Security architecture review helps identify access control flaws and ensures that proper access controls are implemented and enforced in the application's design and architecture.
2	Cryptographic Failures (A02-2021)	Security architecture review assesses the implementation of cryptographic mechanisms, such as encryption and key management, to identify weaknesses and ensure secure cryptographic practices are followed.

#	Name	Description
3	Injection (A03-2021)	Security architecture review helps identify design flaws and implementation weaknesses that could lead to injection vulnerabilities. It ensures that input validation, parameterization, and secure coding practices are implemented to prevent injection attacks.
4	Injection (A03-2021)	Security architecture review helps identify design flaws and implementation weaknesses that could lead to injection vulnerabilities. It ensures that input validation, parameterization, and secure coding practices are implemented to prevent injection attacks.
5	Vulnerable and Outdated Components (A06-2021)	Security architecture review examines the use of third-party libraries and components in the application's architecture to identify outdated or known vulnerable versions. It ensures that proper component management and patching practices are in place.
6	Security Misconfiguration (A05-2021)	Security architecture review assesses the application's configuration to identify insecure settings, default configurations, or unnecessary functionality that may expose the application to security risks. It helps enforce secure configuration practices.
7	Identification and Authentication Failures (A07-2021)	Security architecture review focuses on the design and implementation of identification and authentication mechanisms to ensure secure authentication processes, strong password policies, and protection against authentication bypass or impersonation attacks.
8	Software and Data Integrity Failures (A08-2021)	Security architecture review assesses the application's design and implementation to identify vulnerabilities that could compromise the integrity of software components, configurations, or data. It helps enforce secure coding techniques and data protection measures.
9	Software and Data Integrity Failures (A08-2021)	Security architecture review assesses the application's design and implementation to identify vulnerabilities that could compromise the integrity of software components, configurations, or data. It helps enforce secure coding techniques and data protection measures.
10	Server-Side Request Forgery (A10-2021)	Security architecture review helps identify design flaws and insecure configurations that could lead to server-side request forgery vulnerabilities. It ensures that proper input validation and secure configurations are implemented to prevent unauthorized requests.

10.5.5 Security Requirements Verification

In the SAMM (Software Assurance Maturity Model) Verification domain, Security Requirements Verification is an important process that ensures the software application meets the necessary security requirements. This process involves evaluating the application's security requirements, assessing their implementation, and verifying that the application meets the desired security objectives.

By conducting security requirements verification within the SAMM Verification domain, organizations can ensure that their software applications are developed and configured to meet the necessary security

requirements. This process helps in building more secure and resilient applications by integrating security from the early stages and throughout the software development lifecycle

Here's how Security Requirements Verification contributes to the SAMM Verification domain:

#	Name	Description
1	Requirement Analysis	Security Requirements Verification involves analyzing the security requirements specified for the software application. This analysis ensures that the requirements are clear, comprehensive, and aligned with the organization's security policies and standards.
2	Requirement Validation	During the security requirements verification process, the security requirements are validated to ensure they are complete, consistent, and achievable. This validation helps identify any missing or conflicting requirements that need to be addressed.
3	Security Control Implementation	Security Requirements Verification verifies that the necessary security controls are implemented in the software application as per the defined security requirements. It ensures that the appropriate security measures, such as access controls, encryption, authentication mechanisms, and input validation, are properly integrated into the application.
4	Compliance Verification	Security Requirements Verification ensures that the application's security requirements align with relevant compliance obligations, industry regulations, and standards. It verifies that the application implements the necessary controls to meet specific compliance requirements, such as GDPR, HIPAA, PCI DSS, or ISO 27001.
5	Risk Assessment	The security requirements verification process includes a risk assessment to identify potential security risks associated with the application. It helps in evaluating the effectiveness of the implemented security controls in mitigating the identified risks and ensures that the risks are adequately addressed in the security requirements.
6	Security Testing Alignment	Security Requirements Verification aligns the security testing activities with the defined security requirements. It ensures that the security testing activities, such as vulnerability assessments, penetration testing, and security code reviews, are conducted based on the identified security requirements to validate the effectiveness of the implemented controls.
7	Traceability and Documentation	Security Requirements Verification establishes traceability between the implemented security controls and the defined security requirements. It ensures that the security controls are properly documented, and their implementation can be traced back to the corresponding requirements, facilitating transparency and accountability.
8	Continuous Improvement	The security requirements verification process promotes continuous improvement by providing feedback and insights to enhance the security requirements and control implementation. It helps in identifying areas for improvement, addressing gaps, and refining the security measures to strengthen the overall security posture of the software application.

10.5.5.1 Assessment – OWASP Top 10

The SAMM Verification domain - Security Requirements Verification does not directly address each individual OWASP threat. However, the processes and activities involved in security requirements verification contribute to addressing many of the OWASP threats by ensuring that proper security controls, practices, and mitigations are implemented.

While the SAMM Verification domain - Security Requirements Verification does not explicitly list or address each OWASP threat, it provides a systematic approach to ensuring that security requirements are defined, implemented, and verified, which contributes to addressing many of the common security vulnerabilities and risks highlighted by OWASP.

Here are some examples of how the SAMM Verification domain's Security Requirements Verification process can address certain OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The requirement analysis and validation processes ensure that access control requirements are properly defined and implemented to prevent unauthorized access to resources.
2	Cryptographic Failures (A02-2021)	Security requirements verification helps in validating the correct implementation and usage of cryptographic mechanisms to ensure the confidentiality, integrity, and authenticity of data.
3	Injection (A03-2021)	By verifying security requirements, organizations can ensure that proper input validation and parameterization techniques are implemented to prevent injection attacks.
4	Insecure Design (A04-2021)	Security requirements verification helps in identifying insecure design decisions and ensuring that secure design principles are followed to mitigate potential vulnerabilities.
5	Security Misconfiguration (A05-2021)	Verification of security requirements helps identify and rectify misconfigurations in the application and associated components, reducing the risk of exposing sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	By verifying security requirements, organizations can ensure that proper practices are followed for managing software dependencies, including timely updates and patching to address known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	Verification of security requirements ensures the implementation of secure identification and authentication mechanisms to prevent unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	Security requirements verification includes measures to ensure the integrity and protection of software components, configurations, and data within the application.
9	Security Logging and Monitoring Failures (A09-2021)	Verification of security requirements includes the establishment of logging and monitoring capabilities to detect and respond to security incidents effectively.
10	Server-Side Request Forgery (A10-2021)	Security requirements verification can address SSRF vulnerabilities by validating input validation mechanisms and secure configurations to prevent unauthorized requests to internal or external resources.

10.5.6 Threat Modeling

In the SAMM (Software Assurance Maturity Model) Verification domain, Threat Modeling is a crucial process that helps identify and analyze potential threats and vulnerabilities in a software application. It involves systematically evaluating the security risks associated with the application's design, architecture, and implementation.

By conducting Threat Modeling as part of the SAMM Verification domain, organizations can proactively identify and address potential threats and vulnerabilities in their software applications. This process contributes to building more secure and resilient applications by integrating security considerations throughout the development lifecycle.

Here's how Threat Modeling contributes to the SAMM Verification domain:

#	Name	Description
1	Identify Potential Threats	Threat Modeling helps in identifying potential threats and vulnerabilities specific to the application being assessed. It involves analyzing the application's components, interfaces, and interactions to determine potential entry points and attack vectors.
2	Assess Impact and Likelihood	Threat Modeling evaluates the impact and likelihood of each identified threat. It considers factors such as the sensitivity of data, the value of the affected assets, and the likelihood of successful exploitation. This assessment helps prioritize mitigation efforts based on the most critical threats.
3	Determine Mitigation Strategies	Threat Modeling guides the development of mitigation strategies to address the identified threats. It involves selecting appropriate security controls and countermeasures to reduce the risk associated with each threat. These strategies can include architectural changes, code-level security measures, or additional safeguards.
4	Analyze Security Controls	Threat Modeling assesses the effectiveness of existing security controls in mitigating the identified threats. It helps identify gaps or weaknesses in the application's security measures and highlights areas where additional controls may be required.
5	Integration with Secure Development Lifecycle	Threat Modeling is integrated into the software development lifecycle to ensure that security considerations are incorporated from the early stages of application design. It helps foster a proactive security mindset and enables the identification and resolution of potential security issues early in the development process.
6	Support Decision-Making	Threat Modeling provides valuable insights for decision-making related to resource allocation, risk management, and security investments. It helps stakeholders understand the potential impact of different threats and make informed decisions regarding security priorities and investments.

10.5.6.1 Assessment – OWASP Top 10

Threat Modeling in the SAMM Verification domain contributes to addressing a wide range of OWASP threats by identifying vulnerabilities, guiding mitigation strategies, and integrating security considerations throughout the software development lifecycle.

Here's how the SAMM Verification domain, specifically Threat Modeling, addresses the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Threat Modeling helps identify potential access control weaknesses and design flaws, enabling organizations to implement proper access controls and prevent unauthorized access.
2	Cryptographic Failures (A02-2021)	Threat Modeling assesses the implementation of cryptographic mechanisms, identifying weaknesses and guiding organizations to follow recommended practices for secure encryption, key management, and configuration.
3	Injection (A03-2021)	Threat Modeling identifies input validation vulnerabilities, helping organizations implement secure coding practices and proper input handling to prevent injection attacks.
4	Insecure Design (A04-2021)	Threat Modeling highlights insecure design decisions, enabling organizations to address architectural flaws, consider security from the early stages of development, and make informed design choices.
5	Security Misconfiguration (A05-2021)	Threat Modeling identifies insecure configurations, guiding organizations to implement secure configuration practices and eliminate unnecessary functionality, reducing the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Threat Modeling helps identify dependencies and assess their security, enabling organizations to manage software dependencies, stay updated with security patches, and address known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	Threat Modeling evaluates identification and authentication mechanisms, allowing organizations to implement secure authentication methods, enforce strong password policies, and prevent unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Threat Modeling assesses the integrity of software and data, helping organizations implement secure coding techniques, ensure data integrity, and protect against unauthorized modifications.
9	Security Logging and Monitoring Failures (A09-2021)	Threat Modeling identifies deficiencies in logging and monitoring practices, guiding organizations to implement comprehensive logging, effective log analysis, and real-time monitoring mechanisms.
10	Server-Side Request Forgery (A10-2021)	Threat Modeling identifies vulnerabilities that can lead to SSRF, enabling organizations to implement input validation and secure configurations to prevent unauthorized requests to internal or external resources.

10.5.7 Secure Deployment Verification

In the SAMM (Software Assurance Maturity Model) Verification domain, Secure Deployment Verification focuses on ensuring that software applications are securely deployed and configured in production environments. It aims to validate that the deployment process adheres to security best practices and mitigates potential risks.

By focusing on these aspects, the Secure Deployment Verification domain contributes to the overall security of the software application throughout its deployment lifecycle. It helps organizations ensure that their applications are deployed securely, reducing the risk of security breaches, data leaks, and other security incidents in production environments.

Here are some key aspects and contributions of the Secure Deployment Verification domain:

#	Name	Description
1	Deployment Process Validation	Secure Deployment Verification involves validating the deployment process to ensure it follows secure practices. This includes verifying that the deployment is performed using secure configurations, securely transferring software components, and utilizing secure deployment tools and mechanisms.
2	Secure Configuration Management	It emphasizes the importance of securely managing configuration settings for the deployed application and its supporting infrastructure. This includes maintaining a secure baseline configuration, securely storing and managing secrets and credentials, and regularly updating and patching deployed components.
3	Infrastructure Security Assessment	Secure Deployment Verification includes assessing the security of the underlying infrastructure where the application is deployed. This assessment helps identify any vulnerabilities or misconfigurations in the infrastructure components, such as servers, networks, databases, and other supporting systems.
4	Secure Data Transfer and Storage	It focuses on ensuring the secure transfer and storage of sensitive data within the deployed application. This includes encrypting data in transit using secure protocols, implementing secure data storage mechanisms, and enforcing access controls to protect sensitive information.
5	Secure Network Configuration	Secure Deployment Verification addresses the configuration of network components to establish secure communication channels between different application components and external entities. This includes implementing secure network segmentation, firewall rules, intrusion detection and prevention systems, and secure network protocols.
6	Authentication and Authorization Validation	It involves validating the authentication and authorization mechanisms implemented in the deployed application. This includes verifying that strong and secure authentication mechanisms are used, access controls are properly enforced, and privilege escalation risks are mitigated.

10.5.7.1 Assessment – OWASP Top 10

The Software Assurance Maturity Model (SAMM) Verification Domain - Secure Deployment Verification indirectly addresses several OWASP threats by focusing on secure deployment practices. While it does not specifically mention OWASP threats, the contributions of Secure Deployment Verification align with mitigating various OWASP vulnerabilities.

It's important to note that while Secure Deployment Verification addresses these aspects, it is also essential to apply other security practices and conduct comprehensive security testing to fully address OWASP threats and vulnerabilities.

Here's how the domain addresses some of the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Secure Deployment Verification ensures that access controls are properly implemented and enforced during the deployment process, reducing the risk of unauthorized access to resources.
2	Cryptographic Failures (A02-2021)	Secure Deployment Verification emphasizes secure configuration management, which includes managing cryptographic settings and ensuring the correct implementation of cryptographic algorithms and mechanisms.
3	Injection (A03-2021)	Secure Deployment Verification validates the deployment process to ensure that input validation and parameterization techniques are implemented correctly, reducing the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	Secure Deployment Verification emphasizes the importance of securely configuring and deploying the application's architecture and infrastructure, reducing the likelihood of insecure design decisions.
5	Security Misconfiguration (A05-2021)	Secure Deployment Verification focuses on securely configuring the deployed application and associated components, reducing the risk of security misconfigurations that can lead to vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Secure Deployment Verification includes assessing the security of the deployed infrastructure, which can help identify and address vulnerabilities in third-party components used in the application.
7	Identification and Authentication Failures (A07-2021)	Secure Deployment Verification validates the implementation of authentication and authorization mechanisms during the deployment process, reducing the risk of identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Secure Deployment Verification ensures the integrity of software components and data during deployment, reducing the risk of unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	Secure Deployment Verification emphasizes the importance of implementing proper logging and monitoring mechanisms, enhancing the detection and response capabilities to security incidents.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	While not directly mentioned, Secure Deployment Verification's focus on secure network configuration and access controls indirectly helps mitigate the risk of server-side request forgery by ensuring that unintended requests to internal or external resources are prevented.

10.6 Security by Operations

10.6.1 Introduction

SAMM's Operations domain focuses on the activities and practices related to managing the secure operations of software systems. This domain encompasses various aspects that contribute to ensuring the ongoing security and reliability of software applications.

By implementing the practices outlined in SAMM's Operations domain, organizations can establish robust operational processes and practices to support the secure and reliable operation of software systems. This domain helps organizations address the operational aspects of software security and minimize the risk of security incidents and vulnerabilities throughout the software's lifecycle.

Here are the key components of SAMM's Operations domain:

#	Name	Description
1	Environment Hardening	SAMM emphasizes the importance of hardening the software environment by implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations. This helps protect the software system from known vulnerabilities and security weaknesses.
2	Secure Build	SAMM promotes the use of secure build practices, including secure coding standards, secure build tools, and secure software development frameworks. These practices help ensure that the software is developed with security in mind, reducing the likelihood of introducing vulnerabilities during the build process.
3	Configuration Management	SAMM advocates for effective configuration management practices, such as version control, change management, and documentation of software configurations. These practices enable organizations to maintain the integrity of software systems and ensure that unauthorized changes or misconfigurations are detected and remediated.
4	Vulnerability Management	SAMM emphasizes the importance of implementing a robust vulnerability management process, including vulnerability scanning, vulnerability assessment, and penetration testing. These activities help identify and prioritize vulnerabilities, enabling organizations to address them in a timely manner and reduce the risk of exploitation.
5	Incident Management	SAMM recognizes the need for a well-defined incident management process to handle security incidents effectively. This includes incident detection, response, containment, and recovery activities. By implementing incident management practices, organizations can minimize the impact of security incidents and restore normal operations promptly.

#	Name	Description
6	Secure Deployment	SAMM promotes secure deployment practices, including secure software distribution, secure installation procedures, and secure configuration of software components. This helps ensure that software is deployed in a secure and controlled manner, reducing the risk of unauthorized access or compromise during deployment.
7	Security Testing	SAMM encourages the integration of security testing throughout the software development life cycle. This includes activities such as static code analysis, dynamic application security testing (DAST), and security code reviews. By conducting regular security testing, organizations can identify and address security vulnerabilities before software systems are deployed.
8	Operational Enablement	SAMM emphasizes the importance of providing operational support and guidance to ensure the ongoing security and reliability of software systems. This includes activities such as security training for operational staff, documentation of operational procedures, and establishing incident response capabilities.

10.6.2 Environment Hardening

The SAMM (Software Assurance Maturity Model) Operations domain - Environment Hardening focuses on implementing security measures to harden the software development and operational environments. It aims to minimize vulnerabilities and protect software systems from various security risks.

These aspects of environment hardening in the SAMM Operations domain aim to reduce the attack surface, mitigate vulnerabilities, and enhance the overall security of the software development and operational environments. It helps organizations establish a robust and secure foundation for their software systems.

Here are some aspects covered by the Environment Hardening domain in SAMM:

#	Name	Description
1	Secure Configuration Management	Establishing and enforcing secure configuration standards for software components, platforms, and infrastructure. Implementing change management processes to ensure that configurations are reviewed, approved, and properly managed.
2	System Hardening	Applying security guidelines and best practices to secure the underlying operating systems, databases, servers, and other infrastructure components. Disabling unnecessary services, removing default accounts, and applying appropriate access controls.
3	Network Security	Implementing network segmentation to isolate critical systems and protect against lateral movement by attackers. Configuring firewalls, intrusion detection/prevention systems, and other network security controls to monitor and protect the environment.

#	Name	Description
4	Secure Deployment	Implementing secure deployment practices to ensure that software is deployed in a secure manner. Automating deployment processes and incorporating security checks to minimize human errors and vulnerabilities.
5	Data Protection	Implementing appropriate encryption and data protection mechanisms to safeguard sensitive data at rest and in transit. Applying access controls and encryption for backups, archives, and other data storage systems.
6	Incident Response Readiness	Establishing incident response plans and procedures to effectively respond to and mitigate security incidents. Conducting regular incident response exercises and simulations to test and improve the readiness of the response team.
7	Patch and Vulnerability Management	Establishing processes to regularly identify, assess, and remediate vulnerabilities in software components, frameworks, and infrastructure. Keeping systems up to date with security patches and updates to minimize the risk of known vulnerabilities being exploited.
8	Security Monitoring and Auditing	Implementing monitoring mechanisms to detect and respond to security events and anomalies within the environment. Logging and auditing activities to facilitate forensic analysis, compliance requirements, and security incident investigations.
9	Access Control and Authentication	Implementing strong access controls and authentication mechanisms to ensure that only authorized personnel can access critical systems and resources. Implementing multi-factor authentication and secure identity and access management practices.
10	Physical Security	Implementing physical security measures to protect the physical infrastructure hosting the software systems, such as data centers, server rooms, and storage facilities.

10.6.2.1 Assessment – OWASP Top 10

The Software Assurance Maturity Model (SAMM) Operations domain - Environment Hardening indirectly addresses several OWASP threats by implementing security measures and best practices.

While the Environment Hardening domain in SAMM addresses many security aspects, it is important to note that it may not directly cover all the specific details of each OWASP threat. However, by implementing the recommended practices and measures within the SAMM Operations domain, organizations can significantly improve their security posture and reduce the risk of OWASP-related vulnerabilities.

Here is a list of contributions made by the Environment Hardening domain in SAMM that address specific OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Implementing secure configuration management and enforcing access control standards helps prevent unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Implementing secure configuration management ensures proper encryption and data protection mechanisms are in place.
3	Injection (A03-2021)	Implementing secure deployment practices, including input validation and parameterization, helps mitigate the risk of injection attacks.
4	Insecure Design (A04-2021)	Implementing secure design principles and following security best practices helps address vulnerabilities stemming from insecure application design.
5	Security Misconfiguration (A05-2021)	Implementing secure configuration management and adhering to secure practices ensures that web applications and associated components are properly configured, reducing the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Implementing patch and vulnerability management processes helps identify and remediate vulnerabilities in software components, reducing the risk of using outdated or vulnerable components.
7	Identification and Authentication Failures (A07-2021)	Implementing access control and authentication mechanisms helps address weaknesses in identification and authentication, reducing the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	Implementing secure deployment practices and following secure coding techniques helps ensure the integrity of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	Implementing security monitoring and auditing mechanisms helps detect and respond to security events and anomalies within the environment, enhancing the logging and monitoring capabilities of web applications.
10	Server-Side Request Forgery (A10-2021)	Implementing secure configuration management and secure design practices helps mitigate server-side request forgery vulnerabilities by validating and sanitizing user input and implementing proper access controls.

10.6.3 Secure Build

The SAMM (Software Assurance Maturity Model) Operations domain - Secure Build focuses on establishing and implementing secure build practices within the software development and deployment processes. It aims to ensure that the software is built securely from the ground up, reducing the risk of vulnerabilities and weaknesses that could be exploited by attackers.

By incorporating these aspects into the software development and deployment processes, the Secure Build domain of SAMM helps organizations build and release software with a strong focus on security, reducing the likelihood of vulnerabilities and increasing the overall software assurance.

The Secure Build domain covers the following aspects:

#	Name	Description
1	Secure Development Environment	Establishing a secure development environment with secure workstations, development tools, and libraries. Implementing secure coding practices and guidelines to ensure that developers follow secure coding standards.
2	Secure Coding Techniques	Promoting the use of secure coding techniques, such as input validation, output encoding, and proper error handling, to prevent common vulnerabilities like injection attacks, cross-site scripting (XSS), and buffer overflows.
3	Secure Development Frameworks:	Encouraging the use of secure development frameworks and libraries that have undergone security testing and are known to follow secure coding practices.
4	Secure Code Reviews	Conducting regular code reviews to identify and address security flaws and vulnerabilities in the codebase. Implementing automated code analysis tools to assist in identifying potential security issues.
5	Secure Testing	Integrating security testing into the software development lifecycle, including techniques such as static code analysis, dynamic application security testing (DAST), and penetration testing. Regularly testing the application for security vulnerabilities and weaknesses.
6	Secure Build and Deployment	Implementing secure build processes to ensure that the software is built from trusted sources and is free from malicious code. Utilizing secure build tools and techniques to protect against code tampering or unauthorized modifications during the build and deployment process.
7	Secure Configuration Management	Establishing secure configuration management practices to ensure that the deployed software is configured securely. Managing and securing configuration files, including sensitive information such as passwords and cryptographic keys.
8	Secure Release Management	Implementing secure release management practices to ensure that the software is released securely and that the deployment process follows secure procedures. Verifying the integrity and authenticity of the software releases and ensuring that they are properly signed and protected against tampering.
9	Secure Supply Chain	Managing the security of the software supply chain by ensuring that all dependencies and third-party components used in the software are trusted and regularly updated. Conducting security assessments and due diligence on third-party suppliers and vendors.

10.6.3.1 Assessment – OWASP Top 10

The SAMM Operations domain - Secure Build addresses several OWASP threats directly.

The Secure Build domain of SAMM directly addresses multiple OWASP threats by incorporating secure coding practices, secure design principles, secure configuration management, security testing, and supply chain management, among other practices.
Here is a list of contributions from the Secure Build domain that align with specific OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Implementing secure coding practices and guidelines to ensure proper access controls are enforced.
2	Cryptographic Failures (A02-2021)	Encouraging the use of secure development frameworks and libraries that follow recommended cryptographic practices. Implementing secure configuration management practices, including secure handling of cryptographic keys.
3	Injection (A03-2021)	Promoting secure coding techniques, such as input validation and parameterization, to prevent injection vulnerabilities. Integrating security testing, including static code analysis and dynamic application security testing, to identify and mitigate injection vulnerabilities.
4	Insecure Design (A04-2021)	Implementing secure design principles and considering security from the early stages of development to prevent insecure design decisions.
5	Security Misconfiguration (A05-2021)	Establishing secure configuration management practices to avoid security misconfigurations. Regularly updating software components and conducting security assessments to address misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Managing the security of the software supply chain by ensuring the use of trusted and updated third-party components.
7	Identification and Authentication Failures (A07-2021)	Implementing secure authentication mechanisms and enforcing strong password policies to address identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Implementing secure coding techniques to ensure the integrity of software components and data. Regularly testing for vulnerabilities and unauthorized modifications to prevent integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Establishing effective logging and monitoring practices within the software development and deployment processes. Implementing real-time monitoring mechanisms to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	Implementing secure coding practices and input validation techniques to prevent server-side request forgery vulnerabilities.

10.6.4 Configuration Management

The SAMM (Software Assurance Maturity Model) Operations domain - Configuration Management focuses on establishing and maintaining secure configuration management practices within the software

development and deployment processes. It aims to ensure that the software is configured securely throughout its lifecycle, reducing the risk of misconfigurations that could lead to vulnerabilities or security breaches.

By incorporating these aspects into the software development and deployment processes, the Configuration Management domain of SAMM helps organizations establish and maintain secure configuration practices, ensuring the integrity and security of the software throughout its lifecycle.

The Configuration Management domain covers the following aspects:

#	Name	Description
1	Configuration Management Policies and Procedures	Defining and documenting policies and procedures for configuration management. Establishing guidelines for secure configuration practices.
2	Configuration Item Identification	Identifying and documenting the configuration items within the software system. Maintaining an inventory of software and hardware components.
3	Configuration Baseline Management	Establishing and managing configuration baselines for different software versions or releases. Ensuring that the configuration baselines are properly documented and controlled.
4	Configuration Change Management	Implementing a formal process for managing and controlling configuration changes. Evaluating and approving configuration change requests based on their impact and security considerations.
5	Configuration Item Version Control	Implementing version control mechanisms for configuration items to track changes and maintain integrity. Applying version control practices to both software source code and related configuration files.
6	Configuration Verification and Audit	Conducting regular verification and audit activities to ensure that the software configuration is compliant with established policies and standards. Performing configuration audits to identify and address any deviations or non-compliance.
7	Configuration Documentation and Reporting	Documenting the configuration management processes, including configurations, dependencies, and relationships. Generating reports on the configuration status, changes, and compliance for effective monitoring and decision-making.
8	Secure Configuration Deployment	Implementing secure deployment practices to ensure that the software is deployed with the correct and secure configuration settings. Protecting configuration files and sensitive information during the deployment process.

10.6.4.1 Assessment – OWASP Top 10

The Software Assurance Maturity Model (SAMM) Operations domain - Configuration Management indirectly addresses several OWASP threats by promoting secure configuration practices and controls within the software development and deployment processes.

It's important to note that while SAMM provides guidance on secure configuration management practices, organizations should also consider additional security measures and specific OWASP guidelines to comprehensively address each threat.

While SAMM does not explicitly mention OWASP threats, the implementation of secure configuration management practices can help mitigate the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	By establishing and enforcing proper access control mechanisms as part of configuration management policies and procedures, SAMM helps reduce the risk of broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	SAMM promotes secure configuration practices, including the correct usage and configuration of cryptographic algorithms and mechanisms. This can help mitigate cryptographic failures in web applications.
3	Injection (A03-2021)	Proper configuration management, including input validation and secure coding practices, can reduce the risk of injection vulnerabilities by ensuring untrusted data is handled securely.
4	Insecure Design (A04-2021)	SAMM encourages secure design principles and the consideration of security throughout the software development lifecycle. By adopting secure design practices, organizations can mitigate insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	SAMM emphasizes secure configuration practices and the regular assessment of software configurations. This helps reduce the risk of security misconfigurations in web applications.
6	Vulnerable and Outdated Components (A06-2021)	SAMM promotes the management of software dependencies, including keeping them updated and applying security patches. This reduces the risk of using outdated or vulnerable components within web applications.
7	Identification and Authentication Failures (A07-2021)	By implementing secure authentication mechanisms and enforcing strong password policies, SAMM helps mitigate identification and authentication failures in web applications.
8	Software and Data Integrity Failures (A08-2021)	SAMM promotes secure practices and configurations to ensure the integrity and protection of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	While not directly mentioned, SAMM encourages the implementation of comprehensive logging and monitoring mechanisms, which can help address security logging and monitoring failures in web applications.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Although not explicitly mentioned, SAMM promotes secure configuration practices, which can help mitigate server-side request forgery vulnerabilities by ensuring that servers are configured securely and access to internal or external resources is properly controlled.

10.6.5 Vulnerability Management

The SAMM (Software Assurance Maturity Model) Operations domain - Vulnerability Management focuses on establishing effective practices for identifying, assessing, and mitigating vulnerabilities in software applications. It aims to ensure that organizations have robust vulnerability management processes in place to proactively identify and address security weaknesses, reducing the risk of exploitation and potential damage.

By incorporating these aspects into their software development and operations processes, organizations can enhance their ability to identify, assess, and mitigate vulnerabilities in their software applications, thereby improving the overall security posture and reducing the risk of exploitation.

The Vulnerability Management domain covers the following aspects:

#	Name	Description
1	Vulnerability Assessment:	Conducting regular vulnerability assessments to identify potential vulnerabilities in software applications. Using automated scanning tools and manual techniques to identify security weaknesses. Prioritizing vulnerabilities based on their severity and potential impact on the system.
2	Vulnerability Remediation	Establishing processes for remediation of identified vulnerabilities. Assigning responsibility for addressing vulnerabilities and establishing timelines for remediation. Tracking and verifying the completion of remediation activities.
3	Patch Management	Implementing a patch management process to address vulnerabilities in software components and dependencies. Regularly monitoring for available patches and updates. Testing and deploying patches in a timely manner to mitigate known vulnerabilities.
4	Secure Development Practices	Incorporating secure coding practices and secure development lifecycle (SDL) methodologies to prevent the introduction of vulnerabilities during the software development process. Educating developers on secure coding practices and providing them with tools and resources to identify and address common vulnerabilities.

#	Name	Description
5	Security Testing	Conducting regular security testing, including penetration testing, to identify vulnerabilities that may not be caught by automated scanning tools. Performing code reviews and security assessments to identify and address vulnerabilities in the application's source code.
6	Incident Response Planning	Developing an incident response plan that includes procedures for responding to and mitigating vulnerabilities and security incidents. Establishing communication channels and responsibilities for reporting and addressing vulnerabilities promptly.
7	Continuous Monitoring	Implementing continuous monitoring mechanisms to detect and respond to new vulnerabilities as they emerge. Staying informed about the latest security threats and vulnerabilities through vulnerability intelligence sources.

10.6.5.1 Assessment – OWASP Top 10

The Software Assurance Maturity Model (SAMM) Operations domain - Vulnerability Management indirectly addresses several OWASP (Open Web Application Security Project) threats by incorporating effective practices for identifying, assessing, and mitigating vulnerabilities in software applications. Although SAMM does not directly mention the OWASP threat names and codes, its focus on vulnerability management aligns with addressing these threats.

It's important to note that while SAMM provides a framework for establishing effective vulnerability management practices, it is still necessary for organizations to align these practices with specific OWASP recommendations and guidelines to comprehensively address the corresponding threats.

Here's how SAMM contributes to mitigating OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's emphasis on vulnerability assessments, secure development practices, and incident response planning helps identify and address access control weaknesses, reducing the risk of unauthorized access.
2	Cryptographic Failures (A02-2021)	SAMM promotes secure development practices and secure coding methodologies, which include the proper implementation of cryptographic algorithms and mechanisms, reducing the risk of cryptographic vulnerabilities.
3	Injection (A03-2021)	SAMM's emphasis on security testing, code reviews, and secure development practices helps identify and mitigate injection vulnerabilities by ensuring proper input validation and handling.
4	Insecure Design (A04-2021)	SAMM's focus on secure development practices, secure coding, and incorporating security considerations into the design phase helps address insecure design vulnerabilities and ensures the adoption of secure architectural decisions.

#	Name	Description
5	Security Misconfiguration (A05-2021)	SAMM's emphasis on secure configuration practices, vulnerability assessments, and continuous monitoring helps organizations identify and address security misconfigurations, reducing the risk of insecure configurations.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's patch management practices, continuous monitoring mechanisms, and secure development practices help organizations manage and update software components, reducing the risk of using vulnerable or outdated dependencies.
7	Identification and Authentication Failures (A07-2021)	SAMM's focus on secure authentication mechanisms, secure development practices, and security testing helps address identification and authentication vulnerabilities, reducing the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	SAMM's secure coding practices, secure development lifecycle, and vulnerability assessments contribute to mitigating software and data integrity vulnerabilities by ensuring the integrity and protection of software components and data.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's emphasis on continuous monitoring, incident response planning, and secure development practices helps organizations improve their logging and monitoring capabilities, reducing the risk of security incidents going undetected.
10	Server-Side Request Forgery (A10-2021)	While SAMM does not explicitly mention SSRF, its emphasis on secure development practices, input validation, and security testing contributes to mitigating vulnerabilities that could lead to SSRF attacks.

10.6.6 Incident Management

The SAMM (Software Assurance Maturity Model) Operations domain - Incident Management focuses on establishing effective practices for detecting, responding to, and mitigating security incidents related to software applications. It aims to ensure that organizations have robust incident management processes in place to promptly identify and address security breaches, minimizing the impact and potential damage.
By incorporating these aspects into their incident management processes, organizations can effectively detect, respond to, and mitigate security incidents related to their software applications, minimizing the impact and reducing the risk of future incidents.
The Incident Management domain covers the following aspects:

#	Name	Description
1	Incident Response Planning	Developing an incident response plan that outlines the procedures, roles, and responsibilities for responding to security incidents. This includes establishing communication channels, defining escalation processes, and identifying key stakeholders.

#	Name	Description
2	Incident Detection	Implementing mechanisms and tools for detecting security incidents, such as intrusion detection systems (IDS), security information and event management (SIEM) systems, and log monitoring. This involves setting up alerts and triggers to identify potential security breaches.
3	Incident Response	Defining procedures and workflows for responding to security incidents in a coordinated and timely manner. This includes steps for containment, eradication, and recovery, as well as evidence gathering and preservation for forensic analysis.
4	Incident Reporting and Communication	Establishing channels and processes for reporting security incidents internally within the organization and, if necessary, externally to relevant stakeholders, such as customers, partners, or regulatory bodies. This ensures timely and accurate communication to facilitate effective incident management.
5	Incident Analysis and Investigation	Conducting thorough investigations and analysis of security incidents to understand the root causes, impact, and potential vulnerabilities exploited. This includes leveraging forensic techniques, analyzing logs, and collaborating with relevant teams to identify lessons learned and improve security practices.
6	Continuous Improvement	Incorporating the findings from incident analysis and investigations into the organization's security practices. This involves updating incident response plans, implementing preventive measures to mitigate similar incidents in the future, and enhancing the overall security posture.

10.6.6.1 Assessment – OWASP Top 10

The SAMM Operations domain - Incident Management indirectly addresses some of the OWASP (Open Web Application Security Project) Top 10 threats by establishing effective incident management practices. While it does not directly address each individual threat, it provides a framework for organizations to detect, respond to, and mitigate security incidents related to software applications, which can help in addressing various OWASP threats.

While the SAMM Operations domain - Incident Management provides a framework for incident management, organizations should also incorporate specific measures and controls to directly address each OWASP threat in their software development lifecycle and security practices.

Here's how it aligns with some of the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Incident management processes can help identify and respond to unauthorized access attempts, ensuring access controls are properly enforced and preventing privilege escalation.
2	Cryptographic Failures (A02-2021)	Incident management can assist in identifying and mitigating security incidents related to cryptographic weaknesses, such as improper usage or key management issues.

#	Name	Description
3	Injection (A03-2021)	Incident management processes can aid in detecting and responding to injection attacks, facilitating timely containment and recovery to prevent further exploitation.
4	Insecure Design (A04-2021)	Incident management helps identify security incidents resulting from insecure design decisions, enabling organizations to take corrective actions to improve the design and architecture of their applications.
5	Security Misconfiguration (A05-2021)	Incident management practices can help detect and respond to security incidents caused by misconfigurations, allowing organizations to address these issues promptly and prevent further exploitation.
6	Vulnerable and Outdated Components (A06-2021)	Incident management processes assist in identifying and mitigating security incidents related to the use of vulnerable or outdated software components, enabling organizations to update or replace them to mitigate risk.
7	Identification and Authentication Failures (A07-2021)	Incident management can aid in detecting and responding to incidents related to authentication weaknesses, helping organizations identify unauthorized access attempts and take appropriate actions.
8	Software and Data Integrity Failures (A08-2021)	Incident management practices assist in identifying and responding to incidents that compromise the integrity of software components and data, enabling organizations to restore integrity and strengthen their security measures
9	Security Logging and Monitoring Failures (A09-2021)	Incident management processes emphasize the importance of logging and monitoring security-related events, helping organizations detect and respond to incidents more effectively.
10	Server-Side Request Forgery (A10-2021)	Incident management can aid in identifying and responding to incidents related to server-side request forgery, enabling organizations to mitigate the risk of unauthorized access and potential exploitation.

10.6.7 Secure Deployment

Addressing these aspects of secure deployment, organizations can enhance the security of their software applications and minimize the risk of security incidents or vulnerabilities during the deployment phase.
The Secure Deployment activity in the SAMM Security by Verification domain contributes to addressing the following aspects:

#	Name	Description
1	Security Testing	Secure deployment involves conducting security testing activities to validate the security of the software application before it is deployed. This includes activities such as penetration testing, vulnerability scanning, code review, and security testing automation. By conducting these tests, organizations can identify and address vulnerabilities or weaknesses in the software application, ensuring that it is secure before deployment.
2	Configuration Management	Secure deployment includes implementing effective configuration management practices. This involves managing and documenting software configurations, including version control and change management. By ensuring that the software is deployed with the correct and secure configuration settings, organizations can minimize the risk of misconfigurations or unauthorized changes that could introduce vulnerabilities.
3	Environment Hardening	Secure deployment also focuses on environment hardening, which involves implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations. By hardening the software environment, organizations can protect the software application from known vulnerabilities and security weaknesses, reducing the risk of exploitation.
4	Secure Build	Secure deployment includes following secure build practices, such as adhering to secure coding standards, using secure build tools, and leveraging secure software development frameworks. By incorporating secure build practices during the deployment process, organizations can ensure that the software is developed with security in mind, reducing the likelihood of introducing vulnerabilities during the deployment phase.
5	Secure Software Distribution	Secure deployment involves implementing secure software distribution practices. This includes ensuring the integrity and authenticity of the software packages during distribution, verifying digital signatures, and using secure channels for distribution. By securely distributing the software, organizations can prevent tampering or unauthorized modifications to the software during the deployment process.
6	Secure Installation Procedures	Secure deployment includes defining and implementing secure installation procedures for the software application. This involves providing clear instructions and guidelines for installing the software securely, including appropriate access controls, authentication mechanisms, and encryption measures. By following secure installation procedures, organizations can ensure that the software is deployed in a secure and controlled manner.

10.6.7.1 Assessment – OWASP Top 10

Security By Verification domain - Secure Deployment activity addresses each item in the OWASP Top 10:

#	Name	Description
1	Broken Access Control (A01-2021)	SAMM's Secure Deployment activity emphasizes access control and secure installation procedures, which can help address broken access control vulnerabilities by implementing appropriate authorization mechanisms and securely configuring access controls.
2	Cryptographic Failures (A02-2021)	SAMM's Secure Deployment activity does not explicitly address cryptographic failures. However, organizations can incorporate secure coding practices, code review, and secure deployment procedures during this activity to mitigate cryptographic vulnerabilities.
3	Injection (A03-2021)	SAMM's Secure Deployment activity focuses on secure coding practices, code review, and secure deployment procedures, which can help address injection vulnerabilities by implementing proper input validation and secure coding techniques to prevent code injection attacks.
4	Insecure Design (A04-2021)	SAMM's Secure Deployment activity does not explicitly address insecure design. However, organizations can consider secure design principles and secure coding practices during the software deployment process to mitigate insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	SAMM's Secure Deployment activity emphasizes configuration management and environment hardening, which can help address security misconfiguration vulnerabilities by implementing secure configurations and effectively managing system settings.
6	Vulnerable and Outdated Components (A06-2021)	SAMM's Secure Deployment activity includes secure distribution and installation procedures, which can help address this issue by ensuring that software components are obtained from trusted sources and regularly updated to avoid using vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	SAMM's Secure Deployment activity focuses on secure installation procedures and appropriate access controls, which can help address identification and authentication failures by enforcing strong authentication mechanisms and securely managing user credentials.
8	Software and Data Integrity Failures (A08-2021)	SAMM's Secure Deployment activity does not explicitly address software and data integrity failures. However, organizations can incorporate secure deployment procedures, including secure distribution and installation, to ensure the integrity of software and data.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM's Secure Deployment activity does not explicitly address security logging and monitoring failures. However, organizations can include logging and monitoring practices as part of their overall secure deployment processes to detect and respond to security incidents effectively.
10	Server-Side Request Forgery (A10-2021)	SAMM's Secure Deployment activity focuses on secure coding practices, code review, and secure deployment procedures, which can help address server-side request forgery vulnerabilities by implementing proper input validation and securely handling user-supplied data.

10.6.8 Security Testing

The SAMM (Software Assurance Maturity Model) Operations domain - Security Testing focuses on establishing effective practices for testing the security of software applications. It aims to ensure that organizations have robust security testing processes in place to identify vulnerabilities and weaknesses in their applications, enabling them to address these issues and enhance the overall security posture.
By incorporating these aspects into their security testing processes, organizations can identify vulnerabilities and weaknesses in their software applications, enabling them to prioritize and implement appropriate security controls and measures. This, in turn, helps enhance the overall security of the applications and protects against potential security breaches and attacks.
The Security Testing domain covers the following aspects:

#	Name	Description
1	Security Testing Policy	Developing a comprehensive policy that outlines the objectives, scope, and approach for security testing activities. This policy defines the guidelines and standards to be followed during testing.
2	Security Testing Methodology	Establishing a structured and systematic approach to security testing, including techniques, tools, and processes to be used. This methodology guides testers in conducting thorough assessments of application security.
3	Security Testing Requirements	Defining specific security requirements that applications must meet, including compliance with relevant regulations, industry standards, and best practices. These requirements serve as a basis for security testing activities.
4	Security Test Planning	Creating a detailed plan for conducting security tests, including defining test objectives, test scenarios, test cases, and test data. This plan ensures that security testing activities are well-organized and comprehensive.
5	Security Test Execution	Executing security tests according to the defined plan, utilizing various techniques such as vulnerability scanning, penetration testing, code review, and security-focused testing. This involves assessing the application for vulnerabilities and weaknesses.
6	Security Test Reporting	Documenting and communicating the results of security tests, including identified vulnerabilities, their severity, and recommended remediation actions. This reporting facilitates understanding of the security posture and prioritization of security improvement efforts.
7	Security Test Automation	Incorporating automation tools and techniques into security testing processes to enhance efficiency, coverage, and repeatability. Automated testing helps identify common vulnerabilities and enables more frequent testing.
8	Security Test Metrics and Measurement	Defining metrics and measurement criteria to assess the effectiveness and efficiency of security testing activities. This enables organizations to evaluate the maturity of their security testing processes and identify areas for improvement.

#	Name	Description
9	Security Test Environment	Establishing a dedicated test environment that closely resembles the production environment to ensure realistic testing conditions. This environment allows for thorough assessment of the application's security controls.
10	Security Test Training and Awareness	Providing training and awareness programs to ensure that testers have the necessary knowledge and skills to effectively conduct security testing. This helps improve the quality and accuracy of security testing activities.

10.6.8.1 Assessment – OWASP Top 10

The Software Assurance Maturity Model (SAMM) Operations domain - Security Testing indirectly addresses the OWASP threats by promoting a comprehensive and systematic approach to security testing. While SAMM does not explicitly mention each OWASP threat, it provides a framework for organizations to establish effective security testing practices, which can help identify and address vulnerabilities that align with the OWASP Top 10 threats.

While the SAMM Operations domain - Security Testing does not directly mention each OWASP threat, it provides a framework and guidance for organizations to establish robust security testing practices, which can contribute to addressing vulnerabilities aligned with the OWASP Top 10 threats. It is essential for organizations to align their security testing efforts with the specific threats and vulnerabilities relevant to their applications and environments.

Here's how the SAMM Operations domain - Security Testing can contribute to addressing some of the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The Security Testing domain emphasizes the establishment of security testing policies, methodologies, and requirements, which can help identify and assess access control vulnerabilities
2	Cryptographic Failures (A02-2021)	The Security Testing domain promotes the use of proper security testing techniques, such as code review and vulnerability scanning, which can help identify cryptographic implementation weaknesses and vulnerabilities.
3	Injection (A03-2021)	Through security testing methodologies and techniques like penetration testing, SAMM can contribute to identifying and addressing injection vulnerabilities by assessing how untrusted data is handled by the application.
4	Insecure Design (A04-2021)	SAMM's focus on security testing planning and methodologies can help identify insecure design decisions, architectural flaws, and inadequate threat considerations that may contribute to insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	SAMM emphasizes the establishment of security testing policies, requirements, and secure configuration practices. This can help identify security misconfigurations during the testing process.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	By incorporating vulnerability scanning and code review as part of security testing, SAMM can contribute to identifying and addressing vulnerabilities in third-party components used in web applications.
7	Identification and Authentication Failures (A07-2021)	Security testing methodologies and techniques recommended by SAMM can help identify weaknesses in identification and authentication mechanisms, enabling organizations to improve their authentication processes.
8	Software and Data Integrity Failures (A08-2021)	Security testing activities, including code review and penetration testing, can help identify vulnerabilities related to software and data integrity, allowing organizations to address potential weaknesses.
9	Security Logging and Monitoring Failures (A09-2021)	SAMM promotes security testing metrics and measurement, including the effectiveness of logging and monitoring capabilities. By assessing these aspects, organizations can identify weaknesses and enhance their logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	The Security Testing domain focuses on comprehensive testing methodologies and techniques, such as penetration testing, which can help identify and address server-side request forgery vulnerabilities.

10.6.9 Operational Enablement

The SAMM (Software Assurance Maturity Model) Operations domain - Operational Enablement focuses on enabling the operational aspects of software assurance within an organization. It encompasses various aspects related to establishing and improving the operational capabilities necessary for effective software assurance practices.

The Operational Enablement domain of SAMM addresses these aspects to enable organizations to establish and enhance their operational capabilities for effective software assurance practices. By incorporating these aspects into their operations, organizations can improve their ability to manage software security risks and ensure the development and deployment of secure software applications.

The Operational Enablement domain covers the following aspects:

#	Name	Description
1	Software Security Training and Awareness	This aspect focuses on providing training and awareness programs to the organization's staff regarding software security best practices, secure coding guidelines, and understanding the importance of software assurance.
2	Software Security Guidance	It involves the development and dissemination of software security guidelines, standards, and policies within the organization. This aspect ensures that employees have access to the necessary resources and guidance to incorporate software security into their daily activities.

#	Name	Description
3	Software Security Assessment and Verification	This aspect involves conducting regular security assessments and verification activities to evaluate the security posture of software applications. It includes techniques such as security code reviews, vulnerability scanning, penetration testing, and security testing to identify potential vulnerabilities and weaknesses.
4	Security Architecture and Design	It focuses on incorporating security considerations into the software architecture and design process. This aspect ensures that security controls and mechanisms are properly defined, implemented, and integrated throughout the software development lifecycle.
5	Secure Software Deployment	This aspect deals with the secure deployment and configuration of software applications. It includes practices such as secure installation, hardening of systems and networks, and secure configuration management to minimize the risk of vulnerabilities during deployment.
6	Incident Response and Recovery	It covers the establishment of incident response plans and procedures to handle security incidents effectively. This aspect ensures that the organization is prepared to respond promptly and efficiently to security breaches or incidents and recover from them.
7	Security Operations Monitoring and Management	This aspect involves implementing monitoring and management capabilities to detect, analyze, and respond to security events and threats. It includes practices such as security event logging, security information and event management (SIEM), and security incident management.
8	Supplier Security Assurance	This aspect focuses on ensuring that software security requirements are considered and addressed when engaging with third-party suppliers or vendors. It includes evaluating the security practices of suppliers, establishing security requirements in contracts, and monitoring supplier performance.
9	Software Assurance Metrics and Reporting	It involves defining and tracking metrics to measure the effectiveness and efficiency of software assurance activities. This aspect enables organizations to assess their software assurance maturity and progress over time. It also includes reporting on software security metrics to stakeholders to provide transparency and accountability.

10.6.9.1 Assessment – OWASP Top 10

The SAMM Operations domain - Operational Enablement indirectly addresses some of the OWASP threats by providing operational capabilities and practices that contribute to the overall security of software applications.

It's important to note that while the Operational Enablement domain of SAMM provides a framework and practices that can help address certain OWASP threats indirectly, it does not cover all specific OWASP threats comprehensively. Organizations should consider additional measures and guidance specific to each OWASP threat to ensure comprehensive software security.

Here's how the Operational Enablement domain of SAMM relates to certain OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The aspect of "Security Architecture and Design" within Operational Enablement focuses on incorporating security considerations into the software architecture and design process. This helps in addressing access control vulnerabilities by ensuring that proper access controls are defined and enforced.
2	Cryptographic Failures (A02-2021)	The aspect of "Security Architecture and Design" also covers the implementation of secure cryptographic practices. By following recommended cryptographic practices, organizations can minimize the risk of cryptographic failures and vulnerabilities.
3	Injection (A03-2021)	The aspect of "Software Security Assessment and Verification" within Operational Enablement includes techniques such as security code reviews and vulnerability scanning. These practices help in identifying and addressing injection vulnerabilities by ensuring proper input validation and handling of untrusted data.
4	Insecure Design (A04-2021)	The aspect of "Security Architecture and Design" focuses on incorporating secure design principles and considerations. By adopting secure design practices, organizations can mitigate the risk of insecure design vulnerabilities and improve the overall security of their applications.
5	Security Misconfiguration (A05-2021)	The aspect of "Secure Software Deployment" within Operational Enablement addresses the secure deployment and configuration of software applications. By following secure configuration practices, organizations can reduce the risk of security misconfigurations and associated vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	The aspect of "Secure Software Deployment" includes practices such as actively managing software dependencies and staying updated with security patches. This helps in addressing vulnerabilities that may arise from using outdated or known vulnerable software components.
7	Identification and Authentication Failures (A07-2021)	The aspect of "Security Architecture and Design" covers the implementation of secure identification and authentication mechanisms. By enforcing strong authentication practices and regularly testing for vulnerabilities, organizations can mitigate the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	The aspect of "Security Architecture and Design" emphasizes the integrity and protection of software components and data. By following secure practices and implementing mechanisms to ensure data integrity, organizations can reduce the risk of integrity-related vulnerabilities.
9	Security Logging and Monitoring Failures (A09-2021)	The aspect of "Security Operations Monitoring and Management" within Operational Enablement focuses on implementing monitoring and management capabilities. This includes practices such as security event logging and real-time monitoring, which contribute to addressing security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	It does not address Server-Side Request Forgery.

11 Security Practices

11.1 Architecture Design Components

The Solution Architecture represents the “Security By Design” approach, the first and the preferred method to secure systems.
Regarding the solution architecture, security assessment and threat modelling are only possible if the architecture and the system are defined in enough detail to postulate threats and assess their potential risks.
Therefore, this document is based on the adopted baselined architecture the organisation has adopted and baselined. The architecture is based on accumulated experiences and knowledge transfer, which has been iterated multiple times until it becomes solid enough to have it defined to be a toolkit that can be used in combinations to conform the backbone of the solution architecture, comprising technologies of preferences, platforms and design patterns in which the security is embedded by design at different levels on the architecture stack.
Since not all the systems are the same, the proposed architecture will be tailored to the requirements. Therefore the hardening of the security elements will also be a factor that will be tuned accordingly.
This section contains a list of designs considered part of the security. These designs belong to different levels of architecture and are embedded into their respective architecture domains.
It means that security is not a feature of the architecture, and it is all embedded in the architecture's design that follows security's best practices.

11.1.1 Desktop Application authentication

Introduction
Any desktop application requires stronger authentication and authorization mechanisms to comply with large enterprises where the security policies are broader, more specific, and standardized.
One specific requirement for desktop applications is that they may not want an explicit authentication besides the one already provided by the User to authenticate to the Windows Operating System and the one created by Microsoft Authentication mechanisms used for Microsoft Office 365.
The application may not require the end-user to authenticate but must have authentication and authorization control mechanisms.
So, the overall design objective is that any desktop application must enable gathering the session tokens already created by the User at the time of authentication, validate them if required, and use them as their own without prompting the User for permission or extra credentials. This method is called "silent authentication".
These two technologies can provide the foundation for "silent authentication and authorization". These are:
• Integrated Windows Authentication (IWA)
• Microsoft Authentication Library (MSAL) with Web Account Manager (WAM)

Integrated Windows Authentication (IWA) and Microsoft Authentication Library (MSAL) with Web Account Manager (WAM) are different components in the Microsoft ecosystem that serve distinct purposes in the realm of authentication and access management.

In any case, the application backend must verify the credentials against the identity provider. These id in enterprise environments are one of the following:

#	Name	Description
1	Active Directory (AD)	Microsoft's directory service is used for User and identity management in Windows environments.
2	Azure Active Directory (Azure AD)	Microsoft's cloud-based identity and access management service can be used for authentication and authorization.
3	ADFS (Active Directory Federation Services)	A Microsoft technology that enables single sign-on and identity federation between different systems and applications.

Integrated Windows Authentication (IWA)

IWA is a Microsoft Windows operating system mechanism that enables single sign-on (SSO) for users accessing network resources within a Windows domain.

IWA is a built-in Windows authentication mechanism specific to Windows environments.

It relies on the Kerberos authentication protocol and allows users to authenticate automatically without re-entering their credentials.

IWA is specific to Windows environments and primarily focuses on providing seamless authentication for Windows domain users.

It simplifies the authentication process within a Windows domain, enhancing security through Kerberos encryption.

With IWA, the browser would obtain the User's Windows login information from the operating system and pass it to the web server, allowing for seamless authentication without the need for the User to enter their credentials again

IWA was explicitly created for IE, but it can be enabled for modern browsers like Chrome and Firefox.

IWA technology enables a web browser to authenticate a user with a Windows domain controller, allowing the web application to obtain the User's Windows session credentials.

IWA uses the Negotiate protocol to authenticate users with the security provider configured on the server, Kerberos or NTLM. Once the User is authenticated, their Windows session is passed to the web application, allowing it to determine the User's identity and provide appropriate access to resources.

Microsoft Authentication Library (MSAL) with Web Account Manager (WAM)

MSAL is a developer library provided by Microsoft that simplifies integrating authentication and authorization capabilities into applications.

MSAL supports multiple platforms and programming languages, allowing developers to incorporate authentication functionality across various applications and devices.
WAM is a component used by MSAL on Windows platforms to provide authentication and access management capabilities.
WAM integrates with the underlying Windows operating system, and leverages features such as the Credential Manager, Windows Hello, and other Windows security mechanisms.
By utilizing WAM, MSAL can provide features such as SSO, credential caching, and integration with Windows authentication protocols.
MSAL with Web Account Manager (WAM) component provides integration with Windows authentication and access management features and expands the authentication capabilities of MSAL by utilizing the underlying Windows operating system's security mechanisms.
Desktop or mobile application that runs on Windows and a machine connected to a Windows domain - AD or AAD joined - can use Integrated Windows Authentication (IWA) to acquire a token silently. No UI is required when using the application.
WAM can log in to the current Windows user silently.

Process

In any case, the Silent authorization process typically involves the following steps:

#	Name	Description
1	Retrieve the Token	Obtain the IWA or WAM session token from the client application. This token is typically provided as part of the HTTP request headers, such as the "Authorization" header
2	Identify the Identity Provider	Determine the identity provider responsible for validating the token. In the case of IWA, the identity provider is typically the Active Directory domain controller or a similar Windows authentication server. For WAM, the identity provider can vary depending on the system or technology used.
3	Token Validation	Perform the token validation against the identified identity provider. The exact steps may depend on the specific technology and protocols involved. Still, the general process involves verifying the token's integrity, checking its expiration, and ensuring that a trusted authority issues it.
4	Token Decoding	Decode the token to extract relevant information such as user identity, claims, or any additional data embedded within the token. The decoding process may involve using cryptographic algorithms and keys to verify the token's signature and integrity.
5	Verification with Identity Provider	Validate the token with the identity provider by requesting an endpoint or service provided by the identity provider. This typically involves passing the token to the identity provider's validation service and receiving a response indicating the validity and authenticity of the token.
6	Handle Validation Result	Based on the validation result received from the identity provider, you can make decisions within your application on how to handle the authenticated User or whether to grant access to requested resources.

11.1.1.1 Assessment – SAMM

The integration of IWA and MSAL with WAM in the desktop application authentication architecture helps address SAMM's security standards and domains by providing seamless authentication, secure access management, and alignment with security policies, requirements, and best practices.
The architecture design components mentioned, Integrated Windows Authentication (IWA) and Microsoft Authentication Library (MSAL) with Web Account Manager (WAM), help address the SAMM security standards and its domains in the following ways:

#	Name	Description
7	SAMM Security by Governance Policies and Standards	The integration of IWA and MSAL with WAM allows organizations to establish policies and standards for authentication and access management. These components provide mechanisms for seamless authentication and authorization, aligning with the organization's security policies.
8	SAMM Security by Governance Secure Development Training	IWA and MSAL with WAM can be utilized as part of secure development training programs. Developers can learn to integrate these components into desktop applications, ensuring secure authentication and access management practices.
9	SAMM Security by Governance Secure Architecture	The use of IWA and MSAL with WAM aligns with secure architecture practices, contributing to secure software implementation. It helps mitigate vulnerabilities associated with OWASP threats such as Insecure Design, Cryptographic Failures, and Security Misconfiguration.
10	SAMM Security by Governance Security Coding Guidelines	IWA and MSAL with WAM can be integrated into secure coding guidelines, providing developers with recommendations for implementing authentication and authorization controls effectively.
11	SAMM Security by Design Security Requirements	IWA and MSAL with WAM enable the integration of security requirements into the design phase. They provide authentication and authorization capabilities that align with security requirements, addressing threats such as Broken Access Control and Insecure Design.
12	SAMM Security by Design Secure Architecture	IWA and MSAL with WAM contribute to a secure architecture by providing authentication mechanisms and leveraging Windows security features. They help address threats such as Insecure Design and Security Misconfiguration by incorporating robust security controls.
13	SAMM Security by Implementation Secure Development Training	IWA and MSAL with WAM can be utilized as part of secure development training programs. Developers can learn to integrate these components into desktop applications, ensuring secure authentication and access management practices.
14	SAMM Security by Implementation Secure Architecture	The use of IWA and MSAL with WAM aligns with secure architecture practices, contributing to secure software implementation. It helps mitigate vulnerabilities associated with OWASP threats such as Insecure Design, Cryptographic Failures, and Security Misconfiguration.

#	Name	Description
15	SAMM Security by Implementation Security Coding Guidelines	IWA and MSAL with WAM can be integrated into secure coding guidelines, providing developers with recommendations for implementing authentication and authorization controls effectively.
16	SAMM Security by Verification Security Testing	IWA and MSAL with WAM can be included in security testing activities to assess the effectiveness of authentication and authorization mechanisms. They help identify vulnerabilities and weaknesses related to authentication and access management.
17	SAMM Security by Verification Code Review	By incorporating IWA and MSAL with WAM, code reviews can focus on the integration and implementation of these components, ensuring that secure coding practices are followed for authentication and authorization.
18	SAMM Security by Verification Security Architecture Review	IWA and MSAL with WAM contribute to the security architecture review process by providing authentication and access management capabilities. The review can assess the integration and effectiveness of these components.
19	SAMM Security by Operations	Environment Hardening: IWA and MSAL with WAM can be considered as part of environment hardening practices by providing secure authentication mechanisms. They contribute to securing the software environment by minimizing vulnerabilities in authentication and access management.
20	SAMM Security by Operations Configuration Management	IWA and MSAL with WAM can be incorporated into configuration management practices, ensuring that the authentication and authorization configurations are properly documented and controlled.
21	SAMM Security by Operations Vulnerability Management	By integrating IWA and MSAL with WAM, organizations can address vulnerabilities related to authentication and access management, enhancing vulnerability management processes.
22	SAMM Security by Operations Incident Management	IWA and MSAL with WAM assist in incident management by providing secure authentication and access management capabilities. They contribute to incident detection and response, minimizing the impact of security incidents.
23	SAMM Security by Operations Secure Deployment	IWA and MSAL with WAM can be utilized in secure deployment practices, ensuring that authentication and access management mechanisms are securely deployed and configured

11.1.1.2 Assessment – OWASP Top 10

The mentioned architecture design principle of "Desktop Application authentication" does not directly address the OWASP threats. However, it can contribute to addressing some of the threats indirectly.
It's important to note that while the principle indirectly contributes to addressing these threats, it may not fully eliminate the risks associated with them. Implementing additional security measures and following secure coding practices are essential to comprehensively address the OWASP threats.
Here are the contributions of this principle to specific OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The principle emphasizes the need for stronger authentication and authorization mechanisms in desktop applications. By implementing proper authentication and authorization controls, it helps mitigate the risk of unauthorized access and privilege escalation, which are key aspects of Broken Access Control vulnerabilities.
2	Identification and Authentication Failures (A07-2021)	The principle focuses on enabling silent authentication by leveraging technologies like Integrated Windows Authentication (IWA) and Microsoft Authentication Library (MSAL) with Web Account Manager (WAM). By implementing these mechanisms, the desktop application can enhance the identification and authentication process, reducing the risk of bypassing or manipulating the authentication process.

11.1.2 Service Mesh – Data Plane

Kubernetes will have per pod a runtime Proxy using Envoy. This is referred as the “Data Plane”. This is the preferred runtime proxy for all the Pods, deployed as a sidecar. This is part of the "service mesh" which manages all the network traffic inside a Kubernetes cluster. The service mesh is responsible for controlling East to West traffic in the cluster.

These services should be registered via the Istio Ingress controller in the specific environment. The Istio Controller is part of the core Kubernetes infrastructure to manage traffic routing inside a cluster.

The Service Mesh - Data Plane capability, by leveraging Envoy as the runtime proxy for pods and integrating with Istio, enhances the security, control, resilience, observability, and operational efficiency of the Kubernetes cluster. It establishes a robust foundation for managing East to West traffic within the cluster, facilitating the development and deployment of secure and scalable applications.

The Service Mesh - Data Plane capability, which involves using Envoy as the runtime proxy for each pod in a Kubernetes cluster, contributes to enhancing the security and control of network traffic within the cluster.

#	Name	Description
1	Enhanced Traffic Control	By deploying Envoy as a sidecar proxy alongside each pod, the Service Mesh - Data Plane capability provides fine-grained control over network traffic within the Kubernetes cluster. Envoy acts as a data plane proxy, intercepting and managing all incoming and outgoing requests, enabling advanced traffic routing, load balancing, and observability capabilities.
2	Improved Security	With Envoy as the runtime proxy, the Service Mesh - Data Plane capability strengthens the security of the Kubernetes cluster. Envoy's features include secure communication through transport layer security (TLS), mutual TLS (mTLS), and the ability to enforce policies and access controls at the network level. This helps prevent unauthorized access, protect sensitive data, and mitigate potential vulnerabilities.

#	Name	Description
3	Fault Tolerance and Resilience	Envoy, as part of the service mesh, enhances fault tolerance and resilience within the Kubernetes cluster. It includes features such as circuit breaking, retries, timeouts, and health checks, ensuring that the services within the cluster can handle failures and recover gracefully. Envoy's load balancing capabilities also help distribute traffic efficiently, improving performance and overall system availability.
4	Observability and Telemetry	The Service Mesh - Data Plane capability enables comprehensive observability and telemetry features through Envoy. Envoy collects rich metrics, traces, and logs, providing deep insights into the traffic patterns, service behavior, and performance characteristics of the applications running in the cluster. This data can be utilized for troubleshooting, performance optimization, and detecting anomalies or security incidents.
5	Seamless Integration with Istio	The Service Mesh - Data Plane capability aligns with the broader Istio service mesh framework. Istio leverages the Envoy proxies deployed as sidecars to control and manage the traffic within the cluster. The Istio Ingress controller acts as the entry point for external traffic into the cluster, providing a centralized and consistent approach for routing and load balancing. It enables powerful traffic management features like traffic splitting, canary deployments, and fault injection.
6	Simplified Operations and Maintenance	Adopting the Service Mesh - Data Plane capability brings operational benefits by abstracting away the complexity of managing network traffic within the Kubernetes cluster. The runtime proxy deployment pattern using Envoy simplifies the configuration, control, and monitoring of services, reducing the burden on developers and providing a standardized approach to network communication.

11.1.2.1 Assessment – SAMM

The Service Mesh architecture components, when properly configured and integrated into the software development lifecycle, can help address various aspects of SAMM security standards. They provide a robust framework for secure communication, traffic management, and access control, supporting governance, design, implementation, verification, and operations-related activities.
The architecture design components of the Service Mesh, specifically the Data Plane and Control Plane, can help address the SAMM security standards and its domains in the following ways:

#	Name	Description
1	SAMM Security by Design	Security Requirements: The Service Mesh can help enforce security requirements by providing a secure communication channel between microservices. It ensures that microservices communicate through the Istio internal gateway, which can be configured to enforce security measures such as authentication, authorization, and encryption.
2	SAMM Security by Implementation	Secure Architecture The Service Mesh, with its Data Plane and Control Plane components, provides a secure architecture for microservices. It helps address OWASP threats such as Insecure Design, Cryptographic Failures, and Security Misconfiguration by implementing secure communication channels, traffic management, and access control policies. Security Testing Integration The Service Mesh can integrate with security testing activities during the development phase. For example, it can be configured to collect telemetry data and metrics for monitoring and detecting anomalies or potential security vulnerabilities. This helps organizations identify and address security issues early in the development process.
3	SAMM Security by Verification	Security Testing The Service Mesh can facilitate security testing activities by providing visibility into the network traffic between microservices. It can capture and analyze the traffic to detect security vulnerabilities or anomalies, supporting activities such as penetration testing, vulnerability scanning, and code review.
4	SAMM Security by Operations	Environment Hardening The Service Mesh can contribute to environment hardening by implementing secure configurations and infrastructure setups. It helps protect the software system from known vulnerabilities and security weaknesses by enforcing secure communication channels and access control policies. Configuration Management The Service Mesh can assist in managing and documenting the configuration of microservices, ensuring that unauthorized changes or misconfigurations are detected and remediated. It provides centralized control over the configuration of microservices within the mesh.

11.1.2.2 Assessment - OWASP Top 10

While the mentioned architecture design principle of using a service mesh does not directly address all OWASP threats, it contributes to mitigating several of them by providing secure communication channels, access controls, encryption, authentication, and centralized control over network traffic.

The mentioned architecture design principle of using a Service Mesh (specifically, Envoy and Istio) addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The service mesh helps in enforcing access controls within a Kubernetes cluster by managing network traffic and ensuring that microservices communicate through the designated internal gateway. This helps in preventing unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	The service mesh can facilitate secure communication between microservices by providing encryption, authentication, and authorization mechanisms. Envoy and Istio support mutual TLS (mTLS) authentication, ensuring that only authorized microservices can communicate with each other.
3	Injection (A03-2021)	The service mesh can provide security measures to prevent injection attacks by implementing input validation, parameterization, and secure coding practices within the microservices. It can help in preventing untrusted data from being improperly handled and executed within the application.
4	Insecure Design (A04-2021)	While the service mesh does not directly address insecure design vulnerabilities, it provides an infrastructure layer that can help enforce security controls and policies. By managing network traffic and implementing secure communication channels, the service mesh contributes to the overall security of the architecture.
5	Security Misconfiguration (A05-2021)	The service mesh, when properly configured and integrated, can help in reducing security misconfigurations by providing standardized and centralized control over network traffic, authentication, and authorization. It ensures that microservices adhere to the defined security policies and configurations.
6	Vulnerable and Outdated Components (A06-2021)	The service mesh itself, particularly Envoy and Istio, is actively maintained and regularly updated to address security vulnerabilities. By utilizing the latest versions of Envoy and Istio, organizations can mitigate the risk associated with vulnerable or outdated components within the service mesh infrastructure.
7	Identification and Authentication Failures (A07-2021)	The service mesh can assist in implementing secure identification and authentication mechanisms by providing features like mutual TLS authentication, role-based access control, and integration with identity providers. This helps prevent unauthorized access and impersonation attacks.
8	Software and Data Integrity Failures (A08-2021)	The service mesh, through its encryption and authentication mechanisms, contributes to the integrity and protection of software components and data. It ensures that communication between microservices is secure, reducing the risk of unauthorized modifications, manipulation, or destruction of data.
9	Security Logging and Monitoring Failures (A09-2021)	While the service mesh itself does not directly address logging and monitoring, it provides visibility and observability features that can be leveraged for effective logging and monitoring practices. Envoy and Istio offer metrics, tracing, and logging capabilities that organizations can utilize to detect and respond to security incidents.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	The service mesh can help mitigate Server-Side Request Forgery (SSRF) vulnerabilities by enforcing communication through designated gateways. By controlling and monitoring the network traffic between microservices, the service mesh can prevent unintended requests to internal or external resources, reducing the risk of SSRF attacks.

11.1.3 Service Mesh – Control Plane

Kubernetes will have per pod an internal gateway configured using Istio. This is referred as the “Control Plane. This is part of the "service mesh" which manages all the network traffic inside a Kubernetes cluster. In the context of this architecture, it addresses the communication "microservice to microservice" (also known as "point to point" communication).

Any communication between microservices should use the Istio internal gateway, which will be part of the Private DNS Zone. All microservices with endpoints invoked by others must be registered in the Internal Ingress Gateway, which is the East to West traffic.

Suppose microservice endpoints are invoked by external clients (Single Page Application) and internal microservices. In that case, it must be registered to both Kong API Gateway (Ingress Controller) and the Internal Gateway managed by Istio Gateway. It is known as the North-to-South traffic, and it is partially controlled by Istio, which allows the API Gateway to send traffic to microservices.

Utilizing the Service Mesh - Control Plane capability, organizations can establish a robust and scalable architecture for managing microservice communication within their Kubernetes cluster. It enables efficient and secure data exchange, allows for flexible traffic routing, and provides centralized control and management capabilities to enhance the overall reliability and performance of the cluster.

The Service Mesh - Control Plane capability, which involves configuring an internal gateway using Istio for each pod in a Kubernetes cluster, contributes to managing and controlling the network traffic within the cluster. Here's an extended explanation:

#	Name	Description
1	Efficient Microservice-to-Microservice Communication	The Service Mesh - Control Plane capability focuses on enabling efficient communication between microservices within the Kubernetes cluster. By configuring an internal gateway using Istio, the control plane of the service mesh, it establishes a centralized and standardized approach for managing "point-to-point" or "microservice-to-microservice" communication.
2	Streamlined Traffic Routing	The internal gateway provided by Istio's control plane acts as a traffic routing and management component. It allows for intelligent routing decisions, such as load balancing, traffic splitting, and fault tolerance, which are essential for optimizing the communication flow between microservices. This capability helps ensure reliable and efficient data exchange within the cluster.

#	Name	Description
3	Integration with Private DNS Zone	The microservices' endpoints that are invoked by other services within the Kubernetes cluster should be registered within the Internal Ingress Gateway, which is part of the Private DNS Zone. This integration facilitates seamless and secure communication by leveraging a private domain name system (DNS) and ensuring that microservices can be reached using their registered endpoints.
4	North-to-South Traffic Control	In scenarios where microservice endpoints are invoked by both external clients (such as Single Page Applications) and internal microservices, the Service Mesh - Control Plane capability addresses the North-to-South traffic flow. To handle this traffic, microservices need to be registered in both the Kong API Gateway (acting as the Ingress Controller) and the Internal Gateway managed by Istio. This setup enables Istio to partially control the North-to-South traffic, allowing the API Gateway to forward requests to the appropriate microservices.
5	Enhanced Security and Policy Enforcement	By leveraging the Istio control plane, the Service Mesh - Control Plane capability provides advanced security features and policy enforcement mechanisms. Istio supports secure communication between microservices through features like mutual TLS (mTLS), ensuring that communication is encrypted and authenticated. Additionally, Istio allows for the enforcement of access control policies, rate limiting, and request authentication, enhancing the overall security posture of the cluster.
6	Operational Consistency and Management	Adopting the Service Mesh - Control Plane capability brings operational consistency and ease of management to the Kubernetes cluster. The Istio control plane provides a centralized control point for configuring and monitoring the traffic flow, enabling administrators to set and enforce policies, track performance metrics, and troubleshoot any issues related to microservice communication.

11.1.3.1 Assessment – SAMM

See above, Data Plane Assessment section where both Data Plane and Control plane are assesses as a single design component under the “service mesh” concept.

11.1.3.2 Assessment – OWASP Top 10

See above, Data Plane Assessment section where both Data Plane and Control plane are assesses as a single design component under the “service mesh” concept.

11.1.4 Network Segmentation (Microsegmentation)

Network segmentation, specifically microsegmentation, is a security technique that involves dividing a network into smaller, isolated segments to enhance security and control network traffic flow.

Microsegmentation takes network segmentation to a granular level by creating individual security zones for each workload, application, or even down to the level of individual network devices.
The primary goal of microsegmentation is to limit lateral movement within the network. Lateral movement refers to the ability of an attacker to move horizontally across the network, from one compromised system to another, in an attempt to escalate privileges, access sensitive information, or propagate malware. By implementing microsegmentation, each segment acts as its own security zone, effectively isolating workloads or applications from each other.
Implementing microsegmentation requires careful planning and consideration of network architecture. It involves defining logical boundaries, creating access control policies, and configuring security controls at the segment level. Technologies such as virtual firewalls, software-defined networking (SDN), and network overlays are commonly used to implement microsegmentation in modern networks.
Microsegmentation is an effective security practice that provides improved network visibility, access control, and containment in the event of a security breach. By dividing the network into smaller, isolated segments, organizations can reduce the attack surface and limit the lateral movement of attackers, thus strengthening the overall security posture.
Microsegmentation is also an effective security practice that strengthens the network's security posture by implementing granular security zones and access control policies. It reduces the attack surface, limits lateral movement, enhances network visibility, and facilitates incident containment and response. By adopting microsegmentation, organizations can achieve a higher level of network security and better protect their critical assets and data from unauthorized access or compromise.
The Network Segmentation (Microsegmentation) capability focuses on enhancing security and controlling network traffic flow by dividing a network into smaller, isolated segments. Here's an extended explanation:

#	Name	Description
1	Granular Security Zones	Microsegmentation takes network segmentation to a more granular level, creating individual security zones for each workload, application, or even individual network devices. This approach allows organizations to establish fine-grained security boundaries and isolate different components of the network, thereby reducing the potential impact of a security breach or unauthorized access.
2	Limiting Lateral Movement	One of the primary objectives of microsegmentation is to limit lateral movement within the network. Lateral movement refers to an attacker's ability to move horizontally across the network, potentially compromising multiple systems or escalating privileges. By implementing microsegmentation, each segment acts as its own security zone, effectively isolating workloads or applications from each other. This isolation makes it significantly more challenging for an attacker to traverse the network and gain unauthorized access to critical resources.
3	Access Control Policies	: Implementing microsegmentation involves defining logical boundaries and creating access control policies at the segment level. These policies specify which network traffic is allowed or denied between segments, enabling organizations to enforce a least privilege principle. By carefully configuring access control, organizations can ensure that only authorized communication occurs within each segment, reducing the attack surface and minimizing the risk of unauthorized access or data exfiltration.

#	Name	Description
4	Security Controls and Technologies	To implement microsegmentation effectively, organizations leverage various security controls and technologies. Virtual firewalls, software-defined networking (SDN), and network overlays are commonly used to enforce access control policies and create isolated segments. These technologies enable organizations to define and enforce security rules at the network level, providing a scalable and dynamic approach to network segmentation.
5	Improved Network Visibility	Microsegmentation enhances network visibility by providing insights into the communication patterns and dependencies between different segments. By monitoring the traffic flow within and between segments, organizations can gain a comprehensive understanding of their network's behavior, detect anomalies, and respond promptly to potential security incidents. Enhanced visibility also facilitates effective network troubleshooting and assists in ensuring compliance with regulatory requirements.
6	Containment and Incident Response	In the event of a security breach or compromise within a segment, microsegmentation helps contain the impact and limit the lateral movement of the attacker. By isolating each segment, organizations can minimize the spread of malware or unauthorized access, confining the impact to a specific segment. This containment reduces the overall risk and provides more time for incident response teams to investigate, mitigate, and remediate the issue before it affects other parts of the network.

11.1.4.1 Assessment – SAMM

.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Network segmentation can contribute to defining and implementing access control policies and standards at the segment level, ensuring that each segment has its own security zone. Risk Management: Network segmentation can help in identifying and assessing risks associated with different segments of the network, enabling organizations to implement risk management processes effectively. Compliance Management: Network segmentation can assist in ensuring compliance with regulatory requirements and industry standards by enforcing security controls at the segment level. Security Training and Awareness: Network segmentation can contribute to creating a secure culture by isolating workloads or applications, emphasizing the importance of security training and awareness within each segment. Security Metrics and Reporting: Network segmentation can improve network visibility and reporting by providing insights into the security posture of each segment, enabling effective decision-making at various levels. Security Roles and Responsibilities: Network segmentation can help in clearly defining and assigning security roles and responsibilities within each segment, ensuring accountability for software security.
2	SAMM Security by Design	Security Requirements: Network segmentation can be considered as a security design principle to address security requirements related to secure boundaries and isolation between network segments. Secure Architecture: Network segmentation contributes to designing a secure architecture by creating isolated security zones for each segment, minimizing the impact of potential OWASP threats and providing a strong foundation for security controls. Secure Coding Practices: While network segmentation does not directly address secure coding practices, it can complement them by isolating different components and workloads, reducing the attack surface and the potential impact of vulnerabilities. Threat Modeling: Network segmentation can be considered during threat modeling exercises to identify potential lateral movement paths and design appropriate security controls at segment boundaries. Security Testing: Network segmentation can help in creating separate testing environments and isolating test assets, enabling focused security testing of individual segments and mitigating OWASP threats.
3	SAMM Security by Implementation	Does Not Apply Directly to network segmentation.
4	SAMM Security by Verification	Does Not Apply Directly to network segmentation.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening: Network segmentation can contribute to the overall environment hardening strategy by isolating different components or workloads, making it more difficult for attackers to move laterally within the network and compromising sensitive systems. Secure Build: Network segmentation does not directly impact secure build practices but can provide an additional layer of defense by isolating different components or workloads, reducing the potential impact of vulnerabilities introduced during the build process. Configuration Management: Network segmentation can help in managing and documenting the configuration of individual segments, making it easier to maintain and detect unauthorized changes or misconfigurations within specific areas of the network. Vulnerability Management: Network segmentation can assist in vulnerability management efforts by isolating vulnerable systems or applications, reducing the potential risk of exploitation and enabling more targeted vulnerability scanning and patching. Incident Management: Network segmentation can help contain and mitigate the impact of security incidents by limiting lateral movement within the network, preventing attackers from easily accessing and compromising other systems or applications. Secure Deployment: Network segmentation does not directly affect secure deployment practices but can enhance the overall security of the deployment process by isolating different components or workloads, reducing the risk of unauthorized access or compromise. Security Testing: Network segmentation can facilitate security testing activities by separating different components or workloads, allowing for more targeted testing efforts and reducing the scope of potential vulnerabilities. Operational Enablement: Network segmentation can support operational security by providing a controlled approved environment where the operations team can perform rutine work securely.

11.1.4.2 Assessment – OWASP Top 10

See above, Data Plane Assessment section where both Data Plane and Control plane are assesses as a single design component under the “service mesh” concept.

11.1.5 Network Policies Control and Enforcement

The information flow in Kubernetes security with Calico and OPA involves Calico enforcing network policies to control pod communication, while OPA evaluates and enforces fine-grained authorization and access control decisions based on policies defined as code. This integrated approach ensures that pods meet

security and compliance requirements, mitigates risks, and provides visibility into policy violations, all without compromising the performance and availability of the Kubernetes cluster.

The "Network Policies Control and Enforcement" capability contributes to various SAMM security domains and sub-domains by providing enhanced access control and communication security between pods in a Kubernetes environment. This helps in mitigating OWASP threats related to Injection, Broken Access Control, Security Misconfiguration, and Insecure Design.

#	Name	Description
1	Pod Creation and Admission Control	When a pod is created in Kubernetes, Calico comes into play as the network plugin and enforces network policies. Calico examines the pod's metadata and applies the appropriate network policies based on labels, selectors, and security rules defined in Kubernetes. At this stage, OPA can be integrated into the admission control process. OPA evaluates admission policies to ensure that pods meet security and compliance requirements before they are admitted into the cluster. OPA can check for conditions like image vulnerabilities, resource limits, or labels that define security classifications.
2	Pod-to-Pod Communication	Once pods are running, Calico enforces network policies that control communication between pods. Calico inspects network traffic and applies rules based on the defined policies, including allowing or denying traffic based on IP addresses, ports, protocols, and labels. OPA can be integrated with Calico's network policies to provide additional fine-grained authorization and access control decisions. OPA evaluates policies that define who can communicate with which pods or services based on contextual information such as user identity, pod labels, or service account roles.
3	Compliance and Governance	OPA plays a crucial role in enforcing compliance and governance policies within the Kubernetes environment. OPA evaluates policies defined as code (Rego) against Kubernetes resources, including deployments, services, or pods. OPA checks for compliance with security best practices, regulatory requirements, or internal governance policies, ensuring that resources adhere to the defined policies.
4	Dynamic Policy Evaluation and Updates	OPA's dynamic policy evaluation capabilities allow policies to be updated and enforced in real-time. As policies are updated, OPA evaluates the new policies against Kubernetes resources and enforces the changes without requiring a cluster restart. This dynamic evaluation enables organizations to quickly respond to security changes, adapt to new threats, or address compliance needs without disrupting the cluster's operation.

#	Name	Description
5	Auditing and Visibility	Both Calico and OPA provide auditing and visibility capabilities to enhance Kubernetes security. Calico logs network traffic and activity, enabling security teams to monitor and analyze communication patterns for potential anomalies or threats. OPA provides detailed audit logs of policy evaluations, giving visibility into which policies were triggered, the decisions made, and any violations or non-compliant actions detected.

11.1.5.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Contribution: The capability helps enforce network policies in Kubernetes, which align with the organization's policies and standards for secure communication and access control between pods.
2	SAMM Security by Design	Security Requirements: The capability ensures that security requirements related to network policies and access control are defined and incorporated into the design phase, mitigating OWASP threats like Broken Access Control and Security Misconfiguration. Secure Architecture: By implementing network policies, the capability establishes a secure architecture that controls pod communication and reduces the risk of Insecure Design and Security Misconfiguration. Secure Coding Practices: While this capability focuses on network policies rather than coding practices, it indirectly mitigates OWASP threats by enforcing secure communication and access controls between pods.
3	SAMM Security by Implementation	Secure Development Training: Contribution: This capability enhances the knowledge and skills of developers regarding Kubernetes network policies, which contributes to secure implementation and mitigates OWASP threats. Secure Architecture: By utilizing network policies, the capability reinforces the importance of secure architecture practices, reducing the likelihood of Insecure Design and Security Misconfiguration. Secure Coding Guidelines: Although not directly related to coding, this capability indirectly helps in securing the implementation of network policies, which can mitigate OWASP threats.

#	Name	Description
4	SAMM Security by Verification	Security Testing: The capability enhances security testing by validating and verifying the implementation of network policies, ensuring effective access control and mitigating OWASP threats related to Injection and Security Misconfiguration. Code Review: The capability ensures that the implementation of network policies is reviewed, reducing the risk of OWASP threats that might arise from insecure access control.
5	SAMM Security by Operations	Environment Hardening: The capability supports the hardening of the Kubernetes environment through network policies, improving security and access controls between pods. Configuration Management: The capability enforces network policies as part of the configuration management process, enhancing access control and security within the software environment.

11.1.5.2 Assessment – OWASP Top 10

While the capability of network policies control and enforcement, along with the use of Calico and OPA, directly addresses some of the OWASP threats (such as Broken Access Control, Injection, Insecure Design, Security Misconfiguration, Identification and Authentication Failures), it may indirectly contribute to addressing other threats by providing enhanced visibility, auditing, and access control mechanisms. It is important to note that addressing the complete OWASP Top 10 threats requires a comprehensive security approach that includes multiple security practices and controls beyond the network policies control and enforcement capability.

The capability of network policies control and enforcement, as well as the use of Calico and OPA, address the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The network policies enforced by Calico and the fine-grained authorization and access control decisions evaluated by OPA contribute to preventing unauthorized access and enforcing proper access controls within the Kubernetes cluster. By evaluating admission policies during pod creation, OPA can ensure that only pods meeting security and compliance requirements are admitted, mitigating the risk of access control flaws.
2	Cryptographic Failures (A02-2021)	While the capability of network policies control and enforcement does not directly address cryptographic failures, Calico can provide additional security by encrypting the data in transit between pods using Transport Layer Security (TLS). This helps protect the confidentiality and integrity of data.

#	Name	Description
3	Injection (A03-2021)	The network policies enforced by Calico can help mitigate injection attacks by controlling pod-to-pod communication and preventing unauthorized access between pods. Additionally, OPA can contribute to preventing injection vulnerabilities by evaluating policies during the admission control process and enforcing conditions such as image vulnerabilities and secure configurations.
4	Insecure Design (A04-2021)	Calico's network policies and OPA's fine-grained access control decisions can assist in enforcing secure design principles within the Kubernetes environment. By evaluating policies during pod creation and pod-to-pod communication, potential insecure design flaws can be identified and prevented, reducing the risk of exploitation.
5	Security Misconfiguration (A05-2021)	Calico's network policies and OPA's policy evaluation capabilities can contribute to addressing security misconfigurations within the Kubernetes environment. By enforcing proper network policies and evaluating policies against Kubernetes resources, organizations can ensure that their deployments, services, and pods adhere to secure configurations and industry standards.
6	Vulnerable and Outdated Components (A06-2021)	While the capability of network policies control and enforcement does not directly address vulnerable and outdated components, organizations can use Calico and OPA in combination with other security practices to mitigate this threat. By implementing secure configurations, regularly updating software components, and conducting security assessments, organizations can reduce the risk of using vulnerable or outdated components within their Kubernetes environment.
7	Identification and Authentication Failures (A07-2021)	The integration of OPA into the admission control process allows for evaluating and enforcing authentication and authorization policies during pod creation. By leveraging OPA's capabilities, organizations can ensure that proper identification and authentication mechanisms are in place, reducing the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	The use of Calico and OPA does not directly address software and data integrity failures. However, organizations can leverage auditing and visibility capabilities provided by Calico and OPA to monitor and detect potential integrity issues or unauthorized modifications, enhancing overall security.
9	Security Logging and Monitoring Failures (A09-2021)	Calico and OPA provide auditing and visibility capabilities that can contribute to addressing security logging and monitoring failures. By logging network traffic and activities, as well as auditing policy evaluations, organizations gain visibility into potential security incidents, enabling timely detection and response.
10	Server-Side Request Forgery (A10-2021)	The use of Calico and OPA does not directly address server-side request forgery (SSRF) vulnerabilities. However, organizations can implement additional security measures, such as input validation and secure configurations, to mitigate the risk of SSRF attacks within their web applications running on Kubernetes.

11.1.6 Perimeter network security

The Perimeter Network Security capability ensures that the architecture establishes strong perimeter defenses, restricts inbound and outbound traffic, and enforces access control policies. By implementing these measures, the architecture enhances security, reduces the attack surface, and mitigates the risk of unauthorized access or malicious activities within the clusters.

#	Name	Description
1	Frontend and Backend Networks	The architecture incorporates two Kubernetes clusters to segregate traffic and enhance overall security. The clusters are divided into the Frontend Cluster, which houses internet-facing components responsible for authentication and read-only operations, and the Backend Cluster, which contains applications, services, and data.
2	Access Control	To ensure proper access control, the Backend Cluster is designed to be accessible only by traffic redirected from the Frontend Cluster. This restriction prevents direct access to the backend applications and establishes a controlled path for inbound connections.
3	Inbound Connections - Frontend Cluster	All inbound connections for applications in the Frontend Cluster are routed through a nominated Ingress Controller, which in this case is an NGINX instance configured as a reverse proxy. This setup ensures that all incoming connections to applications in the Frontend Cluster pass through a centralized point of entry, allowing for unified security policies and controls.
4	Inbound Connections - Backend Cluster (North-to-South Traffic)	For the Backend Cluster, inbound connections (restful APIs) are directed through a nominated Ingress Controller, specifically a Kong API Gateway. This gateway serves as a traffic filter, allowing only restful API requests to reach the backend applications. By leveraging an API Gateway, organizations can enforce access controls, perform authentication and authorization checks, and protect the backend services from unauthorized access.
5	Point-to-Point Connections - Backend Cluster (East-to-West Traffic)	Point-to-point connections (restful APIs) between applications within the Backend Cluster are facilitated by an internal gateway known as the Istio Gateway. This gateway enables secure communication between services within the cluster, ensuring that all traffic between microservices is controlled and monitored. By leveraging Istio's features, organizations can apply service-level policies, perform traffic encryption, and enforce authentication and authorization rules for East-to-West traffic.
6	Outbound Connections	To maintain a secure architecture, all outbound connections from the clusters are firewall protected by default. This means that the architecture implements restrictive policies for outgoing traffic, allowing only specific destinations and protocols based on predefined rules. This approach helps prevent unauthorized communication from within the clusters and adds an additional layer of defense against potential threats.
7	Deployment, Administration, and Operational Traffic	The architecture enforces strict controls on inbound connections related to operational traffic, such as DevOps pipelines and system-to-system communication. Role-Based Access Control (RBAC) is utilized with service accounts to restrict traffic by default. Only authorized service accounts with the necessary permissions are allowed to initiate inbound connections for operational purposes. This control ensures that operational traffic is closely monitored and limited to authorized entities, reducing the risk of unauthorized access or malicious activities.

#	Name	Description
8	Individual and Service Account Connections	Inbound connections initiated by individuals and service accounts operating in different environments are subject to strict controls and restrictions. RBAC policies are employed to govern these connections. For example, DevOps operators or system operators using SSH protocols are controlled by RBAC, ensuring that only authorized individuals can access the clusters. The same applies to services using service accounts, with connectivity restricted unless the respective accounts have been granted explicit permission. By enforcing RBAC policies, organizations can maintain a secure environment and minimize the risk of unauthorized access.

11.1.6.1 Assessment – SAMM

The Perimeter network security approach, helps address several SAMM security standards and their domains. This architecture design component aligns with multiple SAMM security domains, contributing to the establishment of a robust security governance framework, secure software design, implementation, verification, and operations. It helps address various security standards and requirements, mitigating risks and vulnerabilities throughout the software development and deployment lifecycle

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: The architecture design component ensures that policies and standards are implemented to define expectations and requirements for software security. It includes secure coding practices, vulnerability management, and data protection Risk Management: The architecture design component includes a segregated architecture with separate frontend and backend clusters, limiting the attack surface and reducing the risk of unauthorized access between clusters. Security Training and Awareness: The architecture design component emphasizes RBAC (Role-Based Access Control) and service accounts, which restrict inbound connections and ensure that only authorized individuals or services can operate in different environments. Security Roles and Responsibilities: The architecture design component defines and assigns security roles and responsibilities within the organization, ensuring accountability for software security and that individuals have the necessary knowledge and skills to fulfill their roles effectively. Security Requirements: The architecture design component incorporates security requirements into the design phase, ensuring that security concerns are identified and addressed, thereby mitigating OWASP threats such as Broken Access Control, Injection, and Insecure Design.

#	Name	Description
2	SAMM Security by Design	Secure Architecture: The architecture design component promotes the adoption of secure architectural patterns and principles, creating a robust foundation for security controls and addressing OWASP threats such as Insecure Design, Cryptographic Failures, and Security Misconfiguration. Threat Modeling The architecture design component includes threat modeling exercises during the design phase, helping identify and mitigate potential security risks associated with OWASP threats. Secure Coding Practices: The architecture design component emphasizes the use of secure coding practices, reducing the likelihood of OWASP threats such as Injection, Insecure Design, and Security Misconfiguration. Security Testing: The architecture design component integrates security testing activities, such as static code analysis and security code reviews, to detect and address vulnerabilities related to OWASP threats and ensure the resilience of the design against potential attacks.
3	SAMM Security by Implementation	Secure Coding Practices The architecture design component emphasizes the use of secure coding practices, reducing the likelihood of OWASP threats such as Injection, Insecure Design, and Security Misconfiguration. Security Testing The architecture design component integrates security testing activities, such as static code analysis and security code reviews, to detect and address vulnerabilities related to OWASP threats and ensure the resilience of the design against potential attacks.
4	SAMM Security by Verification	It does not apply to this domain.
5	SAMM Security by Operations	Environment Hardening: The architecture design component emphasizes secure configurations and firewalls to harden the software environment, protecting against known vulnerabilities and security weaknesses. Configuration Management: The architecture design component includes effective configuration management practices, such as version control and change management, to maintain the integrity of software systems and detect unauthorized changes or misconfigurations. Incident Management: The architecture design component incorporates incident management processes to handle security incidents effectively, including incident detection, response, containment, and recovery activities.

11.1.6.2 Assessment – OWASP Top 10

In summary, the architecture design principle addresses the Broken Access Control (A01-2021) and Identification and Authentication Failures (A07-2021) threats to some extent, but it does not directly address the other OWASP threats mentioned.
Here's how the given architecture design principle addresses the OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The architecture addresses this threat by implementing proper access controls. The Backend cluster is only accessible through traffic redirected from the Frontend cluster, ensuring that unauthorized access is restricted.
2	Identification and Authentication Failures (A07-2021)	The architecture addresses this threat by controlling inbound connections related to operational traffic and using RBAC (Role-Based Access Control) to restrict traffic by default.

11.1.7 External Proxy

The External Proxy capability, utilizing services like Cloudflare or Azure Front Door, strengthens the security of the architecture by providing advanced security features, The data centre can enhance the protection of the infrastructure hosted within the Kubernetes cluster, improve performance, and simplify management and administration tasks.
By introducing an External Proxy to the architecture such as Cloudflare or Azure Front Door, bolster the security of the data center where the Kubernetes cluster resides. By routing traffic through an external proxy, organizations can benefit from additional security features and mitigate potential threats targeting the infrastructure.
The External Proxy component, such as Cloudflare or Azure Front Door, can contribute to addressing various SAMM security domains by providing security features like WAF, DDoS protection, secure routing, secure deployment, and environment hardening. These features help organizations meet security requirements, mitigate OWASP threats, and establish robust operational processes to support secure software development and operations.

#	Name	Description
1	Enhanced Security	The additional security features offered by the external proxy, including WAF, DDoS protection, secure routing, and Zero Trust Security, contribute to a more robust and resilient security posture. These features help prevent and mitigate various types of attacks, ensuring the protection of the infrastructure hosted in the Kubernetes cluster.
2	Improved Performance	External proxies often include features like content caching and intelligent traffic routing, which can optimize the delivery of content and improve overall performance. By leveraging these capabilities, organizations can enhance the user experience and reduce latency for accessing applications and services hosted within the Kubernetes cluster.
3	Scalability and Reliability	External proxies are designed to handle high volumes of traffic and provide scalability and reliability. By offloading certain tasks, such as DDoS protection and traffic routing, to the external proxy, the Kubernetes cluster can focus on serving the applications and services, ensuring optimal performance and availability even during peak traffic periods.

#	Name	Description
4	Simplified Management	The external proxy acts as a centralized point of control and management for security policies, routing configurations, and other related settings. This centralized management simplifies the administration and monitoring of the security measures applied to the architecture, reducing complexity and potential management overhead.

As the features of this architecure component are vendor specific, we take Cloudflare as the case reference for describing the exernal proxy capabilities. So, when utilizing Cloudflare as an external proxy, several security features can be leveraged to protect the architecture:

#	Name	Description
1	Web Application Firewall (WAF)	Cloudflare's WAF provides an additional layer of defense against code injection attacks, such as SQL injection and cross-site scripting (XSS). It inspects incoming traffic and applies security rules to detect and block malicious requests, safeguarding the applications and services hosted within the Kubernetes cluster.
2	DDoS Protection	Cloudflare's DDoS protection mitigates Distributed Denial-of-Service (DDoS) attacks by intelligently filtering incoming traffic and identifying and blocking malicious requests. This helps ensure the availability and reliability of the infrastructure, minimizing the impact of DDoS attacks.
3	DNS (Domain Name System)	By utilizing Cloudflare's DNS service, the data centre can enhance the security and performance of their domain name resolution. Cloudflare's DNS offers advanced security features, including DNSSEC (Domain Name System Security Extensions), which helps prevent DNS spoofing and other attacks targeting the domain name infrastructure.
4	Secure Routing (Argo Tunnel)	Cloudflare's Argo Tunnel establishes a secure and private connection between the Kubernetes cluster and the Cloudflare network. This enables secure routing of traffic from external clients to the cluster while keeping the underlying infrastructure protected. Argo Tunnel encrypts and tunnels traffic over HTTPS, providing an additional layer of confidentiality and integrity.
5	Zero Trust Security	Cloudflare's Zero Trust Security offering, while requiring further investigation, presents an opportunity to implement a robust security model. Zero Trust Security operates on the principle of "trust no one, verify everything" and focuses on granular access controls, continuous authentication, and strict security policies to protect applications and data. By adopting Zero Trust Security principles, organizations can improve the overall security posture of their architecture.

11.1.7.1 Assessment – SAMM

The proposed architecture component, the External Proxy (e.g., Cloudflare or Azure Front door), can help address the SAMM security standards and its domains in the following ways:

#	Name	Description
1	SAMM Security by Governance	Compliance Management The External Proxy can help organizations ensure compliance with relevant regulatory requirements and industry standards by providing features such as WAF (Web Application Firewall) and secure routing. Security Training and Awareness The External Proxy can contribute to security training and awareness by offering features like DDoS protection and DNS security, which help educate employees and stakeholders about potential security threats.
2	SAMM Security by Design	Security Requirements The External Proxy, with its security features like WAF, can help the organization to enforce secure coding practices, vulnerability management, and incident response, thereby addressing security requirements. Secure Architecture By leveraging the External Proxy's secure routing capabilities, organizations can enhance the secure architecture of their systems, mitigating OWASP threats such as Insecure Design, Security Misconfiguration, and Cryptographic Failures.
3	SAMM Security by Implementation	Secure Coding Practices The External Proxy's WAF feature can prevent code injection and protect against vulnerabilities associated with OWASP threats like Injection and Security Misconfiguration. Security Testing Organizations can incorporate security testing activities into the implementation phase by leveraging the External Proxy's features such as DDoS protection and secure routing, which help detect and address vulnerabilities.
4	SAMM Security by Verification	Security Testing The External Proxy's WAF, DDoS protection, and secure routing features contribute to security testing activities by helping the organisation to identify vulnerabilities and weaknesses in software applications. Security Architecture Review The External Proxy's features can be assessed during security architecture reviews to ensure that proper security controls and mechanisms are in place.
5	SAMM Security by Operations	Environment Hardening The External Proxy can assist in hardening the software environment by providing secure network and infrastructure configurations. Secure Deployment The organisation can leverage the External Proxy's features for secure software distribution, installation procedures, and configuration, ensuring that software is deployed in a secure manner.

11.1.7.2 Assessment – OWASP Top 10

In summary, the use of a External Proxy design pattern, using products such as Cloudflare or Azure Front door can address several OWASP threats by providing additional security features, access controls, secure

configurations, and protection against common attack vectors. However, it's important to note that the effectiveness of these solutions depends on their proper configuration and ongoing monitoring and maintenance.

The mentioned architecture design principle can address several OWASP threats.

#	Name	Description
1	Broken Access Control (A01-2021)	The External Proxy component can provide additional access control mechanisms, such as Web Application Firewall (WAF), to prevent code injection and enforce proper access controls, reducing the risk of unauthorized access.
2	Cryptographic Failures (A02-2021)	While The External Proxy component is not directly related to cryptographic practices, they can indirectly contribute to the security of cryptographic components by protecting the underlying infrastructure against attacks and vulnerabilities that could compromise cryptographic mechanisms.
3	Injection (A03-2021)	The External Proxy component can help prevent injection attacks by implementing proper input validation and sanitization mechanisms, reducing the risk of untrusted data manipulation and unintended execution of commands.
4	Insecure Design (A04-2021)	The External Proxy component can enhance the overall security of their architecture by integrating security controls and mechanisms into the design, thus addressing potential insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	The External Proxy component offer secure configurations by default, reducing the risk of misconfigurations that could lead to security vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	The External Proxy component may not directly address vulnerable and outdated components within the web application itself, they can provide protection against known vulnerabilities in the underlying infrastructure, minimizing the risk associated with using outdated or vulnerable components.
7	Identification and Authentication Failures (A07-2021)	The External Proxy component can enhance identification and authentication mechanisms by providing additional layers of security, such as two-factor authentication (2FA) or integration with identity providers, reducing the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	The External Proxy component can contribute to ensuring the integrity of software and data within web applications by protecting against unauthorized modifications, tampering, or destruction of components and configurations.
9	Security Logging and Monitoring Failures (A09-2021)	The External Proxy component can provide logging and monitoring capabilities, including real-time monitoring and analysis of security-related logs, helping organizations detect and respond to security incidents more effectively.
10	Server-Side Request Forgery (A10-2021)	The External Proxy component can help prevent Server-Side Request Forgery (SSRF) attacks by implementing secure configurations and input validation mechanisms that reduce the risk of making unintended requests to internal or external resources.

11.1.8 API Gateway (Secure APIs)

Kong API Gateway has been created as a customised version of NGINX and used as as an API restful API reverser Proxy.

It can play the role of Ingress Controller for Kubernetes. So, it can be configured as the exclusive entry point for all cluster communications.

It provides security features, therefore it would be a strategic implementation for an enahnced security validation checks at the infrastructure level.

Kong API Gateway provides several plugins that must be configured to enhance security. All these security features add-ons provided by Kong will be configured accordingly with the latest detailed design of the system. Some add-ons can overlap; not all are needed in all scenarios. It ensures that only valid and authorized requests are processed while protecting against common web application attacks and unauthorized access attempts. Below a list of some of the security related capabilities that can be added to the gateway as plugins.

#	Name	Description
1	ABAC (Attribute-Based Access Control)	ABAC provides security to incoming requests by verifying the content of the message. This capability prevents attacks where someone with a valid session token crafts a payload to bypass the application's security measures. ABAC ensures that data not normally accessible to the user through the application cannot be leaked.
2	RBAC (Role-Based Access Control)	RBAC capability checks the roles or privileges granted to a user for using the API. It can be implemented as a custom plugin that calls an underlying service to retrieve information about the user's roles and associated privileges for specific endpoints. RBAC enforces access control based on the user's application roles.
3	JWT Session Token Validation	The JWT Session Token Validation plugin validates the session token against the Login Service, which manages the lifecycle of user sessions. It ensures that the session token is valid and has not expired, providing secure authentication for API requests.
4	JWT Session Token Authorization Validation	This custom plugin validates the token claims, including scopes, against the User Access Management component. It checks the user's roles and the APIs they are entitled to access based on the application's user roles and defined authorization rules.
5	Identity Aware Plugin	The Identity Aware plugin aggregates the invocations to the RBAC (Role-Based Access Control) and session management capabilities. It ensures that the access control and session validation are performed collectively, providing comprehensive security checks for API requests.
6	Web Application Firewall (WAF)	If a dedicated WAF instance is not implemented at the forefront of incoming traffic, a custom plugin can be created to connect to a WAF instance. This helps protect the APIs from common web application attacks, such as SQL injection, cross-site scripting (XSS), and more.
7	Payload Syntax Validation	The Payload Syntax Validation plugin allows you to validate the syntax of the request payload using either an out-of-the-box plugin or a custom one. This ensures that the payload adheres to the defined syntax rules and prevents malformed or invalid requests from being processed.
8	Payload Semantic Validation	With the Payload Semantic Validation capability, you can create a custom plugin that performs semantic validation on the request payload. This involves implementing a specialized semantic rule engine, which verifies the payload against a configuration file per endpoint. It ensures that the payload follows the required semantic rules, providing an additional layer of validation.
9	CORS (Cross-Origin Resource Sharing)	The CORS plugin allows you to configure restrictions on cross-origin HTTP requests made by clients that are not in the same domain as the deployed APIs. This prevents unauthorized cross-origin requests and ensures that only allowed applications can invoke the APIs.

#	Name	Description
10	IP Whitelisting	The IP Whitelisting plugin enables you to specify a list of IP addresses or ranges that are allowed to invoke the APIs. This is particularly useful in scenarios involving business-to-business (B2B) integrations where the connections between systems are predefined.
11	Bot Detection	The Bot Detection plugin provides protection against automated bots attempting to access the API endpoints. It uses algorithms to recognize invocation patterns commonly used by bots and offers capabilities to whitelist or blacklist clients temporarily or permanently based on their behavior.
12	JWT Validation (Local Check)	The JWT Validation plugin verifies the integrity and expiration of requests containing signed JSON Web Tokens (JWTs) using the HS256 or RS256 algorithm. It performs a local validation without introspecting the token's content, ensuring that the token has not been tampered with or expired.
13	Rate Limiting	The Rate Limiting plugin allows you to configure and enforce limits on the number of HTTP requests and the frequency at which a client can make requests. It provides control over the rate of incoming requests to prevent abuse and ensure fair usage of the API resources.
14	Correlation ID (Add Correlation ID)	The Correlation ID plugin adds a unique identifier to each incoming request, facilitating logging, traceability, and operational support. It also helps with security-related aspects such as non-repudiation and auditing. If the incoming request does not contain a correlation ID in the header, the plugin generates a new one and adds it to the request payload.

11.1.8.1 Assessment – SAMM

Kong API Gateway contributes to addressing the SAMM security standards and its domains as follows:

#	Name	Description
1	SAMM Security by Governance	Policies and Standards Kong API Gateway helps enforce security policies and standards by providing add-ons such as CORS, IP Whitelisting, and Bot Detection. These add-ons allow organizations to define and enforce restrictions on API access based on predefined policies. Compliance Management Kong API Gateway supports compliance management by providing security features like JWT Validation, Rate Limiting, and Web Application Firewall. These features help organizations meet regulatory requirements and adhere to industry standards. Security Metrics and Reporting Kong API Gateway provides logging and monitoring capabilities, including the Correlation ID add-on. This allows organizations to generate security metrics and reports, providing visibility into the status of software security.

#	Name	Description
2	SAMM Security by Design	Security Requirements Kong API Gateway enables the implementation of security requirements through add-ons like JWT Validation, IP Whitelisting, and Payload Syntax validation. These add-ons ensure that APIs are accessed securely and that requests meet specific security criteria. Secure Architecture Kong API Gateway acts as an API gateway and ingress controller, providing a secure architectural layer for API communication. It helps address OWASP threats by enforcing security controls at the network level, such as CORS, IP Whitelisting, and Web Application Firewall integration. Secure Coding Practices While Kong API Gateway is not directly involved in application-level coding, it contributes to secure coding practices by providing add-ons like JWT Validation, Payload Syntax validation, and Payload Semantic validation. These add-ons help validate the integrity and structure of requests, reducing the risk of security vulnerabilities.
3	SAMM Security by Implementation	Secure Development Training Kong API Gateway can be configured to enforce security-related practices, such as JWT Validation and Rate Limiting. By using these add-ons, organizations can ensure that developers adhere to secure coding practices and implement appropriate security controls. Secure Architecture Kong API Gateway contributes to secure architecture by acting as a secure entry point for API communication. It helps organizations implement secure communication channels and access controls, minimizing the risk of security vulnerabilities. Security Testing Integration Kong API Gateway supports security testing integration by providing logging and monitoring capabilities. These capabilities enable organizations to analyze and detect potential security issues during the development and deployment stages.
4	SAMM Security by Verification	Security Testing Kong API Gateway supports security testing activities through its logging and monitoring capabilities. By integrating with security testing tools and processes, organizations can identify and address security vulnerabilities in their software systems. Code Review Kong API Gateway is not directly involved in code review activities, as it primarily focuses on API gateway functionality. However, organizations can review the configuration and customization of Kong API Gateway to ensure it aligns with secure coding practices. Security Architecture Review Kong API Gateway's configuration and customization can be reviewed as part of the security architecture review process. Organizations can assess how Kong API Gateway is implemented and customized to meet their security requirements.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening Kong API Gateway contributes to environment hardening by providing security features such as IP Whitelisting, CORS, and Web Application Firewall integration. These features help organizations secure their API environments and protect against known vulnerabilities. Secure Build Kong API Gateway is not directly involved in the build process of software systems. However, organizations can ensure that Kong API Gateway is deployed and configured securely as part of their secure build practices. Configuration Management Kong API Gateway's configuration can be managed and documented as part of the overall configuration management process. This ensures that the API gateway's configuration remains consistent and aligned with security requirements. Vulnerability Management Kong API Gateway supports vulnerability management by providing security features like JWT Validation, Rate Limiting, and Web Application Firewall integration.

11.1.8.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	Kong API Gateway acts as an ingress controller and restricts traffic only to RESTful APIs, ensuring that access controls are properly enforced. Kong API Gateway serves as the only entry point for all backend cluster communications, providing centralized access control and preventing unauthorized access to restricted resources. Kong API Gateway implements security validation checks, which help enforce access control policies and prevent unauthorized access attempts.
2	Cryptographic Failures (A02-2021)	It act as a normal reverse proxy, therefore, it handles SSL termnination.
3	Injection (A03-2021)	It does address this capability if the WAF plugin is used.
4	Insecure Design (A04-2021)	It acts as a sanitaiser and repeler of any not well conformed payload.
5	Security Misconfiguration (A05-2021)	Kong API Gateway provides several security-enhancing plugins that need to be configured according to the latest detailed design of the system. By correctly configuring these plugins, security misconfigurations can be avoided.
6	Vulnerable and Outdated Components (A06-2021)	It does not address this capability directly.
7	Identification and Authentication Failures (A07-2021)	Kong API Gateway integrates with the User Access Management component, which validates token claims, scopes, and user roles to ensure proper identification and authentication. The JWT Validation plugin verifies the integrity and expiration of JSON Web Tokens (JWTs) to prevent unauthorized access attempts.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	It acts as a sanitaiser and repeler of any not well conformed payload.
9	Security Logging and Monitoring Failures (A09-2021)	It does address this capability by loggin all traffic and exceptions occurred in the platform.
10	Server-Side Request Forgery (A10-2021)	Kong API Gateway acts as a proxy and can help prevent server-side request forgery (SSRF) attacks by validating and sanitizing incoming requests before forwarding them to backend services.

11.1.9 Infrastructure RBAC access for Developers and Operators

Introduction

Considering the necessary controls to enable access permissions to operators and support personnel in the environments is important.

In particular, if adopting Kubernetes, this platform provides built-in functionality that allows controlling the access of individuals to the cluster if it is integrated with an LDAP-compatible platform. To explain this RBAC access control feature, we assume the organization already counts with an Active Directory (AD). The solution assumes that there is an existing Active Directory in the organization. Or an LDAP-compatible platform will be used as the source of truth for individual and service accounts. Kubernetes will use it for authenticating and authorizing access and privileges in the services hosted on the platform.

Regarding the solution, it is based on Kubernetes, which its clusters are RBAC capable. It means that AD Authentication can restrict access to all resources.

This feature enables access control for developers and operations accessing the environments. It also allows service account access and privileges to perform specific roles in the Kubernetes Cluster.

So, the “RBAC access for Developers and Operators" for infrastructure resources access control, involves considerations for implementing RBAC (Role-Based Access Control) in a Kubernetes environment integrated with an Active Directory (AD) or LDAP-compatible platform. This capability ensures controlled access and permissions for developers, operators, and support personnel in the infrastructure.

Individuals Accounts

The domain's team members will only have access to specific Kubernetes namespaces. This applies to Service Accounts needed to run automated processes, such as the Build Pipeline—for example, Jenkins or Azure DevOps.

Once the RBAC security is configured, the cluster users will be prompted for AD sign-in whenever they try to access a resource in the cluster.

The Native AD will authenticate the User via device login and give access to the Kubernetes API Server. For this, some components play a fundamental role. For example, The Service Account and the AD Groups, the Kubernetes Roles, the Role binding, etc.

Service Accounts

Regarding the Service Account, these are needed for automated processes. For example, the Build Pipeline should connect to the cluster via its associated service accounts. There will be one Pipeline service Account for each domain, and necessary role bindings will be done as the environments grow.

AD Groups

Regarding the AD Groups, these are created to segment privileges and access to different domains within the infrastructure. For this, domain-specific AD groups are created.

The Domain AD groups will for example:

Developers: "Kubernetes Service Cluster User Role", DevOps: "Subscription Contributor".

Infrastructure Admin: "Subscription owner".

Role Binding

Regarding the Role binding, the created groups need to be assigned RBAC permission to Kubernetes resources, so each domain group will be assigned to platform roles that are granted specific permission to resources in the different environments. In this scheme, every domain will have two roles:

Domain developer: With reader access and the domain-admin Role with admin access to the resources in the respective Kubernetes domain namespace.

Infrastructure-Admin: This role will be to manage all the domains and infrastructure.

At the Kubernetes level, DevOps and Infra AD groups will have cluster role binding, allowing them to access all resources and namespaces declared in the cluster. Domain Developers have a Namespace reader role, restricted to specific resources and verbs in the domain-specific namespace, and the Domain Admin has full access to the domain namespace.

Namespaces

If the infrastructure needs to support multitenancy, then it may need extra considerations when designing the AD Groups and matching namespaces. Firstly, there are different multitenancy models, for example, per-customer or enterprise.

In the case of an enterprise, this is used for internal partitioning of environments when it is required to divide the environments logically and create virtual clusters, intending to minimize the need for a cluster per environment. Everything that is non-prod can be managed within a single cluster with namespaces segregation.

In the per-customer model, it is a client based multitenancy model that requires configuring role permissions specifics for each tenant. Resources, network and persmissions are completely segregated.

In both case this can be achieved using Kubernetes hierarchical namespaces but with differnet configurations.

Below a summary of the capabilities and components invovled in the enablement of the “RBAC access for Developers and Operators" for infrastructure resources access control:

#	Name	Description
1	RBAC Integration with Active Directory	The solution assumes the presence of an existing Active Directory or an LDAP-compatible platform as the source of truth for individual and service accounts. Kubernetes integrates with this AD or LDAP platform to authenticate and authorize access to services hosted on the platform.

#	Name	Description
2	RBAC Controls in Kubernetes Clusters	Kubernetes clusters inherently support RBAC functionality, allowing fine-grained access control to all resources. This feature enables administrators to control access and permissions for developers, operators, and service accounts within the Kubernetes clusters.
3	Namespace-Based Access	Team members are granted access only to specific Kubernetes namespaces. This access restriction applies to service accounts required for automated processes such as the Build Pipeline (e.g., Jenkins or Azure DevOps). By limiting access to specific namespaces, security and separation of responsibilities are maintained.
4	AD Authentication for Cluster Users	Once RBAC security is configured, cluster users are prompted for Active Directory sign-in whenever they attempt to access a resource in the cluster. Native AD handles the authentication process, allowing access to the Kubernetes API Server. Key components like Service Accounts, AD Groups, Kubernetes Roles, and Role Bindings play vital roles in this authentication flow.
5	Service Accounts	Service accounts are utilized for automated processes, such as the Build Pipeline. Each domain may have one pipeline service account associated with it, and as the environments grow, the necessary role bindings are established.
6	AD Groups	AD groups are created to segment privileges and access for different domains within the infrastructure. Domain-specific AD groups are established to manage access control efficiently.
7	Role Bindings	The created AD groups need to be assigned RBAC permissions for Kubernetes resources. Each domain group is assigned specific platform roles that grant permissions to resources in different environments. Roles like domain developer (with reader access), domain admin (with admin access to domain namespace), and infrastructure admin (managing all domains and infrastructure) are assigned accordingly.
8	Multitenancy Considerations	If the infrastructure needs to support multitenancy, additional considerations are required when designing AD groups and matching namespaces. Different multitenancy models, such as per-customer or enterprise, can be employed. Hierarchical namespaces can be configured to achieve tenant-specific role permissions, enabling logical division and virtual clusters to minimize the need for separate physical clusters

11.1.9.1 Assessment – SAMM

Infrastructure RBAC access for Developers and Operators does not directly address the SAMM security standards and its domains. However, it can indirectly contribute to the Security by Implementation and Security by Operations domains in SAMM.

While RBAC access control does not directly align with other specific aspects of the SAMM domains, it contributes to establishing secure practices and controls in the development and operation of software systems, thereby indirectly supporting the overall objectives of SAMM.

#	Name	Description
1	SAMM Security by Governance	It does not address this capability directly.
2	SAMM Security by Design	It does not address this capability directly.

#	Name	Description
3	SAMM Security by Implementation	In the Security by Implementation domain, RBAC access control helps ensure that secure development practices are followed by granting developers and operators access only to the resources and namespaces they require. This helps enforce security requirements, such as separation of duties, and reduces the risk of unauthorized access and potential vulnerabilities.
4	SAMM Security by Verification	It does not address this capability directly.
5	SAMM Security by Operations	In the Security by Operations domain, RBAC access control supports the secure operation of software systems by limiting access to sensitive resources and ensuring that only authorized individuals can perform specific actions. It helps enforce secure deployment practices, secure configuration management, and incident management processes by controlling access to relevant operations and resources.

11.1.9.2 Assessment – OWASP Top 10

The RBAC access design contributes significantly to addressing several OWASP threats, particularly Broken Access Control, Insecure Design, Security Misconfiguration, Identification and Authentication Failures, and Security Logging and Monitoring Failures.

#	Name	Description
1	Broken Access Control (A01-2021)	The RBAC (Role-Based Access Control) access control feature provided by Kubernetes addresses this threat. It enables access control for developers and operators accessing the environments by allowing fine-grained control over the permissions and privileges granted to individuals and service accounts. RBAC ensures that only authorized users have access to specific resources and functionalities, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	RBAC access control is not specifically related to cryptographic algorithms and mechanisms within a web application. It focuses on access management rather than cryptographic practices.
3	Injection (A03-2021)	RBAC access control is not directly related to the handling of untrusted data or preventing injection attacks. It primarily focuses on access permissions and authorization within the Kubernetes environment.
4	Insecure Design (A04-2021)	The infrastructure RBAC access design considers security from an architectural perspective. By implementing RBAC controls and mapping them to Active Directory (AD) groups, the design ensures that access permissions are properly defined and enforced. This helps in creating a secure and robust access control architecture, addressing the vulnerabilities associated with insecure design decisions.
5	Security Misconfiguration (A05-2021)	The RBAC access design helps in mitigating security misconfiguration vulnerabilities by providing a standardized and controlled approach to access management. By integrating with Active Directory or an LDAP-compatible platform, the RBAC design ensures that access permissions are centrally managed and consistently applied across the Kubernetes cluster. This reduces the chances of misconfigurations and insecure settings that could lead to potential entry points for attackers.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	RBAC access control does not directly address the usage of outdated or vulnerable software components within a web application. It focuses on access management rather than component vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	RBAC access control plays a significant role in addressing identification and authentication failures. By integrating with Active Directory for authentication, RBAC ensures that only authenticated users with valid AD credentials can access the Kubernetes cluster and its resources. This helps in enforcing strong authentication mechanisms and mitigating vulnerabilities associated with weak or inadequate authentication.
8	Software and Data Integrity Failures (A08-2021)	While RBAC access control helps in enforcing proper access permissions, it does not directly address vulnerabilities related to software and data integrity within web applications. It focuses on access management rather than integrity protection mechanisms.
9	Security Logging and Monitoring Failures (A09-2021)	RBAC access design indirectly contributes to addressing security logging and monitoring failures. By implementing RBAC controls, organizations can establish granular access permissions and track user activities within the Kubernetes cluster. This allows for effective logging and monitoring of user actions, aiding in the detection and response to security incidents.
10	Server-Side Request Forgery (A10-2021)	RBAC access control does not directly address the prevention of server-side request forgery vulnerabilities. It primarily focuses on access permissions and authorization within the Kubernetes environment.

11.1.10 User Authentication (Login Service)

The Login Service component provides authentication functionality

The Login Proxy will be used as a decision and routing mechanism to allow abstracting of which Authentication instance will be used.

11.1.10.1 Assessment – SAMM

The Login Service component does not directly address any specific SAMM security domain. However, it can contribute to overall software security by providing authentication functionality, which is crucial for ensuring that only authorized users can access the system. User authentication helps mitigate the risk of unauthorized access, which is a fundamental aspect of security across all SAMM domains. By implementing a robust login service, organizations can enforce proper authentication mechanisms and protect sensitive resources and data from unauthorized access.

#	Name	Description
1	SAMM Security by Governance	Security Roles and Responsibilities: The Login Service can enforce role-based access control (RBAC) by verifying the user's identity and associating it with specific security roles and responsibilities within the system. This contributes to establishing accountability for software security.

#	Name	Description
2	SAMM Security by Design	Security Requirements: The Login Service can enforce security requirements related to user authentication, such as password complexity, multi-factor authentication, and session management. This helps address security concerns associated with authentication vulnerabilities, such as weak passwords and session hijacking.
3	SAMM Security by Implementation	Secure Development Training: The Login Service implementation can adhere to secure coding practices and guidelines to ensure that authentication mechanisms are implemented securely, reducing the likelihood of common vulnerabilities like credential leakage or brute-force attacks. Security Requirements: The Login Service implementation can integrate security requirements related to user authentication into the software development process, ensuring that the system meets the desired security objectives in terms of authentication controls.
4	SAMM Security by Verification	Security Testing: The Login Service can be subjected to security testing activities, such as vulnerability scanning or penetration testing, to identify potential vulnerabilities or weaknesses in the authentication mechanism and ensure its resilience to attacks.
5	SAMM Security by Operations	Secure Deployment: The Login Service can be securely deployed, following best practices for secure software distribution, installation procedures, and configuration of authentication-related components. While the Login Service component may not directly align with specific SAMM domains, its implementation and adherence to security best practices can contribute to addressing various security concerns within the SAMM framework.

11.1.10.2 Assessment – OWASP Top 10

The User Authentication (Login Service) capability can address several OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The Login Service component can enforce access controls by verifying user credentials and determining their privileges and permissions. By properly implementing and enforcing these access controls, it can prevent unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	It does not address this capability directly.
3	Injection (A03-2021)	It does not address this capability directly.
4	Insecure Design (A04-2021)	The Login Service component can be designed with security in mind, considering secure design principles and best practices. This helps ensure that the authentication mechanisms are implemented correctly and securely, reducing the risk of insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	The Login Service component can be configured securely by following recommended practices and industry standards. This includes configuring secure authentication mechanisms, enforcing strong password policies, and using secure protocols. By avoiding common security misconfigurations, the component reduces the risk of security vulnerabilities.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	It does not address this capability directly.
7	Identification and Authentication Failures (A07-2021)	The Login Service component directly addresses this threat by providing the functionality for user identification and authentication. By implementing secure authentication mechanisms, such as multi-factor authentication, enforcing strong password policies, and preventing authentication bypass vulnerabilities, it reduces the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	It does not address this capability directly.
9	Security Logging and Monitoring Failures (A09-2021)	The Login Service component can incorporate logging and monitoring capabilities to track user authentication activities and detect potential security incidents. By logging relevant events and implementing real-time monitoring mechanisms, it enhances the organization's ability to detect and respond to authentication-related security incidents. Please note that while the Login Service component can contribute to mitigating these threats, it is essential to implement secure coding practices, conduct regular security assessments, and follow other relevant security measures throughout the entire application to provide comprehensive protection against OWASP threats.
10	Server-Side Request Forgery (A10-2021)	It does not address this capability directly.

11.1.11 User Authorisation (User Access Management)

The User Authorization provides a robust mechanism for managing user access and permissions. This capability ensures that only authorized users can access the applications and services hosted within the infrastructure, enhancing security and maintaining control over resource usage.

The User Access Management is a capability that provides a centralised authorization functionality to Applications within an eco-system.

User Access Management (UAM) implements the Identity Access Manager (IAM) system that can be used by any application hosted in the same network domain.

At the time of login-in, the user credentials will be loaded in the session by the Login Service accessing the User Access Management service.

After successful authentication, it the login service will retrieve a Users privileges granted to one or more applications. It will build the token and place it in the session storage.

The extended capability for "User Authorization (User Access Management)" involves the implementation of an Identity Access Manager (IAM) system and User Access Management (UAM) service to provide authorization functionality for applications hosted in the infrastructure. Below, a further explanation of this capability’s components.

#	Name	Description
1	Identity Access Manager (IAM) System	The IAM system serves as the foundation for user access management within the infrastructure. It provides a centralized platform to manage user identities, access rights, and permissions across various applications and services. The IAM system ensures secure and controlled access to resources based on defined policies and roles.
2	User Access Management (UAM) Service	The UAM service is a component within the IAM system that handles user access authorization specifically. It integrates with the Login Service, which is responsible for user authentication during the login process.
3	Loading User Credentials and Session Management	During the login process, the Login Service retrieves user credentials provided by the user and communicates with the UAM service. The UAM service validates the user's credentials and retrieves the privileges granted to the user for one or more applications.
4	Token Generation and Session Storage	Upon successful authentication and authorization, the UAM service generates a token that represents the user's authenticated session. This token is securely stored in the session storage, associated with the user's session. The token contains the necessary information to identify the user and their authorized privileges.
5	Token life-cycle management	The generated token serves as a proof of the user's authorization and is utilized for subsequent interactions with the application or services hosted in the infrastructure. It may contain information such as the user's identity, role, permissions, and other relevant attributes.
6	User authorisation validation	The token-based authentication and session management approach allows applications hosted in the infrastructure to validate and authorize user requests efficiently. The token is passed along with each request, enabling the application to verify the user's identity and ensure they have the necessary privileges to perform the requested actions.

11.1.11.1 Assessment – SAMM

User Access Management plays a crucial role in establishing secure user access controls and contributing to various aspects of SAMM's Security by Operations domains.

#	Name	Description
1	SAMM Security by Governance	It does not address directly to this domain.
2	SAMM Security by Design	It does not address directly to this domain.
3	SAMM Security by Implementation	It does not address directly to this domain.

#	Name	Description
4	SAMM Security by Verification	It does not address directly to this domain.
5	SAMM Security by Operations	Configuration Management User Access Management ensures that user privileges and access rights are properly managed and controlled, contributing to effective configuration management practices. By implementing user access controls and managing user permissions, organizations can maintain the integrity of software systems and prevent unauthorized changes or misconfigurations. Incident Management User Access Management plays a role in incident management by providing mechanisms to control and track user access. It enables organizations to monitor and detect any suspicious or unauthorized activities, contributing to incident detection and response capabilities. Secure Deployment User Access Management helps ensure secure deployment practices by managing user access during the deployment process. It includes secure distribution of user credentials and access controls, secure installation procedures, and configuration of user permissions, minimizing the risk of unauthorized access or compromise during deployment. perational Enablement User Access Management contributes to operational enablement by providing the necessary operational support and guidance for managing user access. This includes activities such as user access training for operational staff, documentation of user access procedures, and establishing user access incident response capabilities.

11.1.11.2 Assessment – OWASP Top 10

User Access Management can address these threats, it should be implemented as part of a comprehensive security strategy that includes other security practices and controls to provide a layered defense against different types of vulnerabilities.

The User Access Management capability addresses several OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	User Access Management ensures proper authorization and access controls are implemented. It verifies and enforces access controls, preventing unauthorized access to restricted resources.
2	Cryptographic Failures (A02-2021)	It does not address directly to this domain.
3	Injection (A03-2021)	It does not address directly to this domain.

#	Name	Description
4	Insecure Design (A04-2021)	User Access Management contributes to secure architecture design by implementing a centralized Identity Access Manager (IAM) system. This ensures that security considerations are taken into account during the design phase and that the architecture is built with robust security controls.
5	Security Misconfiguration (A05-2021)	User Access Management helps prevent security misconfigurations by providing a standardized and centralized approach to user authorization. It ensures that access controls are properly configured and eliminates the potential for insecure settings or exposed sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	User Access Management can indirectly address this threat by implementing secure practices for authentication and authorization. By using up-to-date and secure authentication mechanisms, it reduces the risk of vulnerabilities associated with outdated or insecure components.
7	Identification and Authentication Failures (A07-2021)	User Access Management plays a crucial role in implementing secure identification and authentication mechanisms. It enforces strong password policies and ensures that proper authentication techniques are used, reducing the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	It does not address directly to this domain.
9	Security Logging and Monitoring Failures (A09-2021)	While User Access Management itself may not directly address this threat, it can contribute to a comprehensive security logging and monitoring strategy. By integrating with logging and monitoring systems, it enables better visibility into user access activities, helping to detect and respond to security incidents more effectively.
10	Server-Side Request Forgery (A10-2021)	It does not address directly to this domain.

11.1.12 Identity Aware Proxy

The Identity Aware Proxy is an intelligent built-for-purpose reverser proxy responsible for authenticating users' permissions before the traffic reaches the application. An externalized security checkpoint verifies all the traffic before entering the application.

With this design, the applications do not have to have embedded logic to check for user authorization.

The Identity aware proxy can be extended in security capabilities and verifies more than user permissions (the software proposed already has this functionality as a built-in feature and is part of the out-of-the-box functionality of the software suite).

For example, extended verifications can include IP addresses (geolocation), client machine MAC addresses, etc.

#	Name	Description
1	IP Address Verification	The proxy can verify the IP address of the incoming request and perform geolocation checks to ensure that the request originates from an authorized location or to detect suspicious activities from certain regions.
2	Client Machine MAC Address Verification	The proxy can also validate the MAC address of the client machine sending the request. This verification can help ensure that only authorized devices are allowed to access the application.
3	Device Identification	The Identity Aware Proxy can gather information about the device making the request, such as its operating system, browser, and other attributes. This information can be used to enforce security policies or restrict access based on device characteristics.
4	Integration with Security Services	The extended Identity Aware Proxy can integrate with various security services and technologies to enhance its verification capabilities, including a real time Threat Intelligence Services which is a proxy can leverage threat intelligence feeds to identify and block requests originating from known malicious IP addresses or suspicious network patterns.
5	Malware Detection Services	By integrating with malware detection services, the proxy can scan incoming requests for potential malware or malicious payloads before allowing them to reach the application.
6	Behavioral Analytics	The proxy can utilize behavioral analytics to detect anomalies in user behavior and flag suspicious activities for further investigation. This can help identify potential unauthorized access attempts or compromised user accounts.
7	Customizable Verification Rules	The extended Identity Aware Proxy should provide flexibility in defining and configuring verification rules based on specific security requirements. Administrators should be able to define rules and policies to determine which verifications are applied and under what conditions. This flexibility allows organizations to tailor the proxy's behavior to their unique security needs.

11.1.12.1 Assessment – SAMM

Identity Aware Proxy (IAP) can contribute to addressing the SAMM security standards as follows:

#	Name	Description
1	SAMM Security by Governance	Policies and Standards IAP helps enforce security policies and standards by acting as a security checkpoint before allowing traffic to reach the application. It verifies user permissions and other security capabilities, ensuring that only authorized traffic is allowed. Compliance Management IAP can help organizations comply with regulatory requirements and industry standards by enforcing access controls and verifying user permissions based on policies and standards. Security Training and Awareness IAP can contribute to security training and awareness by providing visibility into user access and permissions, helping organizations identify potential security gaps and take appropriate measures. Security Roles and Responsibilities IAP assists in defining and assigning security roles and responsibilities by providing centralized access control and authentication mechanisms.
2	SAMM Security by Design	Security Requirements IAP can help enforce security requirements by verifying user permissions and ensuring that access controls are properly implemented. Secure Architecture IAP provides a secure architectural component by acting as a reverse proxy that authenticates and authorizes user traffic before reaching the application. It minimizes potential vulnerabilities and creates a strong foundation for security controls. Secure Coding Practices While IAP does not directly address coding practices, it can complement secure coding efforts by offloading user authentication and authorization logic from the application code, reducing the risk of coding mistakes related to access control. Threat Modeling IAP can contribute to threat modeling exercises by providing insights into user access patterns and potential vulnerabilities related to user permissions. Security Testing IAP can be included in security testing activities by verifying the effectiveness of access controls and authentication mechanisms.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training While IAP is not directly related to secure development training, it can support secure development practices by providing a secure access control mechanism and reducing the burden of implementing user authentication and authorization logic. Secure Architecture IAP acts as a secure component during the implementation phase by ensuring that user access is properly controlled and verified. Security Requirements IAP enforces security requirements by verifying user permissions based on defined policies and standards. Security Testing Integration IAP can be integrated into security testing activities by verifying access controls and authentication mechanisms. Security Verification IAP contributes to security verification by validating the effectiveness of access controls and authorization mechanisms. Security Architecture Review IAP can be part of the security architecture review process by evaluating the implementation and effectiveness of access controls. Security Operations Integration IAP integrates with security operations by providing a centralized access control and authentication mechanism.
4	SAMM Security by Verification	Security Testing IAP can be included in security testing activities by verifying the effectiveness of access controls and authentication mechanisms. Code Review IAP does not directly address code review activities. Security Architecture Review IAP can be part of security architecture review activities by evaluating the implementation and effectiveness of access controls. Security Requirements Verification IAP contributes to security requirements verification by verifying user permissions based on defined policies and standards. Threat Modeling IAP can contribute to threat modeling exercises by providing insights into user access patterns and potential vulnerabilities related to user permissions. Secure Deployment Verification IAP ensures secure deployment by acting as a reverse proxy that verifies user access and permissions.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening IAP can contribute to environment hardening by providing a secure access control mechanism and verifying user permissions before traffic reaches the application. Secure Build While IAP is not directly related to secure build practices, it can be considered as part of the overall secure build process by providing a secure access control and authentication component. Configuration Management IAP does not directly address configuration management activities. Vulnerability Management IAP can contribute to vulnerability management by verifying user permissions and access controls.

11.1.12.2 Assessment – OWASP Top 10

While the Identity Aware Proxy can contribute to mitigating these OWASP threats, it is not a comprehensive solution to tackle all OWASP Top10 threats. So, other security measures, such as input validation, secure coding practices, and regular security testing, should also be implemented to achieve a robust security posture.

#	Name	Description
1	Broken Access Control (A01-2021)	The Identity Aware Proxy acts as a security checkpoint that authenticates user permissions before allowing traffic to reach the application. By externalizing the authentication and authorization process, it helps ensure proper access controls are enforced, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	It does not address directly to this domain.
3	Injection (A03-2021)	It does not address directly to this domain.
4	Insecure Design (A04-2021)	The Identity Aware Proxy contributes to a secure architecture design by providing a centralized mechanism for user authentication and authorization. By removing the need for applications to embed access control logic, it helps prevent insecure design decisions related to access control implementation.
5	Security Misconfiguration (A05-2021)	The Identity Aware Proxy can enforce secure configurations for web applications. It acts as a reverse proxy and can apply security policies, such as verifying IP addresses or geolocation, before allowing traffic to reach the application. By ensuring proper configuration and secure defaults, it reduces the risk of security misconfigurations.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	It does not address directly to this domain.
7	Identification and Authentication Failures (A07-2021)	The Identity Aware Proxy improves the identification and authentication mechanisms of web applications. It performs authentication and verification of user permissions before allowing access to the application. By enforcing secure authentication mechanisms and strong password policies, it helps address vulnerabilities related to identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	It does not address directly to this domain.
9	Security Logging and Monitoring Failures (A09-2021)	The Identity Aware Proxy can contribute to logging and monitoring capabilities by capturing and analyzing access logs and traffic. It provides a centralized point for visibility into user authentication and authorization events, helping organizations detect and respond to security incidents. By ensuring effective logging and monitoring practices, it addresses vulnerabilities associated with security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	It does not address directly to this domain.

11.1.13 Environment segregation – Multitenancy design

Multitenancy is the logical segregation of resources in a single physical environment. It can be used for several purposes.

The first is the customer tenant scenario where it is used for collocating more than one external organization services deployed in the same physical environment.

The second one is the enterprise scenario, where it is used to create logical partitions in a physical environment to create several logical ones for one organization, for example, Development, Integration, and UAT.

Adopting the multitenancy framework for this project optimizes the costs of the many environments needed to reuse a single infrastructure cluster.

From the security viewpoint, the multitenancy model ensures total segregation between the logical environments and the precautionary measures to validate this hipotesis and implementation. The most important role of the multitenancy framework in terms of security is to avoid anyone without the required permissions to access resources of other tenants' environments.

So, the multitenancy architecture allows organizations can achieve greater flexibility, efficiency, and security in their shared infrastructure environments. At the same time, the tenants benefit from isolated and customizable environments. The administrators can efficiently manage and secure multiple tenants within a single physical environment.

From architecture perspective this is also known as "Environment Segregation” and these are some of the most important capabilities:

#	Name	Description
1	Resource Isolation and Performance	The multitenancy design can include advanced resource isolation mechanisms to ensure that each tenant's applications and data are fully segregated and do not impact the performance or security of other tenants. This can involve techniques such as containerization, virtualization, or resource quotas to allocate and manage resources effectively.
2	Customization and Configuration	The multitenancy framework can allow each tenant to have their own customizable environment settings, configurations, and policies. This flexibility enables tenants to tailor their environment according to their specific requirements, including application settings, security policies, and integration capabilities.
3	Tenant Management and Self-Service	An extended multitenancy design can include tenant management features, enabling tenants to manage their own users, roles, permissions, and resources within their environment. This self-service capability empowers tenants to have more control and autonomy over their environment while reducing administrative overhead.
4	Scalability and Elasticity	The multitenancy design can incorporate scalability and elasticity features, allowing resources to be dynamically allocated and scaled based on the needs of each tenant. This ensures that tenants have the necessary resources to support their workload demands without affecting other tenants.
5	Auditing and Monitoring	The multitenancy framework can include robust auditing and monitoring capabilities to track and log activities within each tenant's environment. This enables administrators and tenants to monitor and analyze resource usage, access patterns, and security events to ensure compliance, detect anomalies, and respond to incidents effectively.
6	Data Governance and Compliance	Extended multitenancy capabilities can address data governance and compliance requirements by providing mechanisms for data separation, encryption, access controls, and regulatory compliance features specific to each tenant's industry or geographic location. This ensures that each tenant's data is protected and managed in accordance with relevant regulations and policies.
7	Cross-Tenant Collaboration and Integration	The multitenancy design can facilitate controlled collaboration and integration between tenants when required. This can include secure communication channels, API gateways, and data exchange mechanisms that allow authorized interaction and data sharing between tenants while maintaining strict security and privacy boundaries.

11.1.13.1 Assessment – SAMM

The Environment segregation - Multitenancy design does not apply directly to SAMM security standards and its domains. While environment segregation through multitenancy can contribute to overall security by ensuring logical segregation between environments, it is not a specific requirement or recommendation outlined in the SAMM framework.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards Multitenancy design can support the implementation and enforcement of security policies and standards by providing a segregated environment for each organization or logical partition within an organization. This helps establish and maintain the expected security practices outlined in the Policies and Standards domain of SAMM. Compliance Management Multitenancy design can assist organizations in meeting regulatory requirements and industry standards by ensuring that each tenant's environment is isolated and compliant. It supports compliance management efforts within the SAMM framework. Security Roles and Responsibilities Multitenancy design helps define and assign security roles and responsibilities within an organization. It ensures that individuals have access only to the resources of their respective tenant's environment, aligning with the Security Roles and Responsibilities domain of SAMM.
2	SAMM Security by Design	Security Requirements Multitenancy design can contribute to addressing security requirements by providing a segregated environment for each tenant. This helps ensure that security concerns, such as access control and data protection, are considered and implemented appropriately, aligning with the Security Requirements domain of SAMM. Secure Architecture Multitenancy design can facilitate the implementation of a secure architecture by isolating each tenant's environment and enforcing security controls at the architectural level. It helps address various OWASP threats, such as Insecure Design and Security Misconfiguration, as outlined in the SAMM Design domain. Secure Coding Practices While multitenancy design does not directly influence coding practices, it provides a secure foundation for development teams to implement secure coding practices within their respective tenant's environment, aligning with the Secure Coding Practices sub-domain of SAMM.

#	Name	Description
3	SAMM Security by Implementation	Secure Architecture Multitenancy design contributes to implementing a secure architecture by providing logical segregation between tenants' environments. It helps address security concerns during the implementation phase, supporting the Secure Architecture sub-domain of SAMM. Security Requirements Multitenancy design assists in implementing security requirements by ensuring that each tenant's environment adheres to the specified security objectives. It helps align with the Security Requirements domain of SAMM. Security Testing Multitenancy design can facilitate security testing activities by providing separate environments for testing and identifying vulnerabilities. It supports the integration of security testing throughout the development lifecycle, as recommended in the Security Testing Integration sub-domain of SAMM.
4	SAMM Security by Verification	Security Testing Multitenancy design supports security testing activities by providing isolated environments for conducting various types of security tests, such as penetration testing and vulnerability scanning. It aligns with the Security Testing domain of SAMM. Security Architecture Review Multitenancy design enables security architecture reviews by providing a framework for evaluating the security controls and mechanisms implemented within each tenant's environment. It supports the Security Architecture Review sub-domain of SAMM. Threat Modeling Multitenancy design can assist in threat modeling exercises by considering the potential threats and risks specific to each tenant's environment. It aligns with the Threat Modeling sub-domain of SAMM.
5	SAMM Security by Operations	Environment Hardening Multitenancy design contributes to environment hardening by enforcing secure configurations and infrastructure practices within each tenant's environment. It supports the Environment Hardening component of SAMM's Operations domain. Secure Deployment Multitenancy design ensures secure deployment practices by isolating and controlling the deployment of software systems within each tenant's environment. It aligns with the Secure Deployment sub-domain of SAMM. Security Testing Multitenancy design facilitates security testing by providing separate environments for conducting testing activities, such as vulnerability scanning and penetration testing.

11.1.13.2 Assessment – OWASP Top 10

While multitenancy design, particularly environment segregation, can contribute to addressing some OWASP threats indirectly, it is not a comprehensive solution for all the threats.
Each threat requires specific security practices and measures to be implemented within the application itself, in addition to the architectural considerations provided by the multitenancy framework.

#	Name	Description
1	Broken Access Control (A01-2021)	Multitenancy design helps enforce access controls between different logical environments or organizations within a single physical environment, ensuring proper segregation of resources and permissions. It prevents unauthorized access to resources of other tenants' environments by enforcing strict separation and access restrictions.
2	Cryptographic Failures (A02-2021)	The multitenancy framework itself does not directly address cryptographic failures. However, by providing a secure and isolated environment, it allows organizations to implement proper cryptographic practices without interference from other tenants.
3	Injection (A03-2021)	Multitenancy design does not directly address injection vulnerabilities. This threat is more related to input validation and secure coding practices within the application itself.
4	Insecure Design (A04-2021)	Multitenancy design, when implemented securely, can contribute to addressing insecure design vulnerabilities by providing a structured and isolated environment for each tenant. It enables organizations to enforce security controls and considerations at the architectural level, reducing the chances of insecure design decisions.
5	Security Misconfiguration (A05-2021)	Multitenancy design can indirectly help mitigate security misconfigurations by providing a controlled and standardized environment for tenants. It allows organizations to enforce secure configurations and prevent unnecessary or insecure settings across the different logical environments.
6	Vulnerable and Outdated Components (A06-2021)	Multitenancy design itself does not directly address the use of vulnerable or outdated components within a web application. However, by promoting a structured environment, it can facilitate the identification and management of components used by different tenants, making it easier to track and update them.
7	Identification and Authentication Failures (A07-2021)	Multitenancy design does not directly address identification and authentication failures. These vulnerabilities are more related to the implementation of authentication mechanisms within the application itself.
8	Software and Data Integrity Failures (A08-2021)	Multitenancy design does not directly address software and data integrity failures. However, by providing isolated logical environments, it can contribute to maintaining the integrity of the data and configurations associated with each tenant.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Multitenancy design does not directly address logging and monitoring failures. However, it can facilitate the implementation of centralized logging and monitoring mechanisms for each tenant, making it easier to track and detect security incidents.
10	Server-Side Request Forgery (A10-2021)	Multitenancy design does not directly address server-side request forgery vulnerabilities. This threat is more related to the secure implementation of request handling and input validation within the application itself.

11.1.14 Regulatory Compliance

The following architecture capability is about Regulatory Compliance.

This capability will adderess PCI Compliance (Payment Card Industry) and PII (Persona Identifiable Information).

However, similar type of analysis can be conducted for other analagous regulatory compliances for other industries.

In general, the best advice to cicunvent PCI data compliance is to avoid it completely. For this there are specific design principles that need to be considered.

The overall intention is no credit card data should be stored in the system. Only surrogate keys or tokens should be used instead.

For this is important considering tokenisation strategies. See more information about tokenisation in "Tokenization guidance (Wikipedia)".

On this regards, similar type of considerations applies for any sensible data that could be used for fraud. For example, semi-permanent tokens must be used for setting up payment method, such as PayPal, Stripe, CashApp, and other payment wallets providers.

About PII compliance, the strategy is slightly different, since PII usually is required to identify individuals and avoid frauds perpetrations while conducting business.

Identifiable individual data refers to the data that can be used to identify univocally an individual.

This is applicable for all government-issued Ids. The overall strategy is to avoid storing these data sets readable to the naked eyes in the database. This could also be extended to the displaying of these datasets into user interfaces.

So, from the overall implementation perspective, these Ids can be tokenized, so these government-issued Ids should not be capable to be used to identify individuals in systems without applying a de-tokenisation.

The government-issued Ids extend to:

- Health records
- Passport
- Driver licence
- Veteran card id
- Welfare card id
- Student id
- etc

With the tokenisation, nobody, including database administrators, system operators, data engineers, should be able to see these Ids with the naked eye. Only authorized personnel should be able to see the data de-tokenised. Which involves an un-encryption process whch reverse the tokenisation.This can be implemented and accesible through an algorithm exposed as an API or a library which will be available for embedding this into applications. But this will be for exceptional scenarios. For example, when auditing is required (e.g. in a fraud investigation). Or when displaying the ids in the user interface to authorised users. The tokenised ids can be used as surrogate ids, replacing the original ids. Even in scenarios where it is needed to compare ids, the tokenised version can be used.
In general, data compliance involves following design principles and implementing additional measures to ensure the protection and proper handling of sensitive information. By incorporating these capabilities into a regulatory compliance framework, the organization can ensure the secure handling and protection of sensitive data, meet PCI requirements, and comply with regulations regarding non-identifiable individual data. These measures mitigate the risk of data breaches and unauthorized access to sensitive information, maintaining the confidentiality and privacy of individuals' data. Here are some capabilities required:

#	Name	Description
1	Tokenization of Credit Card Data	To achieve PCI compliance, it is essential to avoid storing credit card data in the system. Instead, the system should tokenize the credit card information by replacing it with surrogate keys or tokens. These tokens can be securely stored and used for necessary transactions, while the actual credit card data remains protected in a separate, compliant payment gateway or third-party service.
2	Payment Gateway Integration	Payment gateway accounts, such as PayPal or other providers, often offer tokenization capabilities themselves. When users set up a payment method, the system can obtain a semi-permanent token from the payment gateway. This token can be used for future transactions, reducing the need to store sensitive payment data locally.
3	Non-Identifiable Individual Data Handling	Compliance with regulations regarding non-identifiable individual data requires avoiding the storage of government-issued IDs in plain form. Instead, these IDs should be tokenized to ensure that even authorized personnel cannot view them in their original form. Tokenization algorithms can be used to generate unique tokens for each ID, enabling secure comparison or verification without exposing the actual IDs.
4	Extended Government-Issued IDs	In addition to government-issued IDs like passports, driver's licenses, and health records, it's important to identify and tokenize other sensitive information, such as veteran card IDs, welfare card IDs, or any other forms of identification that could potentially identify individuals.
5	Data Encryption and Access Controls	Alongside tokenization, sensitive data, including tokenized IDs, should be encrypted using strong encryption algorithms. Only authorized personnel should have access to the decryption process. Strict access controls and user permissions should be implemented to limit the number of individuals who can view decrypted data, ensuring that only those with a legitimate need can access it.

#	Name	Description
6	Exceptional Access for Auditing and Investigations	In certain exceptional scenarios, such as fraud investigations or auditing, there may be a need to unencrypt or reverse tokenized IDs. This access should be strictly controlled and granted only to authorized personnel for specific purposes. Detailed logs and audit trails should be maintained to track and monitor any access to decrypted data.

11.1.14.1 Assessment – SAMM

The architectural design components described contribute to addressing various aspects of SAMM security standards, primarily related to regulatory compliance, secure coding practices, and data protection. They indirectly contribute to addressing other SAMM domains by establishing a foundation for secure software development, verification, and operations.. Here is a breakdown of the contributions of the architectural design components to each SAMM domain:

#	Name	Description
6	SAMM Security by Governance	Regulatory Compliance The use of surrogate keys or tokens instead of storing credit card data helps address compliance with PCI Data standards. Tokenization of government-issued IDs (e.g., health records, passports, driver's licenses) contributes to compliance with non-identifiable individual data standards.
7	SAMM Security by Design	Security Requirements The architectural design component of tokenization contributes to addressing security requirements related to protecting sensitive information, such as personally identifiable information (PII) and government-issued IDs.
8	SAMM Security by Implementation	Secure Architecture The architectural design component of tokenization, along with secure coding practices, helps build a secure architecture that minimizes vulnerabilities associated with OWASP threats such as Insecure Design and Security Misconfiguration. Secure Coding Practices Adhering to secure coding practices, including the use of surrogate keys or tokens instead of sensitive data, helps mitigate OWASP threats like Injection and Insecure Design.
9	SAMM Security by Verification	Security Testing The architectural design components, such as tokenization and secure coding practices, contribute to reducing vulnerabilities and weaknesses that would be identified through security testing activities like penetration testing and code reviews.

#	Name	Description
10	SAMM Security by Operations	Environment Hardening The architectural design components, such as tokenization and secure coding practices, help establish a secure environment by reducing the exposure of sensitive data, minimizing the risk of vulnerabilities. Configuration Management The use of tokens instead of storing sensitive information simplifies configuration management by eliminating the need to manage and protect the storage of sensitive data. Secure Deployment The architectural design components, such as tokenization, contribute to secure deployment by ensuring that sensitive data is not exposed during the deployment process.

11.1.14.2 Assessment – OWASP Top 10

The capability of Regulatory Compliance, specifically focusing on PCI Data Compliance and Non-Identifiable individual data compliancy, addresses several OWASP threats. Here are the contributions of this capability in addressing each threat:

#	Name	Description
1	Broken Access Control (A01-2021)	By enforcing regulatory compliance, the capability ensures that proper access controls are implemented and enforced within the system, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	The capability emphasizes the importance of tokenization for sensitive data, such as credit card information and government-issued IDs. Tokenization helps protect the confidentiality and integrity of the data, mitigating the risk of cryptographic vulnerabilities.
3	Injection (A03-2021)	While the capability does not directly address injection vulnerabilities, it promotes the use of tokenization for data comparison. By tokenizing and encrypting IDs, the system reduces the likelihood of untrusted data manipulation and injection attacks.
4	Insecure Design (A04-2021)	The capability does not directly address insecure design vulnerabilities. However, by focusing on regulatory compliance, it encourages secure architectural decisions and the proper handling of sensitive data, which can contribute to reducing insecure design flaws.

#	Name	Description
5	Security Misconfiguration (A05-2021)	The capability indirectly addresses security misconfiguration by advocating for the appropriate handling of credit card data and government-issued IDs. Following regulatory compliance guidelines helps organizations avoid misconfigurations that may expose sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	The capability does not directly address vulnerable and outdated components. However, by promoting regulatory compliance, organizations are encouraged to implement secure practices, including using up-to-date and secure components, reducing the risk of vulnerabilities stemming from outdated software.
7	Identification and Authentication Failures (A07-2021)	The capability does not directly address identification and authentication failures. However, by emphasizing the use of tokenization and encrypted data, it indirectly contributes to the security of authentication mechanisms and reduces the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	The capability indirectly addresses software and data integrity failures by promoting secure practices, such as tokenization and encryption. By protecting sensitive data and ensuring its integrity, organizations mitigate the risk of unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	The capability does not directly address security logging and monitoring failures. However, by focusing on regulatory compliance, organizations are encouraged to implement comprehensive logging and monitoring mechanisms to meet compliance requirements, which can help in the detection and response to security incidents.
10	Server-Side Request Forgery (A10-2021)	The capability does not directly address server-side request forgery vulnerabilities. However, by promoting secure design principles and the protection of sensitive data, organizations can reduce the likelihood of SSRF attacks that may exploit internal resources.

11.1.15 Web Application Firewall (WAF) – ModSecurity (OWASP)

It is an industry-standard for protecting applications from code injection. The OWASP ModSecurity Core Rule Set is an open-source initiative of the Open Web Application Security Project® (OWASP).

The ModSecurity Core Rule Set (CRS) is an embedded component to the ModSecurity Core which consist in set of generic attack detection rules. The CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten, with a minimum of false alerts. The CRS covers many common attack categories, including SQL Injection, Cross-Site Scripting, Local File Inclusion, etc.

In the architecture, at least one instance of a WAF capability will be required to inspect all the incoming traffic. This capability can be plugged at different levels in the architecture, but it is advisable to avoid being the DNS main of the organisation.

The deployment options are the following:

- External Cloud-based SaaS Reverse Proxy

  E.g Cloudflare or Azure Frontdoor.
- After the Kubernetes Cluster Ingress controller

It is another layer of proxying. The traffic is redirected directly from the Ingress Controller, and the payloads are verified before the applications end-points.

- As a plugin of Kong API Gateway
  Kong in this case is playing the role of Kubernetes Cluster Ingress controller.

By implementing a WAF, the organization can enhance their protection against code injection and other web application vulnerabilities. If no SaaS WAF option is considered, then it requires integration with the OWASP ModSecurity Core Rule Set and regular maintenance of rule updates ensure that the WAF solution remains effective against emerging threats. Below, a summary of the

#	Name	Description
1	Architecture Integration Points	The WAF capability can be plugged into different levels, such as the Frontend Kubernetes Cluster Ingress controller or the Backend Kubernetes Cluster Ingress controller. The selection of the integration point should be based on the specific requirements and considerations of the project. Each integration point has its advantages and trade-offs, and it is important to evaluate them in terms of scalability, performance, and overall security posture.
2	Cloud-based WAF Solutions	Cloudflare Azure Frontdoor, is an external cloud-based SaaS (Software-as-a-Service) reverse proxy that provides WAF capabilities. Integrating a cloud-based WAF solution like Cloudflare can offer several benefits, including a global network infrastructure, automatic updates, and the ability to leverage their threat intelligence and rule sets. Evaluating the feasibility and compatibility of Cloudflare or similar services is essential for achieving robust security.
3	OWASP ModSecurity Core Rule Set	If considering a self-hosted solution, then the solution must consider the operations and support of the rules set. The OWASP ModSecurity Core Rule Set (CRS) is an open-source initiative that provides a set of attack detection rules for web application firewalls. These rules cover a wide range of common attack categories, including SQL Injection, Cross-Site Scripting (XSS), and Local File Inclusion. Integrating the CRS into the selected WAF solution, whether it's ModSecurity or NAXSI, enhances the protection against known web application vulnerabilities.
4	Regular Rule Updates and Maintenance	To ensure the effectiveness of the WAF solution, it is crucial to keep the rule sets up to date by regularly applying updates and security patches. Monitoring emerging threats and vulnerabilities is essential to promptly address new attack vectors and protect the web applications from evolving risks. This includes staying informed about the latest releases and patches provided by the WAF solution and the OWASP community.

11.1.15.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address directly this domain.
2	SAMM Security by Design	Security Requirements The use of a Web Application Firewall (WAF) such as ModSecurity helps address security requirements by providing a layer of protection against common web application vulnerabilities, including SQL Injection and Cross-Site Scripting (XSS). This contributes to mitigating OWASP threats related to Insecure Design and Injection. Secure Architecture Integrating a WAF into the architecture helps create a more secure architecture by providing additional security controls and protections. It helps address OWASP threats such as Insecure Design and Security Misconfiguration by adding an extra layer of defense to the application. Secure Coding Practices The use of ModSecurity and its rule set helps enforce secure coding practices by automatically detecting and blocking potential code injection attacks. This contributes to addressing OWASP threats related to Injection, Insecure Design, and Security Misconfiguration. Threat Modeling The inclusion of a WAF in the architecture enables organizations to consider potential threats and attack vectors during the design phase. By configuring the WAF to protect against known vulnerabilities and attack patterns, organizations can address OWASP threats more effectively. Security Testing: A WAF like ModSecurity can be used as a component of security testing activities during the design phase. It allows organizations to simulate attacks and evaluate the effectiveness of their security controls in mitigating OWASP threats.
3	SAMM Security by Implementation	It does not address directly this domain.
4	SAMM Security by Verification	Security Testing ModSecurity can be utilized as part of security testing activities to assess the effectiveness of security controls. It helps identify vulnerabilities and weaknesses in software applications, contributing to the verification of security measures and addressing OWASP threats. Code Review ModSecurity's rule set and configuration can be reviewed to ensure it aligns with secure coding practices and effectively mitigates OWASP threats related to Injection, Insecure Design, and Security Misconfiguration. Security Architecture Review The use of a WAF like ModSecurity requires reviewing the security architecture of software applications to ensure proper integration and alignment with the overall design. This helps address OWASP threats and verifies the presence of security controls.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening Integrating a WAF like ModSecurity contributes to environment hardening by adding an extra layer of protection against web application vulnerabilities. It helps address OWASP threats and contributes to the secure operation of software systems. Secure Build Including ModSecurity in the software build process ensures that the resulting software has an added layer of security. This contributes to addressing OWASP threats during the operational phase. Configuration Management The configuration of ModSecurity and its rule set requires proper management and documentation, contributing to effective configuration management practices and reducing the risk of misconfigurations that could lead to OWASP threats. Incident Management ModSecurity can assist in incident management by providing detailed logs and alerts when potential attacks are detected. This helps organizations respond to security incidents and mitigate OWASP threats promptly. Secure Deployment By including a WAF like ModSecurity in the deployment process, organizations can ensure that software applications are deployed securely. This contributes to addressing OWASP threats during the deployment phase.

11.1.15.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	The WAF protects against various access control-related vulnerabilities, such as insecure direct object references, insufficient authorization checks, and privilege escalation attempts. By enforcing proper access controls, ModSecurity helps prevent unauthorized access to restricted resources.
2	Cryptographic Failures (A02-2021)	It does not address directly this domain.
3	Injection (A03-2021)	The WAF's rules-set includes specific rules to detect and prevent injection attacks, such as SQL injection and cross-site scripting (XSS). It validates and sanitizes user input, ensuring that untrusted data is not used in a way that can be exploited to execute unintended commands or manipulate the application's execution flow.

#	Name	Description
4	Insecure Design (A04-2021)	The WAF itself does not directly address insecure design vulnerabilities, it provides a layer of protection against common security weaknesses introduced during the design phase. By implementing the WAF with the appropriate rule set, organizations can mitigate some risks associated with insecure design decisions.
5	Security Misconfiguration (A05-2021)	The WAF can help mitigate security misconfigurations by providing a set of default rules and configurations designed to protect web applications from common vulnerabilities. By implementing the WAF and keeping it up to date, organizations can reduce the risk of misconfigurations that could lead to unauthorized access or exposure of sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	The WAF does not directly address the use of vulnerable or outdated components within web applications. It can provide an additional layer of defense against attacks targeting known vulnerabilities in third-party software. The WAF's rule set can help detect and prevent exploitation attempts targeting vulnerable components.
7	Identification and Authentication Failures (A07-2021)	The WAF's rule set includes protections against common authentication-related vulnerabilities, such as brute force attacks, session fixation, and credential stuffing. By implementing WAF, the organization can enhance the security of their authentication mechanisms and reduce the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	The WAF itself does not directly address security logging and monitoring failures. However, it can generate logs and alerts when suspicious or malicious activities are detected. These logs can be integrated with existing logging and monitoring systems to enhance the overall visibility and detection capabilities of the application.
9	Security Logging and Monitoring Failures (A09-2021)	It does not address directly this domain.
10	Server-Side Request Forgery (A10-2021)	It does not address directly this domain.

11.1.16 Auditing – Non-repudiation

The capability of auditablity of a this system is considered a non-functional requirement. It consist in creating the underlying information systems and the mechanisms to audit the use of a system with the intention of proving that no records have been modified or added externally.

This also is used as evidence of a non-repudiable origin (univocally prove the origin of the data). The audit traces can be created from the business events, and all business events must have all the attributes proposed for auditing.

The implementation strategy consist in incorporating digital signatures, implementing an immutable audit trail, utilizing timestamping services, enforcing access controls and user authentication, implementing a comprehensive auditing framework, and ensuring secure storage and monitoring of audit logs, the system can achieve robust auditing capabilities with non-repudiation. These extended capabilities provide assurance that records cannot be modified or added externally and can be traced back to their origin, enhancing the overall integrity and reliability of the system's audit trails.

#	Name	Description
1	Digital Signatures	Digital signatures can be used to provide non-repudiation by proving the origin and integrity of data. Each business event or record can be digitally signed using cryptographic techniques. The digital signature can be generated using the private key of the signer and verified using their public key. By including digital signatures in the auditing process, it becomes possible to prove the authenticity of records and trace their origin.
2	Immutable Audit Trail	Implementing an immutable audit trail ensures that once a record is stored, it cannot be modified or tampered with. This can be achieved through technologies like blockchain or distributed ledger systems, where each transaction or event is cryptographically linked to the previous one, creating an immutable chain of records. With an immutable audit trail, any modifications or additions to the records can be easily detected and proven, ensuring non-repudiation.
3	Timestamping	Adding timestamps to business events and records can further enhance non-repudiation. A trusted timestamp authority can provide timestamping services, which digitally sign and timestamp each event or record, establishing a clear chronological order. Timestamping enables the detection of any tampering attempts and provides evidence of the exact time when an event occurred, adding to the non-repudiation capabilities of the system.
4	Access Controls and User Authentication	To ensure the non-repudiation of records, it is essential to enforce strict access controls and user authentication mechanisms. Each user should have a unique identifier, and their actions within the system should be linked to their identity. By logging and associating user actions with their identities, it becomes easier to track and attribute any modifications or additions to the records, strengthening non-repudiation.
5	Auditing Framework	Implementing a comprehensive auditing framework that captures all relevant attributes and events is crucial. The framework should define what events need to be audited, including both system-level events (e.g., configuration changes, access attempts) and business-level events (e.g., financial transactions, data modifications). It should also ensure that all required attributes are captured and recorded for each event, enabling thorough audit trails and non-repudiation.
6	Secure Storage and Monitoring	The audit records should be securely stored to prevent unauthorized access or tampering. Implementing strong encryption and access controls for the storage of audit logs adds an extra layer of protection. Additionally, continuous monitoring of the audit logs for any suspicious activities or anomalies can help identify potential security breaches or attempts to modify records, ensuring the integrity and non-repudiation of the system.

11.1.16.1 Assessment – SAMM

The capability of Auditing - Non-repudiation applies directly to the Security Practice: Architecture Component domain in SAMM. It indirectly contributes to other domains by supporting the respective activities and objectives.

#	Name	Description
1	SAMM Security by Governance	Security Metrics and Reporting Contribution: Implementing non-repudiation through auditing enables the generation of accurate and reliable security metrics and reporting. By ensuring that records cannot be modified or added externally and can be traced back to their origin, the integrity and reliability of the audit trails are enhanced, providing a solid basis for security metrics and reporting.
2	SAMM Security by Design	Security Requirements Contribution: Incorporating non-repudiation through auditing as a security requirement helps address the design aspect of software security. By defining the need for auditing mechanisms that provide non-repudiation, organizations can proactively mitigate vulnerabilities associated with unauthorized record modifications or additions.
3	SAMM Security by Implementation	Security Verification Contribution: Implementing auditing with non-repudiation capabilities contributes to the security verification aspect of software implementation. By conducting security testing activities, such as code reviews and vulnerability assessments, organizations can verify that the auditing mechanisms effectively prevent non-repudiation issues, ensuring the delivery of secure and resilient software products.
4	SAMM Security by Verification	Code Review Contribution: Conducting code reviews to identify security flaws and adherence to secure coding practices includes assessing the implementation of auditing with non-repudiation capabilities. This helps verify that the code properly incorporates mechanisms to prevent unauthorized modifications or additions to records, enhancing non-repudiation.

#	Name	Description
5	SAMM Security by Operations	Incident Management Contribution: In the context of incident management, auditing with non-repudiation capabilities becomes crucial. By ensuring that audit logs cannot be tampered with and providing evidence of the origin of data, non-repudiation contributes to the accurate investigation, response, containment, and recovery of security incidents. Note: The capability of Auditing - Non-repudiation applies directly to the Security Practice: Architecture Component domain in SAMM. It indirectly contributes to other domains by supporting the respective activities and objectives.

11.1.16.2 Assessment – OWASP Top 10

The Auditing – Non-repudiation capability may not directly address all OWASP threats, however, it provides an essential features for capturing audit traces and ensuring the integrity, accountability, and traceability of data and activities. This, in turn, contributes to enhancing the security posture and reducing the risks associated with various OWASP threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Auditing with non-repudiation helps in verifying and enforcing access controls within a web application. By having auditable records that prove no external modifications or additions have occurred, it ensures the integrity of access controls and reduces the risk of unauthorized access.
2	Cryptographic Failures (A02-2021)	While Auditing – Non-repudiation does not directly address cryptographic failures, it can contribute indirectly by providing evidence of the origin of data. This can help in ensuring the integrity and authenticity of cryptographic operations and preventing tampering.
3	Injection (A03-2021)	Auditing with non-repudiation contributes to addressing injection vulnerabilities by providing traceability and accountability. By capturing and auditing business events, organizations can track the flow of data and identify potential injection attacks or unauthorized manipulations.
4	Insecure Design (A04-2021)	Auditing with non-repudiation does not directly address insecure design vulnerabilities. However, it can provide evidence of design decisions and their impact on the system. This evidence can help in identifying insecure design practices and implementing appropriate security controls.

#	Name	Description
5	Security Misconfiguration (A05-2021)	Auditing with non-repudiation does not directly address security misconfiguration vulnerabilities. However, it can help in detecting and identifying misconfigurations by capturing and analyzing the audit traces. This can enable organizations to rectify misconfigurations and improve the overall security posture.
6	Vulnerable and Outdated Components (A06-2021)	Auditing with non-repudiation does not directly address vulnerable and outdated components. However, by capturing audit traces, organizations can identify the usage of outdated or vulnerable components and take appropriate actions such as patching or replacing them.
7	Identification and Authentication Failures (A07-2021)	Auditing with non-repudiation contributes to addressing identification and authentication failures by providing evidence of user activities and their associated authentication. This can help in detecting unauthorized access attempts, impersonation attacks, or weaknesses in the authentication mechanisms.
8	Software and Data Integrity Failures (A08-2021)	Auditing with non-repudiation directly addresses software and data integrity failures by ensuring the integrity of audit records. By proving that no records have been externally modified or tampered with, it establishes the trustworthiness and integrity of the software and data.
9	Security Logging and Monitoring Failures (A09-2021)	Auditing with non-repudiation directly addresses security logging and monitoring failures by providing comprehensive logging and traceability. By capturing audit traces and retaining them securely, organizations can analyze the logs, detect security incidents, and respond timely to mitigate risks.
10	Server-Side Request Forgery (A10-2021)	Auditing with non-repudiation does not directly address server-side request forgery vulnerabilities. However, by capturing and auditing request-related information, organizations can identify potential SSRF attacks and investigate their origin.

11.1.17 Business Continuity

Introduction

From this Threat Assessment perspective, the Business Continuity is considered a type of security risk for the Business.

In terms of solutions proposed, in all cases must meet the SLA requirements, usually expressed as non-functional requirements. For example, it must be considered the following metrics:

Service Level Agreements (SLAs)

The minimum levels of performance, availability, and other metrics that customers can expect when using the provider's solution

Service level objectives (SLOs)

Internal goals that the provider wants to reach to meet those SLAs

Service level indicators: The provider uses actual metrics to measure how close it is to meeting both SLOs and SLAs.

Recovery Time Objectives (RTOs)
It is the goal for the maximum length of time it should take to restore normal operations following an outage or data loss.

Recovery Point Objectives (RPOs)
The maximum amount of data the organization can tolerate losing.

Environments
Regarding environments and business continuity, the solution provided to the Customer must consider Disaster Recovery sites and Disaster Recovery Plans that operationalize transitioning to the DR environment.

Operability
The solution must also consider the operations and services that allow diagnosis and troubleshooting of unexpected situations. For example, Monitoring, Service Support and Backup and restore.

The "Business Continuity" as a capability involves implementing various measures and processes to ensure the system's availability, resilience, and ability to recover from disruptions. For this it is necessary to incorporate high availability principles in the architecture, establishing disaster recovery sites and plans, implementing monitoring and alerting systems, setting up service support and incident management processes. In addition take care of the data by ensuring robust backup and restore procedures, with these measuress, the systems can enhance its business continuity capabilities.
These extended capabilities provide the means to meet SLAs, SLOs, RTOs, and RPOs requirements, minimize downtime, and recover quickly from disruptions, ensuring the continuity of critical business operations.

#	Name	Description
1	High Availability Architecture	Designing a high availability architecture ensures that the system remains operational even in the event of hardware failures, network issues, or other disruptions. This can be achieved through redundant components, such as load balancers, replicated databases, and multiple instances of critical services. By distributing the workload across multiple servers or data centers, the system can continue functioning seamlessly even if one or more components fail.

#	Name	Description
2	Disaster Recovery Sites	Establishing disaster recovery sites is crucial for business continuity. These sites serve as backup locations where the system can be quickly restored and brought back online in the event of a catastrophic failure or major outage at the primary site. The disaster recovery sites should be geographically separate from the primary site to minimize the risk of simultaneous disruptions. Regular backups and replication of data to the disaster recovery site ensure that data can be restored and business operations can resume promptly.
3	Disaster Recovery Plans	Developing comprehensive disaster recovery plans is essential to operationalize the transition to the disaster recovery environment. These plans outline the steps and procedures to be followed in the event of a disaster, including the roles and responsibilities of the recovery team, the sequence of actions to be taken, and the communication protocols. Regular testing and updating of the disaster recovery plans ensure their effectiveness and readiness when needed.
4	Monitoring and Alerting	Implementing robust monitoring and alerting systems enables proactive detection of issues and timely response. Monitoring tools can continuously monitor the system's health, performance, and availability, generating alerts or notifications when predefined thresholds or anomalies are detected. This allows the operations team to quickly identify and address any potential issues, minimizing downtime and ensuring business continuity.
5	Backup and Restore	Implementing regular backup and restore procedures is crucial for data protection and recovery. Automated backup processes should be in place to ensure that critical data is regularly backed up and securely stored. These backups should be tested periodically to verify their integrity and the ability to restore data effectively. Having well-documented and tested restore procedures ensures that data can be recovered to a known and usable state in the event of data loss or corruption.
6	Service Support and Incident Management	Establishing a service support framework and incident management processes enables efficient diagnosis and troubleshooting of unexpected situations. This includes the availability of dedicated support personnel who can respond to incidents, analyze the root causes, and initiate appropriate remedial actions. Well-defined incident management processes, such as incident categorization, prioritization, and escalation, ensure a systematic approach to resolving issues and minimizing their impact on business operations.

11.1.17.1 Assessment – SAMM

Business Continuity is about ensure the system's availability, resilience, and ability to recover from disruptions. Implementing measures such as high availability architecture, disaster recovery sites, disaster recovery plans, monitoring and alerting, service support and incident management, and backup and restore procedures enhances the system's business continuity capabilities. These measures ensure that the

system can meet SLA requirements, minimize downtime, and recover quickly from disruptions, thereby ensuring the continuity of critical business operations.

#	Name	Description
1	SAMM Security by Governance	Risk Management Identify and assess risks associated with software development and deployment, to help the organization to identify and mitigate risks related to service disruptions, data loss, and system unavailability. By considering recovery time objectives (RTO) and recovery point objectives (RPO), organizations can assess the impact of potential disruptions and develop risk mitigation strategies. Compliance Management Ensure compliance with relevant regulatory requirements, industry standards, and best practices related to software security to align with regulatory requirements and industry best practices that often mandate organizations to have plans in place to ensure the availability and continuity of critical systems and services.
2	SAMM Security by Design	Security Requirements Define and incorporate security requirements into the design phase, such as RTO and RPO, ensuring that the system's design considers the necessary measures for resilience and recovery.
3	SAMM Security by Implementation	Secure Architecture Promote the adoption of secure architectural patterns and principles, such as high availability architecture and disaster recovery sites, to ensure that these are integral to designing a secure architecture that ensures the system's availability and resilience in the face of disruptions.
4	SAMM Security by Verification	Security Verification Verify the effectiveness of security controls implemented in the software system, ensures that the system's resilience and recovery capabilities are adequately validated and aligned with the desired security objectives. Security Testing Conduct various types of security testing to identify vulnerabilities and weaknesses, to assess the system's resilience and recovery capabilities, therefore validating the effectiveness of business continuity measures.
5	SAMM Security by Operations	Environment Hardening Implement secure configurations and infrastructure to protect the software system, such as backup and restore procedures, contribute to the overall security and resilience of the software system's environment. Incident Management Establish incident detection, response, containment, and recovery processes, such as service support and incident management, ensure that organizations have the necessary processes and capabilities to address disruptions and restore normal operations promptly.

11.1.17.2 Assessment – OWASP Top 10

The primary goal of Business Continuity as a capabiltiy focus on maintaining operations, its implementation can indirectly contribute to addressing various OWASP threats by considering security measures and ensuring the availability and integrity of critical systems and services, in the event of disruptions or disasters. However, by implementing robust business continuity practices, organizations can indirectly mitigate the impact of various OWASP threats by minimizing downtime, data loss, and unauthorized access. The Business Continuity capability addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Business Continuity planning includes identifying critical assets and their dependencies, which can help in prioritizing access control measures. By ensuring the availability of access control mechanisms during business continuity scenarios, organizations can minimize the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Business Continuity planning should consider the integrity and availability of cryptographic components and key management systems. By including cryptographic infrastructure in disaster recovery plans and ensuring their resilience, organizations can prevent cryptographic failures and maintain the security of sensitive data.
3	Injection (A03-2021)	Business Continuity planning can involve monitoring and auditing mechanisms that detect unusual behavior, such as injection attempts. By incorporating such mechanisms into the continuity strategy, organizations can identify and respond to injection attacks effectively.
4	Insecure Design (A04-2021)	While Business Continuity does not directly address insecure design, it provides an opportunity to reassess and improve the security of the overall architecture during the planning and implementation of continuity measures. By considering security best practices and secure design principles, organizations can mitigate insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Business Continuity planning includes documenting and maintaining configuration information for critical systems and services. This documentation can help identify and address security misconfigurations during recovery and restoration processes, reducing the risk of misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Business Continuity planning involves identifying critical dependencies, including software components and libraries. By including vulnerability management and patching processes in the continuity strategy, organizations can address the risks associated with vulnerable and outdated components.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Business Continuity planning should consider the availability and resilience of identification and authentication systems. By ensuring the continuity of these mechanisms during disruptions, organizations can reduce the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	Business Continuity planning includes data backup and restoration processes. By verifying the integrity of backed-up data and implementing secure data storage practices, organizations can minimize the risk of software and data integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Business Continuity planning can incorporate logging and monitoring mechanisms to detect and respond to security incidents during disruptions. By ensuring the availability and effectiveness of these mechanisms, organizations can address security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	Business Continuity planning can involve secure configurations that restrict server-side requests to trusted resources. By implementing robust access controls and validating user input, organizations can mitigate the risk of server-side request forgery vulnerabilities during continuity scenarios.

11.1.18 Operations - Monitoring

Introduction

The architecture that implements monitoring, usually address this from two different viewpoints. Top-down and Bottom-up.

Bottom-up monitoring approach

When using Kubernetes, all the workloads and platforms have been implemented with internal probes for self-healing the infrastructure and platforms.

Top-down monitoring approach

In addition, it is also advidable to any platform to be instrumented top-down to verify the availability of services that the Customer could experience through the user interface. This type of monitoring is Synthetic Monitoring and mimics the user behaviour through the User interfaces at an interval basis using scripting bots that can interact with the 'Applications' user interfaces as a real human being would.

Implementing a bottom-up and top-down approach for monitoring allows covering a large set of scenarios that otherwise are is not possible. In bottom-up monitoring with enhanced probes, performance monitoring, and log monitoring, and implementing top-down monitoring through synthetic monitoring, user experience monitoring, and application performance monitoring, the system can achieve comprehensive visibility into its internal health, performance, and user-facing services. These extended capabilities enable proactive monitoring, rapid issue detection, and efficient troubleshooting, leading to improved system reliability, performance, and user satisfaction.

#	Name	Description
1	Bottom-up Monitoring	Enhanced Probes Building on the Kubernetes-defined standards for implementing probes (liveness, readiness, startup), additional custom probes can be implemented to monitor specific components or services within the infrastructure and platforms. These probes can provide detailed insights into the internal health and status of individual workloads, containers, or services, allowing for proactive self-healing and automatic recovery mechanisms
2	Bottom-up Monitoring	User Experience Monitoring In addition to synthetic monitoring, real user monitoring (RUM) techniques can be employed to capture and analyze actual user interactions with the system. This involves collecting data from client-side scripts or JavaScript tags embedded in web pages to capture performance metrics, such as page load times, rendering times, and user interactions. This data provides insights into the actual user experience and helps identify areas for improvement.
3	Bottom-up Monitoring	Log Monitoring Log aggregation and analysis tools can be employed to collect and analyze logs generated by different components of the platform. This enables the detection of errors, anomalies, and security events, providing valuable insights for troubleshooting, debugging, and identifying potential security threats.
4	Top-down Monitoring	Synthetic Monitoring Synthetic monitoring involves simulating user interactions with the system to monitor the availability and performance of services as experienced by end-users. Scripting bots or synthetic transactions can be created to mimic user behavior, accessing the user interfaces of applications at regular intervals. This allows for the measurement of response times, functional correctness, and user experience metrics. Tools like Selenium, JMeter, or commercial synthetic monitoring services can be utilized to perform these synthetic tests.
5	Top-down Monitoring	User Experience Monitoring In addition to synthetic monitoring, real user monitoring (RUM) techniques can be employed to capture and analyze actual user interactions with the system. This involves collecting data from client-side scripts or JavaScript tags embedded in web pages to capture performance metrics, such as page load times, rendering times, and user interactions. This data provides insights into the actual user experience and helps identify areas for improvement.
6	Top-down Monitoring	Application Performance Monitoring (APM) APM tools can be employed to monitor the performance of individual applications, including tracking transaction times, database queries, external service integrations, and code-level performance metrics. APM tools provide detailed insights into application performance bottlenecks, exceptions, and potential optimizations, allowing for proactive identification and resolution of issues.

11.1.18.1 Assessment – SAMM

Operations Monitoring does not directly address the SAMM security domains and their sub-domains. While the capability of monitoring is essential for overall security and risk management, it is not explicitly aligned with any specific SAMM security domain. Instead, monitoring is a cross-cutting practice that supports various aspects of software security and helps organizations gain visibility into the health, performance, and security of their software systems.

While monitoring is not explicitly tied to any single SAMM domain, it is an essential practice that spans multiple domains and contributes to the overall security posture of the software development and operational processes.

#	Name	Description
1	SAMM Security by Governance	Security Metrics and Reporting Monitoring provides valuable data and metrics that can be used to assess the effectiveness of software security practices. By collecting and analyzing security-related metrics, organizations can track their security posture and report on key security performance indicators.
2	SAMM Security by Design	Threat Modeling Monitoring can provide data that helps validate and refine the threat model used during the design phase. By monitoring for actual security events and incidents, organizations can gain insights into potential threats and vulnerabilities not previously identified.
3	SAMM Security by Implementation	Security Testing Integration Monitoring is crucial for tracking the results of security testing activities, such as penetration testing or dynamic application security testing. It helps organizations assess the effectiveness of their security controls and identify areas for improvement.
4	SAMM Security by Verification	Security Testing Monitoring supports security testing activities by providing real-time data on security events, such as successful attacks or unauthorized access attempts. This data helps validate the effectiveness of security controls and identifies potential weaknesses.
5	SAMM Security by Operations	Incident Management Monitoring plays a critical role in incident management by providing real-time alerts on security incidents. This enables organizations to respond quickly to security breaches and mitigate their impact.

11.1.18.2 Assessment – OWASP Top 10

It's important to note that while the Architecture Design Component's Operations - Monitoring capability contributes to addressing certain OWASP threats, it is not a comprehensive solution. Additional security measures, such as secure coding practices, input validation, access control enforcement, and secure configuration management, are necessary to provide a holistic approach to web application security.

The Architecture Design Component capability of Operations - Monitoring addresses several OWASP threats as follows:

#	Name	Description
1	Broken Access Control (A01-2021)	The top-down monitoring implemented in the platform helps verify the availability of services that the Customer could experience through the user interface. By continuously monitoring the user interfaces, the platform can detect and prevent unauthorized access attempts or privilege escalation, which are key aspects of broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	The Operations - Monitoring capability does not directly address cryptographic failures. Cryptographic failures are primarily related to the implementation and management of cryptographic algorithms and mechanisms, whereas the monitoring capability focuses on detecting and responding to security incidents rather than directly influencing the cryptographic aspects.
3	Injection (A03-2021)	The Operations - Monitoring capability does not directly address injection vulnerabilities. Injection vulnerabilities primarily involve improper handling of untrusted data and can be mitigated through proper input validation, secure coding practices, and security testing, which are different aspects from monitoring.
4	Insecure Design (A04-2021)	The Operations - Monitoring capability does not directly address insecure design vulnerabilities. Insecure design vulnerabilities relate to flaws and weaknesses in the overall design and architecture of a web application, while monitoring focuses on detecting security incidents and ensuring the availability and performance of services.
5	Security Misconfiguration (A05-2021)	The bottom-up monitoring implemented in the platform, using probes for self-healing infrastructure and platforms, can help detect security misconfigurations. By continuously monitoring the workloads and platforms, the platform can identify and remediate misconfigurations that may expose sensitive information or create potential entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	The bottom-up monitoring implemented in the platform can contribute to addressing vulnerabilities related to vulnerable and outdated components. By actively monitoring the workloads and platforms, the platform can detect outdated or insecure versions of third-party software components and trigger remediation actions, such as updating software dependencies or applying security patches.
7	Identification and Authentication Failures (A07-2021)	The top-down monitoring implemented in the platform, through synthetic monitoring, helps verify the availability of services that users interact with through the user interface. This type of monitoring can help detect and prevent identification and authentication failures by mimicking user behavior and detecting any anomalies or issues in the authentication process.
8	Software and Data Integrity Failures (A08-2021)	The Operations - Monitoring capability does not directly address software and data integrity failures. Software and data integrity failures relate to unauthorized modifications, manipulation, or destruction of software components, configurations, or data. While monitoring can detect security incidents that may indicate integrity issues, ensuring data integrity requires other security measures such as secure coding techniques and data protection mechanisms.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	The Operations - Monitoring capability directly addresses security logging and monitoring failures. By implementing comprehensive logging and real-time monitoring mechanisms, the platform can enhance its ability to detect, analyze, and respond to security incidents. Proper logging and monitoring practices are crucial for timely incident detection and effective incident response.
10	Server-Side Request Forgery (A10-2021)	The top-down monitoring implemented in the platform, through synthetic monitoring that mimics user behavior, can help detect server-side request forgery (SSRF) vulnerabilities. By monitoring the interactions with external and internal resources, the platform can identify unauthorized requests made on behalf of the server and implement appropriate controls to prevent SSRF attacks.

11.1.19 Operations - Service Support

All systems new systems need to have a proper service support plan.
In general the systems and platforms do not have a dedicated service support platform, but leverage a single instance adopted by the organsiation. Therefore, all teams supporting new systems can be on-boarded to any existent service support platform.
About the Service Request Management platform, it may require customizing it to align with specific requirements, integrating it with other systems, and leveraging incident, change, problem management, and reporting capabilities, the support team can efficiently handle service requests, incidents, and changes. This ensures timely response, effective problem resolution, and continuous improvement of service support processes.

#	Name	Description
1	Fit-for-purpose	It is assume that the Service Request Management Platform platform counts with features such as, scalability, integration capabilities, ease of use, cost-effectiveness, and vendor support. Also that th platform aligns with the organization's requirements and provide robust capabilities for managing service requests, incident tracking, problem management, and change management.
2	Configurability	The selected Service Request Management platform should offer customization and configuration options to align with the organization's specific workflows, processes, and terminology. This allows the support team to tailor the platform to their requirements and establish consistent and efficient service support practices.
3	Integration Capabilities	The platform should have integration capabilities with other tools and systems used within the organization's IT landscape. This enables seamless data exchange, automates workflows, and ensures a unified view of incidents, changes, and service requests across different systems. Integration with monitoring tools, asset management systems, and communication channels (e.g., email, chat) can improve efficiency and streamline the support processes.

#	Name	Description
4	Incident Management	Ticketing and Escalation The Service Request Management platform should provide ticketing capabilities for efficient incident tracking and management. It should support the creation, assignment, and tracking of incidents, allowing support team members to easily collaborate, escalate, and prioritize incidents based on their impact and urgency. Automated notifications and alerts can be configured to ensure timely response and resolution of incidents.
5	Incident Management	Knowledge Base The platform should include a centralized knowledge base where support team members can document known issues, solutions, workarounds, and best practices. This knowledge base can serve as a valuable resource for quick incident resolution, reduce the need for repetitive troubleshooting, and improve the overall support efficiency. The knowledge base should support search functionality, categorization, and versioning to ensure accurate and up-to-date information.
6	Change and Problem Management	Change Request Handling The Service Request Management platform should support the handling of change requests, including change submission, assessment, approval, scheduling, and implementation. It should provide workflow automation, change impact analysis, and change tracking capabilities to ensure controlled and coordinated changes in the environment.
7	Change and Problem Management	Problem Management The platform should enable the recording and tracking of problem records, allowing the support team to identify and address the root causes of recurring incidents. It should support problem investigation, root cause analysis, and the creation of problem records linked to relevant incidents. Collaboration features, such as discussion threads and knowledge base integration, can facilitate effective problem resolution.
8	Reporting and Analytics	Performance Metrics The Service Request Management platform should provide comprehensive reporting and analytics capabilities to measure and monitor service support performance. It should offer pre-defined and customizable reports to track key performance indicators (KPIs), such as incident response times, resolution times, customer satisfaction ratings, and support team workload. These metrics can help identify areas for improvement, monitor SLA compliance, and drive continuous service improvement initiatives.
9	Reporting and Analytics	Trend Analysis The platform should enable trend analysis by aggregating and analyzing support data over time. This allows the support team to identify recurring patterns, emerging issues, and potential areas of risk. Trend analysis can also provide insights into the effectiveness of support processes, resource allocation, and the impact of changes and improvements implemented in the environment.

11.1.19.1 Assessment – SAMM

While the specific contributions of the Service Request Management platform may depend on its capabilities and features, these are some potential ways it can align with the activities in the SAMM Security by Operations domain.

#	Name	Description
1	SAMM Security by Governance	Does Not Apply Directly.
2	SAMM Security by Design	Does Not Apply Directly.
3	SAMM Security by Implementation	Does Not Apply Directly.
4	SAMM Security by Verification	Does Not Apply Directly.
5	SAMM Security by Operations	Environment Hardening The platform can support the implementation of secure configurations and infrastructure configurations to ensure the software environment is hardened against known vulnerabilities and security weaknesses. Configuration Management The platform can help with version control, change management, and documentation of software configurations, enabling effective configuration management practices. Incident Management The platform can facilitate incident detection, response, containment, and recovery activities, supporting a well-defined incident management process for handling security incidents. Secure Deployment The platform can assist in secure software distribution, installation procedures, and configuration of software components, ensuring that software is deployed securely and reducing the risk of unauthorized access or compromise during deployment. Operational Enablement The platform can provide operational support and guidance by documenting operational procedures, offering security training for operational staff, and facilitating the establishment of incident response capabilities.

11.1.19.2 Assessment – OWASP Top 10

The Operations - Service Support capability within the Architecture Design Component does not directly address the specific OWASP threats mentioned.

#	Name	Description
1	Broken Access Control (A01-2021)	The Operations - Service Support capability, which involves using a Service Request Management platform, does not directly address broken access control vulnerabilities. It focuses on providing service support rather than enforcing access controls within the web application itself.
2	Cryptographic Failures (A02-2021)	The Operations - Service Support capability does not directly address cryptographic failures. It primarily focuses on managing service requests and providing support, rather than implementing or managing cryptographic algorithms and mechanisms.
3	Injection (A03-2021)	The Operations - Service Support capability does not directly address injection vulnerabilities. It focuses on managing service requests and providing support, while injection vulnerabilities are related to improper handling of untrusted data within the application.
4	Insecure Design (A04-2021)	The Operations - Service Support capability does not directly address insecure design vulnerabilities. It primarily involves using a Service Request Management platform for support activities, whereas insecure design vulnerabilities relate to flaws in the overall design and architecture of the web application.
5	Security Misconfiguration (A05-2021)	The Operations - Service Support capability does not directly address security misconfiguration vulnerabilities. It focuses on providing service support rather than configuring and securing the web application itself.
6	Vulnerable and Outdated Components (A06-2021)	The Operations - Service Support capability does not directly address vulnerabilities related to vulnerable and outdated components. It primarily involves managing service requests and providing support, rather than actively managing software dependencies or addressing specific vulnerabilities in components.
7	Identification and Authentication Failures (A07-2021)	The Operations - Service Support capability does not directly address identification and authentication failures. It focuses on managing service requests and providing support, while authentication-related vulnerabilities require specific measures within the web application itself.
8	Software and Data Integrity Failures (A08-2021)	The Operations - Service Support capability does not directly address software and data integrity failures. It primarily involves using a Service Request Management platform for support activities, while integrity-related vulnerabilities require measures such as secure coding techniques and data protection mechanisms within the application.
9	Security Logging and Monitoring Failures (A09-2021)	The Operations - Service Support capability does not directly address security logging and monitoring failures. It primarily focuses on managing service requests and providing support, while logging and monitoring vulnerabilities require dedicated logging and monitoring mechanisms within the web application.
10	Server-Side Request Forgery (A10-2021)	The Operations - Service Support capability does not directly address server-side request forgery vulnerabilities. It primarily involves using a Service Request Management platform for support activities, while addressing SSRF vulnerabilities requires implementing robust input validation and secure configurations within the web application.

11.1.20 Operations - Backup and Restore

It is required that all system has a Backup and Restore strategy.
For this, it is required that the system counts with procedures to back up the data on a time interval basis and procedures to recover the data in the case of unfortunate data lost from those backups.
For example, considering that the project has the scope of building a complete Disaster Recovery, it could be convenient to use the proposed data replication scheme from the Production environment to the DR environment for Backup and Restore.

Extending the capability of "Operations - Backup and Restore" involves enhancing the backup and restore procedures to ensure the protection and recoverability of data. Here are some extended capabilities:

By implementing a robust backup and restore strategy, leveraging data replication for backups, defining restore procedures, and monitoring and auditing the backup process, the system can ensure the availability and recoverability of data, minimizing the impact of data loss or system failures.

#	Name	Description
1	Backup Strategy	Data Backup Frequency Define the appropriate backup frequency based on the criticality of the data and the Recovery Point Objective (RPO). This determines how often backups are taken to ensure minimal data loss in the event of a failure. The backup frequency can vary for different types of data or databases within the system.
2	Backup Strategy	Incremental and Full Backups Implement a backup strategy that combines incremental and full backups. Incremental backups capture only the changes made since the last backup, reducing the backup window and storage requirements. Full backups capture all data, providing a baseline for recovery. A combination of both ensures a balance between backup efficiency and restore capabilities.
3	Backup Strategy	Backup Retention Define the retention period for backups, considering compliance requirements, business needs, and data growth patterns. Retaining backups for a specific duration allows for point-in-time recovery and long-term data retention if necessary.
4	Backup Strategy	Backup Verification Regularly validate the integrity and completeness of backup data by performing backup verifications. This ensures that the backups are usable and can be relied upon for data restoration when needed.

#	Name	Description
5	Backup Mechanism	Data Replication Leverage the proposed data replication scheme from the Production environment to the Disaster Recovery (DR) environment for backup purposes. Replicating data from the Production environment to the DR environment ensures that a synchronized copy of the data is available for restoration in the event of data loss or system failure.
6	Backup Mechanism	Backup Storage Determine the appropriate backup storage solution based on factors such as data volume, retention requirements, scalability, and cost. This can include on-premises backup storage, cloud-based storage solutions, or a combination of both. Implement data encryption and access controls to ensure the security and confidentiality of the backup data.
7	Backup Mechanism	Backup Validation Regularly test the backup and restore process to ensure its effectiveness and reliability. Perform restoration tests from backups to validate the integrity and completeness of the backup data. This helps identify any issues or gaps in the backup and restore procedures, allowing for timely remediation.
8	Restore Procedures	Recovery Point Objective (RPO) Define the desired Recovery Point Objective, which specifies the maximum amount of acceptable data loss during the restore process. Ensure that the restore procedures are designed to meet the defined RPO, allowing for data recovery up to the desired point in time.
9	Restore Procedures	Disaster Recovery (DR) Environment Establish operational procedures for restoring data from the DR database to the Production instances. This includes defining the steps, dependencies, and roles/responsibilities involved in the restore process. Conduct regular drills and exercises to validate the effectiveness of the restore procedures in the DR environment.
10	Restore Procedures	Data Validation Validate the restored data to ensure its accuracy and consistency. Perform data integrity checks and reconciliation processes to verify that the restored data matches the original data. This helps identify any potential data discrepancies or inconsistencies that may have occurred during the restore process.
11	Monitoring and Auditing	Backup Monitoring Implement monitoring mechanisms to track the status and health of the backup process. Monitor backup completion, success/failure rates, storage utilization, and any backup-related alerts or notifications. This enables proactive identification of issues and timely resolution to maintain the integrity of the backup data.

#	Name	Description
12	Monitoring and Auditing	Backup Auditing Establish auditing procedures to ensure compliance with backup and restore policies. Regularly review backup logs, restore activities, and backup storage access logs to monitor and track any unauthorized or unusual activities. This helps maintain data security and ensures that backup and restore processes align with regulatory requirements.

11.1.20.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	It does not address it directly.
3	SAMM Security by Implementation	It does not address it directly.
4	SAMM Security by Verification	It does not address it directly.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening The Backup and Restore strategy contributes to the environment hardening by ensuring that backups are securely stored and protected from unauthorized access. This helps protect the software system from potential vulnerabilities and security weaknesses. Incident Management The Backup and Restore strategy supports incident management by providing a mechanism to recover data in the case of data loss or system failure. It enables organizations to restore normal operations promptly and minimize the impact of security incidents. Secure Deployment The Backup and Restore strategy ensures that the data is securely deployed and restored during the deployment process. It helps ensure that data is protected during the restore process and reduces the risk of unauthorized access or compromise. Security Testing The Backup and Restore strategy can be tested through security testing activities, such as recovery testing and backup integrity checks. This helps identify vulnerabilities or weaknesses in the backup and restore mechanisms and ensures their effectiveness in maintaining the security of the software system. Operational Enablement The Backup and Restore strategy contributes to operational enablement by providing operational staff with guidelines and procedures for performing backup and restore operations. It ensures that operational staff have the necessary knowledge and skills to handle backup and restore processes securely.

11.1.20.2 Assessment – OWASP Top 10

Backup and Restore capability may not directly address all OWASP threats, but it plays a crucial role in mitigating the risks associated with data loss, system failures, and potential security incidents. It indirectly contributes to the overall security posture of web applications by providing a means to recover from security breaches, maintain data integrity, and adhere to secure practices during the restore process.

#	Name	Description
1	Broken Access Control (A01-2021)	While the Backup and Restore capability itself does not directly address this threat, it indirectly contributes to mitigating the risk associated with broken access control. By implementing a backup and restore strategy, organizations can ensure the availability and integrity of data in the event of a security incident. This reduces the impact of potential access control failures and helps maintain the confidentiality and integrity of sensitive information.

#	Name	Description
2	Cryptographic Failures (A02-2021)	The Backup and Restore capability does not directly address cryptographic failures. However, if encryption or cryptographic mechanisms are used in the backup process, it is essential to ensure that proper cryptographic practices are followed to protect the backup data. This includes using strong encryption algorithms, securely managing encryption keys, and implementing secure protocols for data transfer and storage.
3	Injection (A03-2021)	The Backup and Restore capability does not directly address injection vulnerabilities. However, during the restore process, it is crucial to validate and sanitize any user or external input to prevent potential injection attacks. By implementing secure coding practices and input validation mechanisms, organizations can reduce the risk of injection vulnerabilities when restoring data from backups.
4	Insecure Design (A04-2021)	The Backup and Restore capability does not directly address insecure design vulnerabilities. However, it can contribute to overall security by incorporating secure design principles into the backup and restore procedures. This includes ensuring that backups are stored securely, access controls are properly implemented for backup data, and backup processes are designed to minimize the attack surface and potential vulnerabilities.
5	Security Misconfiguration (A05-2021)	The Backup and Restore capability indirectly addresses security misconfigurations by providing an opportunity to validate and update the system's configuration during the restore process. Organizations can ensure that the restored system follows secure configuration practices, including secure default settings, disabling unnecessary functionality, and protecting sensitive information
6	Vulnerable and Outdated Components (A06-2021)	The Backup and Restore capability indirectly addresses vulnerable and outdated components by allowing organizations to restore systems to a known secure state. If vulnerabilities are discovered in the current system, the restore process can involve updating or replacing the components with the latest secure versions, reducing the risk associated with known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	The Backup and Restore capability does not directly address identification and authentication failures. However, during the restore process, organizations should ensure that proper authentication and access controls are in place to prevent unauthorized access to the restored system. This includes verifying the identity of users or administrators involved in the restore process and enforcing strong authentication mechanisms.
8	Software and Data Integrity Failures (A08-2021)	The Backup and Restore capability directly addresses software and data integrity failures. By regularly backing up data and restoring it from trusted backups, organizations can ensure the integrity and protection of software components, configurations, and data within the web application. This helps mitigate the risk of unauthorized modifications, data tampering, and other integrity-related issues.
9	Security Logging and Monitoring Failures (A09-2021)	The Backup and Restore capability indirectly contributes to addressing security logging and monitoring failures. During the backup and restore processes, organizations should ensure that appropriate logs are generated and stored securely. This helps in detecting any anomalies, security incidents, or unauthorized activities related to the backup and restore operations. Proper logging and monitoring mechanisms can assist in identifying potential security breaches and enable timely incident response

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	The Backup and Restore capability does not directly address server-side request forgery vulnerabilities. However, during the restore process, organizations should ensure that the restored system does not introduce or reintroduce vulnerabilities that could be exploited for server-side request forgery. This includes validating and sanitizing any user or external input to prevent unintended requests to internal or external resources.

11.1.21 File upload malware scanning

Introduction

The main risk of file uploads is that they can contain malware that can produce a ransomware attack. The platform system has several scenarios where it is required to synchronize data from different applications. For example, when Users upload documents to an application, or a file is sent by third parties, the file must be scanned of virus and malware. Any file must be scanned before sending or distributing the file to another applications or platforms. So any application dealing with this type of scenarios, must scan the files on the spot and determine if the file must be put in quarentine before sending it to another applications or storing it any repository. So, ideally the file upload malware scanning must consist in:

• Creating a segregated network segment where the files uploaded

• Scanned the files with an antivirus

• Quarantine, or delete and replace the files with a notification alert file if they are infected.

• If the files are clean, follow the business, flow, forward them or store them in the final destination on the system.

Any platform dealing with file upload must adress this risk. In this way, the platform can significantly reduce the risk of malware-infected files being stored and minimize the potential impact of ransomware attacks.

Regarding the deployment options for this solution, ideally it requires an antivirus software to run where the file is received without having to transfer the file to a centralised location or other networks. Therefore it is may be convenient to have a distributed architecture and have one deployment of the same file-scanning solution per channel or wherever is needed.

#	Name	Description
1	Segregated Network Segment	Set up a segregated network segment specifically for file uploads. This helps isolate the uploaded files from the rest of the system, minimizing the potential impact of any malware present in the files.
2	Antivirus Scanning	Integrate antivirus software, into the file upload process. Configure the system to automatically scan each uploaded file using the anti-virus to detect any potential malware or viruses.
3	Real-Time Scanning	Implement real-time scanning capabilities to scan files as they are uploaded to the platform. This helps detect and block malware in real-time, reducing the risk of storing infected files on the system.

#	Name	Description
4	Malware Detection Actions	If the anti-virus detects malware in a file, there must be a default or custom setting to determine how to handle the infected file. immediately quarantine or delete the infected file from the system to prevent any further spread of the malware. Replace it with a notification alert file or message indicating that the upload was blocked due to malware detection.
5	Periodic Antivirus Updates	Ensure regular updates of the antivirus software to keep up with the latest malware signatures and detection techniques. Schedule automatic updates or establish a process to manually update the antivirus software periodically.
6	Logging and Auditing	Implement logging mechanisms to capture and record the results of file scans. Maintain audit logs that include information such as the upload timestamp, file name, scan results (infected/clean), and actions taken. These logs can be used for monitoring, analysis, and audit purposes.
7	Scalability and Performance	Ensure the file upload malware scanning capability can scale to handle increasing file upload volumes efficiently. Optimize the scanning process and infrastructure to minimize any impact on system performance or user experience.
8	Third-Party Application Integration	If third-party applications are introduced into the platform's scope, reevaluate and rectify the file upload malware scanning solution. Collaborate with the third-party application developers to ensure that files sent from these applications undergo the necessary scanning and security measures before being stored in the central application.

11.1.21.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Compliance Management File upload malware scanning helps organizations comply with relevant regulatory requirements and industry standards by ensuring that uploaded files are scanned for malware before storing them in the central application.
2	SAMM Security by Design	Security Requirements File upload malware scanning addresses the security requirement of preventing the upload and storage of infected files, mitigating the risk of malware-related threats. Secure Architecture Incorporating file upload malware scanning into the design phase helps establish a secure architecture.

#	Name	Description
3	SAMM Security by Implementation	Security Testing Integration File upload malware scanning is a security testing activity that can be integrated into the implementation phase to identify and address vulnerabilities associated with file uploads, specifically malware-related risks. Security Verification Verifying the effectiveness of the file upload malware scanning process ensures that the implemented security control is reliable and contributes to the overall security of the system
4	SAMM Security by Verification	Security Testing File upload malware scanning is a type of security testing that helps detect and address security vulnerabilities related to malicious files during the verification phase.
5	SAMM Security by Operations	Vulnerability Management Incorporating file upload malware scanning as part of vulnerability management practices helps identify and mitigate vulnerabilities associated with malware-infected files. Security Testing File upload malware scanning is a security testing activity that supports ongoing security operations by continuously monitoring and mitigating malware-related risks.

11.1.21.2 Assessment – OWASP Top 10

The File Upload Malware Scanning capability may not directly address all OWASP threats, however, it plays a crucial role in mitigating the risk of malware-related security breaches and compromises introduced through insecure design decisions or the use of vulnerable components. By implementing malware scanning for file uploads, organizations can enhance the overall security posture of their applications and protect against potential threats associated with malicious files.

The capability of File Upload Malware Scanning directly addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	It does not address it directly.
2	Cryptographic Failures (A02-2021)	It does not address it directly.
3	Injection (A03-2021)	It does not address it directly.

#	Name	Description
4	Insecure Design (A04-2021)	This capability indirectly addresses insecure design vulnerabilities by implementing a segregated network segment for file uploads and scanning them for malware. By ensuring that uploaded files are scanned for malware, organizations can mitigate the risk of potential security breaches and compromises introduced through insecure design decisions.
5	Security Misconfiguration (A05-2021)	This capability indirectly addresses security misconfiguration vulnerabilities by enforcing secure practices during the file upload process. By scanning files for malware, organizations can prevent the misconfiguration of systems that could lead to the execution of malicious code or the exposure of sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	This capability indirectly addresses vulnerabilities associated with vulnerable and outdated components. By scanning uploaded files for malware, organizations can reduce the risk of using infected files that may contain vulnerabilities or exploit outdated components within the system.
7	Identification and Authentication Failures (A07-2021)	It does not address it directly.
8	Software and Data Integrity Failures (A08-2021)	This capability indirectly addresses software and data integrity failures by scanning uploaded files for malware. By detecting and removing infected files, organizations can mitigate the risk of unauthorized modifications, data tampering, and potential integrity-related issues.
9	Security Logging and Monitoring Failures (A09-2021)	It does not address it directly.
10	Server-Side Request Forgery (A10-2021)	It does not address it directly.

11.1.22 Multi-factor Authentication

When a solution requires enhances the security, one of the using two-factor or more factor authentication, it This second token must be typed into the application and uses a time-based one-time password algorithm.

Extending the capability of "Multi-factor Authentication" involves implementing a two-factor authentication (2FA) model using a time-based one-time password (TOTP) algorithm. Here are some additional details to consider:

By implementing a two-factor authentication model using a time-based one-time password algorithm, the proposed solution enhances security by requiring users to provide an additional factor during the authentication process. This helps protect against unauthorized access, even if passwords are compromised or stolen.

#	Name	Description
1	Security profile	Conduct a thorough analysis to identify the users or user roles that require an additional level of security. This analysis may consider factors such as the sensitivity of the data they can access, their roles and responsibilities, and any regulatory or compliance requirements.
2	Two-Factor Authentication	Consider implementing extra factor authentication mechanisms. For two-factor authentication, it requires the users to provide two separate factors to authenticate their identity. The first factor is typically something the user knows, such as a password or PIN, while the second factor is something the user possesses, such as a time-based one-time password token or a mobile device.
3	Time-Based One-Time Password Algorithm	If it is suitable, implement a time-based one-time password (TOTP) algorithm, such as the widely adopted Time-based One-Time Password (TOTP) algorithm specified in RFC 6238. This algorithm generates a unique one-time password based on a shared secret key and the current time. The password changes periodically, typically every 30 seconds, providing an additional layer of security.
4	Token Entry	Users will be prompted to enter the one-time password generated by their TOTP token into the application during the authentication process. This ensures that only authorized users with both the correct password and the currently valid one-time password can access the system.
5	TOTP Token Delivery	Determine the method of delivering TOTP tokens to users. This can include dedicated hardware tokens, software-based authenticator apps (such as Google Authenticator or Authy), or even SMS-based authentication if it meets the necessary security requirements.
6	User Experience	Consider the user experience during the authentication process. Provide clear instructions to users on how to set up and use the TOTP tokens. Implement user-friendly interfaces that guide users through the process and provide feedback on the status of their authentication attempts.
7	Security and Key Management	Establish secure practices for storing and managing the shared secret keys used in the TOTP algorithm. Use appropriate encryption and key management techniques to protect these sensitive keys from unauthorized access or misuse.

#	Name	Description
8	User Enrollment and Onboarding	Define a process for enrolling users in the two-factor authentication system and ensuring their successful onboarding. This process may include user education, providing step-by-step instructions for setting up TOTP tokens, and handling any potential issues or challenges during the enrollment process.
9	Compliance and Audit Trail	Ensure that the implementation of multi-factor authentication aligns with any regulatory or compliance requirements relevant to the system or organization. Maintain audit logs of authentication events, including the use of the second factor, for security monitoring and audit trail purposes.

11.1.22.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: The use of Two-factor Authentication can contribute to the policy and standard development for software security. It establishes the expectation and requirement of using an extra factor for authentication, enhancing security. Risk Management: By implementing Two-factor Authentication, organizations can mitigate the risk associated with unauthorized access and credential theft, addressing the risk of compromised authentication. Compliance Management: Two-factor Authentication helps organizations comply with industry standards and best practices that recommend the use of multi-factor authentication for enhanced security. Security Training and Awareness: The implementation of Two-factor Authentication requires educating users about the importance of using an extra factor for authentication, raising awareness about the risks associated with weak authentication methods. Security Metrics and Reporting: Organizations can include the measurement of Two-factor Authentication adoption and usage as a security metric to assess the effectiveness of software security practices. Security Roles and Responsibilities: Two-factor Authentication contributes to assigning security roles and responsibilities by specifying the requirement for users to follow the authentication process and protect their authentication factors.

#	Name	Description
2	SAMM Security by Design	Security Requirements: Two-factor Authentication can be considered as a security requirement during the design phase to mitigate OWASP threats related to weak authentication. Secure Architecture: Two-factor Authentication enhances the security of the overall architecture by minimizing the risk of unauthorized access and addressing OWASP threats such as Insecure Design and Cryptographic Failures. Secure Coding Practices: Two-factor Authentication can be implemented using secure coding practices to prevent OWASP threats like Injection, Insecure Design, and Security Misconfiguration. Threat Modeling: Two-factor Authentication addresses the threat of compromised authentication by incorporating an additional factor that mitigates potential risks. Security Testing: The implementation of Two-factor Authentication contributes to security testing by validating the effectiveness of the authentication process and identifying vulnerabilities related to OWASP threats.
3	SAMM Security by Implementation	Secure Development Training: Two-factor Authentication can be included in secure development training programs to educate developers about the importance of implementing multi-factor authentication. Secure Architecture: Two-factor Authentication ensures that the implemented architecture includes appropriate security controls and mechanisms to protect against OWASP threats. Secure Coding Guidelines: Implementing Two-factor Authentication aligns with secure coding guidelines by addressing vulnerabilities associated with weak authentication and access control. Security Requirements: Two-factor Authentication can be specified as a security requirement to ensure its implementation during the software development process. Security Testing Integration: Two-factor Authentication can be incorporated into security testing activities to assess its effectiveness and identify any vulnerabilities related to authentication. Security Verification: Two-factor Authentication can be verified to ensure it meets the desired security objectives and provides the intended level of protection. Security Architecture Review: Two-factor Authentication can be included in security architecture reviews to assess its implementation and effectiveness. Security Operations Integration: Two-factor Authentication contributes to secure deployment practices and supports incident management, vulnerability management, and secure deployment.

#	Name	Description
4	SAMM Security by Verification	Security Testing: Two-factor Authentication can be tested to assess its effectiveness and ensure that it addresses potential vulnerabilities in software applications. Code Review: The implementation of Two-factor Authentication can be reviewed to identify any flaws and ensure adherence to secure coding practices. Security Architecture Review: Two-factor Authentication can be assessed during security architecture reviews to validate the presence of appropriate security controls and compliance with standards. Security Requirements Verification: Two-factor Authentication can be verified against security requirements to ensure its proper implementation. Threat Modeling: Two-factor Authentication can be considered during threat modeling exercises to evaluate its impact on the identified threats and risks. Secure Deployment Verification: Two-factor Authentication can be verified to ensure its secure deployment and configuration.
5	SAMM Security by Operations	Environment Hardening: Two-factor Authentication contributes to the hardening of the software environment by providing an additional layer of security for authentication. Secure Build: Two-factor Authentication can be included in secure build.

11.1.22.2 Assessment – OWASP Top 10

The Multi-factor Authentication capability can address several OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Multi-factor authentication can help mitigate this threat by adding an extra layer of authentication, reducing the risk of unauthorized access. By requiring users to provide a second token, such as a time-based one-time password, the system can verify the user's identity more securely and prevent unauthorized access attempts.
2	Cryptographic Failures (A02-2021)	It does not address it directly.
3	Injection (A03-2021)	Although multi-factor authentication does not directly address injection vulnerabilities, it can provide an additional layer of defense against unauthorized access resulting from successful injection attacks. Even if an attacker manages to inject malicious code, they would still need to bypass the multi-factor authentication to gain access to the system.

#	Name	Description
4	Insecure Design (A04-2021)	Multi-factor authentication does not directly address insecure design vulnerabilities. However, it can complement secure design principles by adding an additional security control to mitigate the risk of unauthorized access. Properly integrating multi-factor authentication within the system's architecture can enhance the overall security of the application.
5	Security Misconfiguration (A05-2021)	Multi-factor authentication does not directly address security misconfigurations. However, it can act as a compensating control by providing an additional layer of security in case misconfigurations occur. Even if a misconfiguration exposes certain authentication mechanisms, the multi-factor authentication requirement adds an extra barrier for attackers.
6	Vulnerable and Outdated Components (A06-2021)	It does not address it directly.
7	Identification and Authentication Failures (A07-2021)	Multi-factor authentication directly addresses identification and authentication failures by introducing an additional factor to verify the user's identity. It helps mitigate vulnerabilities related to weak or compromised credentials, password guessing, and credential stuffing attacks, thus reducing the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	It does not address it directly.
9	Security Logging and Monitoring Failures (A09-2021)	Multi-factor authentication does not directly address security logging and monitoring failures. However, it can indirectly contribute to improved logging and monitoring by providing an additional data point for user authentication. Logging successful and failed authentication attempts, including multi-factor authentication events, can aid in detecting and responding to potential security incidents.
10	Server-Side Request Forgery (A10-2021)	It does not address it directly.

11.1.23 OTP (One-time-password) token

The OTP token One-time-password) token" is a security model where a hardware device or software program can produce a single-use password or PIN passcode.
Incorporating OTP tokens as a security model, adds an extra layer of protection to normal username and passwords, which are possible to to guess or brute force. OTP tokens enhances the overall security of the system, it mitigates the risk of password-based attacks, and helps ensure that only authorized users can access sensitive resources.

#	Name	Description
1	Hardware-based Tokens	These are physical devices, often in the form of key fobs or smart cards, that generate one-time passwords. These devices typically have a small display that shows the current password and may require the user to press a button to generate a new password.
2	Software-based Tokens:	These are typically applications or software programs installed on a user's device, such as smartphones or computers. Software-based tokens generate one-time passwords within the application, eliminating the need for a physical device.
3	Password Generation	OTP, TOTP, and HOTP are all related to one-time passwords and are commonly used for authentication and security purposes. OTP (One-Time Password) TOTP (Time-Based One-Time Password) HOTP (HMAC-Based One-Time Password)
4	OTP (One-Time Password)	OTP, short for One-Time Password, is a single-use password generated for authentication, typically valid for only one login session or transaction. Once used, the OTP becomes invalid, providing an additional layer of security. OTPs are often used as a second factor of authentication in multi-factor authentication (MFA) systems, enhancing the security of user accounts.
5	TOTP (Time-Based One-Time Password)	TOTP, which stands for Time-Based One-Time Password, is a type of OTP that is generated based on the current time and a shared secret key. The shared secret key is known to both the authentication server and the user's device (usually a mobile app). Using a cryptographic algorithm, the server and the device independently generate the same TOTP at any given time. The generated TOTP is then used for authentication purposes. TOTP has a defined time validity window (usually 30 seconds) during which the generated code is valid. The time-based nature of TOTP means that both the server and the user's device must be in sync regarding the current time for successful authentication. Common applications of TOTP include services like Google Authenticator, Microsoft Authenticator, and Authy, where users receive time-based OTPs to secure their accounts.

#	Name	Description
6	HOTP (HMAC-Based One-Time Password)	HOTP, or HMAC-Based One-Time Password, is another type of OTP that is based on a cryptographic hash function called HMAC (Hash-based Message Authentication Code). Similar to TOTP, HOTP also relies on a shared secret key known to both the server and the user's device. Instead of being time-based, HOTP generates a sequence of OTPs based on a counter value. Each time a new OTP is required, the counter value is incremented, and the new OTP is generated using the HMAC algorithm. The generated OTPs are typically valid until they are used, and the server keeps track of the last counter value to prevent replay attacks. HOTP is commonly used in scenarios where time synchronization between the server and the user's device might be challenging or impractical. However, since HOTP doesn't rely on time, it might not provide the same level of convenience as TOTP, as users have to manually advance the counter on their devices.
7	Backup and Recovery	Implement mechanisms for backup and recovery of OTP tokens. This ensures that users can regain access to their accounts in case of token loss or device failure. Backup options may include generating backup codes or linking multiple devices to the same OTP token.
8	Revocation and Deactivation	Have processes in place to handle token revocation or deactivation. In case of token compromise or user account suspension, the system should be able to revoke or deactivate the associated OTP token to prevent unauthorized access.
9	Compliance and Audit	Ensure that the implementation of OTP tokens adheres to relevant regulatory or compliance requirements. Maintain audit logs of OTP usage for security monitoring, compliance audits, and traceability.

11.1.23.1 Assessment – SAMM

While the OTP token may not directly align with every specific activity within the SAMM security domains, it can contribute to the overall security posture of an organization by enhancing authentication mechanisms and protecting against unauthorized access.

The OTP token, as a security model, does not directly address the specific activities within the SAMM security domains. However, it can contribute to enhancing security in various ways, such as:

#	Name	Description
1	SAMM Security by Governance	While the OTP token may not directly contribute to this domain, it can be used as an additional authentication factor, strengthening the overall security posture of the organization.
2	SAMM Security by Design	The OTP token can be integrated into the design of software applications as an additional security control for user authentication and access management, mitigating the risk of unauthorized access and identity theft.

#	Name	Description
3	SAMM Security by Implementation	In the implementation phase, the OTP token can be implemented as part of the secure authentication mechanisms, helping to protect against common security threats such as credential theft and password guessing.
4	SAMM Security by Verification	During security verification activities, the OTP token can be tested for its effectiveness in providing secure authentication and preventing unauthorized access.
5	SAMM Security by Operations	In the operations phase, the OTP token can be used to enhance the secure operation of software systems by providing an additional layer of authentication for users accessing sensitive resources.
6	Password Validation	The system receiving the OTP should have the capability to validate the entered password against the expected value generated by the OTP token. This validation ensures that only users with the correct one-time password can proceed with the authentication process.
7	Synchronization	For TOTP-based OTP tokens, it is important to synchronize the time between the token generator and the system validating the passwords. This synchronization ensures that both parties generate and validate passwords using the same time intervals, allowing for successful authentication.
8	Token Binding	To enhance security, consider implementing token binding techniques. Token binding ensures that the OTP token is bound to a specific user or device, reducing the risk of interception or unauthorized use. This can be achieved through cryptographic techniques or device-specific identifiers.
9	User Enrollment	Define a user enrollment process for issuing OTP tokens. This process may involve securely distributing hardware tokens or guiding users through the installation and setup of software-based tokens. Proper user education and onboarding materials should be provided to ensure a smooth transition to using OTP tokens.

11.1.23.2 Assessment – OWASP Top 10

The OTP (One-time-password) token capability can address the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The OTP token adds an additional layer of authentication, which can help prevent unauthorized access to resources. By requiring users to provide a one-time password or PIN, the OTP token helps enforce access controls and reduces the risk of broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	The OTP token relies on cryptographic algorithms to generate and verify one-time passwords. When implemented correctly, it can ensure the confidentiality and integrity of the generated passwords. By using proper cryptographic practices, such as strong encryption algorithms and secure key management, the OTP token can address cryptographic failures.

#	Name	Description
3	Injection (A03-2021)	The OTP token, when implemented securely, can help mitigate injection vulnerabilities. By validating and sanitizing user input used to generate OTPs, the application can prevent attackers from injecting malicious code or unintended commands into the system.
4	Insecure Design (A04-2021)	The OTP token itself does not directly address insecure design vulnerabilities. However, if integrated into the overall architecture of the application with secure design principles in mind, it can contribute to a more secure system by providing an additional layer of authentication.
5	Security Misconfiguration (A05-2021)	The OTP token capability does not directly address security misconfiguration vulnerabilities. However, proper configuration and management of the OTP token system, such as securing the token generation and verification processes, can help mitigate the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	The OTP token capability does not directly address vulnerable and outdated components vulnerabilities. However, when selecting and implementing the OTP token solution, organizations should consider using up-to-date and secure hardware or software components to minimize the risk of vulnerabilities in the token system itself.
7	Identification and Authentication Failures (A07-2021)	The OTP token addresses identification and authentication failures by providing an additional factor for user authentication. By requiring users to provide a one-time password or PIN along with their regular credentials, the OTP token helps ensure stronger authentication and reduces the risk of authentication-related vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	The OTP token capability does not directly address software and data integrity failures. However, the use of OTPs can help protect the integrity of user accounts and sensitive data by adding an extra layer of authentication, making it more difficult for attackers to tamper with or manipulate the data.
9	Security Logging and Monitoring Failures (A09-2021)	The OTP token capability does not directly address security logging and monitoring failures. However, the OTP token system can generate logs of OTP generation and verification activities, which can be monitored and analyzed as part of an overall logging and monitoring strategy to detect any suspicious or unauthorized activities.
10	Server-Side Request Forgery (A10-2021)	The OTP token capability does not directly address server-side request forgery vulnerabilities. However, implementing proper input validation and secure configurations within the application can help mitigate the risk of SSRF attacks. The OTP token itself, while not a direct solution, can be part of a broader security strategy to prevent unauthorized requests and protect against SSRF.

11.1.24 Biometrics

The biometric solution consist in a specialized hardware connected via Bluetooth to mobile devices that is used to capture and validate biometric data from Individuals.

The solution allows the measurement and analysis of unique physical or behavioral characteristics of individuals for identification or authentication purposes.
Incorporating biometrics into the system architecture, it allows leveraging unique physical or behavioral characteristics of individuals for secure and convenient authentication. The biometric authentication provides an additional layer of security, reduces reliance on traditional authentication methods, and enhances the overall user experience.

#	Name	Description
1	Biometric Data Capture	The system should integrate specialized hardware devices capable of capturing biometric data. These devices could include fingerprint scanners, facial recognition cameras, iris scanners, voice recognition microphones, or other biometric sensors. The hardware is typically connected to mobile devices via Bluetooth or other communication protocols.
2	Biometric Data Processing	The system should include algorithms and processing capabilities to analyze and extract relevant features from the captured biometric data. These features are then used for comparison and matching during the authentication process.
3	Biometric Template Management	The system should manage and store biometric templates, which are mathematical representations derived from the captured biometric data. These templates are used for comparison and matching during authentication, rather than storing the raw biometric data itself.
4	Biometric Authentication:	The captured biometric data is used as a means of authentication. Instead of relying solely on passwords or PINs, users can authenticate themselves using their unique biometric traits. The system compares the captured biometric data with stored reference data to verify the user's identity.
5	Authentication Integration	Integrate biometric authentication into the existing authentication framework of the system. This could involve integrating with identity and access management systems, single sign-on solutions, or other authentication mechanisms.
6	User Enrollment	Establish a user enrollment process to capture and register users' biometric data. During enrollment, the system captures the biometric data and associates it with the user's account, creating a reference template for future authentication.
7	Mobile Device Integration	Mobile devices, such as smartphones or tablets, act as the platform for biometric data capture and authentication. The system should have mobile applications that facilitate the interaction between the specialized hardware and the mobile devices.
8	User Experience and Accessibility	Ensure a seamless and user-friendly experience for biometric authentication. Design the user interfaces to guide users through the biometric data capture process, provide feedback on successful authentication, and handle any errors or failures gracefully. Consider accessibility requirements to accommodate users with disabilities.

#	Name	Description
9	Biometric Data Storage and Encryption	Biometric data is sensitive and requires secure storage. The system should employ robust encryption techniques to protect the stored biometric data. Additionally, consider compliance with privacy regulations and guidelines regarding the storage and protection of biometric information.
10	Security and Anti-Spoofing Measures	Implement security measures to prevent spoofing or fraudulent use of biometric data. This may include liveness detection techniques to ensure that the captured biometric data comes from a live person and not a replica or photo.
11	Compliance and Privacy	Consider compliance with applicable regulations and standards regarding the collection, storage, and use of biometric data. Ensure compliance with privacy laws and obtain appropriate consent from users for collecting and processing their biometric information.

11.1.24.1 Assessment – SAMM

In summary, adding the proposed biometrics solution that includes data capture and processing contributes to addressing security requirements.

Implementing a biometric solution secures the architecture overall, enhancing authentication, access control, and overall security. However, it requires considering specific measures to address the unique risks and considerations associated with biometric data.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards The use of biometrics in the solution helps address the policy and standard development related to secure coding practices, incident response, and data protection. Risk Management Biometrics can contribute to risk management activities by providing an additional layer of authentication and reducing the risk of unauthorized access. Compliance Management The solution's implementation of biometrics can help organizations comply with regulatory requirements and industry standards related to identity verification and access control. Security Training and Awareness Biometrics can be part of security training programs and awareness campaigns to educate employees and stakeholders about the importance of strong authentication measures. Security Metrics and Reporting The use of biometrics can contribute to security metrics and reporting by providing data on the effectiveness and usage of biometric authentication. Security Roles and Responsibilities Biometrics can be used to assign and enforce security roles and responsibilities, ensuring that individuals have the necessary credentials to access sensitive information or perform specific actions.

#	Name	Description
2	SAMM Security by Design	Security Requirements Biometrics can be considered as a security requirement to address concerns related to authentication and access control. Secure Architecture The use of specialized hardware and Bluetooth connectivity in capturing biometric data helps establish a secure architecture that minimizes potential vulnerabilities and provides a strong foundation for security controls. Secure Coding Practices Biometrics can be implemented using secure coding practices to prevent vulnerabilities associated with the OWASP threats. Threat Modeling Biometrics can be incorporated into threat modeling exercises to assess potential risks and identify appropriate security controls. Security Testing Biometrics can be included in security testing activities to evaluate its resilience against potential attacks and vulnerabilities.
3	SAMM Security by Implementation	Secure Development Training Biometrics can be part of secure development training to educate developers on how to integrate biometric authentication securely. Secure Architecture The use of specialized hardware and Bluetooth connectivity in capturing biometric data contributes to a secure architecture, minimizing vulnerabilities. Secure Coding Guidelines Biometrics can be implemented following secure coding guidelines, reducing the likelihood of introducing vulnerabilities. Security Requirements Biometrics can be defined as a security requirement and implemented accordingly. Security Testing Integration Biometrics can be included in security testing activities to identify and address vulnerabilities related to its implementation. Security Verification Biometrics can be subject to security verification techniques to validate its effectiveness and adherence to security objectives. Security Architecture Review Biometrics implementation can be reviewed as part of the security architecture review process to ensure proper security controls and mechanisms are in place. Security Operations Integration Biometrics can be integrated into operational processes such as incident management, vulnerability management, and secure deployment practices to support ongoing security.

#	Name	Description
4	SAMM Security by Verification	Security Testing Biometrics can be subject to security testing activities such as penetration testing, vulnerability scanning, and code review to identify vulnerabilities. Code Review The implementation of biometrics can be reviewed to identify security flaws and ensure adherence to secure coding practices. Security Architecture Review Biometrics implementation can be assessed to ensure that proper security controls are in place. Security Requirements Verification: Biometrics can be reviewed against security requirements to validate their proper implementation. Threat Modeling Biometrics can be included in threat modeling exercises to assess potential threats and risks. Secure Deployment Verification The deployment of biometric solutions can be verified to ensure secure configurations and access controls are in place.
5	SAMM Security by Operations	Secure Deployment The deployment of the proposed architecture, which includes specialized hardware for capturing biometric data, should follow secure deployment practices. This includes secure software distribution, secure installation procedures, and secure configuration of the biometric hardware and software components. Security Testing Ongoing security testing, including regular assessments of the biometric authentication system's effectiveness and identification of any vulnerabilities or weaknesses, should be integrated into the operational processes of the software system. Incident Management The operational processes should include incident management procedures specifically addressing incidents related to biometric data, such as breaches or unauthorized access attempts. These procedures should outline the steps for incident detection, response, containment, and recovery in the context of biometric authentication.

11.1.24.2 Assessment – OWASP Top 10

Biometrics solutions can address these some security threats; however, their effectiveness depends on the proper implementation and integration within the overall architecture design. Additionally, it's essential to consider other security measures, such as secure coding practices, secure configurations, and regular security testing, to comprehensively address OWASP threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Biometrics can contribute to addressing this threat by providing a strong authentication mechanism. By capturing biometric data from individual applicants, the system can enforce access controls based on biometric verification, reducing the risk of unauthorized access or privilege escalation.

#	Name	Description
2	Cryptographic Failures (A02-2021)	It does not address it directly.
3	Injection (A03-2021)	Biometrics, when implemented correctly, can help mitigate injection vulnerabilities. Since biometric data is captured using specialized hardware connected via Bluetooth to mobile devices, it reduces the reliance on user-supplied input that could potentially be manipulated by attackers to inject malicious code. Biometric verification provides a more secure and tamper-resistant means of user authentication.
4	Insecure Design (A04-2021)	It does not address it directly.
5	Security Misconfiguration (A05-2021)	It does not address it directly.
6	Vulnerable and Outdated Components (A06-2021)	It does not address it directly.
7	Identification and Authentication Failures (A07-2021)	Biometrics can enhance the identification and authentication mechanisms of web applications. By capturing biometric data, the system can establish a stronger link between the user and their identity, reducing the risk of impersonation or unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Biometrics can contribute to ensuring the integrity of software and data within web applications. By utilizing biometric data for authentication, organizations can verify the integrity of user identity and protect against unauthorized modifications or tampering of software components, configurations, and data.
9	Security Logging and Monitoring Failures (A09-2021)	It does not address it directly.
10	Server-Side Request Forgery (A10-2021)	It does not address it directly.

11.1.25 OpenID Connect and OAuth 2.0

OpenID Connect and OAuth 2.0 are the most secure protocols for interchanging and authorizing resource access. It standardise the integration protocols with Identity Providers and other Resource Service Provider parties.

OpenID Connect

It is a protocol is an open standard for authentication and authorization that provides a secure way to authenticate users and exchange user identity information between applications.

Incorporating OpenID Connect into the system architecture, the proposed solution leverages a widely adopted and standardized authentication and authorization protocol. It enables secure and seamless user authentication, promotes interoperability with other systems, and provides a foundation for building SSO capabilities within the system.

#	Name	Description
1	Identity Provider (IdP)	Implement an Identity Provider that supports OpenID Connect. The IdP is responsible for authenticating users and issuing identity tokens. It could be a dedicated Identity Provider solution or integrated within the system's authentication infrastructure.
2	Authentication Flow	Define the authentication flow using OpenID Connect. This typically involves redirecting the user to the IdP's login page, where they authenticate themselves using their credentials. Upon successful authentication, the IdP issues an identity token.
3	Identity Token	Handle the identity token returned by the IdP. The identity token contains information about the authenticated user, such as their unique identifier (subject), name, email, and other optional claims. The system should validate and decode the token to extract the necessary user information.
4	Authorization	Leverage OpenID Connect's authorization capabilities to control access to protected resources. The system can use the identity token to determine the user's identity and authorization privileges. This can be done by mapping the user's identity to roles or permissions within the system.
5	Integration with Resource Servers	Ensure that the system's resource servers, such as APIs or web services, are integrated with OpenID Connect. Resource servers can validate incoming requests by verifying the identity token's signature and extracting the necessary user information for authorization checks.
6	Single Sign-On (SSO)	Leverage OpenID Connect's SSO capabilities to enable users to authenticate once and access multiple applications within the system without re-entering their credentials. The system should support the sharing of authentication sessions and tokens among different applications.
7	Token Management	Implement token management mechanisms to handle token expiration, refresh, and revocation. OpenID Connect defines token types, such as access tokens and refresh tokens, which have specific lifetimes and usage patterns. The system should handle these tokens securely and efficiently
8	Federation and Interoperability	Consider the ability to federate with external Identity Providers or enable interoperability with other systems that support OpenID Connect. This allows users from different organizations or systems to authenticate seamlessly and access shared resources within the system.
9	User Consent and Privacy	Implement mechanisms for user consent and privacy management. OpenID Connect allows users to grant or deny access to their information through consent screens. The system should handle user preferences and ensure compliance with privacy regulations.
10	Developer Tools and Libraries	Provide development resources, SDKs, or libraries that simplify the integration of OpenID Connect into the system. This could include client libraries for different programming languages, documentation, and code samples.

OAuth

It is an open standard for authorization that allows users to grant third-party applications limited access to their resources without sharing their credentials directly.
OAuth enables secure and controlled access to user resources by third-party applications. It allows users to share their data without compromising their credentials, promotes interoperability with external systems, and provides a standardized approach to authorization. OAuth enhances security, user privacy, and developer productivity in building integrations with the system.

#	Name	Description
1	OAuth Roles	Identify and define the roles involved in the OAuth flow. The main roles are the Resource Owner (user), Client (third-party application), and Authorization Server (handles authentication and authorization).
2	Client Registration	Implement a mechanism for clients to register with the system. This includes providing client credentials (e.g., client ID and client secret) to enable secure communication between the client and the authorization server.
3	Authorization Grant Types	Support different authorization grant types defined by OAuth. This includes the authorization code grant, implicit grant, client credentials grant, and resource owner password credentials grant. Choose the appropriate grant type based on the security requirements and the nature of the client application.
4	Authorization Endpoint	Implement an authorization endpoint where users can authenticate and authorize the client application to access their resources. The system should handle the authentication process, consent management, and issue an authorization code or access token based on the grant type.
5	Token Endpoint	Set up a token endpoint where the client can exchange the authorization code for an access token or obtain a refresh token. The system should validate the authorization code, authenticate the client, and issue the appropriate tokens.
6	Access Tokens	Handle access tokens securely. Access tokens are used by the client to access protected resources on behalf of the user. The system should validate access tokens, enforce token expiration, and protect them from unauthorized access or tampering.
7	Resource Server Protection	Protect the system's resources (APIs or services) using OAuth access tokens. Resource servers should validate incoming access tokens, verify their authenticity and permissions, and grant or deny access accordingly.
8	Refresh Tokens	If using the refresh token grant type, manage refresh tokens securely. Refresh tokens are used to obtain new access tokens when the current one expires. The system should handle refresh token storage, rotation, and revocation.
9	Scopes and Permissions	Define and enforce scopes and permissions that control the level of access granted to clients. Scopes define the specific resources or actions that the client can request access to. The system should validate requested scopes and ensure that clients only receive the necessary access.

#	Name	Description
10	Token Revocation	Implement a mechanism to revoke access and refresh tokens when needed. This allows users to revoke access granted to a client or mitigate security risks in case of a compromised token.
11	Token Introspection	Provide a token introspection endpoint for resource servers to validate access tokens. This endpoint allows resource servers to verify the validity and details of an access token without making additional requests to the authorization server.
12	Developer Tools and Libraries	Offer development resources, SDKs, or libraries to facilitate OAuth integration for client applications. This could include client libraries, documentation, and code samples to help developers implement OAuth flows correctly.

11.1.25.1 Assessment – SAMM

OpenID Connect and OAuth 2.0 can be utilized as integration methods with Identity Providers and other Resource Service Provider parties; however, they do not inherently contribute to the activities within the SAMM security domains. Organizations would need to combine the use of these standards with other practices and methodologies to address the specific activities within each SAMM domain effectively.

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	It does not address it directly.
3	SAMM Security by Implementation	It does not address it directly.
4	SAMM Security by Verification	It does not address it directly.
5	SAMM Security by Operations	It does not address it directly.

11.1.25.2 Assessment – OWASP Top 10

The OpenID Connect and OAuth 2.0 architecture capability can address the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	OpenID Connect and OAuth 2.0 provide a secure method for enforcing access controls within a web application. These protocols define mechanisms for user authentication and authorization, allowing for fine-grained control over resource access based on user privileges and permissions. By properly implementing and utilizing OpenID Connect and OAuth 2.0, organizations can mitigate the risk of broken access control vulnerabilities by ensuring that only authorized users have access to specific resources.

#	Name	Description
2	Cryptographic Failures (A02-2021)	OpenID Connect and OAuth 2.0 incorporate strong cryptographic mechanisms for securing the exchange of identity and access tokens. These protocols utilize cryptographic algorithms and best practices to protect sensitive information, such as user identities and authorization data, during authentication and authorization processes. By leveraging the security features provided by OpenID Connect and OAuth 2.0, organizations can reduce the likelihood of cryptographic failures and vulnerabilities in their web applications.
3	Injection (A03-2021)	OpenID Connect and OAuth 2.0 can help mitigate injection vulnerabilities by providing a standardized and secure method for exchanging identity and access tokens. By using these protocols, web applications can avoid the need to handle and process user-supplied input directly, reducing the risk of injection attacks. Additionally, the authentication and authorization processes facilitated by OpenID Connect and OAuth 2.0 involve proper input validation and parameterization, further mitigating the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	OpenID Connect and OAuth 2.0 can contribute to addressing insecure design vulnerabilities by providing a well-defined and standardized architecture for implementing identity and access management in web applications. These protocols promote secure design principles and best practices, ensuring that security considerations are taken into account during the implementation of authentication and authorization mechanisms. By adhering to the architectural guidelines of OpenID Connect and OAuth 2.0, organizations can mitigate the risk of insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	OpenID Connect and OAuth 2.0 provide clear specifications and guidelines for implementing secure authentication and authorization mechanisms. By following these guidelines, organizations can avoid common security misconfigurations associated with authentication and authorization processes. OpenID Connect and OAuth 2.0 also promote secure configuration practices for identity providers and resource service providers, reducing the likelihood of misconfigurations that could lead to security vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	OpenID Connect and OAuth 2.0 are widely adopted and well-maintained protocols with strong community support. By utilizing these protocols for integrating with identity providers and other resource service providers, organizations can benefit from the security features and updates provided by the protocols' implementations. This helps to mitigate the risk of using outdated or known vulnerable components, as the protocols themselves undergo regular security reviews and updates.
7	Identification and Authentication Failures (A07-2021)	OpenID Connect and OAuth 2.0 provide robust mechanisms for user identification and authentication. These protocols support various authentication methods, including multifactor authentication, and enforce secure authentication practices. By implementing OpenID Connect and OAuth 2.0, organizations can reduce the risk of identification and authentication failures, such as weak or compromised user credentials, and enhance the overall security of their web applications.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	OpenID Connect and OAuth 2.0 contribute to addressing software and data integrity failures by providing secure methods for exchanging identity and access tokens. The protocols ensure the integrity and protection of user identities and authorization data during authentication and authorization processes. By utilizing the secure token exchange mechanisms of OpenID Connect and OAuth 2.
9	Security Logging and Monitoring Failures (A09-2021)	It does not address it directly.
10	Server-Side Request Forgery (A10-2021)	It does not address it directly.

11.1.26 Runtime malware security

Runtime container security means vetting all activities within the container application environment, from container and host activity analysis to monitoring network connection protocols and payloads.
This solution can be based on open-sourced antivirus such as ClamAV which is a software accepted by the industry for this type of scenarios.
Adding an anti-virus to the containers may cause conflicts or slow down critical processes. It will require a detailed design to prove the compatibility with other deployed components, and also prove that the interoperability with Kubernetes threat detection engine, in the case the solution considers it.
The solution for runtime malware scanning, can also be based on vulnerability docker container scanning. For this, one potential candidate is the "Dagda Vulnerability Scanner for Docker Containers" project.

In any of the options, the runtime malware security scanning ensures proactive detection and mitigation of malware threats within the container environment. It enhances the security of containerized applications, protects against potential vulnerabilities, and contributes to a more robust and secure runtime environment.
The solution considers the following elements:

#	Name	Description
1	Container Security Platform	Implement a container security platform that provides runtime malware security protection. This platform should have the ability to analyze container and host activities, monitor network connection protocols, and detect and mitigate any potential malware threats.
2	Malware Detection	Integrate a malware detection engine into the container security platform. This engine should be capable of scanning containers for known malware signatures and behaviors. One option is to leverage ClamAV, an open-source antivirus solution known for its effectiveness in detecting and mitigating malware threats.

#	Name	Description
3	Vulnerability Management	Establish a repository or database where all known vulnerabilities are stored. This repository should include information about vulnerabilities, such as their severity, impact, and recommended remediation actions. The container security platform should utilize this repository to identify and address vulnerabilities in container images and runtime environments.
4	Threat Intelligence Integration	Integrate threat intelligence feeds into the container security platform. These feeds provide up-to-date information about emerging malware threats and attack patterns. By incorporating threat intelligence, the platform can proactively detect and respond to new and evolving malware threats.
5	Kubernetes Threat Detection Engine	Ensure compatibility and interoperability between the runtime malware security solution and the proposed Kubernetes threat detection engine. This ensures seamless integration and coordination between the different components of the architecture, allowing for comprehensive threat detection and mitigation capabilities.
6	Container Vulnerability Scanning	Integrate a container vulnerability scanning tool, such as the "Dagda Vulnerability Scanner for Docker Containers," into the architecture. This tool can analyze container images and runtime environments to identify known vulnerabilities and provide recommendations for remediation.
7	Real-Time Monitoring and Alerting	Implement real-time monitoring of container activities and network connections. This allows for immediate detection of any suspicious or malicious behavior within the container environment. The container security platform should generate alerts and notifications to relevant stakeholders when potential malware threats are detected.
8	Incident Response and Remediation	Develop an incident response plan and procedures to handle malware incidents. This includes defining the steps to be taken when a malware threat is identified, isolating affected containers, mitigating the impact, and restoring normal operations. Regularly test and update the incident response plan to ensure its effectiveness.
9	Security Auditing and Reporting	Enable auditing and reporting capabilities within the container security platform. This allows for the generation of security reports, including information on detected malware threats, vulnerabilities, and overall security posture. These reports help assess the effectiveness of the runtime malware security measures and support compliance requirements.

11.1.26.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	Secure Architecture The runtime malware security component indirectly contributes to the Secure Architecture activity in the SAMM Security by Design domain. By providing runtime malware security protection within the container application environment, it helps establish a secure architecture for the software system. This contributes to addressing the Insecure Design OWASP threat by minimizing potential vulnerabilities and creating a strong foundation for security controls. Threat Modeling The runtime malware security component indirectly contributes to the Threat Modeling activity in the SAMM Security by Design domain. By analyzing container and host activities and monitoring network connection protocols and payloads, it helps identify potential security risks and threats associated with malware. This information can be used during threat modeling exercises to assess the impact of malware-related threats and implement appropriate security controls and countermeasures.
3	SAMM Security by Implementation	It does not address it directly.
4	SAMM Security by Verification	Security Testing The runtime malware security component contributes to the Security Testing activity in the SAMM Security by Verification domain. By providing runtime malware security protection and monitoring network connection protocols and payloads, it enables organizations to conduct security testing activities to detect and address malware vulnerabilities. This component helps organizations verify the effectiveness of their security controls and ensure the security of the software throughout its lifecycle.
5	SAMM Security by Operations	Security Testing The runtime malware security component contributes to the Security Testing activity in the SAMM Security by Operations domain. It enables organizations to conduct security testing activities during the runtime of the software systems. By monitoring network connection protocols and payloads, and analyzing container and host activities, it helps identify and mitigate potential malware threats. This component ensures that security testing is performed to detect and address runtime malware vulnerabilities and weaknesses

11.1.26.2 Assessment – OWASP Top 10

The Architecture Design Component's capability of runtime malware security contributes to addressing several OWASP threats. Here's how it aligns with each threat:

#	Name	Description
1	Broken Access Control (A01-2021)	Runtime malware security helps prevent unauthorized access and privilege escalation by monitoring activities within the container application environment, ensuring that proper access controls are enforced.
2	Cryptographic Failures (A02-2021)	While runtime malware security is not directly related to cryptographic failures, it can indirectly contribute to mitigating such vulnerabilities by ensuring the integrity of the container environment. By preventing malware attacks, the solution helps maintain the confidentiality and integrity of data protected by cryptographic algorithms.
3	Injection (A03-2021)	Runtime malware security can detect and prevent injection attacks by monitoring network connection protocols and payloads. It helps identify and block malicious code or unintended commands injected into the web application, mitigating the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	Runtime malware security, as a part of the overall architecture design, contributes to addressing insecure design vulnerabilities by providing a layer of protection against malicious activities within the container environment. It helps ensure that the application's design and architecture are not compromised by malware or unauthorized activities.
5	Security Misconfiguration (A05-2021)	Runtime malware security aids in preventing security misconfigurations by actively monitoring the container and host activity. It helps detect and mitigate any misconfigurations that could lead to insecure settings, exposed sensitive information, or unnecessary functionality, reducing the risk of security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	While not directly addressing vulnerable and outdated components, runtime malware security indirectly contributes to their mitigation. By continuously monitoring activities within the container application environment, the solution can help identify and prevent exploitation attempts targeting known vulnerabilities in components used within the containerized application.
7	Identification and Authentication Failures (A07-2021)	Runtime malware security does not directly address identification and authentication failures. Its primary focus is on detecting and preventing malware activities within the container environment. Proper implementation of identification and authentication mechanisms should be handled separately to mitigate these vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Runtime malware security helps ensure the integrity of software components and data within the container environment by actively monitoring for any unauthorized modifications, manipulations, or destruction. By detecting and mitigating such activities, it helps address software and data integrity vulnerabilities.
9	Security Logging and Monitoring Failures (A09-2021)	Runtime malware security, as a part of the overall security architecture, contributes to addressing security logging and monitoring failures. By providing monitoring capabilities for container and host activity analysis, it helps ensure that security-related logs are collected and analyzed effectively, enabling timely detection and response to security incidents.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Runtime malware security does not directly address server-side request forgery vulnerabilities. Its primary focus is on detecting and preventing malware activities within the container environment. Implementing input validation, secure configurations, and regular security testing are more relevant in mitigating SSRF vulnerabilities.

11.1.27 Configuration Management

All projects must have a proper source code life-cycle management defined. The configuration management capability consist in version control. A tool that tracks and manages changes to software code. The tool allows software teams to manage changes to source code over time.

The software integrates with the DevOps CI/CD platform, reduces development time, and increases successful deployments. The Source Code Management keeps track of every modification to the code and relates these changes to the business requirements. If any mistake is made, developers can return the version deployed in an environment to earlier versions.

The source code life-cycle management capability ensures efficient code collaboration, version control, and traceability throughout the software development process. It enhances development productivity, promotes code quality, and enables effective management of code changes across environments and deployments.

The source code life-cycle management, involves incorporating the following elements:

#	Name	Description
1	Version Control System	Implement a robust version control system as part of the configuration management capability. This system allows software teams to track, manage, and control changes to the source code over time. It provides a centralized repository for storing code and ensures that developers have access to the latest version of the codebase.
2	Branching and Merging Strategies	Define branching and merging strategies within the version control system. These strategies determine how code changes are managed in different development environments, such as feature branches, release branches, and main branches. By following established branching and merging practices, teams can effectively collaborate and manage code changes across the software development life cycle.
3	Code Review and Approval Workflow	Establish a code review and approval workflow within the source code management process. This ensures that all code changes undergo a review process by peers or designated reviewers. Code reviews help identify potential issues, improve code quality, and ensure compliance with coding standards and best practices.
4	Integration with CI/CD Platform	Integrate the source code management tool with the DevOps CI/CD platform. This integration allows for seamless code integration, continuous integration, and automated deployment processes. Developers can trigger builds and deployments directly from the version control system, reducing development time and increasing the success rate of deployments.

#	Name	Description
5	Auditing and Traceability	Enable auditing and traceability features within the source code management tool. This allows for tracking and recording every modification made to the code, including the author, timestamp, and associated changeset. Auditing and traceability support compliance requirements, facilitate troubleshooting, and provide a historical record of code changes for future reference.
6	Rollback and Versioning	Implement mechanisms for code rollback and versioning. In case of mistakes or issues introduced in the code, developers should be able to revert to previous versions of the code that were deployed in specific environments. This ensures that the software can be rolled back to a known stable state, reducing the impact of errors and enabling rapid recovery.
7	Integration with Issue Tracking System	Integrate the source code management tool with an issue tracking system or project management tool. This integration allows for seamless linking between code changes and associated tasks, bugs, or user stories. Developers can easily navigate between code changes and the corresponding requirements, facilitating collaboration and ensuring alignment between code modifications and business requirements.
8	Security and Access Control	Implement appropriate security measures and access controls within the source code management system. This includes defining user roles, permissions, and authentication mechanisms to ensure that only authorized individuals can access and modify the codebase. Regularly review and update access controls to maintain the integrity and confidentiality of the source code.

11.1.27.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards The use of Source Code Management (GitHub) contributes to this activity by providing a platform to implement and enforce secure coding practices, vulnerability management, and incident response policies and standards. Developers can track and manage changes to source code, ensuring compliance with established security requirements. Risk Management Source Code Management helps address this activity by allowing organizations to track and manage changes to source code, which enables better risk assessment and management. By maintaining a version history of the code, organizations can identify potential risks associated with code changes and take appropriate mitigation measures. Compliance Management Source Code Management supports compliance management by providing a centralized repository to store source code and related documentation. It facilitates compliance assessments, regulatory reporting, and alignment with security frameworks and guidelines by ensuring proper version control and traceability of code changes. Security Training and Awareness

#	Name	Description
		Although Source Code Management itself does not directly contribute to this activity, it enables security training and awareness by providing a platform for collaboration and knowledge sharing among developers. Developers can access code repositories, review changes, and learn from each other's secure coding practices, fostering a culture of security within the organization. Security Metrics and Reporting Source Code Management indirectly supports this activity by capturing and tracking code modifications. It enables the measurement of key security metrics related to code changes, such as the number of vulnerabilities addressed or introduced, which can be used for reporting and decision-making purposes. Security Roles and Responsibilities Source Code Management helps address this activity by providing role-based access controls and permissions. Organizations can assign specific roles and responsibilities to individuals within the Source Code Management platform, ensuring accountability for software security and granting appropriate access privileges based on job roles.
2	SAMM Security by Design	Security Requirements Source Code Management supports this activity by allowing organizations to define and incorporate security requirements into the development process. It provides a platform to track and manage changes to source code based on identified security requirements, ensuring that the code aligns with the specified security objectives. Secure Architecture Source Code Management contributes to this activity by enabling the adoption and enforcement of secure architectural patterns and principles. It helps ensure that code changes adhere to the established secure architecture guidelines, minimizing potential vulnerabilities and promoting a robust foundation for security controls. Secure Coding Practices Source Code Management directly contributes to this activity by facilitating the adoption and enforcement of secure coding practices. It provides developers with a platform to collaborate, review code changes, and ensure adherence to secure coding guidelines, reducing the likelihood of introducing vulnerabilities. Threat Modeling Although Source Code Management does not directly address this activity, it can support threat modeling efforts by providing a centralized repository for storing threat models and associated code changes. Threat modeling results can be documented and tracked alongside code modifications, improving visibility and traceability of security considerations. Security Testing Source Code Management indirectly supports this activity by integrating with security testing tools and processes. It allows organizations to incorporate security testing activities, such as static code analysis and security code reviews, into the development workflow, facilitating the identification and resolution of vulnerabilities during the design phase.
3	SAMM Security by Implementation	Secure Development Training GitHub supports providing training and awareness programs to developers on secure coding practices, contributing to building secure software from the early stages of development. Secure Architecture

#	Name	Description
		GitHub encourages the use of secure architecture practices, including threat modeling and secure design patterns, helping organizations address OWASP threats by identifying and addressing potential vulnerabilities early on. Secure Coding Guidelines GitHub enables the establishment and adoption of secure coding guidelines and standards, providing developers with recommendations to mitigate vulnerabilities associated with OWASP threats. Security Requirements GitHub helps organizations define and document security requirements, providing a basis for implementing security controls and addressing OWASP threats. Security Testing Integration GitHub facilitates the integration of security testing activities throughout the software development life cycle, aiding in identifying and addressing vulnerabilities related to OWASP threats. Security Verification GitHub supports security verification techniques to validate the effectiveness of security controls, ensuring that the software system meets desired security objectives. Security Architecture Review GitHub assists in conducting security architecture reviews to evaluate the design and implementation of security controls, helping identify and mitigate potential vulnerabilities or weaknesses. Security Operations Integration GitHub emphasizes the integration of security considerations into operational processes, contributing to incident management, vulnerability management, and secure deployment practices.
4	SAMM Security by Verification	Security Testing GitHub facilitates various types of security testing, such as penetration testing and vulnerability scanning, to identify vulnerabilities and weaknesses in software applications. Code Review GitHub supports reviewing the source code for security flaws and adherence to secure coding practices, aiding in addressing OWASP threats. Security Architecture Review GitHub assists in assessing the security architecture of software applications, ensuring proper security controls and mechanisms are in place to mitigate OWASP threats. Security Requirements Verification GitHub helps verify that security requirements are properly defined and implemented, aiding in addressing OWASP threats. Threat Modeling GitHub supports conducting threat modeling exercises to identify and prioritize security threats, contributing to addressing OWASP threats. Secure Deployment Verification GitHub facilitates verifying secure deployment and configuration of software applications, reducing the risk of OWASP threats during deployment.
5	SAMM Security by Operations	Environment Hardening GitHub helps organizations implement secure configurations and infrastructure practices, protecting software systems from known vulnerabilities and security weaknesses. Secure Build GitHub promotes secure build practices, including secure coding standards and tools, reducing the likelihood of introducing vulnerabilities during the build process. Configuration Management

#	Name	Description
		GitHub supports effective configuration management practices, enabling organizations to maintain software system integrity and address unauthorized changes or misconfigurations. Vulnerability Management GitHub assists in implementing a robust vulnerability management process, including scanning and assessment, aiding in identifying and addressing vulnerabilities. Incident Management GitHub contributes to a well-defined incident management process, including detection, response, containment, and recovery activities, minimizing the impact of security incidents. Secure Deployment GitHub facilitates secure deployment practices, ensuring consistency and adherence to approved standards in the organisation.

11.1.27.2 Assessment – OWASP Top 10

The Configuration Management capability that provides Source code life-cycle management to projects may not directly address all OWASP threats, it provides essential support for implementing secure coding practices, maintaining proper access controls, and facilitating secure development processes.it can help in addressing the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The capability indirectly addresses this threat by providing version control and tracking of source code changes. Proper configuration management ensures that access controls related to source code are maintained and enforced, reducing the risk of unauthorized access or privilege escalation.
2	Cryptographic Failures (A02-2021)	While the capability itself may not directly address cryptographic failures, it plays a supporting role in ensuring secure development practices. By integrating with the DevOps CI/CD platform, the capability facilitates the implementation of secure coding practices, including proper usage of cryptographic algorithms and mechanisms. This indirectly reduces the risk of cryptographic failures.
3	Injection (A03-2021)	The capability indirectly addresses injection vulnerabilities by enabling source code management. By tracking and managing changes to source code, it promotes secure coding practices, including input validation, sanitization, and parameterization. These practices help prevent injection attacks by ensuring untrusted data is handled correctly.
4	Insecure Design (A04-2021)	The capability indirectly addresses insecure design vulnerabilities by promoting proper architecture and design practices. With configuration management, development teams can maintain an organized and controlled structure for source code, which facilitates secure design decisions and architectural considerations from the early stages. This reduces the risk of insecure design choices

#	Name	Description
5	Security Misconfiguration (A05-2021)	The capability directly addresses security misconfiguration vulnerabilities by providing a controlled approach to managing source code configurations. By ensuring that the source code is properly configured and aligned with secure practices, the capability reduces the risk of exposing sensitive information, enabling unnecessary functionality, or using insecure default settings.
6	Vulnerable and Outdated Components (A06-2021)	The capability indirectly addresses this threat by promoting secure coding practices and integration with the DevOps CI/CD platform. It allows developers to manage dependencies and update software components effectively. By keeping the source code up to date with security patches and avoiding the use of known vulnerable components, the capability reduces the risk of exploiting vulnerabilities associated with outdated or insecure software components.
7	Identification and Authentication Failures (A07-2021)	The capability indirectly addresses this threat by providing a controlled approach to source code management. By enforcing proper access controls and authentication mechanisms within the source code itself, it helps prevent identification and authentication failures in web applications.
8	Software and Data Integrity Failures (A08-2021)	The capability indirectly addresses software and data integrity failures by promoting secure coding practices. By maintaining a version-controlled and auditable history of source code changes, the capability helps ensure the integrity and protection of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	Although the capability itself does not directly address logging and monitoring, it contributes to the overall security posture of web applications. By enforcing proper source code management and ensuring secure coding practices, it allows for the inclusion of logging and monitoring capabilities within the codebase. This facilitates effective security logging and monitoring practices, helping detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	The capability indirectly addresses this threat by promoting secure coding practices. By enforcing proper input validation and sanitization within the source code, it helps prevent server-side request forgery vulnerabilities. Additionally, by integrating with the DevOps CI/CD platform, it facilitates regular security testing, which can identify and mitigate SSRF vulnerabilities during the development life cycle.

11.1.28 Hardware Security Modules (HSM)

Hardware Security Modules (HSM) are specialized devices that are designed to securely store and manage cryptographic keys and perform cryptographic operations. HSMs provide a high level of security by leveraging tamper-resistant hardware-based protection, ensuring the confidentiality and integrity of sensitive data.

The HSMs generates cryptographic keys, store them, and manage them within the HSM device. The HSM acts as a secure vault, protecting the keys from unauthorized access and tampering. The keys never leave the HSM in plaintext form, providing an additional layer of security.

The Hardware Security Modules (HSMs) as a security architecture feature enhances the protection of cryptographic keys and ensures the secure execution of cryptographic operations. HSMs provide a trusted and tamper-resistant environment for key management, cryptographic operations, and compliance, contributing to the overall security posture of an organization.

Here are some key aspects and benefits of using HSMs in a security architecture:

#	Name	Description
1	Key Management	HSMs offer a secure environment for generating, storing, and managing cryptographic keys. They provide a robust key lifecycle management process, including key generation, distribution, rotation, and destruction. By centralizing key management within the HSM, organizations can enforce strict access controls and audit trails for key operations.
2	Cryptographic Operations	HSMs are designed to perform cryptographic operations efficiently and securely. These operations include encryption, decryption, digital signing, and verification. By offloading cryptographic operations to the HSM, organizations can benefit from dedicated hardware that is specifically designed to handle these operations, improving performance and reducing the risk of key exposure.
3	Tamper Resistance	HSMs are built with physical security measures to protect against tampering and unauthorized access. They employ techniques such as tamper-evident seals, secure enclosure designs, and active tamper detection mechanisms. If an attempt is made to tamper with the HSM, it can trigger self-destruct mechanisms, rendering the keys and sensitive data unusable.
4	Compliance and Auditing	HSMs play a crucial role in meeting compliance requirements for cryptographic key management. They provide the necessary controls and safeguards to demonstrate compliance with regulations such as PCI-DSS, HIPAA, and GDPR. HSMs also offer comprehensive audit logs, enabling organizations to monitor and track key management activities for compliance and forensic purposes.
5	Secure Application Integration	HSMs provide APIs and cryptographic toolkits that allow applications to securely interact with the device. Applications can request cryptographic operations from the HSM without exposing the sensitive keys or data. This ensures that cryptographic operations are performed in a secure and controlled manner, minimizing the risk of key leakage or misuse.

11.1.28.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards Incorporating Hardware Security Modules (HSMs) in the security architecture aligns with the secure coding practices, vulnerability management, and data protection policies and standards by providing a secure environment for key generation, storage, and management. Risk Management HSMs contribute to risk management by offering tamper-resistant hardware-based protection, ensuring the confidentiality and integrity of sensitive data. Compliance Management HSMs support compliance with regulatory requirements such as PCI-DSS, HIPAA, and GDPR by providing controls and safeguards for cryptographic key management. Security Training and Awareness HSMs can be included in security training programs and awareness campaigns to educate employees and stakeholders about secure key management practices. Security Metrics and Reporting HSMs provide comprehensive audit logs that support security metrics and reporting, enabling organizations to monitor and track key management activities for compliance and forensic purposes. Security Roles and Responsibilities HSMs help enforce accountability for software security by providing a secure and controlled environment for cryptographic operations and key management.
2	SAMM Security by Design	Security Requirements HSMs contribute to addressing security requirements by providing a secure environment for key generation, storage, and management, ensuring the confidentiality and integrity of sensitive data. Secure Architecture Incorporating HSMs into the design phase helps establish a secure architecture by leveraging tamper-resistant hardware-based protection for cryptographic keys and operations. Secure Coding Practices HSMs enhance secure coding practices by offloading cryptographic operations to dedicated hardware designed to handle these operations securely. Threat Modeling HSMs can be considered as part of threat modeling exercises to assess the impact of potential attacks on cryptographic keys and operations. Security Testing HSMs provide a secure and controlled environment for cryptographic operations, which contributes to the resilience of the design against potential attacks.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training HSMs can be integrated into secure development training programs to educate developers on secure key management practices. Secure Architecture Incorporating HSMs into the implementation phase helps establish a secure architecture by leveraging tamper-resistant hardware-based protection for cryptographic keys and operations. Secure Coding Guidelines HSMs enhance secure coding practices by offloading cryptographic operations to dedicated hardware designed to handle these operations securely. Security Requirements HSMs contribute to implementing security requirements by providing a secure environment for key generation, storage, and management. Security Testing Integration HSMs provide a secure and controlled environment for cryptographic operations, which supports security testing activities such as static code analysis and security code reviews. Security Verification HSMs can be used to verify the effectiveness of security controls implemented in the software system by assessing the security of cryptographic key management. Security Architecture Review HSMs can be included in security architecture reviews to evaluate the security of the software system's design and architecture in terms of cryptographic key management. Security Operations Integration HSMs support secure operational practices by providing a trusted and tamper-resistant environment for cryptographic key management and operations.
4	SAMM Security by Verification	Security Testing HSMs can be utilized in security testing activities to assess the security of cryptographic operations and verify the effectiveness of security controls implemented in the software system. Code Review HSMs contribute to code reviews by providing a secure environment for cryptographic operations and ensuring adherence to secure coding practices related to key management. Security Architecture Review HSMs can be included in security architecture reviews to evaluate the security of the software system's design and architecture in terms of cryptographic key management. Security Requirements Verification HSMs support the verification of security requirements by providing a secure environment for key generation, storage, and management. Threat Modeling HSMs can be considered in threat modeling exercises to assess the impact of potential attacks on

#	Name	Description
5	SAMM Security by Operations	Environment Hardening HSMs contribute to environment hardening by securely storing cryptographic keys and protecting them against unauthorized access or compromise, ensuring the security and reliability of software systems. Secure Build HSMs can be considered as part of secure build practices by securely storing cryptographic keys and performing cryptographic operations, minimizing the risk of introducing vulnerabilities during the build process. Configuration Management HSMs provide a secure platform for managing cryptographic keys and supporting secure configurations, helping organizations maintain the integrity of software systems and detect unauthorized changes or misconfigurations. Vulnerability Management HSMs enhance vulnerability management processes by securely generating, storing, and managing cryptographic keys, ensuring the identification and prioritization of vulnerabilities in software applications. Incident Management HSMs contribute to incident management by securely storing cryptographic keys and supporting secure cryptographic operations, enabling organizations to respond effectively to security incidents and minimize their impact. Secure Deployment HSMs play a role in secure deployment practices by securely storing cryptographic keys and ensuring their proper use during software deployment, protecting against unauthorized access or compromise. Security Testing HSMs can be used to securely generate and manage cryptographic keys for security testing activities, such as static code analysis and dynamic application security testing (SAST).

11.1.28.2 Assessment – OWASP Top 10

HSMs may not directly address all OWASP threats. However, it provide a robust foundation for secure key management, cryptographic operations, and secure integration within a security architecture. By leveraging HSMs, organizations can enhance the overall security posture of their systems and mitigate the risks associated with various OWASP threats.

#	Name	Description
1	Broken Access Control (A01-2021)	While HSMs themselves do not directly address this threat, they can contribute to access control by securely storing cryptographic keys and enforcing strict access controls for key operations. This helps in preventing unauthorized access to sensitive data and resources.

#	Name	Description
2	Cryptographic Failures (A02-2021)	HSMs play a crucial role in mitigating cryptographic failures by providing a secure environment for generating, storing, and managing cryptographic keys. HSMs ensure that proper cryptographic algorithms and mechanisms are used, reducing the risk of cryptographic vulnerabilities and strengthening the overall security of the system.
3	Injection (A03-2021)	HSMs do not directly address injection vulnerabilities. However, by securely integrating with applications and providing APIs for cryptographic operations, HSMs contribute to secure application integration. This helps in minimizing the risk of injection attacks by ensuring that cryptographic operations are performed securely and without exposing sensitive keys or data.
4	Insecure Design (A04-2021)	HSMs do not directly address insecure design vulnerabilities. However, they can be incorporated as a secure architectural component within the overall design of a system. By using HSMs, organizations can enforce secure key management practices and protect cryptographic operations, which can help in mitigating the risk associated with insecure design decisions.
5	Security Misconfiguration (A05-2021)	HSMs themselves do not directly address security misconfigurations. However, when properly integrated and configured, HSMs can help organizations adhere to secure configuration practices. HSMs provide a secure environment for key management and cryptographic operations, ensuring that cryptographic components are configured correctly and securely.
6	Vulnerable and Outdated Components (A06-2021)	HSMs do not directly address vulnerabilities in components used within web applications. However, by providing a secure environment for key management and cryptographic operations, HSMs contribute to the overall security of the system. Organizations can use HSMs to protect cryptographic keys and ensure that the cryptographic components they use are not vulnerable or outdated.
7	Identification and Authentication Failures (A07-2021)	HSMs themselves do not directly address identification and authentication failures. However, by securely storing and managing cryptographic keys, HSMs can be utilized to enhance the security of authentication mechanisms. Strong cryptographic keys stored within HSMs can help in ensuring the integrity and confidentiality of user credentials, thereby reducing the risk of authentication failures.
8	Software and Data Integrity Failures (A08-2021)	HSMs do not directly address software and data integrity failures. However, by securely storing cryptographic keys and performing cryptographic operations within a trusted environment, HSMs contribute to the overall integrity of software and data. HSMs help in preventing unauthorized modifications, tampering, or destruction of cryptographic components and data.
9	Security Logging and Monitoring Failures (A09-2021)	HSMs themselves do not directly address security logging and monitoring failures. However, HSMs can generate comprehensive audit logs for key management operations, cryptographic operations, and access controls. These logs can be integrated with centralized logging and monitoring systems, contributing to effective security logging and monitoring practices.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	HSMs do not directly address server-side request forgery vulnerabilities. However, by securely storing and managing cryptographic keys, HSMs can contribute to the overall security of web applications. Strong encryption and secure key management within HSMs can help protect against unauthorized access and manipulation of server-side resources, reducing the risk of server-side request forgery attacks.

11.1.29 Vault – Secret management platform

The Vault is a secret management tool that stores and controls access to sensitive credentials in any environment.

It is used for enhancing the security and control of sensitive credentials. It provides a centralized and secure repository for secrets, enforces access controls, enables dynamic secrets, and facilitates encryption and auditing. Incorporating the Vault into the architecture ensures robust secret management practices and strengthens the overall security posture of the system.

The Vault provides the following features:

#	Name	Description
1	Secret management	The Vault can securely store and manage secrets such as API keys, passwords, and certificates.
2	Dynamic secrets	Vault can generate short-lived dynamic secrets for various services, reducing the risk of long-term exposure.
3	Encryption as a Service	Vault provides a platform for encrypting and decrypting data using various encryption methods.
4	Access control	Vault allows you to control access to secrets by configuring policies for specific users, applications, and environments.
5	Auditing	Vault provides a detailed audit trail of all user and system activity. The Vault is a secret management tool that controls access to sensitive credentials in any environment.
6	Secret Lifecycle Management	The Vault enables secure storage and management of secrets throughout their lifecycle. It provides a central repository for storing sensitive credentials such as API keys, passwords, tokens, and certificates. The platform ensures that secrets are encrypted at rest and during transit, protecting them from unauthorized access.
7	Dynamic Secrets	The Vault offers the capability to generate dynamic secrets on-demand. Instead of using static, long-lived credentials, dynamic secrets are short-lived and automatically generated for specific use cases. This approach minimizes the risk of long-term exposure and unauthorized access to credentials, enhancing security.

#	Name	Description
8	Secret Rotation	The Vault supports secret rotation, which is the process of regularly updating and replacing existing secrets with new ones. By automating secret rotation, the platform helps mitigate the risk associated with compromised or leaked credentials. It ensures that secrets are frequently updated, reducing the window of vulnerability.
9	Encryption as a Service	Vault provides encryption capabilities as a service. It allows applications to encrypt and decrypt sensitive data using various encryption methods and algorithms. This feature ensures that data is protected when stored or transmitted, providing an additional layer of security.
10	Access Control and Role-based Policies	The Vault allows fine-grained access control through role-based policies. Administrators can define policies that specify which users, applications, or environments have access to specific secrets. This ensures that only authorized entities can retrieve and use the sensitive credentials stored in the Vault.
11	Auditing and Logging	The Vault maintains a detailed audit trail of all user and system activities. It logs interactions with secrets, access attempts, and configuration changes, providing visibility into who accessed what secrets and when. Auditing capabilities help with compliance requirements, security investigations, and monitoring the overall system's integrity.
12	Integration with Identity Providers	The Vault can integrate with external identity providers, such as LDAP or OAuth, for user authentication and authorization. This enables centralized user management and ensures that access to secrets is tied to the appropriate user identities and roles.
13	High Availability and Disaster Recovery	The Vault architecture should include provisions for high availability and disaster recovery. This involves deploying Vault instances in a clustered configuration across multiple nodes or regions to ensure fault tolerance and availability. Additionally, regular backups and replication of the Vault's encrypted data should be performed to enable recovery in case of data loss or system failure.

11.1.29.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	Security Requirements The Vault can securely store and manage secrets such as API keys and passwords, which are critical security requirements. Secure Architecture The Vault provides encryption as a service, which contributes to the design of a secure architecture by enabling secure data encryption and decryption. Secure Coding Practices The Vault's secret management capabilities help developers adhere to secure coding practices by securely storing and managing sensitive credentials. Threat Modeling

#	Name	Description
		The Vault's access control and encryption features align with threat modeling principles, ensuring that potential risks associated with credential exposure are mitigated. Security Testing The Vault's auditing feature provides a detailed audit trail of user and system activity, which supports security testing by providing visibility into the status of secrets management.
3	SAMM Security by Implementation	Secure Architecture The Vault's encryption as a service feature contributes to implementing secure architecture practices by providing a platform for encrypting and decrypting data using various encryption methods. Security Requirements The Vault's secret management capabilities align with the implementation of security requirements by securely storing and managing sensitive credentials. Security Testing Integration The Vault's auditing feature and access control capabilities support security testing integration by providing visibility into user and system activity related to secret management. Security Verification The Vault's access control and encryption features can be verified to ensure the effectiveness of security controls implemented in the secret management process. Security Operations Integration The Vault's access control and auditing features support secure operational processes, such as incident management and vulnerability management, by providing control over secrets and maintaining an audit trail of activities.
4	SAMM Security by Verification	Security Testing The Vault's secret management capabilities contribute to security testing by securely storing and managing sensitive credentials and providing an audit trail of user and system activity related to secrets management. Code Review The Vault's secret management platform can be reviewed to assess its adherence to secure coding practices and secure handling of sensitive credentials. Security Architecture Review The Vault's access control and encryption features can be reviewed to assess the security architecture and ensure proper security controls and mechanisms are in place. Security Requirements Verification The Vault's secret management capabilities align with security requirements verification by securely storing and managing sensitive credentials. Threat Modeling The Vault's access control and encryption features can be considered during threat modeling exercises to identify and prioritize security threats related to credential exposure. Secure Deployment Verification The Vault's access control and encryption features can be verified during the secure deployment of software applications to ensure proper implementation and configuration.
5	SAMM Security by Operations	Environment Hardening The Vault's access control and encryption features contribute to environment hardening by protecting sensitive credentials and controlling access to them. Secure Build

#	Name	Description
		The Vault's secret management capabilities align with secure build practices by securely storing and managing sensitive credentials during the software development process. Configuration Management: The Vault's access control and encryption features can be part of the configuration management process to ensure secure handling of sensitive credentials. Vulnerability Management: The Vault's secret management capabilities contribute to vulnerability management by securely storing and managing sensitive credentials, reducing the risk of exposure. Incident Management: The Vault's auditing feature supports incident management by providing a detailed audit trail of user and system activity related to secret management. Secure Deployment: The Vault's access control and encryption features can be integrated into secure deployment practices to ensure the secure distribution and installation of software components. Security Testing: The Vault's auditing feature supports security testing by providing an audit trail of activities related to secret management, aiding in vulnerability detection and mitigation. Operational Enablement: The Vault's access control and encryption features can be used to provide operational support and guidance for secure and reliable software systems.

11.1.29.2 Assessment – OWASP Top 10

The Vault's secret management platform helps address several OWASP threats by securely storing and managing sensitive credentials, enforcing access controls, providing encryption as a service, offering auditing capabilities, and indirectly contributing to the mitigation of various vulnerabilities related to access control, misconfiguration, and data protection. The Vault, as a secret management platform, addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	The Vault helps address broken access control by providing access control mechanisms for secrets. It allows organizations to control access to sensitive credentials by configuring policies for specific users, applications, and environments. This ensures that only authorized entities have access to the secrets, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	While the Vault is not specifically designed to address cryptographic failures, it plays a role in securely managing secrets that are used in cryptographic operations. By securely storing and managing secrets such as encryption keys and certificates, the Vault helps mitigate the risk of cryptographic vulnerabilities that may arise from improper key management or insecure storage of sensitive information.

#	Name	Description
3	Injection (A03-2021)	The Vault does not directly address injection vulnerabilities. However, by securely managing secrets such as API keys and passwords, it contributes to minimizing the risk of injection attacks. By properly storing and controlling access to secrets, the Vault helps ensure that sensitive information is not inadvertently exposed or manipulated through injection vulnerabilities.
4	Insecure Design (A04-2021)	The Vault can indirectly help address insecure design vulnerabilities by providing a secure platform for secret management. By utilizing the Vault's access control mechanisms and policies, organizations can enforce secure design principles and restrict access to sensitive credentials, reducing the attack surface and potential impact of insecure design decisions.
5	Security Misconfiguration (A05-2021)	The Vault directly addresses security misconfiguration by providing a platform for secure secret management. It helps organizations avoid common security misconfiguration pitfalls by offering secure defaults, predefined policies, and access controls for secrets. The Vault's configuration options and auditing capabilities also contribute to identifying and rectifying misconfigurations, thereby improving the overall security posture.
6	Vulnerable and Outdated Components (A06-2021)	The Vault indirectly addresses the risk of vulnerable and outdated components by focusing on secure secret management. It ensures that sensitive credentials are properly protected, reducing the risk of compromised secrets leading to exploitation of vulnerable or outdated components within the application or infrastructure.
7	Identification and Authentication Failures (A07-2021)	The Vault does not directly address identification and authentication failures. However, it contributes to improving the security of identification and authentication processes by securely managing secrets related to authentication mechanisms, such as passwords or authentication tokens. By storing these secrets in a secure manner and enforcing access controls, the Vault helps mitigate the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	While the Vault does not directly address software and data integrity failures, it indirectly contributes to their mitigation. By providing encryption as a service and securely managing encryption keys, the Vault helps protect data integrity by ensuring that encrypted data remains confidential and tamper-proof.
9	Security Logging and Monitoring Failures (A09-2021)	The Vault directly addresses security logging and monitoring failures by providing detailed audit trails of all user and system activities related to secret management. The Vault's auditing capabilities help organizations monitor access to secrets, detect suspicious activities, and maintain comprehensive logs for forensic analysis, contributing to effective security logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	The Vault does not directly address server-side request forgery vulnerabilities. However, by securely managing secrets and controlling access to sensitive resources, it can help mitigate the risk of SSRF attacks. By storing credentials and sensitive information within the Vault and enforcing proper access controls, organizations can reduce the likelihood of SSRF vulnerabilities leading to unauthorized access to internal resources.

11.1.30 Privacy by Design

Privacy by Design is a security feature that promotes the integration of privacy considerations into the architecture of a system from its inception. It aims to ensure that privacy is built-in rather than added as an afterthought.

Consist in integrating these privacy-enhancing features into the architecture. Organizations can demonstrate a commitment to privacy and protect individuals' personal information throughout the system's lifecycle. This helps foster trust, compliance with privacy regulations, and ensures that privacy is a fundamental aspect of the system's design and operation.

Here are some additional details on privacy-enhancing technologies, obtaining user consent, and data anonymization/pseudonymization techniques that can be integrated into the architecture:

Privacy-enhancing technologies (PETs) are tools and techniques designed to protect individual privacy and enhance data security. These technologies can be integrated into the system architecture to safeguard personal information. Some examples of PETs include:

#	Name	Description
1	Encryption	Encrypting data ensures that it is protected and unreadable to unauthorized parties. Strong encryption algorithms and secure key management can be implemented to protect sensitive data both in transit and at rest.
2	Anonymization and Pseudonymization	Anonymization involves removing or obfuscating personally identifiable information (PII) from data sets to prevent the identification of individuals. Pseudonymization replaces direct identifiers with pseudonyms, allowing data to be linked to an individual only with additional information held separately. These techniques can help minimize privacy risks while still allowing data analysis.
3	Tokenization	Tokenization replaces sensitive data with non-sensitive tokens, which are then used to perform operations without exposing the actual data. Tokenization helps protect sensitive information, as the tokens do not reveal any meaningful data to unauthorized parties.
4	Differential Privacy	Differential privacy is a technique that adds statistical noise to query responses or data sets, ensuring that individual data points cannot be distinguished or re-identified. It provides a privacy guarantee while still allowing useful data analysis.
5	User Consent and Transparency	Obtaining user consent is crucial for collecting and processing personal data. The system architecture should include mechanisms to inform users about the data being collected, how it will be used, and any third parties involved. User-friendly interfaces can be implemented to present clear and easily understandable privacy policies and consent options. Additionally, mechanisms for users to manage their consent preferences and exercise their rights, such as the right to access, rectify, and delete their data, should be provided.

#	Name	Description
6	Data Minimization	Data minimization is the principle of collecting and storing only the necessary data for a specific purpose. By reducing the amount of personal data collected, processed, and stored, the privacy risks are minimized. The system architecture should include mechanisms to identify and discard unnecessary data and ensure that data retention policies comply with privacy requirements.
7	Privacy Impact Assessments	Privacy Impact Assessments (PIAs) can be conducted during the design and development stages of the system to identify potential privacy risks and mitigation strategies. PIAs help ensure that privacy considerations are taken into account and privacy controls are integrated into the architecture from the beginning.
8	Secure Data Sharing and Consent Management	If the system involves sharing personal data with third parties, secure data sharing protocols and frameworks can be implemented to protect the privacy of the data during transmission and enforce access controls. Consent management systems can be utilized to track and manage user consent across different data processing activities and third-party collaborations.

11.1.30.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	Security Requirements Privacy-enhancing technologies, user consent mechanisms, and data minimization techniques help address security requirements related to privacy concerns. Secure Architecture Incorporating privacy-enhancing features into the architecture helps address secure architectural design principles and mitigate potential vulnerabilities related to privacy. Secure Coding Practices Implementing privacy-enhancing technologies and techniques aligns with secure coding practices by protecting personal information and preventing privacy-related vulnerabilities. Threat Modeling Considering privacy risks and incorporating privacy controls into threat modeling exercises helps identify and mitigate potential threats to privacy. Security Testing Privacy-related features, such as data anonymization and pseudonymization, can be included in security testing activities to ensure the resilience of privacy measures in the design.
3	SAMM Security by Implementation	Secure Development Training Incorporating privacy-enhancing technologies and techniques into secure development training programs helps developers understand the importance of privacy and how to implement privacy measures. Secure Architecture Privacy-enhancing features integrated into the architecture support secure architectural practices, such as secure design patterns and secure component selection, by addressing privacy concerns. Secure Coding Guidelines

#	Name	Description
		Including guidelines for implementing privacy-enhancing technologies and techniques in secure coding practices helps developers write code that protects personal information and respects privacy requirements. Security Requirements Privacy requirements can be defined and documented as part of the software development process, ensuring that privacy concerns are considered throughout the implementation phase. Security Testing Integration Privacy-related features, such as data anonymization and pseudonymization, can be included in security testing activities to validate their effectiveness and identify any vulnerabilities.
4	SAMM Security by Verification	Security Testing Privacy-enhancing technologies and techniques, such as data anonymization and pseudonymization, can be tested to ensure their effectiveness in protecting personal information and maintaining privacy. Code Review Privacy-related features and implementations can be reviewed to assess their adherence to secure coding practices and verify the proper handling of personal information. Security Architecture Review Privacy-enhancing features and controls integrated into the software architecture can be reviewed to ensure their effectiveness in protecting privacy and complying with privacy requirements. Security Requirements Verification Privacy requirements can be verified against industry standards and best practices to ensure their proper implementation and alignment with privacy regulations. Threat Modeling Threat modeling exercises can consider privacy threats and risks to evaluate the impact on personal information and identify appropriate security controls and countermeasures. Secure Deployment Verification: Privacy measures implemented during deployment, such as secure transmission protocols for personal data, can be verified to ensure their proper configuration and effectiveness in protecting privacy.
5	SAMM Security by Operations	Environment Hardening Implementing privacy-enhancing features and controls as part of the software environment hardening process helps protect personal information from unauthorized access and maintain privacy. Secure Build Integrating privacy-enhancing technologies and techniques into the secure build practices helps ensure that the software is developed and built with privacy in mind. Configuration Management Properly configuring and managing privacy-related settings and controls are essential for maintaining the privacy of personal information throughout the software's operational life cycle. Vulnerability Management Privacy-related vulnerabilities and weaknesses can be included in vulnerability management processes to identify and address potential privacy breaches and protect personal information. Incident Management Privacy incidents and breaches involving personal information should be handled through incident management processes to ensure timely response, containment, and recovery while minimizing the impact on privacy.

11.1.30.2 Assessment – OWASP Top 10

Overall, Privacy by Design is a crucial capability that integrates privacy considerations into the architecture from the beginning, indirectly addressing several OWASP threats while also directly addressing Broken Access Control, Security Misconfiguration, Identification and Authentication Failures, and Security Logging and Monitoring Failures.

#	Name	Description
1	Broken Access Control (A01-2021)	Privacy by Design emphasizes integrating privacy considerations into the architecture from the beginning, which includes access controls. By designing and implementing robust access controls as part of the architecture, organizations can prevent unauthorized access and enforce proper user privileges, addressing the Broken Access Control threat.
2	Cryptographic Failures (A02-2021)	Privacy by Design emphasizes the integration of privacy-enhancing technologies, including encryption, into the system architecture. Proper implementation of encryption techniques and secure key management helps protect sensitive data and prevents cryptographic failures.
3	Injection (A03-2021)	While Privacy by Design doesn't directly address injection vulnerabilities, it promotes secure coding practices and input validation, which are crucial for mitigating injection attacks. By designing the architecture with secure coding principles, organizations can reduce the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	Privacy by Design advocates for secure design principles and considerations, which indirectly contribute to addressing insecure design vulnerabilities. By incorporating security into the architecture from the early stages, organizations can mitigate design flaws that could lead to security vulnerabilities.
5	Security Misconfiguration (A05-2021)	Privacy by Design promotes secure configurations as an essential aspect of the system architecture. By following secure configuration practices, organizations can reduce the risk of security misconfigurations, which can expose sensitive information and create potential entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	Privacy by Design doesn't directly address this threat but can indirectly contribute to mitigating it. By adopting privacy-conscious practices during the selection and integration of components, organizations can minimize the risk of using outdated or vulnerable software components within the system architecture.
7	Identification and Authentication Failures (A07-2021)	Privacy by Design emphasizes the implementation of secure authentication mechanisms as part of the architecture. By designing and enforcing strong authentication mechanisms, organizations can mitigate the risk of identification and authentication failures, preventing unauthorized access and impersonation of legitimate users.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	Privacy by Design promotes secure coding techniques and practices, which indirectly contribute to addressing software and data integrity failures. By implementing secure coding principles, organizations can reduce the risk of unauthorized modifications, data tampering, and other integrity-related issues.
9	Security Logging and Monitoring Failures (A09-2021)	Privacy by Design advocates for effective logging and monitoring practices. By integrating comprehensive logging capabilities and real-time monitoring mechanisms into the system architecture, organizations can detect and respond to security incidents, including those related to privacy breaches.
10	Server-Side Request Forgery (A10-2021)	Privacy by Design doesn't directly address this threat, but it emphasizes secure configurations and robust input validation. By implementing secure configurations and properly validating user input, organizations can mitigate the risk of server-side request forgery vulnerabilities.

11.1.31 Secure Data at rest security

The best possible solution if fits the requirements is to abstract the data encryption to applications. So, Data at rest must be encrypted transparently whenever is possible. For example, adopting products or platforms that provides encryption 256-bit AES or higher.

In the potential scenario, an unauthorized actor copies the data on the disk without using the correct APIs that provide transparent encryption and decryption functionality. In that case, the information is locked without any possibility of being seen by the naked eye.

The cryptographic key used for the encryption is stored in a Key Vault and managed using an agreed security protocol to prevent anyone without proper permission from accessing it.

Implementing secure data at rest security measures, organizations can protect sensitive data even when it is stored or copied on physical storage media. Transparent encryption, secure key management, agreed security protocols, and protection against unauthorized access help maintain the confidentiality and integrity of the data. Additionally, incorporating data resilience, compliance, and auditing mechanisms strengthens the overall security posture of the system.

The capability of secure data at rest security involves the following considerations:

#	Name	Description
1	Transparent Encryption	Data at rest should be encrypted using strong encryption algorithms such as 256-bit AES (Advanced Encryption Standard). Transparent encryption means that the encryption and decryption processes are handled seamlessly by the underlying storage system or database without requiring any additional effort from the application or end-user. This ensures that sensitive data remains protected even if it is copied or accessed directly from the storage medium.

#	Name	Description
2	Key Management	The cryptographic keys used for data encryption should be securely managed. Storing the encryption key in a Key Vault is a recommended practice. A Key Vault is a secure storage location that provides centralized key management, ensuring the confidentiality and integrity of the encryption keys. Access to the Key Vault should be tightly controlled using access controls, authentication mechanisms, and encryption protocols to prevent unauthorized access to the keys.
3	Security Protocols	An agreed security protocol should be established for managing and accessing the encryption keys stored in the Key Vault. This protocol should outline the procedures and policies for key generation, distribution, rotation, revocation, and backup. It should also define the roles and responsibilities of individuals involved in key management, ensuring that only authorized personnel have access to the keys.
4	Protection against Unauthorized Access	The system should employ measures to prevent unauthorized access to the encrypted data. This can include implementing access controls, authentication mechanisms, and authorization policies to restrict access to the data based on user roles and permissions. Additionally, mechanisms such as secure boot, secure file system permissions, and file integrity monitoring can be implemented to detect and prevent unauthorized modifications to the encrypted data.
5	Data Resilience and Disaster Recovery	To ensure data resilience and availability, appropriate backup and disaster recovery mechanisms should be implemented. Regular backups of the encrypted data should be taken and securely stored. These backups should also be encrypted to maintain the security of the data at rest. In the event of a data loss or system failure, the backup data can be restored using the encryption keys stored in the Key Vault.
6	Compliance and Auditing	The secure data at rest solution should support compliance requirements by providing mechanisms for auditing and monitoring access to the encrypted data. This includes maintaining detailed logs of data access and modifications, ensuring data integrity, and enabling compliance audits. These logs can be used to track any unauthorized access attempts or suspicious activities, assisting in security investigations and ensuring accountability.

11.1.31.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards The encryption of data at rest contributes to the data protection aspect of policies and standards, ensuring that sensitive information is adequately protected. Risk Management By encrypting data at rest and storing the cryptographic key in a Key Vault, organizations mitigate the risk of unauthorized access to sensitive data. Compliance Management Implementing transparent encryption and proper key management aligns with compliance requirements related to data protection and privacy regulations. Security Training and Awareness Educating employees about the importance of data encryption and proper handling of cryptographic keys helps create a culture of security and awareness. Security Metrics and Reporting Monitoring the implementation and effectiveness of data encryption measures can be included in security metrics and reporting to assess compliance and risk management. Security Roles and Responsibilities Clearly defining and assigning responsibilities for data protection, including encryption and key management, ensures accountability and appropriate handling of security controls.
2	SAMM Security by Design	Security Requirements Transparent encryption of data at rest addresses the security requirement of protecting sensitive information from unauthorized access. Secure Architecture Incorporating transparent encryption mechanisms into the software architecture ensures data confidentiality, addressing various OWASP threats such as Cryptographic Failures and Insecure Design. Secure Coding Practices Adhering to secure coding practices includes implementing encryption and decryption functionality correctly to prevent vulnerabilities associated with OWASP threats. Threat Modeling Considering threats related to unauthorized data access and ensuring appropriate countermeasures, such as encryption, mitigates risks associated with OWASP threats. Security Testing Including security testing techniques, such as validating the correct implementation of encryption and decryption, helps identify vulnerabilities and ensures the resilience of the design against potential attacks.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training Educating developers on the proper implementation of data encryption and key management contributes to building secure software systems. Secure Architecture Implementing secure architectural patterns and principles, including transparent data encryption, minimizes potential vulnerabilities and strengthens security controls. Secure Coding Guidelines Adhering to secure coding guidelines ensures that encryption and key management practices are correctly implemented, reducing the likelihood of OWASP threats. Security Requirements Including data encryption as a security requirement ensures that the implementation phase addresses the protection of sensitive data at rest. Security Testing Integration Incorporating security testing activities, such as testing the effectiveness of data encryption and decryption, helps identify and address vulnerabilities associated with OWASP threats. Security Verification Verifying the implementation of encryption and key management mechanisms ensures that security controls meet the desired security objectives. Security Architecture Review Assessing the implementation of encryption and key management in the software system's architecture helps identify vulnerabilities and provides recommendations for improvement. Security Operations Integration Integrating encryption and key management practices into operational processes ensures ongoing protection of data at rest.
4	SAMM Security by Verification	Security Testing Conducting security testing activities, such as vulnerability scanning and penetration testing, helps identify vulnerabilities and weaknesses in software applications, including those related to data encryption. Code Review Reviewing the source code for proper implementation of encryption and adherence to secure coding practices ensures the effectiveness of encryption mechanisms. Security Architecture Review Assessing the security architecture helps ensure the correct implementation and use of encryption techniques to protect data at rest. Security Requirements Verification Verifying the implementation of encryption requirements ensures that data protection measures are properly integrated into the software applications. Threat Modeling Including data encryption as part of the threat modeling process helps identify potential threats and risks to data at rest and enables appropriate countermeasures. Secure Deployment Verification Verifying secure deployment practices includes ensuring the proper configuration and use of encryption mechanisms during software deployment.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening By encrypting data at rest using 256-bit AES encryption, organizations can ensure that sensitive data remains protected even if unauthorized actors gain access to the physical storage devices. Configuration Management Storing the cryptographic key used for encryption in a Key Vault and managing it using an agreed security protocol helps prevent unauthorized access to the key, ensuring the confidentiality of the data at rest. Vulnerability Management Implementing transparent encryption and decryption functionality using the correct APIs helps mitigate the risk of unauthorized actors copying the data without encryption, minimizing the potential for data breaches.

11.1.31.2 Assessment – OWASP Top 10

Secure Data at rest security capability directly addresses several OWASP threats, including Broken Access Control, Cryptographic Failures, Insecure Design, Security Misconfiguration, Injection, Software and Data Integrity Failures, and indirectly contributes to mitigating other threats by promoting secure data storage practices.

#	Name	Description
1	Broken Access Control (A01-2021)	The use of transparent encryption and decryption functionality ensures that data at rest is protected and inaccessible to unauthorized actors, mitigating the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	By adopting 256-bit AES encryption and securely managing the cryptographic key in a Key Vault, the Architecture Design Component mitigates the risk of cryptographic failures and weaknesses in encryption implementation.
3	Injection (A03-2021)	Although the Architecture Design Component primarily focuses on securing data at rest, it indirectly helps mitigate injection vulnerabilities by preventing unauthorized actors from accessing the data on the disk without using the correct APIs. This helps protect against injection attacks that rely on manipulating data storage or retrieving sensitive information.
4	Insecure Design (A04-2021)	The Architecture Design Component contributes to addressing insecure design vulnerabilities by ensuring that secure data storage and encryption mechanisms are incorporated into the overall design and architecture of the system. This helps establish a more robust and secure foundation for the application.

#	Name	Description
5	Security Misconfiguration (A05-2021)	The Architecture Design Component helps mitigate security misconfiguration vulnerabilities by providing a standardized and secure approach to data at rest security. By ensuring that data is encrypted transparently and that the cryptographic key is stored and managed securely, the component reduces the risk of misconfigurations that could expose sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	While the Architecture Design Component does not directly address vulnerabilities related to components, it indirectly contributes to reducing the risk of vulnerable and outdated components. By focusing on secure data storage and encryption, it promotes the use of secure practices and technologies, which can also extend to the selection and management of components used for data protection.
7	Identification and Authentication Failures (A07-2021)	The Architecture Design Component, specifically its emphasis on securing data at rest, does not directly address identification and authentication vulnerabilities. However, it complements these security measures by ensuring that sensitive data is adequately protected, thereby reducing the potential impact of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	The Architecture Design Component significantly contributes to addressing software and data integrity failures. By encrypting data at rest using 256-bit AES encryption and securely managing the cryptographic key, it protects against unauthorized modifications, data tampering, and other integrity-related issues.
9	Security Logging and Monitoring Failures (A09-2021)	The Architecture Design Component does not directly address security logging and monitoring failures. However, by incorporating secure data storage practices, it enables better traceability and accountability, facilitating effective logging and monitoring of access to and modifications of data at rest.
10	Server-Side Request Forgery (A10-2021)	While the Architecture Design Component primarily focuses on securing data at rest, it indirectly helps mitigate server-side request forgery vulnerabilities by preventing unauthorized actors from accessing sensitive data through unintended requests. By encrypting the data and enforcing proper access controls, the component reduces the risk of SSRF attacks that aim to retrieve or manipulate stored information.

11.1.32 Secure Communication Protocols

Utilizing secure communication protocols, such as Transport Layer Security (TLS), is a crucial security feature that helps ensure the confidentiality and integrity of data transmitted over networks. These protocols are designed to protect data during transmission by establishing a secure and encrypted channel between communicating entities. The most widely used secure communication protocol is Transport Layer Security (TLS), which is the successor to Secure Sockets Layer (SSL). TLS provides a robust and standardized framework for securing network communications, including web traffic, email transmissions, and other network-based protocols.

Implementing secure communication protocols like TLS, organizations can achieve a higher level of security for their network communications. This is particularly important when transmitting sensitive or

confidential data, such as financial information, personal data, or business-critical information. Secure communication protocols help protect against eavesdropping, data manipulation, and unauthorized access, significantly reducing the risk of data breaches and privacy violations.

It is important to note that implementing secure communication protocols is not a one-time task. Organizations need to stay updated with the latest versions and security patches of the chosen protocol (e.g., TLS 1.3) to address vulnerabilities and weaknesses that may be discovered over time. Regular monitoring and maintenance of the secure communication infrastructure are essential to ensure ongoing protection of data during transmission.

When secure communication protocols are implemented, the following key aspects are addressed:

#	Name	Description
1	Confidentiality	Secure communication protocols encrypt data during transmission, preventing unauthorized entities from intercepting and understanding the content. Encryption ensures that even if an attacker captures the transmitted data, they will not be able to decipher it without the encryption key.
2	Integrity	Secure communication protocols use cryptographic mechanisms to ensure the integrity of data. This means that the data remains unchanged during transit and cannot be modified or tampered with by malicious actors without detection.
3	Authentication	Secure communication protocols support authentication mechanisms to verify the identities of communicating entities. Through the use of digital certificates, public key infrastructure (PKI), and mutual authentication, entities can be assured of the authenticity of the communication partners, reducing the risk of impersonation or man-in-the-middle attacks.
4	Key Exchange	Secure communication protocols facilitate secure key exchange between communicating entities. This ensures that the encryption keys used for securing the communication channel are shared securely and are not susceptible to interception or tampering.
5	Compatibility and interoperability	Secure communication protocols, such as TLS, are widely supported by modern network infrastructure and applications. They offer compatibility with various operating systems, web browsers, and network devices, making it easier to establish secure connections across different platforms and technologies.

11.1.32.1 Assessment – SAMM

Secure Communication Protocols:

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.

#	Name	Description
2	SAMM Security by Design	Secure communication protocols contribute to the Security by Design domain by ensuring the confidentiality, integrity, and authenticity of data transmitted over networks, addressing the activity of "Secure Architecture." Implementing secure communication protocols like Transport Layer Security (TLS) helps establish a secure and encrypted channel between communicating entities, preventing unauthorized interception, tampering, and impersonation.
3	SAMM Security by Implementation	Secure communication protocols contribute to the Security by Implementation domain by addressing the activity of "Security Operations Integration." By implementing secure communication protocols, organizations ensure that secure connections are established during the deployment and operation of software systems, reducing the risk of unauthorized access and ensuring the ongoing protection of data in transit.
4	SAMM Security by Verification	Secure communication protocols contribute to the Security by Verification domain by addressing the activity of "Secure Deployment Verification." By utilizing secure communication protocols, organizations can verify that software applications are securely deployed and configured in the production environment, ensuring that security controls and encryption mechanisms are properly implemented during deployment.
5	SAMM Security by Operations	Secure communication protocols contribute to the Security by Operations domain by addressing the activity of "Secure Deployment." By using secure communication protocols, organizations can deploy software systems in a secure and controlled manner, reducing the risk of unauthorized access or compromise during deployment.

11.1.32.2 Assessment – OWASP Top 10

Implementing secure communication protocols like TLS do not directly address all OWASP threats. However, they play a significant role in ensuring the confidentiality, integrity, and authentication of data during transmission. Implementing secure communication protocols is an essential security practice, but it should be complemented with other measures to address the specific threats outlined in the OWASP Top 10.

#	Name	Description
1	Broken Access Control (A01-2021)	Does Not Apply Directly. Secure communication protocols focus on protecting data during transmission, but they do not directly address access control vulnerabilities within a web application.

#	Name	Description
2	Cryptographic Failures (A02-2021)	Secure communication protocols, such as TLS, contribute to addressing Cryptographic Failures by providing a framework for implementing secure encryption and cryptographic mechanisms. TLS ensures the confidentiality and integrity of data transmitted over networks, preventing unauthorized access and manipulation of sensitive information.
3	Injection (A03-2021)	Does Not Apply Directly. Secure communication protocols do not directly address the improper handling of untrusted data that leads to Injection vulnerabilities. Injection vulnerabilities are primarily mitigated through proper input validation and secure coding practices.
4	Insecure Design (A04-2021)	Does Not Apply Directly. Secure communication protocols focus on securing data transmission and do not directly address flaws and weaknesses in the overall design and architecture of a web application.
5	Security Misconfiguration (A05-2021)	Does Not Apply Directly. Secure communication protocols are not directly related to the configuration of web applications and associated components. Addressing security misconfigurations requires proper configuration practices and regular updates to software and systems.
6	Vulnerable and Outdated Components (A06-2021)	Does Not Apply Directly. Secure communication protocols do not directly address vulnerabilities associated with using outdated or known vulnerable software components within a web application. Managing software dependencies and following secure coding practices are more relevant for mitigating this threat.
7	Identification and Authentication Failures (A07-2021)	Secure communication protocols indirectly contribute to addressing Identification and Authentication Failures by providing a secure channel for transmitting authentication credentials. TLS helps prevent eavesdropping and tampering with authentication data during transmission.
8	Software and Data Integrity Failures (A08-2021)	Secure communication protocols, such as TLS, contribute to addressing Software and Data Integrity Failures by ensuring the integrity of data during transmission. TLS uses cryptographic mechanisms to detect any tampering or modification of data.
9	Security Logging and Monitoring Failures (A09-2021)	Does Not Apply Directly. Secure communication protocols are not directly related to logging and monitoring practices within web applications. Addressing Security Logging and Monitoring Failures requires implementing comprehensive logging and monitoring mechanisms.
10	Server-Side Request Forgery (A10-2021)	Does Not Apply Directly. Secure communication protocols do not directly mitigate the risks associated with Server-Side Request Forgery. Preventing SSRF vulnerabilities requires input validation, secure configurations, and proper security testing.

11.1.33 Secure Data Transmission

Encrypt data in transit to prevent interception and tampering. For this, use secure protocols, such as Secure File Transfer Protocol (SFTP) or Secure Shell (SSH), for transferring data between systems. Adding Secure Data Transmission as a security design into architectures, organizations can safeguard sensitive data while it is being transmitted between systems. These protocols imply addressing the underlying capabilities required to operate Secure Protocols, such as authentication, key management, monitoring, and secure configurations, so it is possible have the required security principles of confidentiality, integrity, and availability of data during transit, mitigating the risks of interception and tampering.

#	Name	Description
1	Secure Protocols	Secure protocols, such as Secure File Transfer Protocol (SFTP) and Secure Shell (SSH), are utilized for transferring data between systems. These protocols provide robust encryption and authentication mechanisms, ensuring the confidentiality and integrity of the transmitted data.
2	Encryption in Transit	All data transmitted over the network is encrypted to prevent unauthorized access. This is achieved through the use of strong encryption algorithms, such as Advanced Encryption Standard (AES) or Transport Layer Security (TLS). Encryption scrambles the data, making it unreadable to anyone who intercepts it during transit.
3	Authentication	Secure data transmission includes robust authentication mechanisms to ensure the identity and integrity of the participating systems. Authentication protocols, such as public-key cryptography or digital certificates, are employed to verify the identities of the communicating parties. This prevents unauthorized systems from intercepting or tampering with the transmitted data.
4	Key Management	Effective key management is essential for secure data transmission. Encryption keys used during the data transfer process must be securely generated, distributed, and stored. Key rotation and periodic renewal should be enforced to minimize the risk of key compromise. Additionally, a secure key management system should be implemented to protect the encryption keys from unauthorized access.
5	Monitoring and Logging	Comprehensive monitoring and logging mechanisms are implemented to track data transmission activities. This includes capturing relevant metadata, such as source and destination addresses, timestamps, and file sizes, to detect any suspicious or anomalous behaviour. Regular monitoring and analysis of these logs help identify potential security incidents and enable timely response and investigation.
6	Secure Configuration	The systems involved in data transmission should be configured securely. This includes disabling unnecessary services and protocols, enforcing strong access controls, and applying patches and updates regularly to address any vulnerabilities. By maintaining secure configurations, the risk of unauthorized access and potential exploits is significantly reduced.

11.1.33.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly
2	SAMM Security by Design	Secure Data Transmission contributes to the "Security Requirements" activity by ensuring that security requirements related to data transmission and protection are addressed in the design phase. Secure Data Transmission contributes to the "Secure Architecture" activity by providing a secure mechanism for transmitting data between systems, addressing the need for secure communication channels in the software architecture. Secure Data Transmission contributes to the "Secure Coding Practices" activity indirectly by ensuring that the design incorporates secure protocols and encryption techniques, which align with secure coding principles.
3	SAMM Security by Implementation	Secure Data Transmission contributes to the "Secure Architecture" activity by providing secure protocols and encryption mechanisms that can be integrated into the software system's architecture. Secure Data Transmission contributes to the "Security Coding Guidelines" activity by demonstrating the importance of secure coding practices related to data transmission and encryption. Secure Data Transmission contributes to the "Security Testing Integration" activity by providing a feature that can be tested during the software development process to ensure the proper implementation of secure data transmission mechanisms.
4	SAMM Security by Verification	Secure Data Transmission contributes to the "Security Testing" activity by providing a specific area of security testing related to data transmission and encryption. Secure Data Transmission contributes to the "Code Review" activity indirectly by ensuring that the implementation of secure protocols and encryption techniques is reviewed for adherence to secure coding practices. Secure Data Transmission contributes to the "Security Architecture Review" activity by demonstrating the incorporation of secure data transmission mechanisms into the overall software architecture.

#	Name	Description
5	SAMM Security by Operations	Secure Data Transmission contributes to the "Environment Hardening" activity by providing a secure mechanism for transmitting data, thereby hardening the software environment against unauthorized interception or tampering. Secure Data Transmission contributes to the "Secure Deployment" activity by ensuring that secure protocols and encryption techniques are implemented during the deployment of the software system. Secure Data Transmission contributes to the "Security Testing" activity by providing a specific area of security testing related to data transmission and encryption. Secure Data Transmission contributes to the "Operational Enablement" activity by enabling operational staff to understand and utilize secure data transmission mechanisms to ensure the ongoing security and reliability of the software system.

11.1.33.2 Assessment – OWASP Top 10

The Secure Data Transmission capability addresses the following OWASP threats:

#	Name	Description
11	Broken Access Control (A01-2021)	The use of secure protocols and encryption in transit helps prevent unauthorized access to transmitted data. By enforcing proper access controls and implementing secure data transmission, organizations can mitigate the risk of broken access control vulnerabilities.
12	Cryptographic Failures (A02-2021)	Secure Data Transmission ensures the use of strong encryption algorithms and proper encryption practices during data transmission. By encrypting data in transit, organizations can prevent cryptographic failures and enhance the overall security of their applications.
13	Injection (A03-2021)	Secure Data Transmission, specifically the use of secure protocols, helps prevent injection attacks that exploit vulnerabilities in data transmission. By ensuring that transmitted data is securely encrypted and authenticated, organizations can reduce the risk of injection vulnerabilities.
14	Insecure Design (A04-2021)	Secure Data Transmission contributes to secure architecture design by incorporating secure protocols, encryption, authentication, and monitoring mechanisms. By considering secure design principles and implementing secure data transmission, organizations can address insecure design vulnerabilities.
15	Security Misconfiguration (A05-2021)	Secure Data Transmission emphasizes secure configurations for systems involved in data transmission. By implementing secure protocols, disabling unnecessary services and protocols, and enforcing strong access controls, organizations can mitigate security misconfiguration vulnerabilities.

#	Name	Description
16	Vulnerable and Outdated Components (A06-2021)	Although Secure Data Transmission primarily focuses on the secure transmission of data, it indirectly contributes to addressing vulnerable and outdated components. By ensuring secure protocols and encryption, organizations can protect against attacks that target vulnerable components during data transmission.
17	Identification and Authentication Failures (A07-2021)	Secure Data Transmission incorporates robust authentication mechanisms to ensure the identity and integrity of the participating systems. By implementing secure authentication protocols, organizations can reduce the risk of identification and authentication failures.
18	Software and Data Integrity Failures (A08-2021)	Secure Data Transmission contributes to data integrity by encrypting data in transit, preventing unauthorized modification or manipulation. By ensuring the integrity of transmitted data, organizations can mitigate software and data integrity failures.
19	Security Logging and Monitoring Failures (A09-2021)	Secure Data Transmission suggests the implementation of monitoring and logging mechanisms to track data transmission activities. By capturing relevant metadata and analyzing logs, organizations can enhance their security logging and monitoring capabilities and address related vulnerabilities.
20	Server-Side Request Forgery (A10-2021)	While Secure Data Transmission does not directly address SSRF vulnerabilities, it can indirectly contribute to mitigating this threat by ensuring secure protocols and encryption during data transmission. By securing the communication channels, organizations can reduce the risk of server-side request forgery attacks.

11.1.34 Secure Data in transit

Regarding data encryption in transit, all data in transit among internal components and all the incoming or outgoing data from third parties or other systems are encrypted It is required to have and end-to-end security for data sent over the internet, and also over internal networks. The solution must ensure the data is encrypted and authenticated and that the information is not tampered with while in transit between applications.

The primary mechanism for securing data in transit is by using the TLS protocol. TLS provides encryption, authentication, and integrity of data exchanged between systems over a network. It ensures that data is protected from unauthorized access and tampering during transmission.

Implementing secure data in transit guarantee that the transmission of data between systems remains confidential, authenticated, and intact. The use of TLS, proper certificate management, secure protocols and cipher suites, and mutual authentication helps establish a secure communication channel. Secure configuration, monitoring, and logging contribute to maintaining the security of data in transit and detecting any potential security issues.

#	Name	Description
1	Encryption Algorithms and Key Exchange	TLS supports various encryption algorithms, such as AES (Advanced Encryption Standard), to encrypt the data in transit. The choice of encryption algorithm should follow industry best practices and compliance requirements. Additionally, TLS utilizes secure key exchange protocols, such as Diffie-Hellman or Elliptic Curve Diffie-Hellman, to establish a secure communication channel and exchange encryption keys securely.
2	Certificate Management	TLS relies on digital certificates to authenticate the identity of the communicating parties. These certificates are issued by trusted certificate authorities (CAs). Proper certificate management involves obtaining valid certificates from trusted CAs, regularly renewing them, and configuring the system to validate the authenticity and validity of certificates during the TLS handshake process.
3	Secure Protocols and Cipher Suites	TLS supports different versions, such as TLS 1.2 or TLS 1.3, with each version providing improvements in security and cryptographic algorithms. It is crucial to use the latest TLS version that is compatible with the system and clients while ensuring backward compatibility. Additionally, the use of secure cipher suites, which define the combination of encryption, hashing, and key exchange algorithms, should be configured to provide the strongest level of security.
4	Secure Configuration	Secure data in transit can be further enhanced by properly configuring the TLS settings. This includes enforcing strong encryption protocols, disabling insecure cipher suites, enabling Perfect Forward Secrecy (PFS), and implementing secure protocol configurations such as HTTP Strict Transport Security (HSTS) and Certificate Pinning.
5	Mutual Authentication	In certain cases, where both the client and the server need to verify each other's identities, mutual authentication can be implemented. This involves the client presenting its digital certificate to the server for authentication, in addition to the server's certificate being validated by the client. Mutual authentication provides an extra layer of security and ensures that both parties can trust each other before establishing a secure connection.
6	Monitoring and Logging	It is essential to implement monitoring and logging mechanisms to track and analyze the security of data in transit. This includes monitoring TLS handshakes, tracking cipher suites and protocol versions used, and logging any security-related events or errors. Monitoring and logging can help detect any potential security incidents or vulnerabilities and provide valuable information for incident response and forensic analysis.

11.1.34.1 Assessment – SAMM

Adding to the architecture the security design of encrypting data in transit helps address the SAMM security domains indirectly. It contributes to the overall security posture of an organization by ensuring the confidentiality and integrity of data during its transmission. While it may not directly align with the specific

activities of each SAMM security domain, it provides a foundational security measure that supports the secure development, deployment, and operation of software systems across all domains. Encryption of data in transit helps protect against eavesdropping, tampering, and unauthorized access, which are common concerns in software security.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: By developing and implementing policies and standards, organizations address the activity of defining expectations and requirements for software security within the Security by Governance domain. Risk Management: The establishment of risk management processes and the implementation of risk mitigation strategies contribute to addressing the activity of identifying and assessing risks associated with software development and deployment. Compliance Management: Ensuring compliance with relevant regulatory requirements, industry standards, and best practices helps address the activity of aligning with security frameworks and guidelines within the Compliance Management activity. Security Training and Awareness: Providing training programs and awareness campaigns contributes to the activity of educating and raising awareness among employees and stakeholders about software security. Security Metrics and Reporting: Defining and measuring security metrics and establishing reporting mechanisms address the activity of measuring key security metrics to assess the effectiveness of software security practices. Security Roles and Responsibilities: Clearly defining and assigning security roles and responsibilities within the organization addresses the activity of establishing accountability for software security and ensuring individuals have the necessary knowledge and skills to fulfill their roles effectively.
2	SAMM Security by Design	Security Requirements: Addressing security requirements helps mitigate vulnerabilities associated with OWASP threats such as Broken Access Control, Injection, Insecure Design, and others within the Security Requirements activity. Secure Architecture: Adopting secure architectural patterns and principles contributes to addressing OWASP threats like Insecure Design, Cryptographic Failures, Security Misconfiguration, and others within the Secure Architecture activity. Secure Coding Practices: Adhering to secure coding practices helps prevent common coding mistakes and vulnerabilities associated with OWASP threats such as Injection, Insecure Design, Security Misconfiguration, and others within the Secure Coding Practices activity. Threat Modeling: Conducting threat modeling exercises helps identify and mitigate potential security risks, including those associated with OWASP threats, within the Threat Modeling activity. Security Testing: Incorporating security testing techniques helps detect and address vulnerabilities related to OWASP threats, ensuring the design is resilient to potential attacks within the Security Testing activity.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training: Providing training on secure coding practices and principles helps developers build secure software from the outset within the Secure Development Training activity. Secure Architecture: Applying secure architecture practices, such as threat modeling and secure design patterns, helps identify and address vulnerabilities related to OWASP threats within the Secure Architecture activity. Secure Coding Guidelines: Establishing and adopting secure coding guidelines and standards helps prevent vulnerabilities associated with OWASP threats like Injection, Insecure Design, Security Misconfiguration, and others within the Secure Coding Guidelines activity. Security Requirements: Defining and documenting security requirements ensures that security objectives and expectations for the software system are met within the Security Requirements activity. Security Testing Integration: Integrating security testing activities throughout the software development life cycle helps detect and address vulnerabilities related to OWASP threats within the Security Testing Integration activity. Security Verification: Verifying the effectiveness of security controls implemented in the software system helps ensure that it meets the desired security objectives within the Security Verification activity. Security Architecture Review: Conducting security architecture reviews helps evaluate the security of the software system's design and architecture within the Security Architecture Review activity. Security Operations Integration: Integrating security considerations into operational processes helps support the secure and reliable operation of the software system within the Security Operations Integration activity.
4	SAMM Security by Verification	Security Testing: Testing the effectiveness of TLS and encryption protocols in securing data during transit is a part of security testing activities. Security Architecture Review: Evaluating the implementation of TLS and encryption protocols in the security architecture helps ensure proper protection of data during transit. Threat Modeling: Considering data in transit as a potential threat helps identify risks and define appropriate security measures, such as encryption, to mitigate those risks. Secure Deployment Verification: Verifying the secure deployment of the software system includes ensuring the proper implementation and configuration of TLS and encryption for data in transit.
5	SAMM Security by Operations	Environment Hardening: Implementing TLS and encryption for data in transit contributes to hardening the software environment and protecting the integrity and confidentiality of the transmitted data. Secure Deployment: The use of TLS for data encryption in transit is part of secure deployment practices to ensure secure software distribution and installation procedures. Security Testing: Verifying the implementation and effectiveness of TLS and encryption protocols for data in transit is a part of security testing activities in the operational phase. Operational Enablement: Including guidelines and procedures for secure data transmission, such as TLS implementation, supports operational staff in ensuring the ongoing security and reliability of software systems.

11.1.34.2 Assessment – OWASP Top 10

The capability of "Secure Data in transit" addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	Secure data in transit helps protect against unauthorized access to sensitive data during transmission. By encrypting data using Transport Layer Security (TLS), it ensures that only authorized entities can access and decrypt the data. This prevents attackers from bypassing access controls and gaining unauthorized access to restricted resources.
2	Cryptographic Failures (A02-2021)	Secure data in transit involves the correct implementation of cryptographic algorithms and mechanisms, such as TLS, to protect data confidentiality, integrity, and authenticity during transmission. It mitigates vulnerabilities related to weak algorithms, improper usage, and insecure configurations. By utilizing TLS, organizations can ensure that data is encrypted properly and transmitted securely, reducing the risk of cryptographic failures.
3	Injection (A03-2021)	While secure data in transit does not directly address injection vulnerabilities, it plays a supporting role in preventing data manipulation during transmission. By encrypting data, it ensures that malicious actors cannot inject unauthorized commands or manipulate data in transit, reducing the risk of injection attacks.
4	Insecure Design (A04-2021)	Secure data in transit, specifically through the use of TLS, does not directly address insecure design vulnerabilities. However, it complements secure design practices by protecting data during transmission. This ensures that even if the design of the application has vulnerabilities, the data remains encrypted and secure while being transported between components, reducing the impact of insecure design on data confidentiality and integrity.
5	Security Misconfiguration (A05-2021)	Secure data in transit helps mitigate security misconfiguration vulnerabilities by ensuring that the communication channels between components are properly configured with secure protocols, such as TLS. This prevents misconfigurations that could expose sensitive data during transmission, reducing the risk of security misconfiguration-related attacks.
6	Vulnerable and Outdated Components (A06-2021)	While secure data in transit does not directly address vulnerable and outdated components, it indirectly contributes to their mitigation. By encrypting data during transmission using TLS, it provides an additional layer of protection against potential attacks targeting vulnerabilities in outdated or insecure software components. Even if a component has vulnerabilities, encrypting the data in transit helps prevent attackers from exploiting those vulnerabilities to intercept or manipulate the data.
7	Identification and Authentication Failures (A07-2021)	Secure data in transit, specifically through the use of TLS, helps protect against identification and authentication failures by ensuring that sensitive information, such as authentication tokens or credentials, is encrypted during transmission. This prevents unauthorized parties from intercepting and gaining access to authentication data, reducing the risk of impersonation or unauthorized access due to identification and authentication vulnerabilities.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	Secure data in transit helps address software and data integrity failures by protecting the integrity of data during transmission. By encrypting data using TLS, it ensures that the data remains intact and unaltered during transit, reducing the risk of unauthorized modifications, tampering, or data integrity issues.
9	Security Logging and Monitoring Failures (A09-2021)	Secure data in transit does not directly address security logging and monitoring failures. However, it contributes indirectly by protecting the confidentiality and integrity of security logs during transmission. By encrypting data in transit, organizations can ensure that security-related logs are not exposed or tampered with, maintaining the integrity and confidentiality of log data.
10	Server-Side Request Forgery (A10-2021)	Secure data in transit does not directly address server-side request forgery vulnerabilities. However, by implementing secure data transmission mechanisms such as TLS, it can help protect against unauthorized requests initiated through SSRF attacks. Encrypting data in transit ensures that any malicious requests made by an attacker are encrypted and cannot be manipulated or intercepted during transmission, reducing the risk of SSRF attacks. In summary, while secure data in transit primarily focuses on protecting data confidentiality and integrity during transmission, it indirectly contributes to addressing several OWASP threats by mitigating vulnerabilities and providing an additional layer of protection for the data being transmitted.

11.1.35 Redundancy and High Availability

Business continuity is considered as a type of security risk. Architecture that consider redundancy and high availability, address it by incorporting system resiliency to mitigate the impact of potential security incidents or system failures. Organizations can enhance their ability to withstand and recover from security incidents or system failures. These measures provide continuity of critical services, protect data integrity, and minimize the impact of potential disruptions, ultimately ensuring a resilient and secure environment. Redundancy refers to the duplication of critical components or resources within a system to ensure that if one component fails, there is another available to take its place.

#	Name	Description
1	Redundant Hardware	Implementing redundant hardware components, such as servers, firewalls, load balancers, or network devices, ensures that if one device fails, another can seamlessly take over its functions. This redundancy minimizes downtime and ensures continuous availability of critical security services.
2	Redundant Data Storage	Redundant data storage mechanisms, such as RAID (Redundant Array of Independent Disks) or distributed storage systems, replicate data across multiple drives or locations. If one drive or storage location fails, the redundant copies can be used to retrieve the data without any loss, ensuring data availability and integrity.

#	Name	Description
3	Redundant Network Paths	Establishing redundant network paths ensures that if one network link or pathway fails, the traffic can automatically be rerouted through an alternative path. This redundancy prevents connectivity issues and ensures continuous network availability.
4	High Availability	High availability focuses on ensuring that critical services and systems are accessible and operational with minimal disruption. It involves designing the architecture in a way that minimizes single points of failure and maximizes system uptime.
5	Load Balancing	It is one of the considerations for high-availability. Implementing load balancers distributes incoming traffic across multiple servers, ensuring that no single server is overwhelmed and can handle requests efficiently. Load balancing helps optimize resource utilization and prevents service degradation or failure due to excessive traffic.
6	Failover Mechanisms	It is one of the considerations for high-availability. Failover mechanisms are designed to automatically switch to redundant components or systems when a failure or performance degradation is detected. For example, if a primary firewall becomes unavailable, a secondary firewall can take over the functions seamlessly, ensuring continuous protection and availability.
7	Disaster Recovery	It is one of the considerations for high-availability. A robust disaster recovery strategy ensures that critical systems can be quickly restored in the event of a catastrophic failure or a major security incident. This may involve off-site backups, redundant infrastructure at geographically separate locations, and well-defined recovery processes.
8	Monitoring and Alerting	It is one of the considerations for high-availability. High availability requires proactive monitoring and alerting mechanisms to detect and respond to potential issues promptly. Continuous monitoring of system performance, security events, and network connectivity helps identify anomalies and triggers appropriate actions to maintain high availability.

11.1.35.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly
2	SAMM Security by Design	It does not apply directly

#	Name	Description
3	SAMM Security by Implementation	Redundancy and High Availability, specifically the aspect of high availability, contributes to the Security by Implementation domain by ensuring that security measures are available and operational even during system failures or incidents. Redundancy and High Availability also support the Security by Implementation domain's aspect of Security Operations Integration by ensuring continuous availability of security controls and mechanisms during the operation of the software system.
4	SAMM Security by Verification	It does not apply directly
5	SAMM Security by Operations	Environment Hardening: Redundancy and high availability measures help ensure the availability and reliability of the software system in the operational environment, contributing to its hardening. Configuration Management: Redundancy and high availability mechanisms can be part of the configuration management process to ensure that redundant components or resources are properly configured and managed. Vulnerability Management: Redundancy and high availability measures can help mitigate the impact of vulnerabilities by providing failover options or redundant systems to maintain service availability while vulnerabilities are being addressed. Incident Management: Redundancy and high availability architectures support incident management by providing continuity of critical services and minimizing the impact of security incidents on the operations of the software system. Secure Deployment: Redundancy and high availability measures can be considered during the deployment phase to ensure that the software system is deployed in a secure and reliable manner, with redundant components and failover mechanisms in place.

11.1.35.2 Assessment – OWASP Top 10

Redundancy and High Availability capability does not directly address all OWASP threats; however, it contributes to mitigating several threats by ensuring system resilience, minimizing the impact of failures, and maintaining the availability of critical security services.

#	Name	Description
1	Broken Access Control (A01-2021)	Redundancy and high availability help maintain access controls during system failures, ensuring that authorized users can still access the resources they are entitled to.
2	Cryptographic Failures (A02-2021)	Redundancy and high availability do not directly address cryptographic failures. However, they provide system resilience, which can help mitigate the impact of cryptographic failures and maintain the availability of critical security services.

#	Name	Description
3	Injection (A03-2021)	Redundancy and high availability measures help protect against injection attacks by ensuring continuous availability of critical services. If one component becomes compromised due to an injection attack, redundant components can take over, minimizing the impact and allowing for quick recovery.
4	Insecure Design (A04-2021)	Redundancy and high availability do not directly address insecure design vulnerabilities. However, they contribute to the overall resilience of the system, which can help mitigate the impact of insecure design decisions and provide an opportunity to recover quickly.
5	Security Misconfiguration (A05-2021)	Redundancy and high availability measures can indirectly mitigate the impact of security misconfigurations by providing redundancy in the system. If a misconfiguration leads to a system failure, redundant components can take over, minimizing the impact and allowing for quick recovery.
6	Vulnerable and Outdated Components (A06-2021)	Redundancy and high availability measures help protect against the exploitation of vulnerabilities in components. If a vulnerable component fails, redundant components can continue providing the required services, reducing the window of opportunity for attackers.
7	Identification and Authentication Failures (A07-2021)	Redundancy and high availability do not directly address identification and authentication failures. However, they ensure continuous availability of authentication services, reducing the impact of failures and potential unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Redundancy and high availability measures indirectly contribute to mitigating software and data integrity failures by providing mechanisms for data replication and recovery. This helps ensure the integrity and availability of critical software components and data.
9	Security Logging and Monitoring Failures (A09-2021)	Redundancy and high availability measures do not directly address security logging and monitoring failures. However, they can help maintain the availability of logging and monitoring systems, ensuring that security incidents are promptly detected and responded to.
10	Server-Side Request Forgery (A10-2021)	Redundancy and high availability measures do not directly address server-side request forgery vulnerabilities. However, they can help mitigate the impact of SSRF attacks by ensuring the availability of critical services and providing an opportunity for quick recovery.

11.1.36 Container Orchestration Security

Secure container orchestration platforms, such as Kubernetes, by configuring RBAC (Role-Based Access Control), enabling network policies, and implementing container runtime security measures.

Container orchestration security ensures that containers and the container orchestration platform are protected against unauthorized access, data breaches, and other security risks. By combining RBAC, network policies, and container runtime security measures, organizations can create a robust security framework for their containerized applications and infrastructure.

Container orchestration security refers to the set of practices and measures implemented to ensure the secure deployment and management of containers within a container orchestration platform, such as

Kubernetes. It focuses on protecting the containerized applications, infrastructure, and data from potential threats and vulnerabilities.

In addition to RBAC, network policies, and container runtime security, there are other security considerations in container orchestration.

#	Name	Description
1	Role-Based Access Control (RBAC)	One important feature of container orchestration security is Role-Based Access Control (RBAC). RBAC allows administrators to define and enforce granular access controls within the container orchestration platform. By assigning specific roles and permissions to users or groups, RBAC ensures that only authorized individuals have access to critical resources and can perform specific actions within the cluster. RBAC helps prevent unauthorized access and reduces the risk of malicious activities.
2	Network Policies	Another key aspect of container orchestration security is network policies. Network policies enable administrators to define rules and restrictions for network traffic between containers and external entities. By configuring network policies, administrators can control communication between containers, enforce encryption and authentication, and restrict access to specific network segments. Network policies help create a secure network environment within the container orchestration platform and protect against unauthorized access or data breaches.
3	Container runtime security	Container runtime security measures are also crucial for container orchestration security. These measures involve implementing security controls at the container runtime level to ensure the integrity and isolation of containers. This includes utilizing secure container runtimes, such as containerd or cri-o, that follow best security practices and provide isolation between containers. Container runtime security also involves regularly updating container runtimes to patch any security vulnerabilities and using container vulnerability scanning tools to identify and mitigate potential risks.
4	Other	Secure image registries, enforcing secure image scanning and validation, monitoring and logging container activities, and implementing secure deployment and configuration practices.

11.1.36.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.
2	SAMM Security by Design	Security Requirements: Container orchestration security contributes to addressing security requirements by implementing RBAC, network policies, and container runtime security measures to ensure secure design and protect against vulnerabilities.

#	Name	Description
		Secure Architecture: Container orchestration security helps establish a secure architecture by implementing security controls at the container orchestration platform level, such as RBAC, network policies, and container runtime security measures. Secure Coding Practices: While container orchestration security focuses more on runtime security, it indirectly contributes to secure coding practices by providing a secure environment for running containerized applications. Threat Modeling: Container orchestration security measures can be incorporated into threat modeling exercises to identify potential threats and risks associated with containerized applications and infrastructure. Security Testing: Container orchestration security measures can be included in security testing activities during the design phase to validate the effectiveness of security controls implemented at the container orchestration platform level.
3	SAMM Security by Implementation	Secure Development Training: Container orchestration security practices can be included in secure development training programs to educate developers about secure container deployment and management. Secure Architecture: Container orchestration security contributes to implementing secure architecture practices by providing security controls at the container orchestration platform level. Secure Coding Guidelines: Container orchestration security measures can be included in secure coding guidelines to address security considerations related to container deployment and management. Security Requirements: Container orchestration security measures help address security requirements by implementing RBAC, network policies, and container runtime security measures to protect containerized applications and infrastructure. Security Testing Integration: Container orchestration security measures can be integrated into security testing activities to ensure that container deployments are tested for vulnerabilities and security weaknesses. Security Verification: Container orchestration security measures contribute to security verification by implementing security controls and measures at the container orchestration platform level. Security Architecture Review: Container orchestration security measures can be reviewed as part of the security architecture review process to assess the effectiveness of security controls at the container orchestration platform level. Security Operations Integration: Container orchestration security measures contribute to integrating security considerations into operational processes, such as secure deployment practices and incident management.
4	SAMM Security by Verification	Security Testing: Container orchestration security measures can be validated and verified through security testing activities, such as penetration testing and vulnerability scanning. Code Review: Code reviews can include reviewing the implementation of container orchestration security measures and their adherence to secure coding practices. Security Architecture Review: Container orchestration security measures can be assessed as part of the security architecture review process to ensure the effectiveness of security controls at the container orchestration platform level. Security Requirements Verification: Container orchestration security measures can be reviewed to verify their alignment with security requirements related to containerized applications and infrastructure.

#	Name	Description
		Threat Modeling: Container orchestration security measures can be incorporated into threat modeling exercises to identify and mitigate potential security risks. Secure Deployment Verification: Container orchestration security measures can be verified to ensure that secure deployment practices are followed and security controls are properly implemented.
5	SAMM Security by Operations	Environment Hardening: Container orchestration security measures contribute to hardening the software environment by implementing secure configurations and infrastructure controls at the container orchestration platform level. Secure Build: Container orchestration security measures can be part of secure build practices by ensuring that the container orchestration platform is securely configured and that containerized applications are built with security in mind. Configuration Management: Container orchestration security measures can be integrated into configuration management practices to ensure that container orchestration platform configurations are properly managed and documented. Vulnerability Management: Container orchestration security measures help in managing vulnerabilities by implementing container runtime security measures and regularly updating container runtimes to address security vulnerabilities. Incident Management: Container orchestration security measures can be integrated into the standards procedures in operations and incident response teams for consistency in trainning and skill management for when needing access to environments and platforms.

11.1.36.2 Assessment – OWASP Top 10

The Container Orchestration Security capability addresses the following OWASP threats:

#	Name	Description
1	Broken Access Control (A01-2021)	RBAC (Role-Based Access Control) helps enforce granular access controls within the container orchestration platform, preventing unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	It does not apply directly.
3	Injection (A03-2021)	It does not apply directly.
4	Insecure Design (A04-2021)	It does not apply directly.
5	Security Misconfiguration (A05-2021)	Proper configuration of container orchestration platforms, such as Kubernetes, ensures that default or insecure settings are not used, reducing the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Regularly updating container runtimes and ensuring the use of secure container technologies, such as containerd or cri-o, helps address vulnerabilities associated with outdated or insecure software components.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Container orchestration security practices can include implementing secure authentication mechanisms, enforcing strong password policies, and regularly testing for authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	By implementing container runtime security measures, organizations can protect the integrity of software components, configurations, and data within containerized applications.
9	Security Logging and Monitoring Failures (A09-2021)	Container orchestration security practices can include comprehensive logging, effective log analysis, and real-time monitoring mechanisms to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	It does not apply directly.

11.1.37 Immutable Infrastructure

Immutable infrastructure is an architecture pattern where components are treated as disposable and replaced with new instances rather than being modified. This reduces the risk of configuration drift and ensures consistent security configurations. It promotes the practice of scripting the architecture and configuration management. So that the architecture can be reproduced with its configurations from code. Infrastructure components of the infrastructure, such as virtual machines, containers, or serverless instances, are treated as disposable entities that are replaced with new instances instead of being modified in-place. This concept is inspired by the immutable nature of functional programming, where data is immutable and any modifications result in the creation of new data structures.
Adopting immutable infrastructure brings benefits such as reduced configuration drift, consistent security configurations, improved scalability and availability, enhanced disaster recovery capabilities, simplified rollbacks and testing, and better alignment with Infrastructure as Code practices. These advantages contribute to a more reliable, secure, and efficient infrastructure management approach.

By adopting immutable infrastructure, organizations can achieve several benefits:

#	Name	Description
1	Reduced Configuration Drift	Configuration drift refers to inconsistencies that can occur between different instances of an infrastructure component due to manual modifications or unauthorized changes. Immutable infrastructure mitigates this risk by ensuring that components are never modified in-place. Instead, when changes are required, new instances are created with the desired configurations, eliminating the possibility of configuration drift.
2	Consistent Security Configurations	Immutable infrastructure enables consistent security configurations across all instances. Since the infrastructure components are immutable, security configurations can be embedded into their build process. This ensures that every new instance is automatically provisioned with the latest security measures and patches, reducing the risk of security vulnerabilities caused by misconfigurations or outdated software.

#	Name	Description
3	Improved Scalability and Availability	Immutable infrastructure facilitates horizontal scaling by allowing organizations to rapidly provision new instances as needed. Instead of scaling up existing instances, additional instances are created with the required configurations, which can be easily replicated. This approach improves the overall availability and responsiveness of the infrastructure, as new instances can be quickly brought online to handle increased workloads or replace faulty components.
4	Enhanced Disaster Recovery	With immutable infrastructure, recovering from disasters becomes more efficient. In the event of an infrastructure failure or a security breach, affected instances can be swiftly replaced with new ones that have known and trusted configurations. This eliminates the need to troubleshoot and fix individual instances, reducing downtime and enabling faster recovery.
5	Simplified Rollbacks and Testing	Immutable infrastructure makes rollbacks and testing easier. If an issue arises after deploying a new version of the infrastructure, rolling back to the previous state is as simple as replacing the new instances with the previous versions. Furthermore, testing new configurations or updates becomes safer and less complex, as the existing infrastructure remains intact and new instances can be deployed for testing purposes without impacting the production environment.
6	Infrastructure as Code (IaC) Alignment	Immutable infrastructure aligns well with Infrastructure as Code (IaC) practices. Infrastructure configurations are defined and managed through code, allowing for version control, automated provisioning, and reproducibility. Immutable infrastructure complements IaC by ensuring that the infrastructure remains consistent and reproducible across deployments, making it easier to manage and track changes over time.

11.1.37.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.

#	Name	Description
2	SAMM Security by Design	Immutable Infrastructure contributes to the Security by Design domain by enabling consistent security configurations (Consistent Security Configurations). It ensures that every new instance is automatically provisioned with the latest security measures and patches, reducing the risk of security vulnerabilities caused by misconfigurations or outdated software. Immutable Infrastructure also facilitates horizontal scaling, which improves the scalability and availability of the infrastructure (Improved Scalability and Availability). Additionally, in the event of a security breach, affected instances can be swiftly replaced with new ones that have known and trusted configurations, enhancing disaster recovery capabilities (Enhanced Disaster Recovery). Immutable Infrastructure also simplifies rollbacks and testing, allowing for easy replacement of instances and safer testing of new configurations or updates (Simplified Rollbacks and Testing). Lastly, by aligning with Infrastructure as Code (IaC) practices, Immutable Infrastructure ensures the infrastructure remains consistent and reproducible, contributing to security governance (Infrastructure as Code (IaC) Alignment).
3	SAMM Security by Implementation	It does not apply directly.
4	SAMM Security by Verification	It does not apply directly.
5	SAMM Security by Operations	Immutable Infrastructure contributes to the Security by Operations domain by providing secure build practices (Secure Build). By ensuring that software is developed with security in mind, Immutable Infrastructure reduces the likelihood of introducing vulnerabilities during the build process. It also promotes secure deployment practices, such as secure software distribution and secure configuration of software components (Secure Deployment). Immutable Infrastructure supports environment hardening by implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations (Environment Hardening). Additionally, Immutable Infrastructure integrates well with security testing activities, such as static code analysis and security code reviews (Security Testing). It also enables the establishment of incident response capabilities by swiftly replacing affected instances with new ones that have known and trusted configurations (Incident Management).

11.1.37.2 Assessment – OWASP Top 10

Immutable Infrastructure offers benefits in terms of consistent security configurations and reducing configuration drift; however, it does not directly address all the OWASP threats mentioned. Organizations should implement additional security measures and best practices at the application layer to comprehensively address these threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Immutable Infrastructure indirectly contributes to addressing this threat by ensuring consistent security configurations across all instances. By embedding security configurations into the build process of infrastructure components, Immutable Infrastructure helps enforce proper access controls and reduces the risk of misconfigurations that could lead to unauthorized access.
2	Cryptographic Failures (A02-2021)	Immutable Infrastructure does not directly address cryptographic failures. While it can help in deploying secure configurations, cryptographic failures involve issues with the implementation of cryptographic algorithms and mechanisms within the application, which are beyond the scope of infrastructure design.
3	Injection (A03-2021)	Immutable Infrastructure does not directly address injection vulnerabilities. Injection vulnerabilities primarily arise from improper handling of untrusted data within the application code, and Immutable Infrastructure focuses on the infrastructure layer rather than the application layer.
4	Insecure Design (A04-2021)	Immutable Infrastructure indirectly contributes to addressing insecure design vulnerabilities by promoting the adoption of secure design principles and aligning with Infrastructure as Code practices. By defining and managing infrastructure configurations as code, organizations can enforce secure design decisions and minimize insecure design choices.
5	Security Misconfiguration (A05-2021)	Immutable Infrastructure indirectly addresses security misconfiguration vulnerabilities by enabling organizations to define and manage infrastructure configurations as code. This approach allows for consistent and reproducible configurations, reducing the risk of misconfigurations that could lead to security vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Immutable Infrastructure does not directly address vulnerabilities related to outdated or vulnerable components within web applications. However, it can indirectly contribute to mitigating this threat by facilitating the deployment of updated and patched instances of infrastructure components, reducing the likelihood of using outdated or vulnerable software.
7	Identification and Authentication Failures (A07-2021)	Immutable Infrastructure does not directly address identification and authentication failures. These vulnerabilities are primarily related to the application layer and involve weaknesses in the mechanisms used to identify and authenticate users, which are outside the scope of infrastructure design.
8	Software and Data Integrity Failures (A08-2021)	Immutable Infrastructure does not directly address software and data integrity failures. These vulnerabilities relate to the integrity and protection of software components, configurations, and data within web applications, which are beyond the scope of infrastructure design.
9	Security Logging and Monitoring Failures (A09-2021)	Immutable Infrastructure does not directly address security logging and monitoring failures. These vulnerabilities primarily involve deficiencies in logging and monitoring capabilities within web applications, which are typically implemented at the application layer rather than the infrastructure layer.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Immutable Infrastructure does not directly address server-side request forgery vulnerabilities. SSRF vulnerabilities are related to the application layer and involve tricking a vulnerable web application into making unintended requests. Immutable Infrastructure focuses on infrastructure design and management rather than application-level vulnerabilities.

11.1.38 Immutable Application Containers

Utilize immutable application containers, such as Docker containers, to ensure consistent and secure deployment of applications. Immutable containers prevent unauthorized modifications and reduce the risk of vulnerabilities.

Immutable Application Containers are a crucial component of a comprehensive security architecture, providing significant benefits in terms of consistent and secure application deployment. By utilizing immutable application containers, such as Docker containers, organizations can enhance their overall security posture and reduce the risk of vulnerabilities and unauthorized modifications.

It helps to maintain consistent and secure deployments of applications. By preventing unauthorized modifications and reducing the attack surface, these containers enhance the overall security posture and reduce the risk of vulnerabilities. Immutable containers also provide benefits such as isolation, vulnerability mitigation, auditing, compliance, and reproducibility, making them a valuable component of a robust security strategy.

#	Name	Description
1	Isolation and Least Privilege	Immutable containers enable isolation of applications and their dependencies, ensuring that each container operates within its own secure environment. By encapsulating the application and its dependencies within the container, it becomes easier to enforce the principle of least privilege, where each container has only the necessary permissions and access to resources. This isolation helps contain any potential security breaches or vulnerabilities, preventing them from affecting the underlying host system or other containers.
2	Reduced Attack Surface	Immutable containers reduce the attack surface by eliminating the need for additional software installations or modifications within the container runtime. Since the container image is built with all the required dependencies and configurations, there is no need to install unnecessary packages or make runtime changes. This minimizes the number of potential entry points for attackers, reducing the risk of exploitation and unauthorized modifications.

#	Name	Description
3	Vulnerability Mitigation	By leveraging immutable containers, organizations can mitigate the risk of vulnerabilities in several ways. Firstly, containers provide a consistent and controlled environment, making it easier to apply security patches and updates uniformly across all instances. Additionally, immutable containers allow for rapid deployment of patched versions, ensuring that any known vulnerabilities are addressed promptly. Moreover, since containers are immutable, any changes introduced by attackers or malware within the container are discarded when the container is terminated, reducing the impact of potential security breaches.
4	Auditing and Compliance	Immutable containers facilitate better auditing and compliance capabilities. Since containers are immutable and cannot be modified once deployed, it becomes easier to track and verify the integrity of applications throughout their lifecycle. This audit trail provides transparency and accountability, allowing organizations to meet regulatory requirements and demonstrate adherence to security standards.
5	Reproducible Builds	Immutable containers promote reproducibility in the build process. By ensuring that container images are created consistently with a known set of dependencies and configurations, it becomes easier to reproduce the same environment in different stages of the software development lifecycle. This reproducibility simplifies the testing and validation of applications, reducing the likelihood of vulnerabilities being introduced due to variations in the deployment process.

11.1.38.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.
2	SAMM Security by Design	Security Requirements: Immutable application containers help address the activity of defining and incorporating security requirements into the design phase. By utilizing immutable containers, organizations can ensure that the containerized applications meet the required security standards and mitigate vulnerabilities associated with OWASP threats. Secure Architecture: Immutable application containers contribute to secure architecture by providing a consistent and controlled environment for software deployment. They help address OWASP threats such as Insecure Design, Cryptographic Failures, and Security Misconfiguration by minimizing potential vulnerabilities and creating a strong foundation for security controls. Secure Coding Practices: Immutable application containers indirectly support secure coding practices by enforcing isolation and encapsulation. They prevent common coding mistakes and vulnerabilities, reducing the likelihood of OWASP threats such as Injection, Insecure Design, and Security Misconfiguration.

#	Name	Description
		Threat Modeling: Immutable application containers can be considered as part of the threat modeling exercise during the design phase. By incorporating containers into the threat modeling process, organizations can assess the impact of OWASP threats and implement appropriate security controls and countermeasures. Security Testing: Immutable application containers aid security testing during the design phase by providing a consistent and controlled environment for conducting security testing activities. They contribute to identifying and addressing vulnerabilities related to OWASP threats, ensuring that the design is resilient to potential attacks.
3	SAMM Security by Implementation	Secure Development Training: Immutable application containers indirectly support secure development training by providing a standardized and controlled environment for developers. Training programs can include the use of immutable containers to promote secure coding practices and emphasize the importance of containerization in software security. Secure Architecture: Immutable application containers contribute to secure architecture by providing a consistent and controlled environment for software development and deployment. They help address the implementation of secure architectural practices and minimize vulnerabilities associated with OWASP threats. Secure Coding Guidelines: Immutable application containers indirectly support secure coding guidelines by enforcing the use of standardized container images with predefined configurations and dependencies. These guidelines can include recommendations for container security and best practices. Security Requirements: Immutable application containers can assist in implementing security requirements by ensuring that the containerized applications meet the defined security objectives. Containers provide a controlled environment where security requirements can be enforced and validated. Security Testing Integration: Immutable application containers aid in the integration of security testing activities by providing a consistent and reproducible environment for conducting tests. Security testing techniques, such as static code analysis and security code reviews, can be performed on containerized applications to detect and address vulnerabilities associated with OWASP threats. Security Verification: Immutable application containers contribute to security verification by providing a controlled and reproducible environment for verifying the effectiveness of security controls implemented in the software system. Containerized applications can undergo security code reviews, vulnerability assessments, and testing against established security requirements. Security Architecture Review: Immutable application containers can be included in the security architecture review process to evaluate the security of the software system's design and architecture. Containers provide a standardized and reproducible environment for assessing security controls and identifying potential vulnerabilities or weaknesses. Security Operations Integration: Immutable application containers aid in integrating security considerations into operational processes by providing a consistent and controlled deployment mechanism. Containers can be securely deployed and configured, aligning with operational security practices such as incident management, vulnerability management, and secure deployment practices.
4	SAMM Security by Verification	It does not apply directly.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening: Immutable application containers help harden the software environment by providing a consistent and controlled deployment mechanism. The use of immutable containers ensures that the deployed applications are built from known, trusted images, reducing the risk of misconfigurations and unauthorized changes. Secure Build: Immutable containers contribute to secure build practices by encapsulating the application and its dependencies in a container image. This helps ensure that the software is developed with security in mind and that the container image is built using secure coding practices and secure component selection. Configuration Management: Immutable containers simplify configuration management by packaging the application and its dependencies into a self-contained unit. The container image serves as a consistent and reproducible artifact, reducing the complexity of managing software configurations and ensuring that the deployed containers are based on approved and secure images. Vulnerability Management: Immutable containers support vulnerability management efforts by enabling organizations to quickly and easily deploy patched versions of the container images. When a vulnerability is identified, organizations can update the container image with the necessary security patches and deploy the updated image, reducing the exposure to known vulnerabilities. Incident Management: Immutable containers can aid in incident management by facilitating the isolation and containment of security incidents. If a container becomes compromised or affected by a security incident, it can be terminated and replaced with a new, clean instance based on a trusted container image. Secure Deployment: Immutable containers provide a secure deployment mechanism by ensuring that the deployed applications are based on verified and trusted container images. This reduces the risk of deploying vulnerable or compromised software and helps maintain the integrity and security of the deployed systems. Security Testing: Immutable containers support security testing activities by providing a consistent and controlled testing environment. Security tests can be performed on the container image, ensuring that the application and its dependencies are tested in a reproducible and isolated environment. Operational Enablement: Immutable containers enable operational teams to manage and support software systems more effectively. The use of containers simplifies deployment, configuration, and maintenance tasks, reducing the complexity and potential for misconfigurations or human errors that can introduce security vulnerabilities.

11.1.38.2 Assessment – OWASP Top 10

Immutable application containers, such as Docker containers, can directly address several OWASP threats by providing secure deployment, isolation, vulnerability mitigation, and consistent environments for web applications. However, it's important to note that while containers contribute to a secure architecture, they are not a silver bullet and should be combined with other security practices and measures to comprehensively address OWASP threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Immutable application containers help enforce access controls by encapsulating applications and their dependencies within a secure environment. This isolation ensures that each container operates within its own context, preventing unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	It does not address it directly.
3	Injection (A03-2021)	Immutable containers mitigate injection vulnerabilities by ensuring that untrusted data is properly handled and not used to manipulate the application's execution flow or interact with data stores. The containerization process helps enforce secure coding practices and input validation, reducing the risk of injection attacks.
4	Insecure Design (A04-2021)	Immutable containers contribute to secure design principles by providing a controlled and consistent environment for applications. By using containers, organizations can ensure that security considerations are incorporated from the early stages of development, mitigating the risk of insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Immutable containers reduce the likelihood of security misconfigurations by providing a predefined and consistent environment. Since containers are built with all required dependencies and configurations, the risk of default or insecure settings is minimized, enhancing the overall security posture.
6	Vulnerable and Outdated Components (A06-2021)	Immutable containers help address this threat by promoting the active management of software dependencies. With containerization, organizations can easily update and replace container images with newer versions that include security patches, reducing the risk of using outdated or vulnerable components.
7	Identification and Authentication Failures (A07-2021)	Immutable containers contribute to secure identification and authentication mechanisms by providing a secure execution environment for authentication-related components. By isolating authentication processes within containers, organizations can enhance the integrity of user identification and prevent unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Immutable containers assist in ensuring the integrity of software components, configurations, and data within web applications. By enforcing immutability, any unauthorized modifications or tampering attempts are discarded when the container is terminated, reducing the risk of integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Immutable containers do not directly address this threat. While containers can generate logs and be monitored, ensuring effective logging and monitoring practices is more reliant on the logging and monitoring mechanisms implemented at the infrastructure or application level.
10	Server-Side Request Forgery (A10-2021)	Immutable containers can help mitigate SSRF vulnerabilities by enforcing secure configurations and input validation. By controlling the container environment, organizations can prevent attackers from manipulating requests to internal or external resources.

11.1.39 Secure Default Configurations

Setting up secure default configurations for systems, applications, and devices to minimize the attack surface and ensure that security controls are enabled by default.

This architectural security feature involves predefining settings and configurations that prioritize security and enable necessary security controls from the moment a system, application, or device is deployed or initialized.

It minimises the exposure to potential vulnerabilities and entry points that malicious actors can exploit, reducing the risk of unauthorized access, data breaches, and other security incidents.

Also improves time-to-deployment security, meet compliance requirements, enhance resilience, and provide a user-friendly experience.

#	Name	Description
1	Reduced Vulnerabilities	Secure default configurations minimize the number of security vulnerabilities present in a system, application, or device from the outset. By enabling security controls by default, organizations can eliminate common misconfigurations or insecure default settings that could be exploited by attackers.
2	Improved Time-to-Deployment Security	By adopting secure default configurations, organizations can enhance their overall security posture without significant manual intervention. Instead of relying on users or administrators to configure security settings, systems are preconfigured with secure defaults, ensuring that necessary security measures are in place from the start. This approach saves time, reduces human error, and enhances security readiness.
3	Compliance with Best Practices	Secure default configurations align with industry best practices and security standards. By following established guidelines and recommendations, organizations can meet compliance requirements and demonstrate their commitment to security. This can be particularly crucial for industries with stringent regulatory frameworks, such as finance, healthcare, or government sectors.
4	Enhanced Resilience	Secure default configurations contribute to the resilience of systems, applications, and devices. By enabling essential security controls by default, organizations can better withstand attacks, mitigate potential risks, and quickly recover in the event of a security incident.
5	User-Friendly Experience	While security is the primary objective, implementing secure default configurations should also consider the user experience. By striking the right balance between security and usability, organizations can provide a seamless and intuitive user interface while maintaining robust security measures. This encourages users to adopt secure practices and reduces the likelihood of circumventing security controls.
6	For effective implementation of Secure Default Configurations	Threat Modeling: Conduct a comprehensive analysis of potential threats and vulnerabilities specific to the system, application, or device being deployed. This helps identify the most critical security controls that should be enabled by default.

#	Name	Description
7	For effective implementation of Secure Default Configurations	Security Baselines: Establish security baselines or standards that define the minimum set of security configurations for different types of systems, applications, or devices. These baselines should reflect industry best practices and take into account the organization's unique security requirements.
8	For effective implementation of Secure Default Configurations	Automation and Configuration Management: Leverage automation tools and configuration management frameworks to ensure consistent deployment of secure default configurations. This streamlines the process and reduces the risk of human error or inconsistencies in configuring security settings.
9	For effective implementation of Secure Default Configurations	Ongoing Monitoring and Updates: Continuously monitor systems, applications, and devices to identify any deviations from the secure default configurations. Regularly update and patch the configurations as new vulnerabilities are discovered or security requirements evolve.
10	For effective implementation of Secure Default Configurations	User Education and Awareness: Provide user education and awareness programs to help individuals understand the importance of secure default configurations and promote best practices. This empowers users to make informed security decisions and reinforces the organization's overall security culture.

11.1.39.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Secure Default Configurations: By setting secure default configurations, organizations ensure that security controls are enabled by default, aligning with the organization's policies and standards (SAMM's Policies and Standards). This helps establish a robust security governance framework and supports effective decision-making and risk management (SAMM's Risk Management). It also contributes to compliance management by ensuring that secure configurations meet regulatory requirements and industry standards (SAMM's Compliance Management). Additionally, secure default configurations promote security training and awareness, as they enforce secure coding practices and data protection measures (SAMM's Security Training and Awareness). Finally, secure default configurations enable the measurement and reporting of key security metrics, providing visibility into the organization's security posture (SAMM's Security Metrics and Reporting).
2	SAMM Security by Design	Secure Default Configurations: Implementing secure default configurations sets the foundation for secure software design. By enabling security controls by default, organizations address SAMM's Security Requirements by incorporating security concerns into the design phase. Secure default configurations also contribute to the development of a secure architecture, minimizing vulnerabilities and addressing OWASP threats related to insecure design and security misconfiguration. Additionally, secure default configurations align with SAMM's secure coding practices, as they prevent common coding mistakes and vulnerabilities. Finally, secure default configurations support threat modeling exercises by providing a baseline of secure settings and configurations for analysis.

#	Name	Description
3	SAMM Security by Implementation	Secure Default Configurations: Implementing secure default configurations ensures that security measures are integrated into the software development life cycle. By establishing secure development training, organizations promote the adoption of secure default configurations in the early stages of development (SAMM's Secure Development Training). Secure default configurations align with secure architecture practices by providing a baseline of secure settings and controls (SAMM's Secure Architecture). They also support the adoption of secure coding guidelines and the definition of security requirements (SAMM's Secure Coding Guidelines and Security Requirements). Furthermore, secure default configurations facilitate security testing integration and verification, as they provide a consistent starting point for assessing the effectiveness of implemented security controls (SAMM's Security Testing, Security Verification, and Security Architecture Review). Finally, secure default configurations contribute to the integration of security operations by establishing secure deployment practices and supporting operational enablement (SAMM's Secure Deployment and Security Operations Integration).
4	SAMM Security by Verification	Secure Default Configurations: Secure default configurations are not directly related to SAMM's Security by Verification domain. However, they indirectly contribute to security verification by establishing a baseline of secure settings and configurations for the software applications being verified. This baseline helps ensure that security requirements are properly implemented and that security testing activities, such as code review, architecture review, and threat modeling, can be effectively performed.
5	SAMM Security by Operations	Secure default configurations are not directly related to SAMM's Security by Operations domain. However, they indirectly support secure operations by providing a secure starting point for the software environment (SAMM's Environment Hardening) and ensuring that secure build practices are followed during deployment (SAMM's Secure Build). Secure default configurations also contribute to vulnerability management by reducing the initial attack surface and enabling organizations to focus on addressing specific vulnerabilities (SAMM's Vulnerability Management). Finally, they support the secure deployment of software systems by providing a baseline of secure settings and controls (SAMM's Secure Deployment).

11.1.39.2 Assessment – OWASP Top 10

Secure default configurations, organizations can address these OWASP threats from the initial deployment or initialization of systems, applications, and devices. This approach helps minimize attack surfaces, reduces vulnerabilities, and ensures that necessary security controls are enabled, contributing to an overall enhanced security posture.

#	Name	Description
1	Broken Access Control (A01-2021)	By setting secure default configurations, access controls can be properly enforced from the start, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Secure default configurations can include proper cryptographic settings and algorithms, minimizing the risk of weak or insecure cryptographic practices.

#	Name	Description
3	Injection (A03-2021)	Secure default configurations can enforce input validation and secure coding practices, reducing the risk of injection vulnerabilities where untrusted data is improperly handled.
4	Insecure Design (A04-2021)	Secure default configurations can incorporate secure design principles, ensuring that the overall architecture and design of the application are resilient against common design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Secure default configurations help prevent insecure configurations and ensure that systems, frameworks, and web applications adhere to secure practices, reducing the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Secure default configurations can include practices such as actively managing software dependencies and keeping them updated, minimizing the use of outdated or vulnerable components.
7	Identification and Authentication Failures (A07-2021)	Secure default configurations can enforce secure authentication mechanisms and strong password policies, reducing the risk of identification and authentication vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Secure default configurations can help protect the integrity of software components, configurations, and data by incorporating secure coding techniques and ensuring proper data protection measures.
9	Security Logging and Monitoring Failures (A09-2021)	Secure default configurations can include comprehensive logging and monitoring capabilities, ensuring that security-related logs are collected, analyzed, and retained effectively.
10	Server-Side Request Forgery (A10-2021)	Secure default configurations can implement secure input validation and configurations, minimizing the risk of server-side request forgery vulnerabilities.

11.1.40 Immutable Logs

Store logs in an immutable and tamper-evident manner to ensure their integrity. Immutable logs prevent unauthorized modification and can serve as a valuable source of forensic evidence during incident investigations. To achieve such a capability, it is required at the very least implementing secure hash algorithms and digital signatures. These are used for strengthening the tamper-evident nature of logs and provides an additional layer of authentication and non-repudiation, making the logs even more reliable for forensic investigations.

In turn, to implement secure hash algorithms, digital signatures, is required a robust key management solution, audit trail mechanisms, and forensic readiness.

The benefit of immutable logs is that strengthens the integrity, authenticity, and reliability of logs created by any platform or applications, making them highly resilient against unauthorized modifications and invaluable for incident response and forensic investigations

Key Components:

#	Name	Description
1	Secure Hash Algorithms	Secure hash algorithms, such as SHA-256 or SHA-3, are employed to generate unique cryptographic hashes for each log entry. These algorithms take the log data as input and produce a fixed-size hash value, which serves as a digital fingerprint for that specific log entry. The use of secure hash algorithms ensures the integrity of the log data by making it computationally infeasible to modify the log entry without changing its hash.
2	Digital Signatures	Each log entry can be digitally signed using asymmetric cryptography. A digital signature is created by encrypting the hash of the log entry with the private key of the signer. This signature is appended to the log entry, along with the signer's public key, forming a digital proof of authenticity and integrity. Verifying the digital signature with the corresponding public key confirms the log entry's origin and ensures that it hasn't been tampered with since it was signed.
3	Key Management	Key management practices is needed to safeguard the private keys used for digital signatures. Private keys should be stored securely, such as in hardware security modules (HSMs), and access to them should be strictly controlled. Key rotation and revocation mechanisms are also implemented to mitigate the impact of compromised or outdated keys.
4	Audit Trail and Monitoring	In addition to immutable logs, an audit trail and monitoring system can be implemented to track access and modifications to the log data. This includes logging activities such as log creation, modification, and access attempts. These logs should themselves be immutable and securely stored to ensure their integrity and reliability for forensic investigations.
5	Forensic Readiness	When considering immutable logs, the solution should consider the requirements for forensic investigations. This includes capturing relevant metadata along with the log entries, such as timestamps, source IP addresses, user identifiers, and any other contextual information that might be valuable during incident investigations. Proper documentation and guidelines should be established to ensure the availability and preservation of log data for forensic purposes.

11.1.40.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.

#	Name	Description
2	SAMM Security by Design	Secure Hash Algorithms and Digital Signatures contributes to the Security by Design domain because it ensures the integrity and authenticity of system activities. In particular: Security Requirements: By incorporating secure hash algorithms and digital signatures, the architecture ensures the integrity and authenticity of log entries, which aligns with the need for security requirements in the design phase.
3	SAMM Security by Implementation	Secure Hash Algorithms and Digital Signatures contributes to the Security by Implementation domain because it provides a mechanism for ensuring the integrity and reliability of log data. In particular: Secure Development Training: The architecture promotes the use of secure hash algorithms and digital signatures, which aligns with the need for secure coding practices and secure design principles. Secure Architecture: The use of secure hash algorithms and digital signatures in the architecture supports the adoption of secure architectural patterns and principles, contributing to the development of a secure architecture. Secure Coding Guidelines: The architecture reinforces secure coding practices by ensuring the integrity and authenticity of log entries through secure hash algorithms and digital signatures. Security Requirements: The architecture component ensures the integrity of log entries, which aligns with the need to define and document security requirements. Security Testing Integration: By providing tamper-evident logs, the architecture supports security testing activities by enabling the identification and analysis of potential vulnerabilities.
4	SAMM Security by Verification	Using Secure Hash Algorithms and Digital Signatures contributes to the Security by Verification domain because it provides a mechanism to validate and verify the integrity and authenticity of log data. In particular: Security Testing: The architecture ensures the integrity of log data, which supports security testing activities such as penetration testing and vulnerability scanning. Code Review: By providing tamper-evident logs, the architecture helps verify the integrity of log entries, which can be used for code reviews to identify security flaws. Security Architecture Review: The architecture reinforces the security architecture review process by ensuring the integrity and authenticity of log entries, supporting the evaluation of security controls and mechanisms. Security Requirements Verification: The architecture component contributes to verifying the integrity of log entries, which aligns with the need to verify security requirements. Threat Modeling: The architecture supports threat modeling exercises by providing reliable log data for analyzing potential threats and risks. Secure Deployment Verification: The architecture enhances the verification of secure deployment by ensuring the integrity and authenticity of log entries, helping to validate the effectiveness of security controls during deployment.

#	Name	Description
5	SAMM Security by Operations	Secure Hash Algorithms and Digital Signatures contributes to the Security by Operations domain because ir provides a mechanism for securely managing and preserving log data. In particular: Environment Hardening: The architecture reinforces the environment hardening process by ensuring the integrity and authenticity of log data through immutable logs with secure hash algorithms and digital signatures. Secure Build: The architecture component supports secure build practices by providing tamper-evident logs that can be used to verify the integrity of the software system during the build process. Configuration Management: The architecture contributes to configuration management by ensuring the integrity and authenticity of log data, helping detect unauthorized changes or misconfigurations. Vulnerability Management: By ensuring the integrity of log entries, the architecture supports vulnerability management activities by providing reliable data for vulnerability scanning and assessment. Incident Management: The architecture component strengthens incident management by providing tamper-evident logs that serve as valuable forensic evidence during incident investigations. Secure Deployment: The architecture reinforces secure deployment practices by ensuring the integrity and authenticity of log entries, contributing to the secure deployment of software systems.

11.1.40.2 Assessment – OWASP Top 10

Immutable Logs capability contributes to addressing multiple OWASP threats by ensuring the integrity, authenticity, and availability of log data, which is crucial for detecting and investigating security incidents, identifying vulnerabilities, and mitigating risks.

#	Name	Description
1	Broken Access Control (A01-2021)	Immutable logs can provide valuable forensic evidence during incident investigations related to access control breaches. They can help identify unauthorized access attempts, privilege escalation, and exposure of sensitive data, which are key concerns in broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	By using secure hash algorithms and digital signatures, immutable logs with tamper-evident properties can ensure the integrity and authenticity of log entries. This addresses cryptographic failures by implementing proper cryptographic practices, such as secure hash algorithms and digital signatures, which help protect the confidentiality, integrity, and authenticity of the log data.
3	Injection (A03-2021)	Immutable logs can be used as a source of evidence during investigations related to injection attacks. By storing logs in an immutable manner, it becomes more difficult for attackers to manipulate log entries to cover up their injection attempts or unauthorized actions.
4	Insecure Design (A04-2021)	Immutable logs can help identify insecure design decisions by providing a reliable record of system activities and events. By analyzing the logs, organizations can detect indicators of insecure design, such as improper handling of user input or architectural flaws, and take appropriate measures to address them.

#	Name	Description
5	Security Misconfiguration (A05-2021)	Immutable logs can help detect security misconfigurations by capturing events related to misconfigured settings or exposed sensitive information. Analyzing the logs can reveal misconfigurations and allow organizations to rectify them promptly, thereby reducing the risk of security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Immutable logs can assist in identifying vulnerabilities and outdated components within a web application. By analyzing the logs, organizations can detect signs of exploitation or suspicious activities related to vulnerable or outdated components, enabling them to take remedial actions such as patching or updating the affected software.
7	Identification and Authentication Failures (A07-2021)	Immutable logs can provide evidence of identification and authentication failures, such as unauthorized access attempts or manipulation of authentication mechanisms. Analyzing the logs can help identify weaknesses in the identification and authentication mechanisms and guide organizations in implementing more secure authentication practices.
8	Software and Data Integrity Failures (A08-2021)	Immutable logs, with their tamper-evident properties, can help detect unauthorized modifications, manipulation, or destruction of software components, configurations, or data. By analyzing the logs, organizations can identify potential integrity failures and take appropriate measures to ensure the integrity and security of their applications and data.
9	Security Logging and Monitoring Failures (A09-2021)	Immutable logs directly address security logging and monitoring failures by providing a reliable and tamper-evident record of security-related events. By ensuring the immutability of logs, organizations can maintain the integrity and availability of log data, facilitating effective monitoring, analysis, and incident response.
10	Server-Side Request Forgery (A10-2021)	Immutable logs can help detect and investigate server-side request forgery attacks by capturing the requests made by the vulnerable web application. Analyzing the logs can provide insights into unauthorized requests and help identify vulnerabilities or misconfigurations that enabled the SSRF attack.

11.1.41 Secure Remote Access

Implement secure remote access solutions, such as VPNs (Virtual Private Networks) or zero-trust network access (ZTNA), to enable secure connectivity for remote users and protect against unauthorized access. These security measures strengthen the overall security remote users while mitigating the risks of unauthorized access and potential data breaches.

#	Name	Description
1	Multi-factor Authentication (MFA)	Implementing MFA adds an extra layer of security by requiring remote users to provide multiple forms of authentication, such as a password and a unique code sent to their mobile device. This mitigates the risk of unauthorized access even if credentials are compromised.

#	Name	Description
2	Access Controls and Privilege Management	Enforce strict access controls by implementing the principle of least privilege, ensuring that remote users only have access to the resources they require for their specific roles and responsibilities. Regularly review and update user privileges to minimize the potential for unauthorized access.
3	Network Segmentation	Employ network segmentation techniques to isolate remote access systems from the rest of the internal network. This ensures that if a remote access solution is compromised, an attacker will have limited access to other critical resources.
4	Intrusion Detection and Prevention Systems (IDPS)	Deploy IDPS solutions to monitor and detect suspicious activities and potential intrusions in real-time. IDPS can help identify and prevent unauthorized access attempts, providing an additional layer of defense for remote access solutions.
5	Security Information and Event Management (SIEM)	Implement a SIEM system to collect and analyze logs from remote access systems, VPNs, firewalls, and other security devices. This allows for centralized monitoring, correlation of events, and timely detection of security incidents related to remote access.
6	Continuous Monitoring and Auditing	Regularly monitor and audit remote access solutions to identify any vulnerabilities, misconfigurations, or unauthorized access attempts. This helps maintain the security and integrity of the remote access infrastructure.
7	Security Awareness Training	Provide regular security awareness training to remote users to educate them about the importance of secure remote access practices, such as avoiding phishing attacks, using strong passwords, and reporting suspicious activities.
8	Patch Management	Keep remote access systems, VPNs, and related software up to date with the latest security patches and updates. Regularly apply patches to address known vulnerabilities and protect against potential exploits.
9	Encryption	Ensure that all remote access connections, including VPN tunnels and data transmissions, are encrypted using strong encryption protocols (e.g., TLS) to protect the confidentiality and integrity of data in transit.
10	Incident Response Planning	Develop an incident response plan specifically tailored for remote access security incidents. Define clear procedures and responsibilities to effectively respond to and mitigate any security breaches or unauthorized access incidents.

11.1.41.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	It does not address it directly.
3	SAMM Security by Implementation	It does not address it directly.
4	SAMM Security by Verification	It does not address it directly.
5	SAMM Security by Operations	Environment Hardening: Implementing secure remote access solutions, such as VPNs or zero-trust network access (ZTNA), helps harden the software environment by establishing secure connectivity for remote users and protecting against unauthorized access. Secure Build: Incorporating secure remote access solutions into the software build process ensures that the software is developed with security in mind, reducing the likelihood of introducing vulnerabilities during the build process. Configuration Management: Implementing access controls and privilege management for remote access solutions helps ensure that software configurations are properly managed, unauthorized changes are detected, and misconfigurations are remediated. Vulnerability Management: Regularly monitoring and auditing remote access solutions helps identify vulnerabilities and prioritize them for remediation, reducing the risk of exploitation. Incident Management: Developing an incident response plan specifically tailored for remote access security incidents helps organizations handle security incidents effectively, including incident detection, response, containment, and recovery activities. Secure Deployment: Securely deploying remote access solutions and ensuring secure installation procedures and configuration help minimize the risk of unauthorized access or compromise during deployment. Security Testing: Integrating security testing activities, such as penetration testing and vulnerability scanning, into the operational processes helps identify and address security vulnerabilities in software systems. Operational Enablement: Providing security training for operational staff, documenting operational procedures, and establishing incident response capabilities contribute to the ongoing security and reliability of software systems during operations.

11.1.41.2 Assessment – OWASP Top 10

Secure remote access indirectly contributes to mitigating several OWASP threats by incorporating secure practices, access controls, authentication mechanisms, secure configurations, and encryption protocols.

#	Name	Description
1	Broken Access Control (A01-2021)	Implementing multi-factor authentication (MFA) as part of the secure remote access solution helps mitigate the risk of unauthorized access by requiring additional authentication factors, reducing the likelihood of broken access controls.
2	Cryptographic Failures (A02-2021)	The use of secure remote access solutions, such as VPNs or zero-trust network access (ZTNA), ensures that cryptographic algorithms and mechanisms are properly implemented, reducing the risk of cryptographic failures.
3	Injection (A03-2021)	While the capability of secure remote access does not directly address injection vulnerabilities, incorporating secure remote access solutions can help mitigate the risk of injection attacks by providing secure channels for remote users to access applications, reducing the attack surface.
4	Insecure Design (A04-2021)	Secure remote access solutions, when designed and implemented following secure design principles, contribute to overall secure architecture. By incorporating secure remote access, organizations can ensure that their design decisions consider the security requirements and reduce the likelihood of insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Secure remote access solutions need to be properly configured to adhere to secure practices and industry standards. By following secure configuration practices for remote access systems, organizations can avoid security misconfigurations and reduce the risk of this vulnerability.
6	Vulnerable and Outdated Components (A06-2021)	Secure remote access solutions, such as VPNs or ZTNA, should use up-to-date and secure software components. By actively managing and updating the software dependencies of remote access solutions, organizations can reduce the risk of using vulnerable or outdated components.
7	Identification and Authentication Failures (A07-2021)	Secure remote access solutions often involve authentication mechanisms to verify the identity of remote users. By implementing secure authentication mechanisms as part of the remote access solution, organizations can reduce the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	Secure remote access solutions contribute to the overall integrity of software and data by providing secure channels for remote users to access and transmit data. By using strong encryption protocols and secure data transmission mechanisms, organizations can protect against integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Secure remote access solutions should include logging and monitoring capabilities to detect and respond to security incidents. By implementing logging and monitoring mechanisms for remote access systems, organizations can address security logging and monitoring failures and improve incident detection and response.
10	Server-Side Request Forgery (A10-2021)	Secure remote access solutions can help mitigate the risk of server-side request forgery (SSRF) by enforcing strict access controls and secure configurations for the remote access infrastructure. By preventing unauthorized requests and properly validating input, the risk of SSRF vulnerabilities can be reduced.

11.1.42 Back Up and Restore (Kubernetes Disaster Recovery)

Velero is an open-source backup and restore tool specifically designed for Kubernetes clusters. It helps in protecting and restoring the state of Kubernetes resources, including volumes, persistent volumes, namespaces, deployments, and more. Velero enables users to easily create backups of their entire Kubernetes cluster or specific resources and restore them in case of data loss or cluster failures.
It simplifies the process of backup and restore in Kubernetes clusters, ensuring data protection, disaster recovery, and application consistency. Its extensibility and integration capabilities make it a versatile tool for various Kubernetes deployment scenarios.

#	Name	Description
1	Backup and Restore	Velero allows you to create backups of your entire cluster or specific resources, including namespaces, deployments, services, config maps, secrets, and persistent volumes. It captures the state of your cluster at a specific point in time and enables you to restore it later if needed.
2	Application Consistency	Velero ensures application consistency during backups by coordinating the capture of persistent volume snapshots with the backup of associated Kubernetes resources. This ensures that the backed-up data is in a consistent state and can be accurately restored.
3	Volume Snapshots	Velero integrates with Volume Snapshot Providers (VSPs) to create snapshots of persistent volumes. VSPs interact with the underlying storage infrastructure to capture point-in-time snapshots of volumes, ensuring data integrity and consistency during backups and restores.
4	Incremental Backups	Velero supports incremental backups, which means it only captures changes made to resources since the last backup. This reduces the backup time and storage requirements, making the backup process more efficient.
5	Scheduling and Retention Policies	Velero allows you to schedule automated backups at regular intervals. You can configure retention policies to control the number of backups to retain, enabling you to manage your storage resources effectively.
6	Cloud Provider Integration	Velero integrates with cloud providers' APIs and services to optimize backup and restore operations. It leverages cloud-specific features like snapshots, object storage, and block storage to improve performance and reliability.
7	Plugin Architecture	Velero supports a plugin architecture that enables extensibility and customization. Plugins can be developed to support specific storage providers, cloud platforms, or custom requirements. This allows Velero to integrate with different storage systems and backup solutions.
8	Disaster Recovery	In the event of a disaster or cluster failure, Velero helps with the recovery process by restoring the entire cluster or specific resources from a backup. This facilitates business continuity and minimizes downtime.

#	Name	Description
9	Multi-cluster and Multi-cloud Support	Velero can be used to back up and restore resources across multiple Kubernetes clusters and even across different cloud providers. This makes it suitable for complex multi-cluster or multi-cloud architectures.
10	Monitoring and Metrics	Velero provides metrics and logs that can be monitored to ensure the health and effectiveness of backup and restore operations. This helps identify any issues or failures in the backup process and enables timely corrective actions.

11.1.42.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Compliance Management: Velero aids in compliance management by providing a backup and restore solution for Kubernetes resources. Organizations can create backups that capture the desired compliance state of their cluster and ensure that the backups are properly stored and protected. In case of compliance assessments or audits, Velero can help demonstrate adherence to regulatory requirements and best practices. Security Metrics and Reporting: Velero contributes to security metrics and reporting by providing visibility into backup and restore operations. It offers logging and monitoring capabilities, allowing organizations to track backup and restore activities, analyze patterns, and generate reports. This visibility helps in assessing the effectiveness of backup and restore processes and supports decision-making at various levels of the organization. Security Roles and Responsibilities: Velero supports security roles and responsibilities by providing role-based access control (RBAC) mechanisms. Organizations can define fine-grained access controls to manage who can perform backup and restore operations, ensuring that individuals have the necessary privileges and responsibilities to fulfill their roles effectively.
2	SAMM Security by Design	Secure Architecture: While Velero itself does not directly contribute to secure architecture practices, it enables organizations to include backup and restore considerations as part of their secure architecture. By incorporating Velero into the design phase, organizations can ensure that backup and restore capabilities are factored in, reducing the risk of data loss and supporting the overall security and availability of the system. In summary, Velero contributes to SAMM security practices in several domains, including Security by Verification, Security by Operations, Security by Governance, and Security by Design. It helps organizations enhance security testing, environment hardening, configuration management, incident management, secure deployment, compliance management, security metrics and reporting, security roles and responsibilities, and secure architecture in the context of Kubernetes clusters.
3	SAMM Security by Implementation	It does not address it directly.

#	Name	Description
4	SAMM Security by Verification	Security Testing: Velero contributes to the security testing aspect by allowing users to test the backup and restore processes. By regularly performing backup and restore tests using Velero, organizations can verify the integrity of their backup data and ensure that the restore process works as expected, reducing the risk of data loss or corruption.
5	SAMM Security by Operations	Environment Hardening: Velero can be used to backup and restore configurations related to environment hardening. It helps organizations ensure that secure configurations, patches, and updates applied to the cluster are properly backed up and can be restored in case of any issues or failures. Secure Build: Although Velero does not directly contribute to secure build practices, it indirectly supports secure operations by providing a reliable mechanism to backup and restore the Kubernetes cluster, including the applications and configurations. This ensures that the software system is deployed in a secure and controlled manner. Configuration Management: Velero assists in configuration management by allowing users to backup and restore configurations of Kubernetes resources. It helps organizations maintain the integrity of software systems by ensuring that configurations are properly documented, version-controlled, and can be restored if changes or misconfigurations occur. Incident Management: Velero plays a role in incident management by providing the ability to restore the cluster state in case of security incidents or failures. By having reliable backups created with Velero, organizations can recover from incidents more effectively and minimize the impact on the operational environment. Secure Deployment: Velero helps in secure deployment practices by providing a backup and restore mechanism for Kubernetes resources. It ensures that software components are deployed securely by allowing organizations to restore to a known good state in case of any unauthorized access or compromise during deployment. Security Testing: Velero indirectly contributes to security testing by enabling users to perform backup and restore tests. By regularly testing the backup and restore processes, organizations can identify any security vulnerabilities or weaknesses in their deployment and take appropriate measures to address them. Operational Enablement: Velero supports operational enablement by providing operational staff with the ability to manage backups and restores effectively. It helps in documenting operational procedures related to backup and restore processes, enabling operational staff to perform their duties efficiently and ensuring the ongoing security and reliability of software systems.

11.1.42.2 Assessment – OWASP Top 10

Velero helps protect Kubernetes resources and data; however, application developers and administrators must also implement secure coding practices and adhere to security guidelines to address these other OWASP threats effectively.

#	Name	Description
1	Broken Access Control (A01-2021)	It does not address it directly.
2	Cryptographic Failures (A02-2021)	It does not address it directly.
3	Injection (A03-2021)	It does not address it directly.
4	Insecure Design (A04-2021)	It does not address it directly.
5	Security Misconfiguration (A05-2021)	Velero ensures that the backup and restore process is configured properly, including authentication and access controls, to protect sensitive backup data. Proper configuration helps prevent unauthorized access to backups and minimizes the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	It does not address it directly.
7	Identification and Authentication Failures (A07-2021)	It does not address it directly.
8	Software and Data Integrity Failures (A08-2021)	Contribution: Velero ensures the integrity of backed-up data by capturing the state of Kubernetes resources and associated persistent volumes at a specific point in time. This allows for accurate restoration of data, reducing the risk of unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	Velero provides logging and metrics that can be monitored to ensure the health and effectiveness of backup and restore operations. By having comprehensive logging and monitoring mechanisms, organizations can detect any potential security incidents related to the backup process.
10	Server-Side Request Forgery (A10-2021)	It does not address it directly.

11.1.43 Backup and Disaster Recovery

Implementing regular backup and disaster recovery processes is essential for ensuring data availability and minimizing downtime in the event of a security incident or system failure. This feature encompasses a comprehensive set of practices and technologies designed to protect data integrity, enable efficient recovery, and maintain business continuity.

#	Name	Description
1	Data Backup Strategy	Develop a robust data backup strategy that includes regular backups of critical systems, applications, and data. This strategy should consider the frequency of backups, the types of data to be backed up, and the storage locations for backups. Employ a combination of full, incremental, and differential backups to optimize storage and recovery times.
2	Offsite Data Storage	Store backup data in secure offsite locations to protect against physical damage, theft, or other on-site incidents that may affect the primary data center. Use encryption and secure protocols for transmitting backup data to offsite locations to maintain confidentiality.
3	Redundancy and Replication	Implement redundancy and data replication mechanisms to ensure that backups are stored in multiple locations. This approach enhances data availability and reduces the risk of data loss in the event of a disaster. Utilize technologies such as RAID (Redundant Array of Independent Disks) and distributed storage systems to achieve data redundancy and fault tolerance.
4	Regular Testing and Validation	Periodically test the backup and disaster recovery processes to validate their effectiveness. Conduct test recoveries to ensure that backups are usable and can be restored successfully. Regular testing helps identify any gaps in the backup strategy and provides an opportunity to fine-tune the recovery procedures.
5	Disaster Recovery Plan	Develop a comprehensive disaster recovery plan that outlines the steps to be taken in the event of a security incident or system failure. The plan should include clear roles and responsibilities, communication protocols, and predefined recovery procedures. Document the recovery time objectives (RTO) and recovery point objectives (RPO) to guide the recovery process and prioritize critical systems and data.
6	Automated Backup and Monitoring	Utilize automated backup solutions that enable scheduled backups and provide real-time monitoring of backup operations. Automated systems can detect backup failures, notify administrators, and initiate remedial actions promptly. Monitor backup logs and regularly review backup reports to identify any anomalies or potential issues.
7	Secure Backup Infrastructure	Ensure that the backup infrastructure is adequately secured to prevent unauthorized access or tampering. Apply access controls, strong authentication mechanisms, and encryption to protect backup data and prevent unauthorized recovery attempts.
8	Employee Training and Awareness	Train employees on the backup and disaster recovery procedures and their roles during recovery operations. Foster awareness about the importance of backups and the potential impact of data loss. Conduct drills and simulations to familiarize personnel with their responsibilities and ensure they can effectively execute recovery plans.

11.1.43.1 Assessment – SAMM

Implement regular backup and disaster recovery processes to ensure data availability and minimize downtime in the event of a security incident or system failure, which enhance data protection capabilities, minimize downtime, and swiftly recover from security incidents or system failures, ultimately maintaining business continuity.

#	Name	Description
1	SAMM Security by Governance	Establish and maintain a robust security governance framework that enables effective decision-making, risk management, and oversight of software security activities. The implementation of regular backup and disaster recovery processes supports the risk management activity within the Security by Governance domain. It ensures data availability and minimizes downtime in the event of a security incident or system failure, contributing to maintaining business continuity.
2	SAMM Security by Design	Design software applications with security in mind, incorporating security controls and considerations into the design phase of software development. The backup and disaster recovery feature indirectly contributes to the Security by Design domain by providing a means to recover and restore software systems in the event of a security incident or system failure. This supports the overall goal of designing secure and resilient software applications.
3	SAMM Security by Implementation	Implement software security measures during the development and deployment of software systems, integrating security into the software development life cycle. The backup and disaster recovery feature indirectly contributes to the Security by Implementation domain by providing a mechanism to restore software systems in case of security incidents or system failures. It ensures the availability and integrity of software applications, supporting the overall objective of building secure and reliable software systems.
4	SAMM Security by Verification	Validate and verify the security of software throughout its lifecycle, detecting and addressing security vulnerabilities and weaknesses. The backup and disaster recovery feature indirectly contributes to the Security by Verification domain by ensuring the availability and integrity of software systems. It helps verify that the software can be recovered and restored successfully, mitigating potential security risks and vulnerabilities.
5	SAMM Security by Operations	Manage the secure operations of software systems, implementing processes and practices to support the ongoing security and reliability of software applications. The backup and disaster recovery feature directly contributes to the Security by Operations domain by enabling organizations to manage the secure operations of their software systems. It ensures data availability and supports incident management and recovery processes, contributing to the overall security and reliability of the software applications.

11.1.43.2 Assessment – OWASP Top 10

Backup and disaster recovery capability does not directly address all the OWASP threats; however, it contributes to mitigating the impact of several threats by ensuring data availability, integrity, and system recovery in the event of security incidents or system failures. Organizations should also consider incorporating additional security practices to address the specific OWASP threats.

#	Name	Description
11	Broken Access Control (A01-2021)	While the backup and disaster recovery capability does not directly address this threat, it indirectly contributes to mitigating the impact of broken access control. In the event of a security incident or system failure, having a robust backup and recovery process helps organizations restore the system to a known secure state, reducing the risk of unauthorized access.
12	Cryptographic Failures (A02-2021)	The backup and disaster recovery capability does not directly address cryptographic failures. However, organizations can ensure the integrity and confidentiality of backup data by employing encryption and secure protocols during the transmission and storage of backup data.
13	Injection (A03-2021)	The backup and disaster recovery capability does not directly address injection vulnerabilities. However, during the recovery process, it is crucial to ensure the integrity of the restored data by validating and sanitizing inputs to prevent injection attacks that may occur during the restoration process.
14	Insecure Design (A04-2021)	The backup and disaster recovery capability does not directly address insecure design vulnerabilities. However, organizations can include secure design principles in their disaster recovery plan, ensuring that the recovered systems and applications are built with proper security controls and considerations.
15	Security Misconfiguration (A05-2021)	The backup and disaster recovery capability can indirectly contribute to addressing security misconfiguration vulnerabilities. During the recovery process, organizations have an opportunity to review and validate the configuration settings of the recovered systems, ensuring that they adhere to secure practices and industry standards.
16	Vulnerable and Outdated Components (A06-2021)	The backup and disaster recovery capability does not directly address vulnerabilities associated with outdated or vulnerable components. However, during the recovery process, organizations can update and patch the recovered systems to address known vulnerabilities in software components, reducing the risk posed by such components.
17	Identification and Authentication Failures (A07-2021)	The backup and disaster recovery capability does not directly address identification and authentication failures. However, organizations can consider including authentication mechanisms in their recovery process to ensure that only authorized personnel can access and restore the backup data.
18	Software and Data Integrity Failures (A08-2021)	The backup and disaster recovery capability does not directly address identification and authentication failures. However, organizations can consider including authentication mechanisms in their recovery process to ensure that only authorized personnel can access and restore the backup data.

#	Name	Description
19	Security Logging and Monitoring Failures (A09-2021)	The backup and disaster recovery capability does not directly address security logging and monitoring failures. However, organizations can include logging and monitoring components in their recovery process to ensure that security-related events are captured and analyzed, aiding in incident detection and response.
20	Server-Side Request Forgery (A10-2021)	The backup and disaster recovery capability does not directly address server-side request forgery vulnerabilities. However, organizations can review and validate the recovery process to ensure that it does not introduce or amplify such vulnerabilities, preventing unintended requests to internal or external resources.

11.1.44 Secure Supply Chain

Establish controls and verification mechanisms to ensure the integrity of the software supply chain. This includes verifying the authenticity of software components, using secure repositories, and implementing secure software distribution processes.

In today's interconnected and globalized world, ensuring the integrity and security of the software supply chain is of utmost importance. A secure supply chain refers to a set of controls and verification mechanisms put in place to guarantee the authenticity, integrity, and safety of software components throughout their lifecycle. This involves implementing measures to prevent tampering, unauthorized access, and the introduction of malicious code or vulnerabilities into the software.

Incorporating these features into the architecture, it enables a secure supply chain that safeguards software components from tampering, ensures authenticity, and mitigates the risk of introducing vulnerabilities or malicious code. This, in turn, enhances trust among stakeholders and reduces the potential for security breaches, thereby safeguarding critical systems and data.

#	Name	Description
1	Component Authentication:	A crucial aspect of a secure supply chain is verifying the authenticity and integrity of software components. This can be achieved through various means such as digital signatures, cryptographic hashes, or certificates. By implementing these authentication mechanisms, organizations can ensure that the software components they receive are from trusted sources and have not been modified or tampered with during transit.
2	Secure Repositories	Establishing secure repositories is another vital feature of a secure supply chain. These repositories serve as centralized platforms for storing and distributing software components. They should be protected with robust access controls, encryption, and integrity checks to prevent unauthorized access, data breaches, or tampering. Additionally, secure repositories can employ technologies like blockchain to provide an immutable record of software transactions and enhance transparency.

#	Name	Description
3	Secure Software Distribution Processes	Secure software distribution processes play a crucial role in maintaining the integrity of the supply chain. Organizations should implement secure channels for distributing software updates and patches, ensuring end-to-end encryption and protection against unauthorized modifications during transit. This may involve digitally signing software packages, using secure protocols (e.g., HTTPS), and employing secure update mechanisms to validate and apply updates securely.
4	Vulnerability Management	A comprehensive secure supply chain strategy should include robust vulnerability management practices. This entails regularly monitoring and assessing software components for known vulnerabilities and promptly addressing any identified issues. Organizations should establish processes for tracking vulnerabilities, applying security patches and updates, and communicating with software vendors to ensure timely remediation.
5	Supplier Management	A secure supply chain also involves managing relationships with software suppliers and vendors. Organizations should establish strict criteria for selecting and onboarding suppliers, including evaluating their security practices and conducting regular audits. Supplier agreements should include clauses on security requirements, adherence to secure development practices, and compliance with industry standards or regulations.
6	Continuous Monitoring and Auditing	Implementing continuous monitoring and auditing mechanisms is essential to maintain the security of the software supply chain. This involves real-time monitoring of software repositories, distribution processes, and supplier activities to detect any anomalies or suspicious behavior. Regular audits can help identify potential weaknesses or gaps in the supply chain and enable organizations to take proactive measures to address them.
7	Incident Response and Recovery	Despite robust preventive measures, security incidents can still occur within the software supply chain. A secure supply chain architecture should include well-defined incident response and recovery plans to minimize the impact of security breaches or supply chain disruptions. This involves establishing incident response teams, defining escalation procedures, and regularly testing and updating incident response plans to ensure their effectiveness.

11.1.44.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Establishing controls and verification mechanisms in the supply chain helps organizations define and implement policies and standards related to software security, ensuring that secure coding practices, vulnerability management, incident response, and data protection are addressed. Risk Management: Implementing controls in the supply chain helps organizations identify and assess risks associated with software components, enabling them to establish risk management processes and implement risk mitigation strategies.

#	Name	Description
		Compliance Management: Controls and verification mechanisms in the supply chain contribute to ensuring compliance with relevant regulatory requirements, industry standards, and best practices related to software security. Security Training and Awareness: By incorporating secure supply chain practices, organizations can raise awareness and provide training programs to employees and stakeholders about software security, promoting a culture of security throughout the organization. Security Metrics and Reporting: Controls and verification mechanisms in the supply chain help organizations define and measure key security metrics, providing visibility into the status of software security and supporting decision-making at various levels of the organization. Security Roles and Responsibilities: Secure supply chain practices contribute to clearly defining and assigning security roles and responsibilities within the organization, ensuring accountability for software security and providing individuals with the necessary knowledge and skills to fulfill their roles effectively.
2	SAMM Security by Design	It does not address it directly.
3	SAMM Security by Implementation	Secure Development Training: Secure supply chain practices help organizations provide training and awareness programs to developers and development teams on secure coding practices, vulnerability management, and incident response. Security Architecture: Controls and verification mechanisms in the supply chain contribute to designing a secure architecture, which is an essential aspect of secure software development and helps address various security requirements. Security Requirements: By implementing controls and verification mechanisms in the supply chain, organizations can define and incorporate security requirements into the software development process, ensuring that the software meets the desired security objectives.
4	SAMM Security by Verification	Security Testing: Controls and verification mechanisms in the supply chain enable organizations to conduct security testing activities, such as penetration testing and vulnerability scanning, to identify vulnerabilities and weaknesses in software applications. Code Review: By implementing controls in the supply chain, organizations can review the source code of software applications to identify security flaws and ensure adherence to secure coding practices.
5	SAMM Security by Operations	Environment Hardening: Secure supply chain practices contribute to hardening the software environment, including implementing secure configurations and applying patches and updates, which enhances the overall security and reliability of software systems. Secure Build: Controls and verification mechanisms in the supply chain help organizations adopt secure build practices, including secure coding standards and secure build tools, which contribute to building secure software systems. Configuration Management: Implementing controls in the supply chain supports effective configuration management practices, ensuring that software configurations are properly documented, version-controlled, and protected against unauthorized changes or misconfigurations. Vulnerability Management: Secure supply chain practices contribute to implementing a robust vulnerability management process, including vulnerability scanning and penetration testing, to identify and address vulnerabilities in software systems.

#	Name	Description
		Incident Management: Controls and verification mechanisms in the supply chain help organizations establish a well-defined incident management process, which includes incident detection, response, containment, and recovery activities, ensuring the effective handling of security incidents. Secure Deployment: Secure supply chain practices contribute to secure deployment practices, including secure software distribution and secure installation procedures, which help ensure that software is deployed in a secure and controlled manner. Security Testing: Controls and verification mechanisms in the supply chain enable organizations to integrate

11.1.44.2 Assessment – OWASP Top 10

The Secure Supply Chain capability contributes to mitigating several OWASP threats; however, it is part of a broader security strategy that should include other practices and measures to comprehensively address the entire OWASP Top 10 list.

#	Name	Description
1	Broken Access Control (A01-2021)	The Secure Supply Chain capability contributes to addressing this threat indirectly by ensuring that software components are authenticated and verified before being integrated into the application. By implementing controls and verification mechanisms in the supply chain, organizations can prevent unauthorized access to software components and mitigate the risk of broken access controls.
2	Cryptographic Failures (A02-2021)	The Secure Supply Chain capability directly addresses this threat by emphasizing the use of authentication mechanisms such as digital signatures, cryptographic hashes, and certificates. By ensuring the authenticity and integrity of software components through cryptographic measures, organizations can mitigate the risk of cryptographic failures within their applications.
3	Injection (A03-2021)	The Secure Supply Chain capability does not directly address the Injection threat. However, by implementing secure software distribution processes and applying security updates and patches securely, organizations can reduce the likelihood of injection vulnerabilities introduced through the supply chain.
4	Insecure Design (A04-2021)	The Secure Supply Chain capability does not directly address the Insecure Design threat. However, by establishing secure repositories and implementing controls to prevent tampering or unauthorized modifications of software components, organizations can reduce the risk of insecure design decisions being introduced into the application through the supply chain.
5	Security Misconfiguration (A05-2021)	The Secure Supply Chain capability indirectly addresses this threat by emphasizing the need for secure configurations and secure software distribution processes. By ensuring that software components are obtained from secure repositories and distributed through secure channels, organizations can reduce the risk of security misconfigurations within their applications.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	The Secure Supply Chain capability directly addresses this threat by focusing on establishing controls and verification mechanisms to ensure the integrity of software components. By verifying the authenticity and applying secure practices in the supply chain, organizations can minimize the risk of incorporating vulnerable or outdated components into their applications.
7	Identification and Authentication Failures (A07-2021)	The Secure Supply Chain capability does not directly address the Identification and Authentication Failures threat. However, by implementing secure software distribution processes and ensuring the authenticity and integrity of software components, organizations can reduce the risk of introducing vulnerabilities that could lead to identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	The Secure Supply Chain capability directly addresses this threat by establishing controls and verification mechanisms to ensure the integrity and protection of software components and data within web applications. By preventing tampering and unauthorized modifications through the supply chain, organizations can mitigate the risk of software and data integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	The Secure Supply Chain capability does not directly address the Security Logging and Monitoring Failures threat. However, by incorporating continuous monitoring and auditing mechanisms in the supply chain, organizations can enhance their overall logging and monitoring capabilities and detect potential security incidents related to the software components obtained through the supply chain.
10	Server-Side Request Forgery (A10-2021)	The Secure Supply Chain capability does not directly address the Server-Side Request Forgery threat. However, by implementing secure software distribution processes and ensuring the integrity of software components, organizations can reduce the likelihood of incorporating components vulnerable to SSRF attacks into their applications.

11.1.45 Secure Data Handling

Implement secure data handling practices, including data minimization, secure data storage, and secure data deletion to ensure that sensitive data is appropriately protected throughout its lifecycle.

It should be implemented a process to document and maintain an inventory of sensitive data, including its location, access controls, and data handling procedures. Regular audits and vulnerability assessments should be conducted to ensure compliance with secure data handling practices. One of the most important practices to be considered is Data Minimisation, which is a strategy that involves collecting and storing only the minimum amount of sensitive data required to fulfill business objectives. So, focusing in minimizing the data collection could be used to reduce the risk of data breaches and limit their exposure to potential security threats. There are several aspect that need to be considered to achieve data minimization.

#	Name	Description
1	Data anonymization	Implement data anonymization or pseudonymization techniques to reduce the sensitivity of stored data.

#	Name	Description
2	Data retention policies	Implement and regularly review and update data retention policies to ensure data that is no longer necessary.
3	Secure data storage	Secure data storage ensures that sensitive data is protected from unauthorized access, disclosure, or tampering while at rest. Here are some practices to consider: Secure data storage ensures that sensitive data is protected from unauthorized access, disclosure, or tampering while at rest. Here are some practices to consider: Encryption: Implement strong encryption algorithms (e.g., AES-256) to encrypt sensitive data before storing it. Use industry best practices for key management, including key rotation and secure key storage. Access Controls: Employ strict access controls to limit access to sensitive data. Implement role-based access controls (RBAC) and enforce the principle of least privilege, ensuring that only authorized personnel can access specific data. Secure Storage Infrastructure: Store sensitive data in secure environments such as encrypted databases, file systems, or cloud storage. Regularly monitor and patch storage systems to address potential vulnerabilities. Data Integrity: Implement measures to ensure the integrity of stored data, such as implementing checksums or digital signatures to detect any unauthorized modifications.
4	Secure Data Deletion	Secure data deletion is essential to ensure that sensitive data is permanently removed from storage media when it is no longer needed. Simply deleting files or data entries may not be sufficient, as they can still be recovered. Consider the following practices: Data Sanitization: Use secure deletion techniques (e.g., overwriting, degaussing, or physical destruction) to ensure that data is irreversibly removed from storage media. Secure Disposal: Establish policies and procedures for the proper disposal of storage media that may contain sensitive data. This includes secure shredding or degaussing of hard drives, erasing data from solid-state drives (SSDs), or engaging professional data destruction services.

11.1.45.1 Assessment – SAMM

The "Secure Data Handling" capability contributes to various aspects of SAMM security domains by promoting secure data storage, data minimization, encryption, access controls, and proper data disposal practices, which collectively enhance software security and data protection throughout the software development lifecycle.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: The implementation of data retention policies and secure data storage practices contributes to defining and implementing policies that ensure data protection and compliance with regulatory requirements. Risk Management: Secure data handling practices, including data minimization and secure data storage, contribute to identifying and mitigating risks associated with the handling of sensitive data. Compliance Management: Secure data handling practices help ensure compliance with data protection regulations and industry standards.

#	Name	Description
		Security Training and Awareness: Training programs on secure data handling educate employees and stakeholders about the importance of protecting sensitive data. Security Metrics and Reporting: Monitoring and audits of secure data handling practices contribute to measuring and reporting security metrics related to data protection.
2	SAMM Security by Design	Security Requirements: Implementing data minimization and secure data storage practices helps address security requirements related to data protection and privacy. Secure Architecture: Secure data storage practices, such as encryption and access controls, contribute to designing a secure architecture that mitigates risks associated with data breaches. Secure Coding Practices: Data anonymization techniques and secure data storage practices can be considered secure coding practices to prevent vulnerabilities related to sensitive data exposure. Threat Modeling: Threat modeling exercises can include analyzing potential threats and risks associated with the mishandling of sensitive data. Security Testing: Testing the security controls related to secure data handling contributes to validating the effectiveness of security measures.
3	SAMM Security by Implementation	Secure Development Training: Secure data handling practices, including data minimization and secure data storage, can be incorporated into training programs for developers. Secure Architecture: Implementing secure data storage practices and data retention policies contributes to building a secure software architecture. Secure Coding Guidelines: Secure coding guidelines can include recommendations for handling sensitive data, such as using encryption and access controls. Security Requirements: Implementation of secure data handling practices ensures compliance with security requirements related to data protection. Security Testing Integration: Integrating security testing of data handling practices helps identify vulnerabilities and weaknesses in software systems.
4	SAMM Security by Verification	Security Testing: Security testing activities can include evaluating the effectiveness of secure data handling practices, such as encryption and data sanitization. Code Review: Code reviews can assess the implementation of secure data handling practices and the proper integration of encryption and access controls. Security Architecture Review: Reviewing the security architecture includes evaluating the design and implementation of secure data storage mechanisms. Security Requirements Verification: Verifying the implementation of secure data handling practices ensures compliance with security requirements. Threat Modeling: Threat modeling can include analyzing potential threats related to data breaches and the effectiveness of data minimization strategies. Secure Deployment Verification: Verification of secure deployment practices includes ensuring that data handling controls are properly configured in the production environment.
5	SAMM Security by Operations	Environment Hardening: Secure data storage practices, including encryption and access controls, contribute to hardening the software environment against data breaches. Secure Build: Secure data handling practices can be integrated into secure build processes to ensure that sensitive data is handled appropriately during development.

#	Name	Description
		Configuration Management: Documentation of data handling procedures and access controls contributes to effective configuration management. Vulnerability Management: Vulnerability management includes assessing vulnerabilities related to data handling practices and ensuring their timely mitigation. Incident Management: Incident management processes include procedures for handling data breaches and unauthorized access to sensitive information. Secure Deployment: Secure deployment practices ensure that data handling controls are maintained during software deployment. Security Testing: Regular security testing activities include assessing the effectiveness of secure data handling measures. Operational Enablement: Providing operational support and guidance includes ensuring that operational staff understands and follows secure data handling procedures.

11.1.45.2 Assessment – OWASP Top 10

Secure Data Handling capability may not directly address all OWASP threats; however, it contributes significantly to mitigating several of them by promoting secure data storage, access controls, encryption, and proper data deletion techniques. It also encourages organizations to conduct regular audits and vulnerability assessments to ensure compliance with secure data handling practices, which can help identify and address potential vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	Secure data handling, specifically the implementation of access controls, helps address this threat. By enforcing strict access controls and employing role-based access controls (RBAC), organizations can prevent unauthorized access to sensitive data, thereby mitigating the risk of broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	The Secure Data Handling capability emphasizes the use of strong encryption algorithms (e.g., AES-256) and proper key management practices. By implementing these measures, organizations can mitigate cryptographic failures and ensure the confidentiality and integrity of sensitive data.
3	Injection (A03-2021)	While the Secure Data Handling capability does not directly address injection vulnerabilities, it indirectly contributes to their mitigation. By implementing secure data handling practices, organizations can reduce the risk of data manipulation and unauthorized access, which are common consequences of injection attacks.
4	Insecure Design (A04-2021)	The Secure Data Handling capability, particularly the emphasis on secure data storage and secure deletion, can help address insecure design vulnerabilities. By securely storing sensitive data and implementing proper data deletion techniques, organizations can prevent attackers from exploiting design flaws to gain unauthorized access or compromise data integrity.

#	Name	Description
5	Security Misconfiguration (A05-2021)	The Secure Data Handling capability indirectly addresses security misconfiguration vulnerabilities by promoting secure data storage and access controls. By securely configuring storage systems and implementing strict access controls, organizations can reduce the risk of misconfiguration and enhance the overall security of their applications.
6	Vulnerable and Outdated Components (A06-2021)	The Secure Data Handling capability does not directly address this threat. However, by implementing secure data handling practices, organizations can minimize the impact of vulnerabilities in components that handle sensitive data.
7	Identification and Authentication Failures (A07-2021)	While the Secure Data Handling capability does not directly address identification and authentication vulnerabilities, it indirectly contributes to their mitigation. By securely handling sensitive data, organizations can prevent unauthorized access and impersonation, thereby reducing the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	The Secure Data Handling capability contributes to addressing software and data integrity failures by implementing measures to ensure the integrity of stored data. By employing encryption, access controls, and data integrity checks, organizations can protect against unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	The Secure Data Handling capability does not directly address this threat. However, by maintaining an inventory of sensitive data and conducting regular audits and vulnerability assessments, organizations can improve their overall security logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	The Secure Data Handling capability does not directly address this threat. However, by implementing secure data handling practices and access controls, organizations can reduce the risk of SSRF vulnerabilities that may result from unauthorized access to sensitive data.

11.1.46 Zero Trust Architecture

Implement a zero trust architecture that assumes no trust by default, regardless of network location. Enforce strict authentication, authorization, and access controls for all users, devices, and applications, regardless of their location or network.

Implementing a zero trust architecture requires a comprehensive understanding of the organization's infrastructure, data flows, and access requirements. It also involves integrating various security technologies, such as identity and access management (IAM), network security controls, and security information and event management (SIEM) systems, to enable seamless enforcement of zero trust principles.

Zero trust security architecture enhances security posture by minimizing the risk of unauthorized access, lateral movement, and data breaches. It provides a more robust and proactive approach to security, irrespective of the location or network environment.

#	Name	Description
1	Strict Authentication	Implement strong authentication mechanisms to verify the identity of users, devices, and applications before granting access to resources. This may include multi-factor authentication (MFA), biometric authentication, or other advanced authentication methods. By enforcing strict authentication, organizations can ensure that only authorized entities gain access to sensitive resources.
2	Authorization and Access Controls	Enforce granular authorization and access controls to limit user and application access to the minimum necessary privileges required to perform their tasks. This involves implementing role-based access controls (RBAC), attribute-based access controls (ABAC), or other access control models based on the principle of least privilege. Regularly review and update access controls to adapt to changing roles and responsibilities
3	Continuous Monitoring	Implement continuous monitoring of user, device, and application behavior to detect and respond to potential security incidents or anomalies. This involves collecting and analyzing data from various sources, such as network logs, endpoint logs, and user behavior analytics. By monitoring activity and detecting suspicious patterns or deviations from normal behavior, organizations can identify potential threats and take appropriate action.
4	Secure Access	Implement secure access mechanisms, such as virtual private networks (VPNs), secure web gateways, or software-defined perimeter (SDP) solutions, to establish secure connections for remote users and devices. This ensures that data and applications are accessed over encrypted and authenticated channels, regardless of the user's location.
5	Continuous Verification and Trust Assessment:	Continuously verify the trustworthiness of users, devices, and applications throughout their lifecycle. This includes periodic re-authentication, device health checks, and assessment of application vulnerabilities. By regularly assessing and verifying trust, organizations can mitigate the risk of compromised entities gaining prolonged access to critical resources.
6	Incident Response and Remediation	Establish an incident response plan that outlines procedures for detecting, responding to, and recovering from security incidents. This includes processes for isolating compromised entities, conducting forensics, and applying necessary patches or updates. Regularly test and update the incident response plan to ensure its effectiveness in addressing evolving threats.

11.1.46.1 Assessment – SAMM

Zero trust architecture is a security approach that provides a foundation for addressing security domains and activities across SAMM by enforcing strict authentication, authorization, and access controls, regardless of network location or user/device/application type. Its contributions to specific SAMM domains and activities may vary based on the organization's implementation and context.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Implementing a zero trust architecture helps organizations establish and enforce policies and standards for software security. It ensures that strict authentication, authorization, and access controls are implemented regardless of network location or user/device/application type. Risk Management: Zero trust architecture contributes to risk management by minimizing the risk of unauthorized access, lateral movement, and data breaches. It establishes a framework that assumes no trust by default and implements strict security measures to mitigate potential risks. Compliance Management: Implementing a zero trust architecture helps organizations comply with relevant regulatory requirements and industry standards related to software security. It ensures that strict authentication, authorization, and access controls are enforced, reducing the risk of non-compliance.
2	SAMM Security by Design	Security Requirements: Zero trust architecture addresses security requirements by enforcing strict authentication, authorization, and access controls as core design principles. It ensures that security is integrated into the design phase by considering potential threats and vulnerabilities. Secure Architecture: Zero trust architecture promotes the adoption of secure architectural patterns and principles, such as network segmentation and secure access mechanisms. These practices contribute to addressing OWASP threats and ensuring a resilient and secure software architecture.
3	SAMM Security by Implementation	Secure Architecture: Zero trust architecture aligns with secure architecture practices by implementing network segmentation and secure access mechanisms. These measures contribute to the implementation of a secure software system and help mitigate OWASP threats. Secure Coding Guidelines: Zero trust architecture enforces strict authentication, authorization, and access controls, which align with secure coding guidelines. By implementing secure coding practices, organizations can reduce the likelihood of OWASP threats such as Injection, Insecure Design, and Security Misconfiguration.
4	SAMM Security by Verification	SAMM Security by Verification, Security Testing: Zero trust architecture supports security testing activities by enforcing strict authentication, authorization, and access controls. By continuously monitoring user, device, and application behavior, organizations can detect and respond to potential security incidents or anomalies.
5	SAMM Security by Operations	Environment Hardening: Implementing a zero trust architecture contributes to environment hardening by enforcing strict authentication, authorization, and access controls. It helps protect the software system from known vulnerabilities and security weaknesses. Configuration Management: Zero trust architecture aligns with effective configuration management practices by integrating secure access controls and secure deployment mechanisms. This helps maintain the integrity of software systems and ensures that unauthorized changes or misconfigurations are detected and remediated.

11.1.46.2 Assessment – OWASP Top 10

The Zero Trust Architecture capability addresses several OWASP threats by implementing a security approach that assumes no trust by default and enforces strict authentication, authorization, and access controls. Zero Trust Architecture may not directly address all OWASP threats; however, it provides a security framework and approach that, when combined with other secure practices and controls, can significantly enhance the overall security posture of applications and mitigate many common vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	Zero Trust Architecture contributes to addressing broken access control by enforcing strict authentication and access controls for all users, devices, and applications. It ensures that access to resources is granted based on proper authorization and privileges, minimizing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	While Zero Trust Architecture itself may not directly address cryptographic failures, it provides a foundation for implementing proper cryptographic practices within the overall security architecture. By adopting a zero trust approach, organizations can ensure that cryptographic mechanisms are used to protect data and communications, reducing the risk of cryptographic vulnerabilities.
3	Injection (A03-2021)	Zero Trust Architecture does not directly address injection vulnerabilities. However, it can contribute to the prevention of injection attacks by implementing secure coding practices, input validation, and regular security testing. These practices, when combined with the zero trust approach, can significantly reduce the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	Zero Trust Architecture indirectly addresses insecure design by promoting a security-focused approach to the design and architecture of applications. By considering security requirements from the early stages and implementing appropriate security controls, organizations can mitigate the risk of insecure design decisions and enhance the overall security of their applications.
5	Security Misconfiguration (A05-2021)	Zero Trust Architecture does not directly address security misconfigurations. However, it emphasizes the importance of proper configuration and secure practices within the security architecture. By following secure configuration practices and conducting regular security assessments, organizations can reduce the likelihood of security misconfigurations within their applications and associated components.
6	Vulnerable and Outdated Components (A06-2021)	Zero Trust Architecture does not directly address vulnerable and outdated components. However, it can contribute to reducing the risk of such vulnerabilities by promoting a proactive approach to managing software dependencies. By actively managing and updating components, organizations can minimize the use of vulnerable or outdated software, enhancing the overall security of their applications.
7	Identification and Authentication Failures (A07-2021)	Zero Trust Architecture directly addresses identification and authentication failures by enforcing strict authentication mechanisms and access controls for all users, devices, and applications. By implementing secure authentication mechanisms and strong password policies, organizations can reduce the risk of unauthorized access and impersonation attacks.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	Zero Trust Architecture indirectly addresses software and data integrity failures by promoting secure practices and regular vulnerability testing. By ensuring secure coding techniques, proper input validation, and integrity checks, organizations can mitigate the risk of unauthorized modifications and data tampering within their applications.
9	Security Logging and Monitoring Failures (A09-2021)	Zero Trust Architecture does not directly address logging and monitoring failures. However, it emphasizes the importance of continuous monitoring as part of the security approach. By implementing comprehensive logging, effective log analysis, and real-time monitoring mechanisms, organizations can detect and respond to security incidents in a timely manner.
10	Server-Side Request Forgery (A10-2021)	Zero Trust Architecture does not directly address server-side request forgery (SSRF). However, it can contribute to mitigating SSRF vulnerabilities by implementing input validation and secure configurations. By ensuring that web applications validate and sanitize user-supplied input, organizations can reduce the risk of SSRF attacks.

11.1.47 Code Obfuscation

Employ code obfuscation techniques to make it harder for attackers to understand and reverse-engineer the application code. This adds an extra layer of protection for intellectual property and sensitive algorithms.

It is important to note that while code obfuscation can significantly increase the complexity of reverse-engineering efforts, it should not be considered a foolproof security measure. It is essential to combine code obfuscation with other security practices, such as strong authentication, secure network communication, and secure coding practices, to create a robust security architecture.

The following practices can be employed to achieve effective code obfuscation:

#	Name	Description
1	Name Obfuscation	Rename variables, functions, classes, and other code elements to non-descriptive or misleading names. This makes it difficult for attackers to comprehend the purpose and functionality of the code.
2	Control Flow Obfuscation	Modify the order of instructions, introduce additional branching, or use techniques like dead code insertion and junk code generation. This obscures the logical flow of the code, making it harder for attackers to follow and analyze.
3	Data Obfuscation	Apply techniques such as data encryption, data encoding, or data splitting to obscure the representation of sensitive data within the code. This prevents attackers from easily extracting or understanding the actual data.
4	Code Encryption:	Encrypt critical parts of the code to protect sensitive algorithms or proprietary logic. This ensures that even if an attacker gains access to the code, it remains unintelligible without the decryption key.

#	Name	Description
5	Anti-Tampering Measures	Introduce integrity checks and anti-tampering mechanisms within the code to detect unauthorized modifications. These measures can include checksum verification, digital signatures, or code obfuscation detectors that trigger alerts when tampering is detected.
6	String Obfuscation	Hide sensitive strings (e.g., API keys, database credentials) by encrypting or encoding them within the code. This makes it harder for attackers to extract these valuable pieces of information.
7	Code Packing	Utilize code packing or binary packing techniques to compress and obfuscate the application's executable files. This not only makes it challenging for attackers to analyze the code but also helps prevent reverse-engineering and unauthorized modifications.
8	Dynamic Code Generation	Generate code dynamically at runtime instead of having it statically present in the application's binaries. This technique makes it harder for attackers to identify and analyze the code as it is not readily available.

11.1.47.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.
2	SAMM Security by Design	Security Requirements: Incorporating security requirements into the design phase helps address OWASP threats such as Broken Access Control, Injection, and Insecure Design by proactively mitigating vulnerabilities. Secure Architecture: Adopting secure architectural patterns and principles helps address OWASP threats such as Insecure Design, Cryptographic Failures, and Security Misconfiguration by minimizing potential vulnerabilities. Secure Coding Practices: Adhering to secure coding practices reduces the likelihood of OWASP threats such as Injection, Insecure Design, and Security Misconfiguration by preventing common coding mistakes and vulnerabilities. Threat Modeling: Conducting threat modeling exercises during the design phase helps identify and mitigate potential security risks associated with OWASP threats. Security Testing: Including security testing activities in the design phase helps detect and address vulnerabilities related to OWASP threats, ensuring a resilient design.
3	SAMM Security by Implementation	Secure Development Training: Providing training on secure coding practices helps address OWASP threats by ensuring developers have the necessary knowledge and skills to build secure software. Secure Architecture: Using secure architecture practices helps address OWASP threats by identifying and mitigating potential vulnerabilities and risks early in the development process. Secure Coding Guidelines: Establishing and following secure coding guidelines reduces the likelihood of OWASP threats by providing recommendations for writing secure code. Security Requirements: Defining and implementing security requirements helps address OWASP threats by setting the foundation for designing and implementing security controls.

#	Name	Description
		Security Testing Integration: Integrating security testing activities throughout the development life cycle helps identify and address vulnerabilities related to OWASP threats. Security Verification: Using security verification techniques validates the effectiveness of security controls in addressing OWASP threats. Security Architecture Review: Conducting security architecture reviews helps evaluate and improve the security of the software system's design and architecture. Security Operations Integration: Integrating security considerations into operational processes helps support and maintain security measures throughout the software's life cycle.
4	SAMM Security by Verification	Security Testing: Conducting security testing activities, such as penetration testing and vulnerability scanning, helps identify vulnerabilities and weaknesses, addressing OWASP threats. Code Review: Reviewing source code helps identify security flaws and adherence to secure coding practices, mitigating OWASP threats. Security Architecture Review: Assessing the security architecture ensures the presence of proper security controls, addressing OWASP threats. Security Requirements Verification: Verifying the implementation of security requirements helps ensure alignment with OWASP best practices. Threat Modeling: Conducting threat modeling exercises helps identify and prioritize security threats, including those outlined in OWASP. Secure Deployment Verification: Ensuring secure deployment and configuration helps mitigate OWASP threats.
5	SAMM Security by Operations	Environment Hardening: Implementing secure configurations and infrastructure protects software systems from vulnerabilities, including those related to OWASP threats. Secure Build: Following secure build practices reduces the introduction of vulnerabilities during development, addressing OWASP threats. Configuration Management: Effective configuration management helps detect and remediate unauthorized changes or misconfigurations, mitigating OWASP threats. Vulnerability Management: Robust vulnerability management processes help identify and address vulnerabilities associated with OWASP threats. Incident Management: Well-defined incident management processes minimize the impact of security incidents, including those arising from OWASP threats. Secure Deployment: Secure deployment practices prevent unauthorized access or compromise during deployment, mitigating OWASP threats. Security Testing: Regular security testing, such as static code analysis and dynamic application security testing, helps identify and address vulnerabilities related to OWASP threats. Operational Enablement: Providing operational support and guidance ensures ongoing security and reliability, minimizing the risk of OWASP threats.

11.1.47.2 Assessment – OWASP Top 10

Code obfuscation provides an additional layer of protection for intellectual property and sensitive algorithms; however, it does not directly address all OWASP threats. It should be combined with other

security practices, such as secure design, secure coding, secure configurations, and proper input validation, to create a robust security architecture that effectively mitigates a broader range of threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Code obfuscation indirectly contributes to mitigating this threat by protecting the application code from unauthorized access and tampering. By obfuscating the code, it becomes harder for attackers to understand the internal workings of the application and find vulnerabilities that could be exploited to bypass access controls.
2	Cryptographic Failures (A02-2021)	Code obfuscation does not directly address cryptographic failures. While it can help protect sensitive algorithms or proprietary logic within the code, it does not address the implementation of cryptographic algorithms or key management issues. To address cryptographic failures, organizations should focus on using secure and well-tested cryptographic libraries, implementing cryptographic algorithms correctly, and managing encryption keys securely.
3	Injection (A03-2021)	Code obfuscation indirectly contributes to mitigating injection vulnerabilities. By obfuscating the code, it becomes more challenging for attackers to identify injection points and craft malicious input that could exploit these vulnerabilities. However, code obfuscation should not be relied upon as the primary defense against injection attacks. Implementing proper input validation, sanitization, and parameterization techniques is crucial to address injection vulnerabilities effectively.
4	Insecure Design (A04-2021)	Code obfuscation does not directly address insecure design vulnerabilities. Insecure design vulnerabilities stem from architectural flaws and poor security considerations during the design phase. While code obfuscation may make it harder for attackers to analyze the code, it does not address the underlying design flaws. To mitigate insecure design vulnerabilities, organizations should follow secure design principles, conduct threat modeling, and ensure security is considered throughout the development lifecycle.
5	Security Misconfiguration (A05-2021)	Code obfuscation does not directly address security misconfiguration vulnerabilities. Security misconfigurations occur due to improper configurations of systems, frameworks, servers, or web applications. Code obfuscation focuses on obscuring the code's functionality and structure but does not address the configuration settings themselves. Organizations should adopt secure configuration practices, regularly update software, and perform security assessments to mitigate security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Code obfuscation does not directly address vulnerabilities arising from the use of outdated or vulnerable components. While obfuscating the code may make it harder for attackers to identify specific components, it does not address the underlying vulnerabilities present in those components. Organizations should actively manage software dependencies, keep them updated with security patches, and use secure coding practices to address vulnerabilities related to components.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Code obfuscation does not directly address identification and authentication failures. These vulnerabilities relate to weaknesses in the mechanisms used to identify and authenticate users. Code obfuscation primarily focuses on protecting intellectual property and sensitive algorithms within the code but does not directly address authentication vulnerabilities. Implementing secure authentication mechanisms, enforcing strong password policies, and implementing multi-factor authentication are essential to mitigate identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	Code obfuscation indirectly contributes to mitigating software and data integrity failures. By obfuscating the code, it becomes harder for attackers to modify or manipulate the software components and data. However, code obfuscation should not be the sole defense against integrity failures. Implementing secure coding practices, input validation, and integrity checks are essential to protect against unauthorized modifications and data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	Code obfuscation does not directly address security logging and monitoring failures. These vulnerabilities relate to deficiencies in the logging and monitoring capabilities of web applications. While code obfuscation may make it harder for attackers to understand the code and potentially evade detection, it does not address the logging and monitoring mechanisms themselves. Organizations should focus on implementing comprehensive logging, log analysis, and real-time monitoring to address these vulnerabilities.
10	Server-Side Request Forgery (A10-2021)	Code obfuscation does not directly address server-side request forgery vulnerabilities. Server-side request forgery occurs due to improper handling of user input and making unintended requests to internal or external resources. Code obfuscation primarily focuses on protecting the code's functionality and structure but does not address the underlying vulnerabilities that enable SSRF attacks. Implementing robust input validation, secure configurations, and thorough security testing are necessary to mitigate SSRF vulnerabilities effectively.

11.1.48 Intrusion Prevention System (IPS)

Deploy Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) solutions to detect and prevent unauthorized access, malicious activities, and network attacks.

An Intrusion Prevention System (IPS) is a crucial component of a comprehensive security architecture that helps detect and prevent unauthorized access, malicious activities, and network attacks. It works in conjunction with an Intrusion Detection System (IDS) to provide real-time protection and enhance the overall security posture of a network.

The primary goal of an IPS is to proactively identify and block potential threats before they can cause any harm to the network or its resources. It accomplishes this by actively monitoring network traffic, analyzing it for suspicious patterns, and taking appropriate action to prevent malicious activities. An IPS can be deployed either as a hardware appliance or as software integrated into network devices such as routers, switches, or firewalls.

Deploying an Intrusion Prevention System (IPS) as part of a security architecture can enhance the ability to detect and prevent network attacks, safeguard sensitive data, and maintain a secure computing

environment. The combination of an IPS and IDS helps create a robust defense-in-depth strategy that complements other security measures, such as firewalls, antivirus software, and secure network configurations.

Key features and capabilities of an Intrusion Prevention System include:

#	Name	Description
1	Real-time Threat Detection	An IPS continuously monitors network traffic and analyzes it in real-time to identify any suspicious behavior or known attack patterns. It uses various techniques such as signature-based detection, anomaly detection, and behavioral analysis to spot potential threats.
2	Prevention of Network Attacks	Upon detecting a threat, an IPS can take immediate action to prevent the attack from succeeding. This can involve blocking malicious traffic, terminating connections, or modifying firewall rules to prevent further access. By actively preventing attacks, an IPS adds an extra layer of security to the network.
3	Signature-based and Behavior-based Detection	An IPS relies on a vast database of known attack signatures to identify and block known threats. It compares network traffic against these signatures to identify matches and take action. Additionally, behavior-based detection techniques are used to identify anomalies in network behavior that may indicate new or unknown threats.
4	Deep Packet Inspection (DPI)	An IPS performs deep packet inspection on network traffic to analyze the content of packets and identify any potential threats. It examines not only the packet headers but also the payload to detect and prevent attacks that may be hidden within legitimate-looking traffic.
5	Integration with Security Information and Event Management (SIEM) Systems	An IPS can integrate with SIEM systems to provide a centralized view of security events and facilitate comprehensive security monitoring. Integration with SIEM allows for better correlation of events, threat intelligence sharing, and streamlined incident response.
6	Automatic Updates and Threat Intelligence	To stay effective against emerging threats, an IPS regularly updates its database of attack signatures and threat intelligence. This ensures that it can detect and prevent the latest known attacks and adapt to new attack techniques and patterns.
7	Customizable Policies	An IPS offers flexible policy management capabilities, allowing administrators to define custom rules and policies based on their organization's security requirements. These policies can be tailored to specific network segments, applications, or user groups, providing granular control over the protection mechanisms.

11.1.48.1 Assessment – SAMM

In the context of the Intrusion Prevention System (IPS) architecture component, here is how it helps address the SAMM security domains and their activities:

It's important to note that while an IPS contributes significantly to addressing the activities of the SAMM security domains, it is just one component of a comprehensive security architecture. Other security measures and components are also necessary to achieve a holistic security posture.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: An IPS helps enforce secure coding practices, vulnerability management, and incident response policies and standards by actively detecting and preventing unauthorized access and network attacks.
2	SAMM Security by Design	Security Requirements: An IPS assists in meeting security requirements by actively monitoring and blocking network traffic that violates the defined security requirements. Secure Architecture: An IPS, as part of a secure architecture, enhances the overall security posture of the system by detecting and preventing security threats and vulnerabilities. Secure Coding Practices: An IPS supports secure coding practices by identifying and blocking network attacks that exploit common coding vulnerabilities, such as injection attacks.
3	SAMM Security by Implementation	Security Testing Integration: An IPS integrates with security testing activities by providing a layer of defense against network attacks during the software development and deployment phases. Security Verification: An IPS contributes to security verification by actively monitoring network traffic and preventing security vulnerabilities from being exploited. Security Operations Integration: An IPS plays a crucial role in security operations by detecting and preventing network attacks, contributing to incident management and vulnerability management processes.
4	SAMM Security by Verification	Security Testing: An IPS actively participates in security testing by identifying and blocking potential vulnerabilities and weaknesses in software applications. Security Architecture Review: An IPS assists in the security architecture review process by demonstrating its capability to detect and prevent network attacks and by providing insights into the effectiveness of security controls. Secure Deployment Verification: An IPS contributes to secure deployment verification by actively monitoring network traffic during the deployment phase to ensure that security measures are properly implemented and enforced.
5	SAMM Security by Operations	Environment Hardening: An IPS helps in environment hardening by actively detecting and preventing unauthorized access and network attacks, thereby reducing known vulnerabilities in the software environment. Vulnerability Management: An IPS plays a crucial role in vulnerability management by actively detecting and preventing network attacks that exploit known vulnerabilities. Incident Management: An IPS supports incident management activities by detecting and preventing network attacks, contributing to incident detection, response, containment, and recovery processes. Secure Deployment: An IPS assists in secure deployment by actively monitoring network traffic and preventing unauthorized access and network attacks during the deployment phase.

11.1.48.2 Assessment – OWASP Top 10

The Intrusion Prevention System (IPS) capability helps address several OWASP threats by detecting and preventing unauthorized access, malicious activities, and network attacks. Here's how it contributes to each OWASP threat:

An Intrusion Prevention System (IPS) does not directly address all OWASP threats; however, it contributes significantly to mitigating several of them by providing real-time detection, prevention, and protection capabilities against network-based attacks targeting web applications. To achieve comprehensive security, organizations should combine an IPS with other security measures, such as secure design, secure coding practices, secure configurations, and regular vulnerability assessments.

#	Name	Description
1	Broken Access Control (A01-2021)	An IPS can monitor and analyze network traffic to identify suspicious behavior and potential attempts to bypass access controls. It can detect unauthorized access attempts and take action to prevent them, thereby enhancing the security of access control mechanisms.
2	Cryptographic Failures (A02-2021)	While an IPS does not directly address cryptographic failures, it can detect and prevent attacks that exploit vulnerabilities resulting from cryptographic failures. For example, if a cryptographic algorithm or mechanism is weak or improperly implemented, an attacker may try to exploit it to gain unauthorized access or manipulate data. An IPS can detect such attacks and prevent them from succeeding.
3	Injection (A03-2021)	An IPS can help mitigate injection attacks by monitoring network traffic and detecting suspicious patterns or known attack signatures associated with injection techniques. It can block or modify malicious traffic attempting to inject unintended commands or code into a web application, thereby preventing injection attacks.
4	Insecure Design (A04-2021)	While an IPS does not directly address insecure design vulnerabilities, it can provide an additional layer of protection by detecting and preventing attacks targeting insecurely designed components. By monitoring network traffic and identifying malicious activities that exploit insecure design decisions, an IPS helps enhance the overall security posture of the application.
5	Security Misconfiguration (A05-2021)	An IPS can identify and prevent attacks that target misconfigured components or insecure configurations. It can detect unauthorized access attempts or suspicious traffic patterns associated with known security misconfigurations and take action to block or mitigate such attacks, helping organizations maintain secure configurations.
6	Vulnerable and Outdated Components (A06-2021)	While an IPS does not directly address the use of vulnerable and outdated components, it can detect attacks that exploit known vulnerabilities present in these components. By monitoring network traffic for patterns associated with known attacks targeting specific vulnerable components, an IPS can block or modify traffic to prevent successful exploitation.
7	Identification and Authentication Failures (A07-2021)	An IPS can help prevent attacks that exploit identification and authentication failures by monitoring network traffic for suspicious authentication patterns or attempts to bypass authentication mechanisms. It can detect unauthorized access attempts and take action to prevent them, thereby strengthening the overall authentication mechanism.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	An IPS can help mitigate software and data integrity failures by monitoring network traffic for malicious activities that may indicate unauthorized modification or tampering of software components or data. It can detect and prevent attacks that attempt to compromise the integrity of the system, thereby enhancing the overall security and integrity of the application.
9	Security Logging and Monitoring Failures (A09-2021)	While an IPS can detect and prevent attacks, it does not directly address logging and monitoring failures. However, an IPS can generate security-related logs and contribute to the overall logging and monitoring capabilities of the system. By integrating with Security Information and Event Management (SIEM) systems, an IPS can provide valuable security event data for analysis and correlation.
10	Server-Side Request Forgery (A10-2021)	An IPS can help detect and prevent server-side request forgery attacks by monitoring network traffic for suspicious requests and patterns associated with SSRF. It can identify unauthorized requests made by an attacker and take action to block or modify the traffic, thereby mitigating the risk of SSRF vulnerabilities.

11.1.49 Threat Intelligence Integration

Integrate threat intelligence feeds and services into your security infrastructure to receive real-time information about emerging threats and indicators of compromise. It is a critical component of a comprehensive security architecture. It involves the integration of threat intelligence feeds and services into an organization's security infrastructure to gather real-time information about emerging threats and indicators of compromise. Threat intelligence integration enhance security monitoring and response capabilities, thereby reducing the risk of successful cyberattacks.

The Threat Intelligence Integration feature empowers organizations to leverage real-time threat intelligence to detect, respond to, and mitigate cyber threats effectively. By integrating threat intelligence into their security infrastructure, organizations can enhance their security capabilities and strengthen their defenses against evolving and sophisticated threats.

#	Name	Description
1	Threat Intelligence Feeds	Organizations can subscribe to various threat intelligence feeds provided by reputable sources such as cybersecurity vendors, government agencies, and industry-specific organizations. These feeds offer up-to-date information on the latest threat actors, attack techniques, vulnerabilities, and indicators of compromise. Integrating these feeds into the security infrastructure allows for continuous monitoring and assessment of the threat landscape.
2	Threat Intelligence Platforms	Implementing a threat intelligence platform enables organizations to aggregate, analyze, and manage the vast amount of threat intelligence data they receive from multiple feeds. These platforms use advanced algorithms and machine learning techniques to process and correlate the information, helping security teams identify patterns, trends, and potential risks. The platforms can also provide automated threat scoring and prioritization, facilitating more efficient incident response.

#	Name	Description
3	Security Information and Event Management (SIEM) Integration	Integrating threat intelligence with a SIEM solution enhances the visibility and correlation capabilities of security monitoring. By combining threat intelligence data with security event logs and network traffic analysis, security teams gain a more comprehensive view of potential threats. This integration enables the detection of suspicious activities, identification of known malicious IP addresses, domains, or files, and the ability to correlate them with internal security events for faster incident response.
4	Incident Response Automation	Threat intelligence integration can facilitate incident response automation by providing real-time indicators of compromise. When integrated with security orchestration, automation, and response (SOAR) platforms, organizations can create automated workflows that trigger immediate actions based on specific threat intelligence indicators. For example, if a threat intelligence feed identifies a known malicious IP address accessing the network, an automated response can be triggered to block the IP address and generate an alert to the security team. Benefits: a. Early Threat Detection: By integrating threat intelligence, organizations can proactively identify emerging threats and stay ahead of potential attacks. Real-time information about evolving attack techniques, vulnerabilities, and indicators of compromise allows security teams to detect and mitigate threats before they cause significant damage. b. Enhanced Incident Response: With threat intelligence integration, security teams gain valuable context and actionable insights when responding to security incidents. The correlation of internal security events with external threat intelligence provides a more accurate understanding of the incident's scope, impact, and potential adversaries involved, enabling faster and more effective incident response. c. Improved Decision-Making: Threat intelligence integration helps security professionals make informed decisions based on the latest threat landscape. By understanding the tactics, techniques, and procedures employed by threat actors, organizations can prioritize their security efforts and allocate resources more effectively to protect critical assets. d. Heightened Security Posture: Integrating threat intelligence feeds into the security infrastructure strengthens an organization's overall security posture. It enables proactive defense mechanisms, such as blocking known malicious IP addresses or domains, updating firewall rules, and deploying patches to address newly discovered vulnerabilities. e. Collaboration and Sharing: Threat intelligence integration encourages collaboration and sharing among organizations. By participating in information-sharing communities and platforms, organizations can contribute their own threat intelligence and benefit from the collective knowledge of the community. This collaboration enhances the accuracy and effectiveness of threat detection and response across industries.

11.1.49.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.
2	SAMM Security by Design	Threat Intelligence Integration contributes to the Security by Design domain by providing real-time information about emerging threats and indicators of compromise, which helps organizations incorporate security controls and considerations into the design phase of software development. It enhances the identification and mitigation of OWASP threats, such as Broken Access Control, Injection, Insecure Design, and others.
3	SAMM Security by Implementation	Threat Intelligence Integration helps organizations implement security measures during the development and deployment of software systems by providing real-time threat intelligence. It supports secure coding practices, secure architecture, threat modeling, and security testing, helping to address OWASP threats and vulnerabilities.
4	SAMM Security by Verification	Threat Intelligence Integration contributes to the Security by Verification domain by providing valuable information for security testing activities. It helps identify and address security vulnerabilities in software applications, including vulnerabilities associated with OWASP threats.
5	SAMM Security by Operations	Threat Intelligence Integration supports the secure operations of software systems by providing real-time information about emerging threats. It enhances environment hardening, vulnerability management, incident management, and secure deployment practices, helping to minimize the risk of security incidents and vulnerabilities throughout the software's lifecycle.

11.1.49.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	Threat Intelligence Integration can provide real-time information about emerging threats, including attack techniques and indicators of compromise. By incorporating this intelligence into the security infrastructure, organizations can enhance their security monitoring and response capabilities, improving the detection and prevention of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	While Threat Intelligence Integration does not directly address cryptographic failures, it can provide organizations with information about new vulnerabilities, attack vectors, and best practices related to cryptography. By staying updated with threat intelligence, organizations can learn about cryptographic weaknesses and take appropriate actions to mitigate vulnerabilities and strengthen their cryptographic implementations.

#	Name	Description
3	Injection (A03-2021)	Threat Intelligence Integration can contribute to addressing injection vulnerabilities by providing information on the latest attack techniques and trends. Organizations can use this intelligence to update their security measures, implement proper input validation and sanitization techniques, and stay aware of emerging injection attack vectors.
4	Insecure Design (A04-2021)	Threat Intelligence Integration indirectly helps address insecure design vulnerabilities by providing organizations with insights into the latest attack patterns and techniques used by threat actors. This information can be used to enhance the design and architecture of web applications, incorporating security best practices and avoiding common design flaws that could lead to security vulnerabilities.
5	Security Misconfiguration (A05-2021)	Threat Intelligence Integration can contribute to addressing security misconfiguration vulnerabilities by providing organizations with information about misconfigurations that are commonly targeted by attackers. By staying informed about the latest security misconfigurations and implementing appropriate configuration practices, organizations can reduce the risk of misconfigurations that could lead to security breaches or unauthorized access.
6	Vulnerable and Outdated Components (A06-2021)	Threat Intelligence Integration plays a significant role in addressing vulnerabilities associated with the use of vulnerable and outdated components. By subscribing to threat intelligence feeds, organizations can receive information about known vulnerabilities in software components and stay updated on the latest patches and security updates. This allows them to actively manage their software dependencies, update vulnerable components promptly, and reduce the risk of exploitation.
7	Identification and Authentication Failures (A07-2021)	Threat Intelligence Integration can help organizations address identification and authentication failures by providing information about emerging authentication bypass techniques, credential stuffing attacks, and compromised credentials. This knowledge can be used to enhance authentication mechanisms, enforce strong password policies, and implement additional security measures to prevent unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	Threat Intelligence Integration contributes to addressing software and data integrity failures by providing information on potential threats and attack vectors that could compromise the integrity of software components and data. By leveraging threat intelligence, organizations can implement security controls, secure coding practices, and regular vulnerability assessments to detect and prevent unauthorized modifications, data tampering, and other integrity-related issues.
9	Security Logging and Monitoring Failures (A09-2021)	Threat Intelligence Integration indirectly helps address security logging and monitoring failures by providing organizations with insights into the latest attack techniques, indicators of compromise, and security events. By integrating threat intelligence with their security logging and monitoring systems, organizations can enhance their capabilities to detect and respond to security incidents, improving their overall logging and monitoring practices.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	While Threat Intelligence Integration does not directly address server-side request forgery vulnerabilities, it can provide organizations with information about emerging SSRF attack techniques and potential SSRF targets. This knowledge can help organizations implement appropriate input validation and security controls to prevent SSRF attacks and protect their web applications from unauthorized access to internal or external resources.

11.2 DevOps

In addition to the Solution Architecture approach, which represents “Security By Design”, DevOps and DevSecOps represent approaching security “By Automation”. In this regard, the security model is complemented by capabilities implemented in automated processes that help ensure that the development process follows the required quality rigour agreed upon by the organisation's stakeholders. In particular, the DevOps and DevSecOps will be capabilities will complement the necessary standardisation of infrastructure definition, hardening, deployment processes, and validation tools applied to different levels of architecture and the artefacts created from scratch, such as custom code, customised, configured, and deployed.

Since not all the systems are the same, the proposed DevOps and DevSecOps capabilities will be tailored to the requirements. Therefore the hardening of the security elements will also be a factor that will be tuned accordingly.

Achieving deployment consistency is crucial to strengthening security aspects in IT lifecycle management. In this regard, DevOps practices ensure that your software deployments are predictable, reliable, and reproducible across different environments.

The key aspects of DevOps and how these capabilities help to enhance security are articulated below.

11.2.1 Infrastructure as Code (IaC)

Use IaC tools like Terraform or CloudFormation to define and manage your infrastructure in a declarative and version-controlled manner. IaC enables you to provision and configure infrastructure consistently across different environments.

By utilizing IaC practices, organizations can enhance IT security by ensuring consistent, auditable, and secure infrastructure deployments. The ability to define infrastructure as code facilitates the implementation of security best practices, reduces human errors, and enables agile and secure infrastructure management throughout the DevOps lifecycle.

General security aspects

#	Name	Description
1	Consistent Infrastructure Configuration	IaC allows organizations to define and configure their infrastructure in a consistent and repeatable manner. This ensures that security controls, settings, and configurations are consistently applied across different environments, reducing the risk of misconfigurations and inconsistencies that could lead to security vulnerabilities.
2	Version Control and Auditing	IaC tools enable version control of infrastructure code, allowing organizations to track and manage changes over time. This facilitates auditing and provides a history of infrastructure modifications, making it easier to identify and address security issues. It also enables rollbacks to previous known-good configurations if security incidents occur.
3	Reduced Human Errors	By automating infrastructure provisioning and configuration through IaC, organizations can reduce the reliance on manual processes prone to human errors. Automated deployments minimize the risk of misconfigurations, eliminate inconsistencies, and help maintain a secure baseline for infrastructure components.
4	Faster Response to Security Requirements	IaC enables agile and iterative infrastructure management, allowing organizations to respond quickly to security requirements and adapt their infrastructure configurations accordingly. Changes can be made to the infrastructure code, reviewed, and deployed in a controlled and efficient manner, reducing the time to implement security improvements.
5	Security Best Practices Implementation	IaC provides a structured approach for implementing security best practices throughout the infrastructure provisioning and management process. Security controls, such as network segmentation, access controls, encryption, and monitoring, can be defined as part of the infrastructure code, ensuring their consistent application across environments.
6	Collaboration and Knowledge Sharing	IaC promotes collaboration between development, operations, and security teams by providing a common language and shared infrastructure codebase. This facilitates knowledge sharing, ensures security considerations are integrated into the development process, and enables cross-functional collaboration to address security requirements effectively.
7	Infrastructure Testing and Validation	IaC tools often provide capabilities for automated testing and validation of infrastructure code. This allows organizations to perform security assessments, vulnerability scans, and configuration checks before deploying the infrastructure. By identifying and addressing security issues early in the development process, organizations can reduce the risk of deploying insecure infrastructure.
8	Disaster Recovery and Resilience	IaC can support disaster recovery and resilience efforts by defining infrastructure configurations that can be easily replicated and restored in case of failures or security incidents. Infrastructure-as-code practices enable organizations to quickly recover from security breaches or outages by re-provisioning the infrastructure based on known good configurations.

11.2.1.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: IaC enables consistent and auditable infrastructure deployments, ensuring that security controls and configurations defined in policies and standards are consistently applied across environments.
2	SAMM Security by Design	Security Requirements: IaC allows organizations to define security requirements as part of the infrastructure code, ensuring that the infrastructure is designed and provisioned to meet specific security needs. Secure Architecture: IaC facilitates the adoption of secure architectural patterns and principles, enabling the design of a robust and secure infrastructure that minimizes potential vulnerabilities and supports the implementation of security controls.
3	SAMM Security by Implementation	Secure Development Training: IaC practices promote secure coding and infrastructure provisioning techniques, which align with secure development training and enable developers to build secure software systems. Security Requirements: IaC allows organizations to define and document security requirements as part of the infrastructure provisioning process, ensuring that security objectives are considered during software development and deployment.
4	SAMM Security by Verification	Security Testing: IaC tools can automate the provisioning of test environments and facilitate the integration of security testing activities into the development process, helping to identify and address vulnerabilities in software applications. Secure Deployment Verification: IaC enables the secure deployment and configuration of software systems, ensuring that security controls, encryption mechanisms, access controls, and other security measures are properly implemented during deployment.
5	SAMM Security by Operations	Environment Hardening: IaC allows organizations to define and enforce secure configurations for the software environment, including infrastructure components, patches, updates, and network configurations, contributing to the hardening of the operational environment. Secure Deployment: IaC practices facilitate secure software deployment by providing mechanisms for secure software distribution, installation procedures, and configuration management, ensuring that software is deployed in a controlled and secure manner. Security Testing: IaC enables the integration of security testing throughout the software development life cycle, supporting activities such as static code analysis, dynamic application security testing (DAST), and security code reviews. Operational Enablement: IaC practices, including documentation of operational procedures and incident response capabilities, contribute to the operational enablement of software systems, ensuring ongoing security and reliability.

11.2.1.2 Assessment – OWASP Top 10

The capability of Infrastructure as Code (IaC) indirectly addresses several OWASP threats by providing a structured and controlled approach to provisioning and managing infrastructure.

While Infrastructure as Code (IaC) does not directly address all OWASP threats, it contributes to the mitigation of several vulnerabilities by enabling secure configurations, promoting secure coding practices, and facilitating consistent and auditable infrastructure deployments, it contributes to the mitigation of specific vulnerabilities as outlined below:

#	Name	Description
1	Broken Access Control (A01-2021)	IaC tools allow for consistent and auditable infrastructure deployments. By defining access controls as part of the infrastructure configuration, organizations can enforce proper access controls consistently across different environments, reducing the risk of broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	IaC enables organizations to implement secure configurations for cryptographic components by defining them as part of the infrastructure code. This helps ensure that cryptographic algorithms, keys, and configurations are correctly implemented, reducing the likelihood of cryptographic failures.
3	Injection (A03-2021)	IaC encourages secure coding practices by allowing organizations to define and enforce input validation and parameterization as part of the infrastructure code. By following secure coding practices, organizations can significantly reduce the risk of injection vulnerabilities in their applications.
4	Insecure Design (A04-2021)	IaC promotes the adoption of secure design principles by making security considerations an integral part of the infrastructure code. By implementing secure design patterns and controls within the infrastructure, organizations can mitigate the risk of insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	IaC tools facilitate the implementation of secure configurations for infrastructure components. By defining and enforcing secure settings and removing unnecessary functionality, organizations can reduce the risk of security misconfigurations in their applications and associated components.
6	Vulnerable and Outdated Components (A06-2021)	IaC enables organizations to manage their software dependencies effectively. By leveraging IaC tools, organizations can automate the process of updating and patching components, reducing the risk of using outdated or vulnerable software components.
7	Identification and Authentication Failures (A07-2021)	IaC allows organizations to define and enforce secure authentication mechanisms as part of the infrastructure code. By implementing strong identification and authentication controls, organizations can reduce the risk of authentication failures and unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	IaC promotes the use of secure coding techniques and practices, which can help ensure the integrity of software components, configurations, and data within web applications. By following secure practices, organizations can mitigate the risk of software and data integrity failures.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	IaC tools can facilitate the implementation of comprehensive logging and monitoring capabilities as part of the infrastructure code. By defining logging and monitoring requirements, organizations can enhance their ability to detect and respond to security incidents, reducing the likelihood of security logging and monitoring failures.
10	Server-Side Request Forgery (A10-2021)	While IaC does not directly address SSRF vulnerabilities, organizations can leverage IaC practices to enforce secure configurations and input validation, reducing the risk of SSRF attacks. By implementing robust input validation and secure configurations in the infrastructure code, organizations can mitigate the risk of SSRF vulnerabilities.

11.2.2 Configuration Management

Utilize configuration management tools such as Ansible, Puppet, or Chef to automate the deployment and configuration of software components. These tools help ensure that the configuration settings remain consistent and are applied uniformly across various environments.

Configuration Management, as an aspect of DevOps, enhances IT security.

By utilizing configuration management tools, organizations can enhance IT security by achieving consistent, auditable, and standardized configurations across their infrastructure. The ability to automate configuration changes, enforce security policies, and roll back changes when necessary improves security posture, reduces vulnerabilities, and enhances overall system resilience.

#	Name	Description
1	Consistency and Standardization	Configuration management tools like Ansible, Puppet, or Chef enable the automation of software deployment and configuration. By defining and managing configuration settings as code, these tools ensure consistency and standardization across different environments. This reduces the risk of misconfigurations and minimizes security vulnerabilities that may arise from inconsistent or incorrect configuration settings.
2	Version Control and Auditing	Configuration management tools integrate with version control systems, allowing configuration code to be stored, tracked, and audited. This facilitates traceability and accountability, enabling organizations to review and monitor changes made to configurations. Version control also helps in identifying unauthorized or unexpected modifications, improving the overall security posture.
3	Rapid and Reliable Configuration Changes	Configuration management tools enable quick and reliable changes to software configurations. Security patches, updates, and required configuration changes can be applied automatically and consistently across multiple environments. This reduces the time and effort required to implement security-related changes, ensuring that systems are properly configured and protected against known vulnerabilities.

#	Name	Description
4	Policy Enforcement	Configuration management tools can enforce security policies and best practices by defining and enforcing configuration standards. Security-related configurations, such as access controls, encryption settings, or system hardening measures, can be codified and applied uniformly across the infrastructure. This helps organizations ensure that security controls are consistently implemented and adhered to, reducing the risk of non-compliance or security gaps.
5	Change Management and Rollbacks	Configuration management tools provide mechanisms for change management and rollbacks. Changes to configurations can be tracked, reviewed, and tested in a controlled manner before being applied to production environments. In the event of an issue or security breach, rollbacks can be performed quickly and reliably, reverting configurations to a known good state, thereby minimizing the impact on security.
6	Automation and Efficiency	Configuration management automates the deployment and configuration of software components, reducing manual intervention and human errors. This increases operational efficiency and reduces the likelihood of misconfigurations that could lead to security vulnerabilities. Automation also enables rapid scaling and provisioning of resources while ensuring that security configurations are consistently applied.

11.2.2.1 Assessment – SAMM

The use of configuration management tools contributes to several SAMM security domains and their activities, including enhancing consistency and standardization, enforcing policies and standards, enabling secure architecture, supporting security testing, and facilitating configuration management in software operations.

#	Name	Description
1	SAMM Security by Governance	Utilizing configuration management tools such as Ansible, Puppet, or Chef can contribute to the "Policies and Standards" activity by automating the enforcement of configuration standards and ensuring consistent and standardized configurations across the infrastructure.
2	SAMM Security by Design	Configuration management tools help address the "Security Requirements" activity by enabling the automation and enforcement of security-related configuration settings, ensuring that software applications are designed with the necessary security requirements in mind.
3	SAMM Security by Implementation	Configuration management tools contribute to the "Secure Architecture" activity by enabling the automated deployment and configuration of software components, ensuring that secure architectural patterns and principles are consistently applied throughout the development process.

#	Name	Description
4	SAMM Security by Verification	Configuration management tools can assist in the "Security Testing" activity by automating the deployment and configuration of test environments, allowing for easier setup and execution of security testing activities such as penetration testing, vulnerability scanning, and code review.
5	SAMM Security by Operations	Configuration management tools can support the "Configuration Management" activity by automating the management of software configurations, including version control, change management, and documentation. This helps organizations maintain the integrity of software systems and detect unauthorized changes or misconfigurations.

11.2.2.2 Assessment – OWASP Top 10

Configuration Management capability does not directly address all the OWASP threats; however, it contributes to enhancing the security posture of web applications by ensuring consistent and standardized configurations, enforcing security policies, and automating configuration changes.

#	Name	Description
1	Broken Access Control (A01-2021)	Configuration management tools like Ansible, Puppet, or Chef enable the automation of software deployment and configuration. By defining and managing configuration settings as code, these tools ensure consistency and standardization across different environments, reducing the risk of misconfigurations and minimizing security vulnerabilities (contribution: Consistency and Standardization).
2	Cryptographic Failures (A02-2021)	Configuration management tools integrate with version control systems, allowing configuration code to be stored, tracked, and audited. This facilitates traceability and accountability, enabling organizations to review and monitor changes made to configurations, improving overall security (contribution: Version Control and Auditing).
3	Injection (A03-2021)	Configuration management tools enable quick and reliable changes to software configurations, including applying security patches, updates, and required configuration changes automatically and consistently across multiple environments. This helps ensure that systems are properly configured and protected against known vulnerabilities (contribution: Rapid and Reliable Configuration Changes).
4	Insecure Design (A04-2021)	Configuration management tools can enforce security policies and best practices by defining and enforcing configuration standards. Security-related configurations, such as access controls, encryption settings, or system hardening measures, can be codified and applied uniformly across the infrastructure, reducing the risk of insecure design decisions (contribution: Policy Enforcement).

#	Name	Description
5	Security Misconfiguration (A05-2021)	Configuration management tools, when used to adhere to secure configuration practices and regularly update software components, can help organizations reduce the risk of security misconfigurations. This contributes to enhancing the overall security posture of web applications (contribution: General security aspects).
6	Vulnerable and Outdated Components (A06-2021)	Configuration management tools can actively manage software dependencies and ensure that up-to-date and secure versions of components are used. By following secure coding practices and keeping software dependencies updated, organizations can reduce the risk of using vulnerable or outdated components (contribution: General security aspects).
7	Identification and Authentication Failures (A07-2021)	While configuration management tools do not directly address authentication vulnerabilities, they contribute to security by ensuring consistent and standardized configurations across different environments. This indirectly helps in maintaining secure identification and authentication mechanisms (contribution: Consistency and Standardization).
8	Software and Data Integrity Failures (A08-2021)	Configuration management tools help organizations enforce secure coding practices and ensure consistent configurations, reducing the risk of unauthorized modifications and potential integrity-related issues (contribution: General security aspects).
9	Security Logging and Monitoring Failures (A09-2021)	Although configuration management tools do not directly address logging and monitoring vulnerabilities, they contribute to security by facilitating consistent and standardized configurations, which can include logging and monitoring settings. Proper configuration management ensures that logging and monitoring mechanisms are effectively implemented (contribution: General security aspects).
10	Server-Side Request Forgery (A10-2021)	Configuration management tools can enforce secure configurations, including input validation and secure coding practices, reducing the risk of vulnerabilities that could lead to Server-Side Request Forgery (SSRF) attacks (contribution: General security aspects).

11.2.3 Version Control

Keep your application code, configuration files, scripts, and infrastructure definitions under version control. This allows you to track changes, roll back if needed, and maintain a single source of truth for your deployments.

Utilizing version control systems, organizations can enhance IT security by promoting a structured and controlled approach to code, configuration, and infrastructure management. The ability to track changes, roll back when needed, facilitate collaboration, enforce access control, and document modifications improves the security and reliability of deployments. Additionally, version control serves as a foundation for other DevOps practices, such as continuous integration and continuous deployment, further enhancing the overall security of the software development lifecycle.

#	Name	Description
1	Change Tracking and Auditing	Version control systems, such as Git, provide a centralized repository for storing application code, configuration files, scripts, and infrastructure definitions. By keeping these artifacts under version control, organizations can track changes made to them over time. This enables effective change management, enhances accountability, and supports auditing requirements. In the event of a security incident, the ability to review the history of changes can aid in identifying the source and impact of potential vulnerabilities.
2	Reproducibility and Rollbacks	Version control allows for the recreation of previous states of code, configurations, and infrastructure. This capability is crucial in maintaining a reliable and secure environment. In case of an issue or security breach, having a single source of truth for deployments enables organizations to roll back to a known good state quickly. This helps mitigate the impact of security incidents, minimize downtime, and restore systems to a secure state.
3	Collaboration and Code Review	Version control facilitates collaboration among team members by providing a centralized platform for sharing and reviewing code and configurations. This allows for peer code review, which helps identify security vulnerabilities and best practices. By leveraging the collective knowledge and expertise of the team, potential security risks can be detected early in the development process, leading to more secure deployments.
4	Access Control and Permissions	Version control systems offer robust access control mechanisms, allowing organizations to define and manage user permissions. Access can be granted based on roles or individuals, ensuring that only authorized personnel can modify code, configurations, or infrastructure definitions. Granular access control helps prevent unauthorized changes and reduces the risk of introducing security vulnerabilities due to improper modifications.
5	Branching and Release Management	Version control systems support branching, which enables the isolation of different development efforts. This is particularly important for security-related changes or hotfixes that need to be applied separately from regular development activities. By maintaining separate branches, organizations can manage the release of security patches or updates without impacting ongoing development work. This ensures that security fixes can be deployed promptly and independently, improving the overall security posture.
6	Change Documentation and Communication	Version control systems provide mechanisms for documenting and communicating changes. Commit messages, pull requests, and code comments allow developers and operations teams to document the purpose and impact of changes. This documentation is valuable for understanding the context of modifications, aiding troubleshooting, and providing a historical record of changes. Clear communication within the team supports knowledge sharing and helps maintain a secure and well-documented environment.

11.2.3.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not apply directly.

#	Name	Description
2	SAMM Security by Design	Security Requirements: Version control helps in tracking changes made to security requirements over time, ensuring that they are incorporated into the design phase. Secure Architecture: Version control allows for the management and tracking of changes to the software's architecture, enabling the adoption of secure architectural patterns and principles. Secure Coding Practices: Version control facilitates collaboration and code review, which helps enforce secure coding practices during the design phase. Threat Modeling: Version control enables the documentation and tracking of changes made during threat modeling exercises, providing a historical record of security considerations. Security Testing: Version control supports the integration of security testing activities, such as code review and static code analysis, into the design phase.
3	SAMM Security by Implementation	Secure Development Training: Version control allows for the tracking of changes made during training programs, ensuring that developers have access to up-to-date secure coding practices and guidelines. Secure Architecture: Version control supports the adoption and implementation of secure architecture practices, such as threat modeling and secure design patterns. Secure Coding Guidelines: Version control helps enforce the use of secure coding guidelines by providing a centralized platform for sharing and reviewing code. Security Requirements: Version control facilitates the tracking of changes to security requirements and their implementation, ensuring that they are properly documented and integrated into the software development process. Security Testing Integration: Version control enables the integration of security testing activities, such as code analysis and security code reviews, throughout the development life cycle. Security Verification: Version control supports the documentation and tracking of security verification activities, such as security code reviews and vulnerability assessments. Security Architecture Review: Version control allows for the tracking of changes made during security architecture reviews, ensuring that recommendations for improvement are documented and addressed. Security Operations Integration: Version control facilitates the tracking of changes made during the integration of security considerations into operational processes, such as incident management and secure deployment practices.
4	SAMM Security by Verification	Security Testing: Version control enables the tracking of changes made during security testing activities, such as penetration testing and vulnerability scanning. Code Review: Version control provides a centralized repository for source code, allowing for code reviews to identify security flaws and adherence to secure coding practices. Security Architecture Review: Version control supports the documentation and tracking of security architecture reviews, ensuring that proper security controls are in place.

#	Name	Description
		Security Requirements Verification: Version control helps track changes made during the verification of security requirements, ensuring compliance with standards and best practices. Threat Modeling: Version control allows for the documentation and tracking of changes made during threat modeling exercises, facilitating the identification and prioritization of security threats. Secure Deployment Verification: Version control enables the tracking of changes made during the verification of secure deployment practices, ensuring that security measures are properly implemented.
5	SAMM Security by Operations	Environment Hardening: Version control helps track changes made to environment configurations, ensuring that secure configurations and patches are applied. Secure Build: Version control supports the tracking of changes made during the build process, ensuring adherence to secure coding standards and practices. Configuration Management: Version control provides a centralized repository for configuration files, allowing for proper management and documentation of software configurations. Vulnerability Management: Version control facilitates the tracking of changes made during vulnerability management activities, such as vulnerability scanning and assessment. Incident Management: Version control allows for the tracking of changes made during incident response and recovery activities, ensuring proper documentation and resolution of security incidents. Secure Deployment: Version control enables the tracking of changes made during the deployment of software systems, ensuring secure

11.2.3.2 Assessment – OWASP Top 10

Version control systems do not directly address all OWASP threats; however, they provide valuable capabilities that support secure development practices, change tracking, collaboration, and documentation. These practices contribute to enhancing the overall security posture of applications by reducing the risk of vulnerabilities and providing a structured approach to code, configuration, and infrastructure management.

#	Name	Description
1	Broken Access Control (A01-2021)	Change Tracking and Auditing: Version control systems allow organizations to track changes made to application code, configurations, and infrastructure definitions. This helps in identifying potential vulnerabilities and understanding the source and impact of unauthorized access.
2	Cryptographic Failures (A02-2021)	Version control systems do not directly address cryptographic failures. However, they can indirectly contribute to overall security by ensuring that cryptographic-related code changes are properly tracked, reviewed, and audited.

#	Name	Description
3	Injection (A03-2021)	Version control systems do not directly address injection vulnerabilities. However, they can contribute to secure development practices by facilitating collaboration and code review, which can help identify and mitigate injection vulnerabilities at an early stage.
4	Insecure Design (A04-2021)	Version control systems do not directly address insecure design vulnerabilities. However, they can support secure development practices by providing a platform for documenting and communicating design decisions, facilitating collaboration, and enabling code review.
5	Security Misconfiguration (A05-2021)	Version control systems do not directly address security misconfigurations. However, they can indirectly contribute to mitigating security misconfiguration vulnerabilities by enforcing access control and permissions, documenting changes, and supporting collaboration among team members.
6	Vulnerable and Outdated Components (A06-2021)	Version control systems do not directly address vulnerabilities in third-party components. However, they can indirectly contribute to managing software dependencies by documenting and tracking the versions of libraries, frameworks, and other components used in the application.
7	Identification and Authentication Failures (A07-2021)	Version control systems do not directly address identification and authentication failures. However, they can indirectly support secure authentication mechanisms by enforcing access control, documenting changes to authentication-related code, and facilitating code review.
8	Software and Data Integrity Failures (A08-2021)	Version control systems do not directly address software and data integrity failures. However, they can indirectly contribute to maintaining data integrity by providing the ability to track changes, roll back if needed, and document modifications to application code and configurations.
9	Security Logging and Monitoring Failures (A09-2021)	Version control systems do not directly address security logging and monitoring failures. However, they can indirectly contribute to security logging by allowing developers to document the purpose and impact of changes through commit messages, pull requests, and code comments.
10	Server-Side Request Forgery (A10-2021)	Version control systems do not directly address server-side request forgery vulnerabilities. However, they can indirectly contribute to secure development practices by facilitating code review and collaboration, which can help identify and mitigate vulnerabilities related to request handling.

11.2.4 Continuous Integration and Delivery (CI/CD)

Implement CI/CD pipelines to automate the build, testing, and deployment processes. CI/CD pipelines provide a consistent and repeatable mechanism for packaging and deploying applications across different environments.

Implementing CI/CD pipelines, organizations can enhance IT security by automating the build, testing, and deployment processes, ensuring consistency, early detection of security issues, faster remediation, and the incorporation of automated security controls. The use of immutable infrastructure and the ability to roll back deployments quickly also contribute to a more secure and resilient environment. Overall, CI/CD practices promote a security-focused mindset and help integrate security measures seamlessly into the software development lifecycle.

#	Name	Description
1	Consistency and Repeatability	CI/CD pipelines automate the build, testing, and deployment processes, ensuring consistency and repeatability across different environments. By automating these steps, the risk of human error and configuration drift is reduced, leading to more secure deployments. Consistent and repeatable deployments make it easier to enforce security policies, apply security patches, and maintain a standardized and secure infrastructure.
2	Early Detection of Security Issues	CI/CD pipelines typically include automated testing, including security testing, as part of the build and deployment process. This allows for the early detection of security vulnerabilities, such as code flaws, configuration issues, or known security weaknesses. By catching these issues early in the development cycle, organizations can address them promptly, reducing the overall exposure to security risks.
3	Faster Time-to-Remediation	CI/CD pipelines enable rapid feedback loops, allowing developers and operations teams to identify and resolve security issues more quickly. Automated testing and continuous monitoring help identify vulnerabilities and weaknesses, enabling prompt remediation. By shortening the time between identifying a security issue and resolving it, organizations can minimize the window of opportunity for potential attacks and reduce the potential impact on the system.
4	Automated Security Controls	CI/CD pipelines can incorporate security controls, such as vulnerability scanning, static code analysis, or security code reviews, as automated steps in the deployment process. This ensures that security checks are consistently applied, reducing the risk of overlooking critical security measures. Automated security controls help enforce security best practices and ensure that applications and infrastructure are deployed with the necessary security safeguards in place.
5	Immutable Infrastructure	CI/CD pipelines often utilize infrastructure as code principles, where infrastructure configurations are treated as code and stored in version control systems. This enables the creation of immutable infrastructure, where infrastructure changes are deployed by creating new instances rather than modifying existing ones. Immutable infrastructure reduces the risk of unauthorized changes and provides better control over the deployment process, enhancing security by minimizing configuration drift and making rollbacks easier.

#	Name	Description
6	Rollback and Recovery	CI/CD pipelines, coupled with version control, provide the ability to roll back deployments quickly in case of a security incident or unexpected issues. The automated nature of CI/CD pipelines allows for swift recovery by deploying known good versions or triggering additional security measures, such as rollbacks to previous states or activating incident response plans. The ability to roll back deployments efficiently contributes to minimizing the impact of security breaches and ensuring system availability.

11.2.4.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address it directly.
2	SAMM Security by Design	Security Requirements: Implementing CI/CD pipelines ensures that security requirements are consistently incorporated into the design phase of software development, mitigating OWASP threats through early identification and proactive measures. Secure Architecture: CI/CD promotes the adoption of secure architectural patterns and principles, which helps address OWASP threats by minimizing vulnerabilities and establishing a strong foundation for security controls. Secure Coding Practices: CI/CD encourages the use of secure coding practices, reducing the likelihood of OWASP threats by preventing common coding mistakes and vulnerabilities. Threat Modeling: CI/CD supports the integration of threat modeling exercises during the design phase, aiding in the identification and mitigation of potential security risks associated with OWASP threats. Security Testing: CI/CD incorporates security testing activities, such as static code analysis and security code reviews, during the design phase, ensuring that vulnerabilities related to OWASP threats are detected and addressed.
3	SAMM Security by Implementation	Secure Development Training: CI/CD facilitates the integration of secure development training programs, raising awareness and providing knowledge to developers on secure coding practices, mitigating vulnerabilities during implementation. Secure Architecture: CI/CD promotes the use of secure architecture practices, such as threat modeling and secure design patterns, reducing the risk of OWASP threats by addressing potential vulnerabilities early in the implementation process. Secure Coding Guidelines: CI/CD supports the establishment and adoption of secure coding guidelines, enabling developers to follow best practices and avoid common coding mistakes that could lead to OWASP threats. Security Requirements: CI/CD helps ensure that security requirements are implemented during the development process, guiding developers to build software systems that address OWASP threats effectively.

#	Name	Description
		Security Testing Integration: CI/CD integrates security testing activities throughout the development process, allowing for the detection and remediation of vulnerabilities associated with OWASP threats. Security Verification: CI/CD incorporates security verification techniques to validate the effectiveness of security controls implemented during implementation, reducing the impact of OWASP threats. Security Architecture Review: CI/CD enables the review of the security architecture during implementation, helping identify potential vulnerabilities or weaknesses related to OWASP threats. Security Operations Integration: CI/CD supports the integration of security considerations into operational processes, ensuring that security measures are implemented and maintained throughout the software's lifecycle.
4	SAMM Security by Verification	Security Testing: CI/CD facilitates various types of security testing, including penetration testing, vulnerability scanning, and code review, to identify and address vulnerabilities associated with OWASP threats. Code Review: CI/CD supports code review activities to identify security flaws and adherence to secure coding practices, helping mitigate vulnerabilities related to OWASP threats. Security Architecture Review: CI/CD enables the assessment of the security architecture to ensure proper security controls and mechanisms are in place, addressing vulnerabilities associated with OWASP threats. Security Requirements Verification: CI/CD helps verify the implementation of security requirements, ensuring alignment with industry standards and best practices, mitigating OWASP threats. Threat Modeling: CI/CD supports the incorporation of threat modeling exercises to identify and prioritize security threats, addressing vulnerabilities associated with OWASP threats. Secure Deployment Verification: CI/CD assists in verifying secure deployment practices, ensuring that security controls and measures are properly implemented during deployment and minimizing vulnerabilities related to OWASP threats.
5	SAMM Security by Operations	Environment Hardening: CI/CD pipelines enable organizations to implement secure configurations and apply patches and updates consistently across different environments, contributing to environment hardening. Secure Build: CI/CD pipelines automate the build process, ensuring adherence to secure coding standards, secure build tools, and secure software development frameworks, thereby supporting secure software builds. Configuration Management: CI/CD pipelines, coupled with version control systems, provide effective configuration management by maintaining the integrity of software configurations and facilitating change management. Vulnerability Management: CI/CD pipelines can incorporate vulnerability scanning and assessment as automated steps, enabling organizations to identify and prioritize vulnerabilities early in the development and deployment process.

#	Name	Description
		Incident Management: CI/CD pipelines, along with version control and rollback capabilities, support incident management by facilitating swift recovery through rollbacks to known good versions or activating incident response plans. Secure Deployment: CI/CD pipelines automate the deployment process, ensuring secure software distribution, installation procedures, and configuration of software components, contributing to secure deployments. Security Testing: CI/CD pipelines include automated testing as part of the build and deployment process, enabling the integration of security testing activities such as static code analysis, dynamic application security testing (DAST), and security code reviews. Operational Enablement: CI/CD pipelines provide operational support by streamlining and automating software operations, reducing the likelihood of human error and promoting operational reliability and security.

11.2.4.2 Assessment – OWASP Top 10

The CI/CD capability addresses multiple OWASP threats by automating security practices, enforcing secure coding standards, promoting consistent configurations, facilitating vulnerability scanning, and integrating security controls into the deployment process. These practices contribute to enhancing the overall security posture of web applications.

#	Name	Description
1	Broken Access Control (A01-2021)	CI/CD promotes consistent and repeatable deployments, ensuring that access controls are properly implemented and enforced across different environments. It reduces the risk of misconfigurations and unauthorized access to resources, enhancing the overall security posture
2	Cryptographic Failures (A02-2021)	CI/CD facilitates the implementation of secure coding practices and the integration of cryptographic mechanisms into the deployment process. It enables automated security controls, such as vulnerability scanning and static code analysis, which can identify cryptographic weaknesses and ensure the correct usage of encryption algorithms.
3	Injection (A03-2021)	CI/CD supports secure coding practices, including input validation and sanitization, as automated steps in the deployment process. By catching injection vulnerabilities early in the development cycle, CI/CD helps prevent attackers from injecting malicious code into web applications.
4	Insecure Design (A04-2021)	CI/CD promotes the adoption of secure design principles and security considerations from the early stages of development. By enforcing secure coding standards and conducting regular security assessments, CI/CD helps identify and address insecure design decisions, reducing the likelihood of vulnerabilities being introduced.

#	Name	Description
5	Security Misconfiguration (A05-2021)	CI/CD pipelines enforce consistent configurations and reduce the risk of insecure settings or unnecessary functionality being enabled. By automatically deploying standardized and securely configured infrastructure, CI/CD helps organizations avoid security misconfigurations and maintain a secure application environment.
6	Vulnerable and Outdated Components (A06-2021)	CI/CD enables organizations to actively manage software dependencies and stay up-to-date with security patches. By integrating vulnerability scanning and dependency management into the pipeline, CI/CD helps identify and remediate vulnerable or outdated components, minimizing the risk of exploitation.
7	Identification and Authentication Failures (A07-2021)	CI/CD supports the implementation of secure authentication mechanisms and strong password policies as part of the deployment process. It enables automated testing for authentication vulnerabilities, ensuring that identification and authentication mechanisms are properly implemented and reducing the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	CI/CD promotes secure coding techniques and the regular testing of web applications for vulnerabilities. By ensuring the integrity of software components, configurations, and data, CI/CD helps prevent unauthorized modifications and data tampering, enhancing the overall security of web applications.
9	Security Logging and Monitoring Failures (A09-2021)	CI/CD pipelines can include automated logging and monitoring mechanisms as part of the deployment process. By enforcing comprehensive logging, log analysis, and real-time monitoring, CI/CD helps organizations detect and respond to security incidents more effectively, reducing the impact of potential breaches.
10	Server-Side Request Forgery (A10-2021)	CI/CD supports secure configurations and input validation, which can prevent server-side request forgery vulnerabilities. By validating and sanitizing user input as an automated step in the deployment process, CI/CD helps protect web applications from unauthorized access and potential exploitation.

11.2.5 Immutable Infrastructure

Immutable infrastructure approach allows treating infrastructure components as disposable and are replaced entirely for every deployment. Immutable infrastructure reduces configuration drift and ensures consistent deployments by creating fresh instances with predefined configurations.

Adopting an immutable infrastructure approach, organizations can enhance IT security by ensuring consistent and reproducible deployments, reducing the attack surface, simplifying change management, enabling easy rollback and disaster recovery, facilitating version control and auditing, and maintaining consistent environments across different stages of the software development lifecycle. These benefits contribute to a more secure and resilient infrastructure, reducing the risk of vulnerabilities and providing a solid foundation for secure application deployments.

#	Name	Description
1	Consistent and Reproducible Deployments	By treating infrastructure components as disposable and replacing them entirely for every deployment, immutable infrastructure ensures consistent and reproducible deployments. Each deployment starts from a known, predefined state, eliminating configuration drift and reducing the risk of inconsistent or misconfigured infrastructure. This consistency enhances IT security by reducing the likelihood of vulnerabilities caused by misconfigurations or unintended changes.
2	Reduced Attack Surface	Immutable infrastructure minimizes the attack surface by creating fresh instances for each deployment. These instances are typically built from a secure base image and include only the necessary components and configurations. Any vulnerabilities or weaknesses present in previous instances are eliminated, reducing the risk of attacks that target outdated or compromised components. By reducing the attack surface, organizations can enhance the overall security posture of their infrastructure.
3	Simplified Change Management	With immutable infrastructure, changes are managed by replacing instances rather than modifying existing ones. This simplifies change management by reducing the complexity and potential risks associated with in-place updates or configuration changes. Each change is implemented by deploying a new, fully configured instance, which can be tested and validated before routing traffic to it. This approach allows for easier rollback and faster recovery in case of issues or security incidents.
4	Easy Rollback and Disaster Recovery	: Immutable infrastructure makes it easier to roll back deployments or recover from disasters. Since each deployment creates new instances, rolling back to a previous version is as simple as redirecting traffic to the previous instances. This provides a quick and efficient way to address security incidents or unexpected issues. Additionally, in the event of a disaster, spinning up new instances from predefined configurations becomes a straightforward process, contributing to faster recovery and minimizing downtime.
5	Version Control and Auditing: Immutable	infrastructure is often managed through infrastructure-as-code tools and stored in version control systems. This allows for versioning, auditing, and tracking changes to the infrastructure configurations over time. Version control enables organizations to review and validate infrastructure changes, track who made the changes, and easily revert to previous versions if necessary. These capabilities enhance IT security by promoting transparency, accountability, and the ability to trace and investigate any potential security-related changes.
6	Consistent Environments across Development, Testing, and Production	Immutable infrastructure ensures that environments used for development, testing, and production are consistent. By using the same predefined configurations for all environments, organizations can eliminate discrepancies or variations that can introduce security vulnerabilities or impact the behavior of applications. This consistency helps identify and address security issues early in the development process, promoting a more secure and reliable system.

11.2.5.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address this domain directly.
2	SAMM Security by Design	Immutable Infrastructure contributes to the Security by Design domain by enabling consistent security configurations (Consistent Security Configurations). It ensures that every new instance is automatically provisioned with the latest security measures and patches, reducing the risk of security vulnerabilities caused by misconfigurations or outdated software. Immutable Infrastructure also facilitates horizontal scaling, which improves the scalability and availability of the infrastructure (Improved Scalability and Availability). Additionally, in the event of a security breach, affected instances can be swiftly replaced with new ones that have known and trusted configurations, enhancing disaster recovery capabilities (Enhanced Disaster Recovery). Immutable Infrastructure also simplifies rollbacks and testing, allowing for easy replacement of instances and safer testing of new configurations or updates (Simplified Rollbacks and Testing). Lastly, by aligning with Infrastructure as Code (IaC) practices, Immutable Infrastructure ensures the infrastructure remains consistent and reproducible, contributing to security governance (Infrastructure as Code (IaC) Alignment).
3	SAMM Security by Implementation	It does not address this domain directly.
4	SAMM Security by Verification	It does not address this domain directly.
5	SAMM Security by Operations	Immutable Infrastructure contributes to the Security by Operations domain by providing secure build practices (Secure Build). By ensuring that software is developed with security in mind, Immutable Infrastructure reduces the likelihood of introducing vulnerabilities during the build process. It also promotes secure deployment practices, such as secure software distribution and secure configuration of software components (Secure Deployment). Immutable Infrastructure supports environment hardening by implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations (Environment Hardening). Additionally, Immutable Infrastructure integrates well with security testing activities, such as static code analysis and security code reviews (Security Testing). It also enables the establishment of incident response capabilities by swiftly replacing affected instances with new ones that have known and trusted configurations (Incident Management).

11.2.5.2 Assessment – OWASP Top 10

The Immutable Infrastructure capability contributes to addressing various OWASP threats by promoting consistent and reproducible deployments, reducing configuration drift, eliminating vulnerable components, and enforcing secure configurations across the infrastructure. These practices help mitigate common vulnerabilities and enhance the overall security posture of web applications and associated components.

#	Name	Description
1	Broken Access Control (A01-2021)	Immutable infrastructure helps address this threat by ensuring consistent and reproducible deployments. By starting each deployment from a known, predefined state, it eliminates configuration drift and reduces the risk of misconfigurations or unintended changes that could lead to access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	While Immutable Infrastructure does not directly address cryptographic failures, it indirectly contributes to their mitigation. By providing consistent and reproducible deployments, it enables organizations to apply secure cryptographic configurations consistently across their infrastructure, reducing the risk of weak or insecure cryptographic practices.
3	Injection (A03-2021)	Immutable infrastructure can mitigate injection vulnerabilities by ensuring that deployments are based on secure base images and include only necessary components and configurations. By eliminating outdated or compromised components, it reduces the risk of injection attacks that target vulnerable software.
4	Insecure Design (A04-2021)	Immutable infrastructure promotes secure design principles by allowing organizations to define and enforce predefined configurations for their infrastructure components. By incorporating security considerations into the design of these configurations, organizations can mitigate insecure design vulnerabilities and ensure a more robust and secure system.
5	Security Misconfiguration (A05-2021)	Immutable infrastructure directly addresses security misconfiguration vulnerabilities by enforcing predefined configurations for every deployment. By using secure base images and eliminating configuration drift, it reduces the likelihood of insecure or misconfigured settings that could expose sensitive information or create entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	Immutable infrastructure helps mitigate this threat by ensuring that deployments are based on up-to-date and secure base images. By replacing instances entirely for each deployment, organizations can easily incorporate security updates and patches, reducing the risk of using vulnerable or outdated components.
7	Identification and Authentication Failures (A07-2021)	Immutable infrastructure indirectly contributes to the mitigation of identification and authentication failures by providing consistent and reproducible deployments. This allows organizations to apply secure authentication mechanisms consistently across their infrastructure, reducing the risk of weaknesses or flaws in the identification and authentication processes.
8	Software and Data Integrity Failures (A08-2021)	Immutable infrastructure can mitigate software and data integrity failures by ensuring that deployments start from a known and trusted state. By using secure base images and predefined configurations, it reduces the risk of unauthorized modifications or tampering with software components, configurations, or data.
9	Security Logging and Monitoring Failures (A09-2021)	While Immutable Infrastructure does not directly address logging and monitoring failures, it can indirectly contribute to their mitigation. By promoting consistent and reproducible deployments, it provides a foundation for implementing robust logging and monitoring practices across the infrastructure, enabling organizations to detect and respond to security incidents effectively.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Immutable infrastructure can help mitigate server-side request forgery by ensuring secure configurations and eliminating vulnerable components. By starting deployments from secure base images, organizations can reduce the likelihood of SSRF vulnerabilities that could allow attackers to manipulate requests to internal or external resources.

11.2.6 Environment Parity

Strive to maintain parity between different environments, such as development, staging, and production. Ensure that the underlying infrastructure, configuration settings, and dependencies closely resemble the production environment to minimize surprises and issues during deployment.

The aspect of Environment Parity in DevOps enhances IT security.

By striving to maintain parity between different environments, organizations can enhance IT security by ensuring consistent behavior, early detection of security issues, effective security testing, reduced deployment risks, configuration consistency, and easier troubleshooting and bug fixing. These benefits contribute to a more secure and reliable system, minimizing the risk of security vulnerabilities and providing a smoother and more secure software development and deployment process.

General security aspects

#	Name	Description
1	Consistent Behavior:	By maintaining parity between different environments, such as development, staging, and production, organizations can ensure that applications behave consistently across these environments. This consistency helps identify and address security issues early in the development process, reducing the risk of vulnerabilities caused by environment-specific factors or configuration discrepancies.
2	Early Issue Detection	When the development and staging environments closely resemble the production environment, it becomes easier to detect and address security issues during testing and validation stages. Any vulnerabilities or misconfigurations that are present in the production environment are more likely to be identified in the earlier stages, enabling proactive security measures and reducing the risk of security incidents in the live environment.
3	Effective Testing	aintaining parity between environments allows for more effective testing of security-related features and functionalities. Security testing, such as penetration testing or vulnerability scanning, can be conducted in environments that closely resemble the production setup. This enables a more accurate assessment of the application's security posture and helps identify potential vulnerabilities or weaknesses that could be exploited in the production environment.

#	Name	Description
4	Reduced Deployment Risks	When the development, staging, and production environments have parity, the risk of surprises or unexpected issues during deployment is minimized. Organizations can have more confidence in their deployments, knowing that the application has been thoroughly tested and validated in environments that closely resemble the production setup. This reduces the likelihood of security incidents or disruptions caused by environment-specific factors or configuration discrepancies.
5	Configuration Consistency	Parity between environments ensures that the configuration settings remain consistent throughout the software development lifecycle. This consistency helps prevent misconfigurations that could introduce security vulnerabilities or impact the behavior of the application. By maintaining consistent configurations across different environments, organizations can ensure that security measures, such as access controls, encryption settings, or firewall rules, are applied consistently and accurately.
6	Easier Troubleshooting and Bug Fixing	When the development and staging environments closely resemble the production environment, troubleshooting and bug fixing become easier. Any issues or bugs that arise in the production environment can be replicated and debugged in environments that have parity. This accelerates the resolution process and reduces the time to mitigate security-related issues, enhancing the overall security of the application.

11.2.6.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Environment Parity does not directly address the SAMM Security by Governance domain, as it focuses on establishing and maintaining a robust security governance framework. However, maintaining parity between different environments can indirectly contribute to the Security by Governance domain by ensuring consistent behavior, early issue detection, effective testing, reduced deployment risks, configuration consistency, and easier troubleshooting and bug fixing. These benefits enhance overall software security and support the implementation of security governance practices.
2	SAMM Security by Design	Environment Parity does not directly address the SAMM Security by Design domain, as it focuses on designing software applications with security in mind. However, by maintaining parity between environments, organizations can ensure that the design principles and security controls implemented in the development environment closely resemble the production environment. This contributes to the secure architecture, secure coding practices, threat modeling, and security testing activities, which are key aspects of the SAMM Security by Design domain.

#	Name	Description
3	SAMM Security by Implementation	Environment Parity directly contributes to the SAMM Security by Implementation domain. By maintaining parity between different environments, organizations ensure that the secure development training, secure architecture, secure coding guidelines, security requirements, security testing integration, security verification, security architecture review, and security operations integration activities are performed consistently across all environments. This promotes the implementation of secure software development practices and helps build secure software systems.
4	SAMM Security by Verification	Environment Parity directly contributes to the SAMM Security by Verification domain. By maintaining parity between environments, organizations can perform security testing, code review, security architecture review, security requirements verification, threat modeling, and secure deployment verification activities consistently across different environments. This ensures that security vulnerabilities and weaknesses are identified and addressed throughout the software lifecycle, contributing to the verification of software security.
5	SAMM Security by Operations	Environment Parity directly contributes to the SAMM Security by Operations domain. By maintaining parity between different environments, organizations can ensure that environment hardening, secure build practices, configuration management, vulnerability management, incident management, secure deployment, security testing, and operational enablement activities are performed consistently across all environments. This supports the secure and reliable operation of software systems and helps minimize the risk of security incidents and vulnerabilities.

11.2.6.2 Assessment – OWASP Top 10

Environment parity does not directly address all OWASP threats; however, it contributes to improving security by promoting consistency, early issue detection, effective testing, reduced deployment risks, configuration consistency, and easier troubleshooting and bug fixing. These benefits collectively enhance the overall security posture of web applications.

#	Name	Description
1	Broken Access Control (A01-2021)	Consistent Behavior: By maintaining parity between different environments, organizations can ensure that access controls behave consistently across all environments, reducing the risk of access control flaws specific to certain environments.
2	Cryptographic Failures (A02-2021)	Environment parity does not directly address cryptographic failures. However, by maintaining consistent configurations and settings, organizations can ensure that cryptographic algorithms and mechanisms are implemented consistently across environments, reducing the risk of misconfigurations.

#	Name	Description
3	Injection (A03-2021)	Early Issue Detection: By closely resembling the production environment, early detection of injection vulnerabilities becomes easier during testing and validation stages. This allows organizations to proactively address injection flaws before they can be exploited in the live environment.
4	Insecure Design (A04-2021)	Environment parity does not directly address insecure design vulnerabilities. However, by considering security in the design phase and maintaining consistent configurations, organizations can mitigate the risk of insecure design decisions.
5	Security Misconfiguration (A05-2021)	Configuration Consistency: Environment parity ensures that configurations remain consistent across different environments, reducing the likelihood of security misconfigurations that could introduce vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Environment parity does not directly address the use of vulnerable and outdated components. However, by maintaining consistent environments, organizations can ensure that updates and patches are applied consistently, reducing the risk of using insecure versions of software components
7	Identification and Authentication Failures (A07-2021)	Consistent Behavior: Environment parity helps ensure that identification and authentication mechanisms behave consistently across all environments, reducing the risk of authentication bypass or manipulation specific to certain environments.
8	Software and Data Integrity Failures (A08-2021)	Configuration Consistency: Environment parity ensures that software components and data remain consistent across environments, reducing the risk of unauthorized modifications or data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	Environment parity does not directly address logging and monitoring failures. However, by maintaining consistent configurations and settings, organizations can ensure that logging and monitoring mechanisms are consistently implemented across environments.
10	Server-Side Request Forgery (A10-2021)	Early Issue Detection: By closely resembling the production environment, early detection of server-side request forgery vulnerabilities becomes easier during testing and validation stages. This allows organizations to proactively address SSRF flaws before they can be exploited in the live environment.

11.2.7 Monitoring and Logging

Implement comprehensive monitoring and logging solutions to gain visibility into the deployment process and detect any inconsistencies or issues. Monitor key metrics, logs, and alerts to ensure that the deployed applications are functioning as expected.

Implementing comprehensive monitoring and logging solutions as part of DevOps enhances IT security by enabling early detection of security incidents, proactive security monitoring, compliance and auditing, performance optimization, forensic analysis and incident response, threat detection and intrusion

prevention, and continuous improvement and risk mitigation. These practices contribute to a more secure and resilient IT environment, reducing the likelihood of security breaches and enhancing the overall security posture of the deployed applications and infrastructure.

General security aspects

#	Name	Description
1	Early Detection of Security Incidents	Comprehensive monitoring and logging solutions allow for real-time visibility into the deployment process and application behavior. By monitoring key metrics, logs, and alerts, organizations can quickly identify potential security incidents, such as unusual or suspicious activities, unauthorized access attempts, or system anomalies. Early detection enables prompt response and mitigation measures, reducing the impact of security breaches.
2	Proactive Security Monitoring	Monitoring and logging solutions provide continuous monitoring of the deployed applications and infrastructure. This allows organizations to proactively monitor security-related events, such as failed login attempts, access control violations, or unusual network traffic patterns. Proactive monitoring helps identify security threats and vulnerabilities before they can be exploited, enabling timely remediation and reducing the risk of successful attacks.
3	Compliance and Auditing	Monitoring and logging solutions play a crucial role in meeting compliance requirements and facilitating auditing processes. By capturing and retaining detailed logs of system activities, organizations can demonstrate compliance with regulatory standards and industry best practices. Logs provide an audit trail that can be used for investigating security incidents, tracking user actions, and identifying potential compliance violations.
4	Performance Optimization	Monitoring tools provide insights into the performance of the deployed applications and infrastructure. By monitoring key performance metrics, organizations can identify and address performance bottlenecks, resource utilization issues, or infrastructure vulnerabilities that could impact security. Optimizing system performance reduces the likelihood of security incidents caused by system instability, resource exhaustion, or inefficiencies.
5	Threat Detection and Intrusion Prevention	Monitoring and logging solutions can be integrated with security analytics and intrusion detection systems to enhance threat detection capabilities. By analyzing logs and monitoring for patterns or indicators of compromise, organizations can identify potential threats or intrusions and take proactive measures to prevent security breaches. Real-time alerts and automated response mechanisms can be triggered based on predefined security policies, enabling timely threat mitigation.
6	Continuous Improvement and Risk Mitigation	Monitoring and logging data provide valuable insights into system behavior, performance, and security. By analyzing this data, organizations can identify areas for improvement, enhance security controls, and mitigate risks. Data-driven decision-making based on monitoring and logging information allows organizations to continually refine their security measures and adapt to evolving threats.

#	Name	Description
7	Forensic Analysis and Incident Response	Comprehensive logging allows for effective forensic analysis and incident response in the event of a security breach or incident. Detailed logs provide valuable information for investigating the root cause, understanding the extent of the breach, and implementing appropriate remediation measures. Logs enable security teams to reconstruct events, track the progression of an attack, and identify compromised systems or data

11.2.7.1 Assessment – SAMM

The capability of Monitoring and Logging in DevOps enhances IT security by contributing to early detection of security incidents, proactive security monitoring, compliance and auditing, performance optimization, forensic analysis and incident response, threat detection and intrusion prevention, and continuous improvement and risk mitigation. These contributions align with various activities across the SAMM security domains, enhancing their effectiveness in addressing software security.

The DevOps capability of Monitoring and Logging contributes to the SAMM security domains and their activities as follows:

#	Name	Description
1	SAMM Security by Governance	Security Metrics and Reporting: Implementing comprehensive monitoring and logging solutions enables the collection of security-related metrics and provides visibility into the status of software security. These metrics can be used to assess the effectiveness of software security practices and support decision-making at various levels of the organization.
2	SAMM Security by Design	Security Requirements: Monitoring and logging solutions can help capture and analyze security-related requirements, ensuring that they are defined and incorporated into the design phase. This helps address specific security concerns and mitigate vulnerabilities associated with OWASP threats.
3	SAMM Security by Implementation	Security Testing Integration: Monitoring and logging solutions play a crucial role in security testing activities. They enable the collection of relevant data, such as logs and alerts, which can be used for conducting security testing, including static code analysis, dynamic application security testing (DAST), and security code reviews.
4	SAMM Security by Verification	Security Testing: Monitoring and logging solutions provide valuable data for security testing activities, such as penetration testing, vulnerability scanning, and code review. The logs and metrics collected can be used to identify security vulnerabilities and weaknesses in software applications.

#	Name	Description
5	SAMM Security by Operations	Security Testing: Contribution: Monitoring and logging solutions contribute to security testing activities by providing the necessary data for activities such as static code analysis, DAST, and security code reviews. This helps identify and address security vulnerabilities before software systems are deployed and during ongoing operations. Incident Management: Comprehensive monitoring and logging solutions facilitate effective incident management by providing detailed logs that can be analyzed to investigate security incidents, track the progression of an attack, and identify compromised systems or data. Logs enable organizations to respond to incidents promptly and implement appropriate remediation measures.

11.2.7.2 Assessment – OWASP Top 10

Monitoring and Logging capability contributes to addressing various OWASP threats by enabling early detection of security incidents, proactive security monitoring, compliance and auditing, performance optimization, threat detection and intrusion prevention, continuous improvement, and risk mitigation.

#	Name	Description
1	Broken Access Control (A01-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can detect unauthorized access attempts or suspicious activities, providing early detection of potential access control failures. Compliance and Auditing: Monitoring and logging solutions capture detailed logs that can be used to track user actions and investigate access control violations, helping to ensure compliance with access control requirements.
2	Cryptographic Failures (A02-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can detect anomalies or errors in cryptographic operations, such as failed encryption or decryption attempts. Compliance and Auditing: Detailed logs can provide evidence of proper cryptographic usage and help ensure compliance with cryptographic standards and best practices.
3	Injection (A03-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can identify suspicious input patterns or unexpected behavior that may indicate injection attacks. Threat Detection and Intrusion Prevention: By analyzing logs, organizations can detect patterns or indicators of injection attacks, triggering real-time alerts for immediate response.
4	Insecure Design (A04-2021)	Continuous Improvement and Risk Mitigation: Monitoring and logging data provide insights into system behavior and security. By analyzing this data, organizations can identify design flaws and make improvements to mitigate insecure design vulnerabilities.

#	Name	Description
5	Security Misconfiguration (A05-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can identify misconfigurations by monitoring system settings and configurations, allowing for timely detection and response. Compliance and Auditing: Detailed logs help in auditing configurations and ensuring adherence to secure configuration practices.
6	Vulnerable and Outdated Components (A06-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can detect indicators of known vulnerabilities in software components, enabling prompt patching or replacement. Continuous Improvement and Risk Mitigation: Analysis of logs can help identify outdated or vulnerable components and guide organizations in updating or removing them to reduce the risk of exploitation.
7	Identification and Authentication Failures (A07-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can identify suspicious authentication events or failed login attempts, providing early detection of authentication failures. Compliance and Auditing: Detailed logs help in tracking user authentication events and ensuring compliance with authentication requirements.
8	Software and Data Integrity Failures (A08-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can identify unauthorized modifications or tampering attempts, providing early detection of integrity failures. Threat Detection and Intrusion Prevention: By analyzing logs, organizations can detect patterns or indicators of unauthorized modifications, triggering real-time alerts for immediate response.
9	Security Logging and Monitoring Failures (A09-2021)	Proactive Security Monitoring: Monitoring and logging solutions address this threat directly by ensuring effective logging and monitoring practices, allowing organizations to detect and respond to security incidents in a timely manner.
10	Server-Side Request Forgery (A10-2021)	Early Detection of Security Incidents: Monitoring and logging solutions can detect unusual or unauthorized server-side requests, helping to identify potential server-side request forgery vulnerabilities.

11.2.8 Testing and Validation

Establish a robust testing and validation process to verify the correctness and stability of your deployments. This includes automated testing, integration testing, and performance testing, among others, to ensure consistency and quality across different deployments.

The aspect of Testing and Validation in DevOps enhances IT security.

Implementing a robust testing and validation process as part of DevOps, enhances IT security by identifying security vulnerabilities, ensuring consistency and quality, verifying security controls, conducting performance and resilience testing, validating security policies and compliance, detecting bugs and defects, and promoting continuous improvement. These practices contribute to a more secure and reliable IT

environment, reducing the risk of security incidents and enhancing the overall security posture of the deployed system.

General security aspects

#	Name	Description
1	Identification of Security Vulnerabilities	Robust testing and validation processes help identify security vulnerabilities in the deployed applications and infrastructure. By conducting automated security testing, including vulnerability scanning and penetration testing, organizations can detect weaknesses in code, configuration, or network infrastructure that could be exploited by attackers. Identifying and addressing these vulnerabilities before deployment helps improve the overall security posture of the system.
2	Consistency and Quality Assurance	Testing and validation processes ensure consistency and quality across different deployments. By automating testing procedures, organizations can achieve consistent and repeatable tests, reducing the chances of human error and ensuring that security controls and configurations are correctly implemented. Consistent and high-quality deployments minimize the risk of misconfigurations or insecure deployments that could compromise the security of the system.
3	Verification of Security Controls	Testing and validation processes verify the effectiveness of security controls implemented within the system. This includes testing access controls, authentication mechanisms, encryption algorithms, and other security measures to ensure they function as intended. By validating security controls, organizations can identify gaps or weaknesses in the security architecture and make necessary improvements to enhance protection against potential threats.
4	Performance and Resilience Testing	Testing and validation processes encompass performance testing and resilience testing, which indirectly contribute to IT security. Performance testing ensures that the system can handle expected workloads without degradation or failure, preventing performance-related vulnerabilities or denial-of-service attacks. Resilience testing evaluates the system's ability to recover from failures and withstand potential security incidents, reducing the impact of attacks or disruptions.
5	Validation of Security Policies and Compliance	Testing and validation processes help validate the enforcement of security policies and compliance requirements. By conducting tests against predefined security policies and regulatory standards, organizations can ensure that the deployed system adheres to security best practices and meets compliance obligations. Validating security policies and compliance helps mitigate the risk of non-compliance and potential security breaches.
6	Bug and Defect Detection	Testing and validation processes assist in the detection of bugs, defects, or logical flaws in the deployed applications. While not directly related to security, addressing these issues enhances overall system reliability and stability, indirectly reducing the risk of security incidents caused by system instability or unexpected behavior.

#	Name	Description
7	Continuous Improvement	Testing and validation processes foster a culture of continuous improvement by providing feedback on the security aspects of the deployed applications and infrastructure. Through test results and analysis, organizations can identify areas for improvement, refine security practices, and address emerging threats or vulnerabilities. Continuous improvement enhances the resilience and effectiveness of security measures over time.

11.2.8.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does not address this domain directly.
2	SAMM Security by Design	It does not address this domain directly.
3	SAMM Security by Implementation	
4	SAMM Security by Verification	It does not address this domain directly.
5	SAMM Security by Operations	Environment Hardening: DevOps testing and validation processes help ensure secure configurations and infrastructure by identifying vulnerabilities and weaknesses through automated security testing. Secure Build: DevOps testing and validation processes contribute to secure software development practices by verifying the correctness and stability of software deployments, reducing the likelihood of introducing vulnerabilities during the build process. Configuration Management: DevOps testing and validation processes help ensure consistency and quality across different deployments, reducing the risk of misconfigurations or insecure deployments that could compromise the security of the system. Vulnerability Management: DevOps testing and validation processes, including automated security testing and vulnerability scanning, help identify and address security vulnerabilities before deployment, minimizing the risk of exploitation. Incident Management: DevOps testing and validation processes, by ensuring the correctness and stability of deployments, contribute to incident management by reducing the likelihood of security incidents and minimizing their impact. Secure Deployment: DevOps testing and validation processes promote secure deployment practices by verifying the correctness and stability of software deployments and ensuring the secure configuration of software components. Security Testing: DevOps testing and validation processes encompass automated testing, integration testing, and performance testing, contributing to ongoing security testing throughout the software development life cycle.

#	Name	Description
		Operational Enablement: DevOps testing and validation processes contribute to operational support and guidance by promoting a culture of continuous improvement, providing feedback on security aspects, and identifying areas for improvement to enhance the resilience and effectiveness of security measures over time.

11.2.8.2 Assessment – OWASP Top 10

Overall, the Testing and Validation capability of the Architecture Design Component helps address multiple OWASP threats by identifying vulnerabilities, verifying security controls, ensuring consistency and quality, and promoting continuous improvement in the deployed system's security posture.

#	Name	Description
1	Broken Access Control (A01-2021)	Identification of Security Vulnerabilities: Through robust testing and validation processes, organizations can identify security vulnerabilities related to access controls. This helps detect flaws in the enforcement of access controls within the web application, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Consistency and Quality Assurance: Testing and validation processes ensure that cryptographic algorithms and mechanisms are implemented correctly. By verifying the consistency and quality of cryptographic practices, organizations can mitigate the risk of weak algorithms, improper usage, key management issues, and insecure configurations.
3	Injection (A03-2021)	Verification of Security Controls: Testing and validation processes can verify the effectiveness of security controls related to input validation. By conducting rigorous input validation tests, organizations can reduce the risk of injection vulnerabilities, preventing attackers from injecting malicious code or unintended commands into the web application.
4	Insecure Design (A04-2021)	Consistency and Quality Assurance: Testing and validation processes ensure that secure design principles are implemented consistently across different deployments. By validating the security aspects of the application's design, organizations can identify and address insecure design decisions, minimizing the risk of vulnerabilities that could be exploited by attackers.
5	Security Misconfiguration (A05-2021)	Consistency and Quality Assurance: Testing and validation processes help ensure secure configurations are consistently applied to web applications and associated components. By conducting comprehensive tests against security configurations, organizations can reduce the risk of misconfigurations that may lead to security breaches or unnecessary entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	Verification of Security Controls: Testing and validation processes can verify the usage of up-to-date and secure versions of third-party components. By conducting tests and vulnerability assessments on software dependencies, organizations can identify and address vulnerable or outdated components, minimizing the risk of known vulnerabilities.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Verification of Security Controls: Testing and validation processes can verify the effectiveness of identification and authentication mechanisms within web applications. By conducting tests against authentication processes and enforcing strong password policies, organizations can reduce the risk of unauthorized access and impersonation attacks.
8	Software and Data Integrity Failures (A08-2021)	Verification of Security Controls: Testing and validation processes can verify the integrity and protection of software components, configurations, and data. By conducting tests that assess the application's ability to detect unauthorized modifications or data tampering, organizations can enhance the overall integrity and security of their web applications.
9	Security Logging and Monitoring Failures (A09-2021)	Verification of Security Controls: Testing and validation processes can verify the effectiveness of logging and monitoring capabilities within web applications. By conducting tests to ensure proper collection, analysis, and retention of security-related logs, organizations can enhance their ability to detect and respond to security incidents in a timely manner.
10	Server-Side Request Forgery (A10-2021)	Verification of Security Controls: Testing and validation processes can verify the effectiveness of security controls related to input validation and server-side request handling. By conducting tests that identify and prevent server-side request forgery vulnerabilities, organizations can reduce the risk of unauthorized access to sensitive data and service disruption.

11.3 DevSecOps

Implementing security tools configured in the CI-CD pipeline is essential to the overall security implementation and governance.

The tools proposed to be hooked into the CI-CD pipeline will enhance the software development lifecycle management. Some of these tools mitigate the business continuity risk, a hidden dimension of the risk management portfolio. These tools help improve the certainty and reliability of artefacts promoted to production.

11.3.1 Static Code Analysis

Static Code Analysis is a crucial security capability within the broader context of DevSecOps. It involves examining the source code of an application or software component without executing it. The goal is to identify potential vulnerabilities, security weaknesses, and coding best practices that can lead to security breaches or other issues.

By conducting Static Code Analysis as part of the DevSecOps process, organizations can proactively detect security flaws early in the development lifecycle. This approach is highly beneficial as it allows developers to address vulnerabilities before the code is deployed, reducing the risk of deploying insecure code and minimizing the potential impact of a security breach.

In summary, Static Code Analysis is a vital security capability within DevSecOps. By integrating it into the development process, organizations can identify and address security vulnerabilities early, adhere to coding best practices, improve code quality, and automate security checks. This proactive approach ensures that security is an integral part of the software development lifecycle, reducing the risk of deploying insecure code and enhancing overall application security.

Here are some key aspects of Static Code Analysis that contribute to enhancing security in the DevSecOps framework:

.

#	Name	Description
1	Vulnerability Detection:	Static Code Analysis tools examine the codebase for common programming mistakes and known vulnerabilities. It checks for issues such as buffer overflows, SQL injection, cross-site scripting (XSS), insecure cryptography, and more. By identifying these vulnerabilities before deployment, developers can prioritize fixing them, thereby reducing the attack surface and minimizing the risk of exploitation.
2	Security Best Practices:	Static Code Analysis helps enforce coding best practices and adherence to security guidelines and standards. It can flag coding patterns or techniques that are considered risky or insecure. This enables developers to adopt secure coding practices, follow secure coding guidelines, and adhere to industry-accepted security standards, such as the OWASP (Open Web Application Security Project) Top 10.
3	Code Quality Improvement:	Static Code Analysis not only focuses on security vulnerabilities but also helps improve the overall code quality. It identifies issues like code smells, complexity, maintainability, and potential performance bottlenecks. By addressing these issues, developers can enhance the reliability and maintainability of the codebase while reducing the likelihood of introducing security vulnerabilities due to poorly designed or implemented code.
4	Integration with CI/CD Pipelines	To fully leverage the benefits of Static Code Analysis in the DevSecOps process, it is essential to integrate it into the Continuous Integration/Continuous Deployment (CI/CD) pipelines. By automating the code analysis and incorporating it as part of the build and release process, organizations can ensure that every code change undergoes security checks. This integration allows for faster feedback on security issues, facilitating prompt remediation and reducing the time required to fix vulnerabilities
5	False Positive Reduction	Static Code Analysis tools may generate false positives, flagging potential issues that are not actual vulnerabilities. It is crucial to have processes in place to review and filter these findings to avoid wasting time on non-issues. By reducing false positives through proper configuration, custom rules, and tuning, developers can focus on genuine security concerns, improving the effectiveness of the analysis.

11.3.1.1 Assessment – SAMM

DevSecOps and Static Code Analysis contribute to the SAMM security domains as follows:

#	Name	Description
1	SAMM Security by Governance	Does Not Apply Directly. DevSecOps and Static Code Analysis are not directly related to establishing and maintaining security governance frameworks or defining policies, risk management, compliance, security training and awareness, security metrics and reporting, or security roles and responsibilities.
2	SAMM Security by Design	Security Requirements: Static Code Analysis can help identify potential security vulnerabilities and weaknesses in the codebase, ensuring that security requirements are met during the design phase. Secure Architecture: DevSecOps and Static Code Analysis promote the adoption of secure architectural patterns and principles, contributing to the design of secure software systems. Secure Coding Practices: Static Code Analysis helps enforce secure coding practices, reducing the likelihood of introducing vulnerabilities associated with OWASP threats. Threat Modeling: DevSecOps and Static Code Analysis can be integrated into the threat modeling process to identify and address potential security risks during the design phase. Security Testing: Static Code Analysis is a form of security testing that can be conducted during the design phase to detect and address vulnerabilities.
3	SAMM Security by Implementation	Secure Development Training: DevSecOps and Static Code Analysis support the training and awareness of developers in secure coding practices and common software vulnerabilities. Secure Architecture: DevSecOps and Static Code Analysis contribute to designing and implementing a secure architecture, minimizing potential vulnerabilities. Secure Coding Guidelines: Static Code Analysis helps enforce secure coding guidelines and standards during the implementation phase. Security Requirements: DevSecOps and Static Code Analysis help ensure that security requirements are implemented in the software system. Security Testing Integration: Static Code Analysis is a form of security testing that can be integrated into the development process, allowing for early detection of vulnerabilities. Security Verification: DevSecOps and Static Code Analysis contribute to verifying the effectiveness of security controls implemented during the implementation phase. Security Architecture Review: DevSecOps and Static Code Analysis can be used to review the security architecture and identify potential vulnerabilities or weaknesses.

#	Name	Description
		Security Operations Integration: DevSecOps and Static Code Analysis support the integration of security considerations into operational processes, such as incident and vulnerability management.
4	SAMM Security by Verification	Security Testing: DevSecOps and Static Code Analysis contribute to conducting security testing activities, such as penetration testing, vulnerability scanning, and code review. Code Review: Static Code Analysis is a form of code review that helps identify security flaws and adherence to secure coding practices. Security Architecture Review: DevSecOps and Static Code Analysis can be used to review the security architecture and ensure proper security controls are in place. Security Requirements Verification: DevSecOps and Static Code Analysis support the verification of security requirements against industry standards and best practices. Threat Modeling: DevSecOps and Static Code Analysis can be integrated into the threat modeling process to identify and address security threats. Secure Deployment Verification: DevSecOps and Static Code Analysis can help ensure that software applications are securely deployed and configured.
5	SAMM Security by Operations	Environment Hardening - DevSecOps promotes secure configurations and infrastructure practices, contributing to the hardening of the software environment as outlined in SAMM's Operations domain. Secure Build - DevSecOps emphasizes secure build practices, including secure coding standards and build tools. This aligns with SAMM's Operations domain, which includes secure build practices to ensure the software is developed with security in mind. Configuration Management - DevSecOps incorporates configuration management practices, such as version control and change management, to maintain the integrity of software systems, aligning with SAMM's Operations domain. Vulnerability Management - DevSecOps integrates vulnerability management practices, such as vulnerability scanning and assessment, to identify and prioritize vulnerabilities. This aligns with SAMM's Operations domain, which emphasizes the importance of a robust vulnerability management process. Incident Management - DevSecOps includes incident management practices, such as incident detection and response, to handle security incidents effectively. This aligns with SAMM's Operations domain, which recognizes the need for a well-defined incident management process. Secure Deployment - DevSecOps promotes secure deployment practices, including secure software distribution and configuration. This aligns with SAMM's Operations domain, which emphasizes secure deployment as part of managing the secure operation of software systems.

#	Name	Description
		Security Testing - DevSecOps integrates security testing activities throughout the software development life cycle, including static code analysis, dynamic application security testing (DAST), and security code reviews. This aligns with SAMM's Operations domain, which emphasizes the importance of security testing.

11.3.1.2 Assessment – OWASP Top 10

Static Code Analysis as part of the DevSecOps process contributes to addressing multiple OWASP threats by detecting vulnerabilities, enforcing secure coding practices, identifying insecure configurations, and improving the overall security posture of web applications.

The capability of Static Code Analysis in the context of DevSecOps addresses several OWASP threats by contributing in the following ways:

#	Name	Description
11	Broken Access Control (A01-2021)	Static Code Analysis helps identify potential vulnerabilities and security weaknesses in access control implementations. It can detect code patterns or misconfigurations that may lead to unauthorized access or privilege escalation, thus helping organizations address this threat early in the development process.
12	Cryptographic Failures (A02-2021)	Static Code Analysis can analyze cryptographic code and identify weaknesses in the implementation of cryptographic algorithms. It helps detect issues such as the use of weak algorithms, improper key management, or insecure configurations, enabling organizations to fix these vulnerabilities and enhance the security of cryptographic practices.
13	Injection (A03-2021)	Static Code Analysis plays a crucial role in identifying injection vulnerabilities. It can detect code patterns or input validation issues that may lead to injection attacks, such as SQL injection or command injection. By detecting these vulnerabilities, developers can remediate them before deployment, reducing the risk of injection attacks.
14	Insecure Design (A04-2021)	Static Code Analysis can detect design flaws and insecure architectural decisions in the codebase. It helps identify potential vulnerabilities that may arise from inadequate security considerations during the design phase. By detecting and addressing insecure design issues early on, organizations can improve the overall security posture of their applications.
15	Security Misconfiguration (A05-2021)	Static Code Analysis can identify security misconfigurations in the codebase or associated components. It helps detect instances where default or insecure settings are used, leading to potential entry points for attackers. By identifying and rectifying these misconfigurations, organizations can reduce the risk of security breaches and strengthen the security configuration of their applications.

#	Name	Description
16	Vulnerable and Outdated Components (A06-2021)	Static Code Analysis can detect the usage of vulnerable or outdated third-party software components in the codebase. It helps identify dependencies with known security vulnerabilities, enabling organizations to update or replace them with secure versions. By managing software dependencies effectively, organizations can reduce the risk of exploiting known vulnerabilities in third-party components.
17	Identification and Authentication Failures (A07-2021)	Static Code Analysis can detect weaknesses in the implementation of identification and authentication mechanisms. It helps identify potential vulnerabilities that may allow attackers to bypass or manipulate the authentication process. By addressing these vulnerabilities, organizations can strengthen the security of their identification and authentication mechanisms.
18	Software and Data Integrity Failures (A08-2021)	Static Code Analysis can help detect vulnerabilities related to the integrity of software components and data. It can identify code patterns or configurations that may allow unauthorized modifications or tampering of software or data. By addressing these vulnerabilities, organizations can enhance the integrity and security of their applications.
19	Security Logging and Monitoring Failures (A09-2021)	While Static Code Analysis does not directly address this threat, it can indirectly contribute to improving security logging and monitoring capabilities. By identifying code patterns or configurations that may hinder effective logging and monitoring, developers can rectify these issues during the development process, leading to better security logging and monitoring practices.
20	Server-Side Request Forgery (A10-2021)	Static Code Analysis can help identify code patterns or configurations that may be vulnerable to Server-Side Request Forgery (SSRF) attacks. By detecting potential SSRF vulnerabilities, developers can implement input validation and secure configurations to mitigate this threat.

11.3.2 Container Vulnerability Scanning

Container Vulnerability Scanning is a critical security capability within the DevSecOps framework. It involves scanning container images for known vulnerabilities and security issues before they are deployed into production environments. By conducting container vulnerability scanning as part of the DevSecOps process, organizations can ensure that the containers used in their applications are free from known vulnerabilities, thereby reducing the risk of exploitation and enhancing overall security.

Container Vulnerability Scanning is a vital security capability in the DevSecOps approach. By scanning container images for known vulnerabilities and security issues before deployment, organizations can reduce the risk of exploitation, ensure compliance with security policies, and maintain a secure containerized environment. The integration of container vulnerability scanning into CI/CD pipelines and continuous monitoring allows for proactive vulnerability management and timely remediation, strengthening the overall security posture of containerized applications.

Here are some key aspects of Container Vulnerability Scanning that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Identification of Vulnerabilities	Container vulnerability scanning tools analyze container images to identify known vulnerabilities and security issues. These tools leverage vulnerability databases and security feeds to compare the software components and libraries used within the container against a comprehensive list of known vulnerabilities. By detecting these vulnerabilities early in the deployment pipeline, development teams can take appropriate remediation measures, such as applying patches or using updated versions of the affected components.
2	Compliance with Security Policies	Container vulnerability scanning helps ensure compliance with security policies and standards. Organizations often have specific requirements for the security posture of their containerized applications, which may include restrictions on using vulnerable components or outdated libraries. By scanning container images for vulnerabilities, organizations can enforce compliance with their security policies and take proactive steps to address any identified issues before deploying containers to production.
3	Continuous Monitoring	Container vulnerability scanning is not a one-time activity; it should be performed continuously to account for new vulnerabilities and security updates. As new vulnerabilities are discovered and added to vulnerability databases, scanning tools can alert development teams about the presence of these vulnerabilities in their container images. Continuous monitoring allows for proactive vulnerability management and ensures that containers remain secure even after deployment.
4	Integration with CI/CD Pipelines	To fully integrate container vulnerability scanning into the DevSecOps process, it should be seamlessly integrated into the Continuous Integration/Continuous Deployment (CI/CD) pipelines. This integration enables automated scanning of container images at various stages of the deployment pipeline, such as during the build or release process. By automating vulnerability scanning, organizations can identify and address vulnerabilities in a timely manner, reducing the risk of deploying containers with known security issues.
5	Remediation and Mitigation	Container vulnerability scanning not only identifies vulnerabilities but also provides guidance on remediation and mitigation. Scanning tools often offer actionable recommendations for addressing the identified vulnerabilities, such as suggesting specific patches or updated versions of the affected components. This guidance enables development teams to take appropriate actions to remediate vulnerabilities and ensure that container images are free from known security issues before they are deployed.
6	Integration with Container Security Platforms	Container vulnerability scanning can be integrated with container security platforms that provide comprehensive security capabilities specifically tailored for containerized environments. These platforms often include additional features such as runtime monitoring, network segmentation, access control, and threat detection for containers. Integrating container vulnerability scanning with container security platforms provides a holistic approach to container security within the DevSecOps framework.

11.3.2.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Container vulnerability scanning contributes to the establishment and enforcement of policies and standards related to vulnerability management and secure coding practices. It helps organizations ensure that containerized applications comply with security policies by identifying and addressing vulnerabilities before deployment.
2	SAMM Security by Design	Security Requirements: Container vulnerability scanning aids in addressing specific security requirements by identifying vulnerabilities associated with OWASP threats, such as Injection and Broken Access Control. It helps organizations proactively mitigate these vulnerabilities during the design phase. Secure Architecture: Container vulnerability scanning supports the adoption of secure architectural patterns by identifying vulnerabilities related to Insecure Design, Security Misconfiguration, and Cryptographic Failures. It helps organizations create a robust foundation for security controls. Secure Coding Practices: Container vulnerability scanning promotes secure coding practices by detecting vulnerabilities related to Injection, Insecure Design, and Security Misconfiguration. It helps organizations prevent common coding mistakes and vulnerabilities. Threat Modeling: Container vulnerability scanning aids in threat modeling exercises by identifying potential security risks and vulnerabilities in containerized applications. It helps organizations assess the impact of OWASP threats and implement appropriate security controls. Security Testing: Container vulnerability scanning contributes to security testing activities during the design phase by identifying vulnerabilities associated with OWASP threats. It ensures that the design is resilient to potential attacks.
3	SAMM Security by Implementation	Secure Development Training: Container vulnerability scanning supports secure development training by identifying vulnerabilities and weaknesses in containerized applications. It helps developers and development teams understand the importance of secure coding practices and common software vulnerabilities. Secure Architecture: Container vulnerability scanning aligns with secure architecture practices by identifying vulnerabilities in the design and implementation of security controls. It helps organizations build a secure architecture from the early stages of development. Secure Coding Guidelines: Container vulnerability scanning helps enforce secure coding guidelines by identifying violations and vulnerabilities in containerized applications. It ensures that developers adhere to best practices for writing secure code. Security Requirements: Container vulnerability scanning contributes to the verification of security requirements by identifying vulnerabilities that deviate from the established requirements. It ensures that security objectives are met during implementation. Security Testing Integration: Container vulnerability scanning integrates with security testing activities by identifying vulnerabilities during the development process. It ensures that vulnerabilities are detected and addressed early on.

#	Name	Description
		Security Verification: Container vulnerability scanning aids in security verification by identifying vulnerabilities that may exist in implemented security controls. It helps organizations validate the effectiveness of security measures. Security Architecture Review: Container vulnerability scanning supports security architecture reviews by identifying potential vulnerabilities or weaknesses in the design and architecture of containerized applications. It provides recommendations for improvement. Security Operations Integration: Container vulnerability scanning contributes to the integration of security considerations into operational processes by identifying vulnerabilities that may affect secure deployment practices. It ensures that security measures are supported and maintained throughout the operational life of containerized applications.
4	SAMM Security by Verification	Security Testing: Container vulnerability scanning is a type of security testing that helps identify vulnerabilities and weaknesses in software applications. Code Review: Container vulnerability scanning involves analyzing container images to detect security flaws, aligning with code review activities. Security Architecture Review: Container vulnerability scanning aids in assessing the security architecture of software applications by identifying vulnerabilities present in container images. Security Requirements Verification: Container vulnerability scanning contributes to verifying security requirements by assessing the presence of vulnerabilities that may violate those requirements. Threat Modeling: Container vulnerability scanning helps organizations identify and prioritize potential threats and risks associated with software applications. Secure Deployment Verification: Container vulnerability scanning supports secure deployment verification by ensuring that containers are scanned for vulnerabilities before deployment.
5	SAMM Security by Operations	Environment Hardening: Container vulnerability scanning helps identify vulnerabilities in container images, supporting the process of hardening the software environment. Secure Build: By scanning container images for vulnerabilities, organizations can ensure that the build process includes secure coding practices and secure software development frameworks. Configuration Management: Container vulnerability scanning assists in maintaining the integrity of software systems by identifying unauthorized changes or misconfigurations in container images.

11.3.2.2 Assessment – OWASP Top 10

Container Vulnerability Scanning does not directly address all OWASP threats; however, it provides significant contributions in mitigating vulnerabilities and reducing the risk of exploitation, ultimately enhancing the overall security posture of web applications within the DevSecOps framework.

#	Name	Description
1	Broken Access Control (A01-2021)	While Container Vulnerability Scanning does not directly address this threat, it contributes indirectly by ensuring that container images used in web applications do not contain vulnerabilities that could be exploited to bypass access controls or gain unauthorized access.
2	Cryptographic Failures (A02-2021)	Container Vulnerability Scanning helps mitigate this threat by scanning container images for vulnerabilities in cryptographic libraries and components. It identifies known vulnerabilities in cryptographic algorithms or implementations, enabling organizations to address these weaknesses before deploying containers into production.
3	Injection (A03-2021)	Container Vulnerability Scanning does not directly address Injection vulnerabilities. However, by ensuring that container images do not contain vulnerable software components or libraries, it reduces the risk of using insecure code that could be exploited for injection attacks.
4	Insecure Design (A04-2021)	Container Vulnerability Scanning does not directly address Insecure Design vulnerabilities. However, by identifying vulnerabilities in container images, it helps organizations discover potential design flaws in their applications' containerization strategy and prompts remediation actions to enhance the overall security of the design.
5	Security Misconfiguration (A05-2021)	Container Vulnerability Scanning significantly contributes to addressing Security Misconfiguration vulnerabilities. It identifies misconfigurations in container images, such as exposed sensitive information, insecure default settings, or unnecessary functionality, helping organizations rectify these issues before deploying containers into production.
6	Vulnerable and Outdated Components (A06-2021)	Container Vulnerability Scanning directly addresses this threat. It scans container images for known vulnerabilities in software components, including libraries, frameworks, and plugins. By identifying vulnerable and outdated components, organizations can update or replace them with more secure versions, reducing the risk of exploitation through these components.
7	Identification and Authentication Failures (A07-2021)	Container Vulnerability Scanning does not directly address this threat. However, by ensuring that container images do not contain vulnerabilities that could compromise authentication mechanisms, it indirectly contributes to reducing the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	Container Vulnerability Scanning does not directly address this threat. However, by scanning container images for vulnerabilities, it helps ensure the integrity of software components used in web applications, reducing the risk of unauthorized modifications or tampering that could compromise data integrity.
9	Security Logging and Monitoring Failures (A09-2021)	Container Vulnerability Scanning does not directly address this threat. However, it indirectly contributes to effective logging and monitoring by identifying vulnerabilities in container images that could affect the logging and monitoring capabilities of web applications. By resolving these vulnerabilities, organizations can enhance their overall security logging and monitoring practices.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Container Vulnerability Scanning does not directly address this threat. However, by identifying and mitigating vulnerabilities in container images, it reduces the risk of using insecure code that could be exploited to perform Server-Side Request Forgery attacks.

11.3.3 Unit Test Coverage

Unit Test Coverage is a crucial security capability in the DevSecOps methodology. By ensuring that unit tests cover a significant portion of the codebase, organizations can identify bugs, vulnerabilities, and potential security flaws early in the development process. This approach improves the overall code quality, reduces the risk of security breaches, and enhances the maintainability of the codebase. Integrating Unit Test Coverage into CI/CD pipelines promotes a security-focused development culture and enables automated validation of code units before they are deployed to production.

Here are key aspects of Unit Test Coverage that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Code Validation	Unit Test Coverage serves as a means of code validation, where developers write tests to verify the behavior and correctness of specific code units or components. These units can include functions, classes, methods, or modules. By writing comprehensive unit tests, developers can verify that the code behaves as expected and that potential vulnerabilities or security flaws are identified and addressed early in the development process.
2	Bug and Vulnerability Identification	Unit tests play a crucial role in identifying bugs and vulnerabilities within the codebase. By executing a variety of test cases, including both normal and edge cases, developers can uncover issues such as input validation failures, boundary condition errors, buffer overflows, and other common software vulnerabilities. Identifying these issues through unit tests allows developers to fix them before they become more critical security risks.
3	Security Testing	While unit tests primarily focus on functional correctness, they can also be extended to include security-specific test cases. These security-focused unit tests can help identify potential vulnerabilities and security weaknesses within the code. For example, developers can write unit tests to check for common security issues such as SQL injection, cross-site scripting (XSS), authentication bypass, or insecure handling of sensitive data. By including security-focused unit tests, organizations can strengthen the security of their applications and reduce the risk of security breaches.
4	Early Detection and Mitigation	Unit Test Coverage ensures that bugs and potential security flaws are identified early in the development process. By running tests on code units as they are being developed, developers can catch issues before they propagate to other parts of the system. Early detection allows for timely mitigation and reduces the cost and effort required to fix security vulnerabilities later in the development lifecycle.

#	Name	Description
5	Integration with CI/CD Pipelines	To fully leverage the benefits of Unit Test Coverage, it should be integrated into the Continuous Integration/Continuous Deployment (CI/CD) pipelines. This integration ensures that unit tests are executed automatically as part of the build and deployment process. By incorporating unit tests into the CI/CD pipelines, organizations can enforce the execution of tests before code is promoted to production, providing an additional layer of security and quality assurance.
6	Code Maintainability	Unit Test Coverage contributes to the overall maintainability of the codebase. By writing unit tests, developers document the expected behavior of code units and provide future developers with a safety net for making changes without introducing unintended consequences. This documentation aspect of unit tests improves the maintainability of the codebase and facilitates ongoing security assessments and updates.
7	Test Driven Development (TDD)	Unit Test Coverage aligns with the principles of Test Driven Development (TDD), where tests are written before the corresponding code is implemented. By following a TDD approach, developers are encouraged to think about potential security vulnerabilities and design code that is resilient to attacks from the outset. This proactive mindset helps foster a culture of security awareness and enables developers to address security concerns during the early stages of development.

11.3.3.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	It does Not Apply Directly. Unit Test Coverage is not directly related to the activities within the Security by Governance domain. However, it indirectly supports security governance by improving overall code quality and reducing the risk of security breaches.
2	SAMM Security by Design	Security Requirements: Unit Test Coverage contributes to the identification and validation of security requirements by thoroughly testing code units, helping to ensure that security concerns are addressed during the design phase. Secure Architecture: Unit Test Coverage helps validate the functionality and security of individual code units, supporting the implementation of secure architectural patterns and principles to address various OWASP threats. Secure Coding Practices: Unit Test Coverage aids in verifying that secure coding practices are followed by identifying potential vulnerabilities and common software weaknesses during the design phase. Threat Modeling: Unit Test Coverage can be used to validate the effectiveness of security controls identified during threat modeling exercises, ensuring that the design is resilient to potential OWASP threats. Security Testing: Unit Test Coverage can be extended to include security-focused test cases, helping to identify vulnerabilities related to OWASP threats and ensuring the design is secure.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training: Unit Test Coverage helps developers understand the importance of writing unit tests to validate the security of code units and encourages the adoption of secure coding practices. Secure Architecture: Unit Test Coverage aids in validating the security measures implemented during the development phase, supporting the establishment of a secure architecture. Secure Coding Guidelines: Unit Test Coverage helps ensure that developers adhere to secure coding guidelines by verifying the functionality and security of code units. Security Requirements: Unit Test Coverage contributes to verifying the implementation of security requirements by validating the behavior and security of code units against the specified requirements. Security Testing Integration: Unit Test Coverage is an essential part of security testing activities, providing an automated approach to test individual code units and detect vulnerabilities. Security Verification: Unit Test Coverage aids in the verification of security controls by validating the functionality and security of code units against established security requirements. Security Architecture Review: Unit Test Coverage helps evaluate the security of the software system's design and architecture by validating the behavior and security of code units. Security Operations Integration: Unit Test Coverage ensures that code units are tested for functionality and security during the development process, contributing to the overall security of the software system.
4	SAMM Security by Verification	Security Testing: Unit Test Coverage is a form of security testing that helps identify vulnerabilities and weaknesses in software applications, supporting the validation and verification of the software's security. Code Review: Unit Test Coverage aids in identifying security flaws and adherence to secure coding practices during code reviews by providing insights into the functionality and security of code units. Security Architecture Review: Unit Test Coverage supports the assessment of the security architecture by validating the behavior and security of code units against the defined security controls and mechanisms. Security Requirements Verification: Unit Test Coverage contributes to the verification of security requirements by validating the behavior and security of code units against the specified requirements. Threat Modeling: Unit Test Coverage helps assess the impact of potential threats and risks by validating the resilience of the software system's design to attacks and vulnerabilities. Secure Deployment Verification: Unit Test Coverage aids in ensuring the secure deployment of software applications by validating the behavior and security of code units during the deployment process.
5	SAMM Security by Operations	It does Not Apply Directly. Unit Test Coverage is not directly related to the activities within the Security by Operations domain. However, it indirectly supports secure operations by improving overall code quality and reducing the risk of security vulnerabilities.

11.3.3.2 Assessment – OWASP Top 10

Unit Test Coverage may not directly address all OWASP Top 10 threats; however, it plays a significant role in improving code quality and identifying security issues early in the development process, indirectly contributing to mitigating several OWASP threats. When combined with other security practices and testing methodologies, Unit Test Coverage strengthens the overall security posture of web applications within the DevSecOps framework.

#	Name	Description
1	Broken Access Control (A01-2021)	Unit Test Coverage helps in ensuring that proper access controls are implemented and enforced in the codebase. By writing unit tests that cover different scenarios related to access control, developers can validate that users are only able to access the resources and functionalities they are authorized to use. This prevents unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	While Unit Test Coverage mainly focuses on functional correctness, it indirectly contributes to addressing cryptographic failures by validating the behavior of cryptographic functions or components. For example, unit tests can ensure that encryption and decryption functions work as expected and handle different input scenarios securely.
3	Injection (A03-2021)	Unit Test Coverage helps identify and prevent injection vulnerabilities by thoroughly testing user input handling. By including unit tests for input validation and parameterization, developers can ensure that user-supplied data is properly sanitized and doesn't lead to code execution vulnerabilities.
4	Insecure Design (A04-2021)	Unit Test Coverage can contribute to addressing insecure design vulnerabilities by promoting secure coding practices. Developers can write unit tests that validate the correct implementation of security controls and architectural decisions to prevent potential security flaws stemming from design weaknesses.
5	Security Misconfiguration (A05-2021)	it Test Coverage indirectly addresses security misconfiguration by validating the behavior of different components and configurations. Unit tests can be written to check the correctness of security configurations, ensuring that the application follows secure practices and settings.
6	Vulnerable and Outdated Components (A06-2021)	While Unit Test Coverage does not directly address this threat, it can play a role in identifying vulnerabilities related to third-party components. For example, developers can write unit tests to check for known vulnerabilities in third-party libraries or frameworks used in the codebase.
7	Identification and Authentication Failures (A07-2021)	Unit Test Coverage can help identify authentication-related issues by thoroughly testing authentication mechanisms. By including unit tests for login, logout, and other authentication functions, developers can ensure that user identification and authentication work as intended.
8	Software and Data Integrity Failures (A08-2021)	Unit Test Coverage indirectly addresses software and data integrity failures by validating the correctness of data processing and storage functions. Unit tests can ensure that data is handled securely, and there are no vulnerabilities that could compromise data integrity.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Unit Test Coverage does not directly address this threat, but it can indirectly contribute to logging and monitoring. Developers can include unit tests for logging mechanisms to ensure that the application logs the necessary security-related information effectively.
10	Server-Side Request Forgery (A10-2021)	Unit Test Coverage can help prevent SSRF vulnerabilities by validating how the application handles external requests. By including unit tests for request handling and ensuring that only allowed requests are made, developers can reduce the risk of SSRF attacks.

11.3.4 Static Cloud and Infrastructure Code Analysis

Static Cloud and Infrastructure Code Analysis is a crucial security capability in the DevSecOps approach. By analyzing cloud and infrastructure code for misconfigurations and enforcing security best practices, organizations can detect and prevent security vulnerabilities before they are deployed to production. This capability helps ensure compliance, reduces the attack surface, and enhances the security and resilience of cloud-based environments. By integrating static code analysis into CI/CD pipelines, organizations can achieve continuous security validation and foster a culture of security throughout the development and deployment lifecycle.

Here are key aspects of Static Cloud and Infrastructure Code Analysis that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Misconfiguration Detection	Static analysis tools are used to scan cloud and infrastructure code for potential misconfigurations. These tools analyze the codebase to identify security-related issues, such as overly permissive access controls, exposed sensitive data, weak encryption settings, or improper network configurations. By detecting misconfigurations early in the development process, organizations can proactively address them and prevent security vulnerabilities from being introduced into the production environment.
2	Compliance and Best Practice Enforcement	Static analysis of cloud and infrastructure code helps ensure compliance with security standards, regulatory requirements, and best practices. It allows organizations to define and enforce specific security policies for their cloud-based infrastructure. For example, the analysis can verify that encryption is enabled for data at rest, multi-factor authentication is enforced, or only approved network traffic is allowed. By enforcing compliance and best practices through static code analysis, organizations can reduce the risk of security incidents and maintain a robust security posture.

#	Name	Description
3	Infrastructure as Code (IaC) Security	With the rise of Infrastructure as Code (IaC) tools like Terraform, cloud infrastructure is often defined and provisioned through code. Static analysis of infrastructure code ensures that security considerations are embedded in the codebase. It helps validate that infrastructure resources are properly configured, securely deployed, and that access controls are correctly implemented. By analyzing IaC code, organizations can detect potential security issues early in the development process and address them before deploying the infrastructure.
4	Continuous Security Validation	Static code analysis of cloud and infrastructure code is integrated into the CI/CD pipelines to provide continuous security validation. By automating the analysis process, organizations can ensure that security checks are performed consistently and in a timely manner throughout the development and deployment lifecycle. This enables developers to receive immediate feedback on security issues, facilitating rapid remediation and reducing the time between vulnerability discovery and resolution.
5	Risk Mitigation	Static analysis of cloud and infrastructure code helps mitigate security risks associated with misconfigurations. By identifying and addressing misconfigurations early, organizations can prevent potential security breaches and operational disruptions. It enables them to reduce the attack surface, enhance the security posture of their cloud-based systems, and improve overall resilience against threats and vulnerabilities.
6	Collaboration and Knowledge Sharing	Static code analysis tools provide insights and actionable recommendations to developers, enabling collaboration and knowledge sharing around secure coding practices. These tools can generate reports and documentation highlighting detected issues, along with suggestions for remediation. By sharing this information, developers can learn from each other's experiences, adopt secure coding practices, and continuously improve the security of the codebase.

11.3.4.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Static Cloud and Infrastructure Code Analysis contributes to this activity by ensuring that policies and standards related to secure coding practices, vulnerability management, incident response, and data protection are enforced through the analysis of cloud and infrastructure code. Risk Management: Static Cloud and Infrastructure Code Analysis helps address risk management by identifying potential misconfigurations and security vulnerabilities in cloud and infrastructure code, allowing organizations to assess and mitigate risks associated with software security.

#	Name	Description
		Compliance Management: Static Cloud and Infrastructure Code Analysis supports compliance management by verifying that cloud and infrastructure code adheres to relevant regulatory requirements, industry standards, and best practices, ensuring that security policies and controls are implemented correctly. Security Training and Awareness: While Static Cloud and Infrastructure Code Analysis may not directly contribute to security training and awareness, the insights and recommendations provided by analysis tools can be used to educate developers and stakeholders about secure coding practices and the importance of addressing misconfigurations in cloud and infrastructure code. Security Metrics and Reporting: Static Cloud and Infrastructure Code Analysis generates reports highlighting detected issues and vulnerabilities in cloud and infrastructure code. These reports contribute to security metrics and reporting by providing visibility into the security status of the codebase and supporting decision-making in security governance. Security Roles and Responsibilities: While Static Cloud and Infrastructure Code Analysis does not directly address security roles and responsibilities, it indirectly supports this activity by providing developers with insights and recommendations for secure coding practices, helping them fulfill their security responsibilities effectively.
2	SAMM Security by Design	Security Requirements: Static Cloud and Infrastructure Code Analysis indirectly contributes to addressing security requirements by identifying potential misconfigurations and vulnerabilities in cloud and infrastructure code that may violate security requirements. By analyzing the codebase, organizations can ensure that security requirements are incorporated into the design of the software. Secure Architecture: Static Cloud and Infrastructure Code Analysis helps promote secure architecture by identifying and addressing potential security issues in cloud and infrastructure code. By ensuring that infrastructure resources are properly configured and access controls are correctly implemented, organizations can establish a strong foundation for secure architecture. Secure Coding Practices: Static Cloud and Infrastructure Code Analysis directly contributes to promoting secure coding practices by analyzing cloud and infrastructure code for common coding mistakes and vulnerabilities. It helps developers identify and remediate issues related to OWASP threats such as Injection, Insecure Design, and Security Misconfiguration. Threat Modeling: While Static Cloud and Infrastructure Code Analysis does not explicitly address threat modeling, it indirectly supports this activity by detecting potential vulnerabilities and weaknesses in cloud and infrastructure code, enabling organizations to assess the impact of OWASP threats and implement appropriate security controls. Security Testing: Static Cloud and Infrastructure Code Analysis can complement security testing activities by providing an additional layer of analysis and vulnerability detection during the design phase. It helps identify potential vulnerabilities related to OWASP threats, supporting the development of resilient designs.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training: While Static Cloud and Infrastructure Code Analysis does not directly provide secure development training, it can contribute indirectly by identifying insecure coding practices and providing recommendations for improvement. This information can be used to educate developers and development teams on secure coding practices. Secure Architecture: Static Cloud and Infrastructure Code Analysis supports the implementation of secure architecture by analyzing cloud and infrastructure code for potential security issues. By addressing these issues, organizations can ensure that the software system is built on a secure foundation. Secure Coding Guidelines: Static Cloud and Infrastructure Code Analysis aligns with secure coding guidelines by scanning cloud and infrastructure code for violations of secure coding practices. It helps enforce and reinforce these guidelines by providing feedback on coding mistakes and vulnerabilities. Security Requirements: Static Cloud and Infrastructure Code Analysis indirectly supports security requirements implementation by identifying potential misconfigurations and security vulnerabilities in cloud and infrastructure code. By addressing these issues, organizations can ensure that the implemented software meets the defined security requirements.
4	SAMM Security by Verification	Security Testing: Static code analysis is a type of security testing that helps identify vulnerabilities and weaknesses in software applications. Code Review: Static code analysis is used for code review, enabling the identification of security flaws and adherence to secure coding practices. Security Architecture Review: Static code analysis can assist in reviewing the security architecture of software applications to ensure the presence of proper security controls and mechanisms. Security Requirements Verification: Static code analysis helps verify the proper implementation of security requirements in software applications. Threat Modeling: Static code analysis can be used to analyze potential threats and risks to software applications, aiding in the identification and prioritization of security threats. Secure Deployment Verification: Static code analysis contributes to ensuring secure deployment and configuration of software applications.
5	SAMM Security by Operations	Environment Hardening: Static code analysis helps identify and address security vulnerabilities in software environments, supporting the implementation of secure configurations. Secure Build: Static code analysis aids in ensuring that secure coding standards and practices are followed during the build process, reducing the introduction of vulnerabilities. Configuration Management: Static code analysis can assist in detecting unauthorized changes or misconfigurations, contributing to effective configuration management practices. Vulnerability Management: Static code analysis helps identify vulnerabilities, enabling organizations to prioritize and address them as part of their vulnerability management process.

#	Name	Description
		Incident Management: Static code analysis helps identify security vulnerabilities that could contribute to security incidents, supporting incident detection and response activities. Secure Deployment: Static code analysis contributes to secure deployment practices by detecting vulnerabilities and misconfigurations during the deployment process. Security Testing: Static code analysis is a form of security testing that can be integrated into operational processes to continuously assess and address security vulnerabilities. Operational Enablement: Static code analysis supports the ongoing security and reliability of software systems by identifying and addressing vulnerabilities and providing guidance for operational staff. In summary, Static Cloud and Infrastructure Code Analysis directly contributes to several SAMM security domains and activities, including Compliance Management, Security Requirements, Secure Architecture, Secure Coding Practices, Threat Modeling, Security Testing, and Operational Enablement.

11.3.4.2 Assessment – OWASP Top 10

Static Cloud and Infrastructure Code Analysis does not directly address all OWASP threats; however, it significantly contributes to mitigating several critical vulnerabilities. By detecting misconfigurations, enforcing best practices, and facilitating collaboration among developers, this capability helps organizations build more secure cloud-based environments, reducing the risk of exploitation and maintaining a robust security posture.

#	Name	Description
1	Broken Access Control (A01-2021)	Misconfiguration Detection: Static code analysis can identify misconfigurations in access controls, such as overly permissive access settings or inadequate privilege verification, helping prevent unauthorized access and privilege escalation. Compliance and Best Practice Enforcement: By analyzing cloud and infrastructure code, static code analysis ensures that access control mechanisms are correctly implemented and enforced, reducing the risk of broken access control vulnerabilities.
2	Cryptographic Failures (A02-2021)	It does not address this domain directly.
3	Injection (A03-2021)	It does not address this domain directly.
4	Insecure Design (A04-2021)	It does not address this domain directly.
5	Security Misconfiguration (A05-2021)	Misconfiguration Detection: Static code analysis can identify security misconfigurations in cloud and infrastructure code, such as default or insecure settings, and provide recommendations for proper configurations. Compliance and Best Practice Enforcement: Static code analysis enforces secure configuration practices, ensuring that cloud and infrastructure code adheres to security standards and best practices, reducing the likelihood of security misconfigurations.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	Misconfiguration Detection: Static code analysis can identify the usage of outdated or known vulnerable components within cloud and infrastructure code, allowing organizations to update or replace them with secure versions. Compliance and Best Practice Enforcement: By analyzing the codebase, static code analysis can enforce policies that require the use of up-to-date and secure components, minimizing the risk of vulnerabilities associated with vulnerable or outdated software.
7	Identification and Authentication Failures (A07-2021)	It does not address this domain directly.
8	Software and Data Integrity Failures (A08-2021)	Misconfiguration Detection: Static code analysis can identify potential integrity failures in cloud and infrastructure code, such as insecure data handling or weak integrity checks, helping ensure the integrity of software components and data. Compliance and Best Practice Enforcement: By enforcing secure coding practices through static code analysis, organizations can reduce the likelihood of software and data integrity failures in their cloud-based systems.
9	Security Logging and Monitoring Failures (A09-2021)	Compliance and Best Practice Enforcement: Static code analysis can enforce logging and monitoring requirements, ensuring that cloud and infrastructure code includes proper logging mechanisms and real-time monitoring capabilities to detect and respond to security incidents effectively.
10	Server-Side Request Forgery (A10-2021)	It does not address this domain directly.

11.3.5 Penetration Testing

Penetration Testing, also known as ethical hacking or pen testing, is a crucial security capability within the DevSecOps framework. It involves conducting controlled simulated attacks on a system or application to identify vulnerabilities and weaknesses that could potentially be exploited by malicious actors. Penetration testing goes beyond automated vulnerability scanning and provides a more comprehensive assessment of the security posture.

Penetration Testing is a vital security capability within DevSecOps. By simulating real-world attack scenarios, identifying vulnerabilities, and providing actionable recommendations, organizations can enhance their security posture and reduce the risk of exploitation. By integrating penetration testing into the development lifecycle and conducting regular tests, organizations can continuously improve their security practices and stay resilient against emerging threats.

#	Name	Description
1	Identifying Vulnerabilities	Penetration testing aims to uncover vulnerabilities that may exist in various layers of the system, including the network, applications, infrastructure, and configurations. By actively probing the system, ethical hackers attempt to exploit these vulnerabilities to gain unauthorized access, escalate privileges, or compromise data. By identifying vulnerabilities, organizations can understand their risk exposure and take proactive measures to patch or mitigate them.
2	Exploitation and Impact Analysis	During a penetration test, ethical hackers attempt to exploit vulnerabilities to gain access to sensitive information or compromise the system. This process helps organizations understand the potential impact of a successful attack, including the extent of data exposure, unauthorized access to critical resources, or disruption of services. By analyzing the impact, organizations can prioritize their remediation efforts and allocate resources effectively to mitigate the most significant risks.
3	Vulnerability Verification	Penetration testing provides an additional layer of verification for vulnerabilities identified through other testing methods, such as static code analysis, dynamic application security testing (DAST), or vulnerability scanning. While these methods can identify potential vulnerabilities, penetration testing validates their real-world exploitability. By confirming the existence and impact of vulnerabilities, organizations can prioritize remediation efforts and ensure that resources are allocated to address the most critical security issues.
4	Post-Exploitation Analysis and Recommendations	After successfully exploiting vulnerabilities, ethical hackers provide detailed reports outlining the vulnerabilities identified, the methods used for exploitation, and recommendations for remediation. These reports often include actionable steps to address the identified vulnerabilities and enhance the overall security posture. By leveraging the insights from penetration testing, organizations can make informed decisions about security improvements and implement effective remediation strategies.
5	Real-World Scenario Simulations	Penetration testing simulates real-world attack scenarios to assess the system's resilience against potential threats. Ethical hackers use a combination of automated tools and manual techniques to exploit vulnerabilities and gain unauthorized access to the system. By mimicking the techniques and tactics employed by real attackers, organizations can evaluate the effectiveness of their security controls and incident response capabilities.
6	Compliance and Regulatory Requirements	Penetration testing is often required to meet compliance standards and regulatory requirements. Many industry frameworks and regulations, such as PCI DSS (Payment Card Industry Data Security Standard) or HIPAA (Health Insurance Portability and Accountability Act), mandate regular penetration testing to assess the security of systems handling sensitive data. By conducting penetration tests, organizations can demonstrate their commitment to security and compliance obligations.

#	Name	Description
7	Continuous Testing and Improvement	Penetration testing is not a one-time activity but an ongoing process. In a DevSecOps approach, penetration testing is integrated into the development lifecycle, enabling continuous testing and improvement. By conducting regular penetration tests, organizations can proactively identify and address security vulnerabilities throughout the software development lifecycle. This iterative approach helps maintain a strong security posture and reduces the risk of exploitation by staying ahead of evolving threats.

11.3.5.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Penetration testing contributes to this activity by validating the effectiveness of policies and standards related to software security. It helps identify vulnerabilities and weaknesses in the system, enabling organizations to refine their policies and standards to address these issues effectively. Risk Management: Penetration testing helps assess the risks associated with software development and deployment. By identifying vulnerabilities through penetration testing, organizations can prioritize and mitigate risks, ensuring effective risk management practices. Compliance Management: Penetration testing is often required to meet compliance standards and regulatory requirements. By conducting penetration tests, organizations can ensure compliance with security regulations and guidelines, demonstrating their commitment to security governance. Security Training and Awareness: Penetration testing provides real-world scenarios that help raise awareness among employees and stakeholders about the importance of software security. It highlights the potential risks and encourages organizations to implement security training programs to educate individuals about vulnerabilities and mitigation strategies. Security Metrics and Reporting: Penetration testing contributes to the measurement of security metrics by identifying vulnerabilities and providing insights into the effectiveness of software security practices. The findings from penetration testing can be used to improve security metrics and reporting mechanisms. Security Roles and Responsibilities: Penetration testing helps validate the effectiveness of security roles and responsibilities within the organization. By identifying vulnerabilities and weaknesses, organizations can ensure that individuals responsible for software security are aware of their roles and have the necessary knowledge and skills to fulfill them effectively.
2	SAMM Security by Design	Security Requirements: Penetration testing helps validate the effectiveness of security requirements by identifying vulnerabilities and weaknesses that may arise from inadequate or missing security requirements. Secure Architecture: Penetration testing verifies the resilience of the secure architecture against potential threats. By identifying vulnerabilities through penetration testing, organizations can refine their architectural design to address these vulnerabilities effectively.

#	Name	Description
		Secure Coding Practices: Penetration testing helps identify vulnerabilities resulting from insecure coding practices. It emphasizes the importance of secure coding guidelines and practices by demonstrating the impact of insecure coding on the overall security of the software system. Threat Modeling: Penetration testing validates the effectiveness of threat modeling exercises by identifying potential security risks and vulnerabilities. The findings from penetration testing can be used to refine threat models and ensure comprehensive risk identification and mitigation. Security Testing: Penetration testing is a key activity in security testing, as it provides real-world scenarios to evaluate the effectiveness of security controls. By incorporating penetration testing into the design phase, organizations can identify vulnerabilities early on and address them proactively.
3	SAMM Security by Implementation	Secure Development Training: Contribution: Penetration testing highlights the importance of secure development training by demonstrating the impact of insecure coding practices and vulnerabilities that could be exploited by attackers. Secure Architecture: Contribution: Penetration testing validates the effectiveness of secure architecture practices implemented during the development phase, identifying potential vulnerabilities and weaknesses. Secure Coding Guidelines: Contribution: Penetration testing helps identify deviations from secure coding guidelines and provides feedback on the effectiveness of the guidelines in preventing OWASP-related vulnerabilities. Security Testing Integration: Contribution: Penetration testing is a crucial component of security testing, providing real-world validation of security controls and detecting vulnerabilities that may have been missed by other testing methods. Security Verification:Contribution: Penetration testing contributes to security verification by validating the effectiveness of security controls implemented in the software system and ensuring that security requirements are met.
4	SAMM Security by Verification	Security Testing: Penetration testing, as a key capability within DevSecOps, contributes to the Security by Verification domain of SAMM by conducting controlled simulated attacks on software systems. It helps identify vulnerabilities and weaknesses, ensuring the effectiveness of security verification processes. Code Review: While not explicitly mentioned in the DevSecOps capabilities provided, code review is an essential aspect of security testing. It helps identify security flaws and adherence to secure coding practices, contributing to the verification of software security. Security Architecture Review: Similarly, while not directly mentioned, security architecture review is a crucial aspect of penetration testing. By assessing the security controls and mechanisms in place, it validates the effectiveness of security architecture and supports the verification process. Security Requirements Verification: Penetration testing also indirectly contributes to verifying security requirements by validating the implementation of security controls and ensuring compliance with industry standards and best practices. Threat Modeling: Although not explicitly mentioned, threat modeling is often part of the penetration testing process. It helps identify potential threats and risks to software applications, contributing to the verification of security requirements and supporting the overall verification process.

#	Name	Description
		Secure Deployment Verification: Penetration testing can also contribute to the verification of secure deployment by assessing whether security controls, encryption mechanisms, access controls, and other security measures are properly implemented during deployment.
5	SAMM Security by Operations	Environment Hardening: Penetration testing indirectly contributes to environment hardening by identifying vulnerabilities and weaknesses in the software environment. By addressing these issues, organizations can strengthen their operational security practices. Secure Build: Although not explicitly mentioned, the DevSecOps capability of penetration testing supports the concept of secure build practices. By identifying vulnerabilities and weaknesses during testing, organizations can improve the secure build process and minimize the introduction of vulnerabilities. Configuration Management: While not directly addressed by penetration testing, it indirectly contributes to configuration management by identifying misconfigurations or insecure configurations during testing. By addressing these issues, organizations can improve their configuration management practices. Vulnerability Management: Penetration testing plays a crucial role in vulnerability management by identifying vulnerabilities and prioritizing them for remediation. By integrating penetration testing into the operational processes, organizations can effectively manage vulnerabilities throughout the software's lifecycle. Incident Management: Although not explicitly mentioned, the insights gained from penetration testing can help organizations improve their incident management processes. By identifying potential security incidents and vulnerabilities, organizations can respond effectively and minimize the impact of security incidents. Secure Deployment: Penetration testing indirectly contributes to secure deployment by identifying vulnerabilities and weaknesses that could be exploited during the deployment process. By addressing these issues, organizations can enhance their secure deployment practices. Security Testing: Penetration testing, as a key DevSecOps capability, supports the overall security testing aspect of SAMM's Operations domain. By conducting regular security testing, including penetration testing, organizations can identify and address vulnerabilities, ensuring the ongoing security of software systems. Operational Enablement: Although not explicitly mentioned, the insights and recommendations provided by penetration testing contribute to operational enablement. By implementing the recommended remediation measures, organizations can improve their operational practices and enhance the security and reliability of their software systems.

11.3.5.2 Assessment – OWASP Top 10

Penetration Testing significantly contributes to addressing the OWASP Top 10 threats by proactively identifying vulnerabilities and providing actionable recommendations to enhance the overall security posture of web applications and systems. It helps organizations stay ahead of emerging threats and improve their security practices continuously.

#	Name	Description
1	Broken Access Control (A01-2021)	Penetration testing helps identify flaws in access controls and authorization mechanisms. By attempting to bypass access controls during the testing, ethical hackers can reveal areas where improper enforcement of access controls could lead to unauthorized access, privilege escalation, or exposure of sensitive data.
2	Cryptographic Failures (A02-2021)	Penetration testing includes an analysis of cryptographic implementations. It helps identify issues like weak encryption algorithms, improper key management, or insecure usage of cryptography within the application. By identifying these weaknesses, organizations can address cryptographic vulnerabilities and enhance data protection.
3	Injection (A03-2021)	Penetration testing attempts to exploit injection vulnerabilities by providing malicious inputs to the application. By successfully identifying and demonstrating injection vulnerabilities, organizations can understand the risks and implement secure coding practices to prevent such attacks.
4	Insecure Design (A04-2021)	Penetration testing can reveal insecure design decisions, such as insufficient input validation or improper session management. By assessing the application's design, ethical hackers can help organizations make necessary improvements to avoid potential security flaws.
5	Security Misconfiguration (A05-2021)	Penetration testing assesses the application and associated components for security misconfigurations. By identifying misconfigurations, organizations can correct default or insecure settings, minimizing potential entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	Penetration testing may uncover vulnerabilities related to outdated or vulnerable third-party components used in the application. By identifying these weaknesses, organizations can update or replace the affected components, reducing the risk of exploitation.
7	Identification and Authentication Failures (A07-2021)	Penetration testing evaluates the effectiveness of authentication mechanisms. By attempting to bypass or manipulate authentication, ethical hackers can uncover weaknesses that may lead to unauthorized access or impersonation.
8	Software and Data Integrity Failures (A08-2021)	Penetration testing can identify vulnerabilities related to data tampering or unauthorized modifications. By assessing the integrity of software and data, organizations can strengthen the protection of critical resources.
9	Security Logging and Monitoring Failures (A09-2021)	Penetration testing helps identify deficiencies in logging and monitoring capabilities. By detecting gaps in logging, analysis, or real-time monitoring, organizations can enhance their incident detection and response capabilities.
10	Server-Side Request Forgery (A10-2021)	Penetration testing includes testing for SSRF vulnerabilities. By simulating SSRF attacks, organizations can identify areas where attackers could manipulate the application to make unintended requests, potentially leading to unauthorized access or data exposure.

11.3.6 Performance Testing

Performance Testing is an important security capability within the DevSecOps framework. It focuses on evaluating the performance of a system or application under various conditions to identify potential bottlenecks, vulnerabilities, and performance-related security risks. By conducting performance testing, organizations can ensure that their systems can handle expected workloads, provide optimal user experience, and mitigate the risk of performance-related security issues, such as denial-of-service attacks. Performance Testing is a crucial security capability within DevSecOps. By evaluating the system's performance under various conditions, identifying bottlenecks, and mitigating performance-related security

risks, organizations can ensure that their systems can handle expected workloads, provide optimal user experience, and remain resilient against performance-related security incidents. By integrating performance testing into the development and deployment processes, organizations can continuously optimize the system's performance, enhance its security, and deliver reliable and efficient applications or services to users.

Here are key aspects of Performance Testing that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Workload Analysis	Performance testing involves analyzing the expected workloads and usage patterns to simulate realistic scenarios. By understanding the anticipated traffic and load on the system, organizations can design appropriate performance tests that mimic the real-world environment. This analysis helps identify potential bottlenecks and performance-related vulnerabilities that may impact the system's security and overall performance.
2	Stress Testing	In addition to evaluating the system under expected workloads, performance testing includes stress testing to assess its behavior and resilience when subjected to extreme or unexpected conditions. This testing helps identify the system's breaking points, resource limitations, and potential vulnerabilities that could be exploited under stress or overload. By uncovering these weaknesses, organizations can take proactive measures to enhance the system's robustness and mitigate the risk of performance-related security incidents.
3	Scalability Assessment	Performance testing also evaluates the system's scalability by assessing its ability to handle increasing workloads and user demands. By simulating scenarios where the user base grows rapidly or the workload spikes suddenly, organizations can identify scalability limitations that could lead to performance degradation or security vulnerabilities. This assessment helps optimize the system's scalability and ensures it can adapt to changing demands while maintaining security and performance.
4	Response Time Analysis	Performance testing measures the response time of the system under different loads and conditions. By monitoring the response time, organizations can identify potential latency issues, processing bottlenecks, or inefficient resource utilization that could impact the system's performance and security. Detecting and resolving such issues ensures that the system remains responsive and resistant to attacks that exploit delays or performance bottlenecks.

#	Name	Description
5	Load Balancing and Resource Management	Performance testing helps evaluate the effectiveness of load balancing mechanisms and resource management strategies. Load balancing distributes the incoming traffic across multiple servers or resources to optimize performance and prevent overload. By testing load balancing configurations and resource allocation strategies, organizations can ensure that the system can handle high volumes of traffic, evenly distribute the workload, and mitigate the risk of performance-related security issues.
6	Denial-of-Service (DoS) Mitigation	Performance testing plays a crucial role in mitigating DoS attacks. By subjecting the system to high loads and stress conditions, organizations can assess its resilience against DoS attacks and identify potential vulnerabilities or weak points that could be exploited to disrupt or degrade the system's performance. This testing helps organizations implement appropriate measures, such as rate limiting, traffic shaping, or caching, to mitigate the impact of DoS attacks and maintain the system's availability and performance.
7	Infrastructure Optimization	Performance testing not only focuses on the application layer but also includes evaluating the underlying infrastructure and network components. By analyzing the performance of servers, databases, network connections, and other infrastructure elements, organizations can identify potential performance bottlenecks or security vulnerabilities that may arise due to suboptimal configurations or resource constraints. This optimization helps ensure that the infrastructure supports the expected workload and maintains the desired performance and security levels.
8	Continuous Performance Monitoring	Performance testing is an ongoing process that should be integrated into the DevSecOps lifecycle. By continuously monitoring the system's performance metrics, organizations can proactively detect any deviations or degradation in performance that may indicate potential security risks or vulnerabilities. Continuous performance monitoring enables organizations to quickly identify and address performance-related issues, ensuring the system remains secure, stable, and performant throughout its lifecycle.

11.3.6.1 Assessment – SAMM

DevSecOps capability of Performance Testing contributes to the following SAMM security domains and sub-domains:

#	Name	Description
1	SAMM Security by Governance	It does not apply directly to this domain.

#	Name	Description
2	SAMM Security by Design	Security Requirements: Performance testing helps validate the performance requirements defined in the security requirements, ensuring that the system meets the desired performance levels and remains resilient against performance-related security incidents. Secure Architecture: Performance testing helps evaluate the system's performance and identify potential bottlenecks or vulnerabilities that could impact the secure architecture. By addressing these performance-related issues, organizations can enhance the system's security. Secure Coding Practices: Performance testing can reveal performance-related coding issues, such as inefficient resource utilization or processing bottlenecks, that could impact the system's security. By optimizing the code for performance, organizations can also improve the system's security posture.
3	SAMM Security by Implementation	Secure Development Training: Performance testing helps developers understand the impact of their code on the system's performance and security. By incorporating performance testing into the development training, developers can learn to write code that not only meets functional requirements but also performs well and maintains security. Secure Architecture: Performance testing contributes to ensuring that the implemented architecture meets the desired performance and security objectives. By identifying architectural weaknesses or performance bottlenecks, organizations can make informed decisions to enhance the system's security. Secure Coding Guidelines: Performance testing can uncover performance-related coding issues that violate secure coding guidelines. By addressing these issues, organizations can ensure that the code follows secure coding practices and performs optimally.
4	SAMM Security by Verification	Security Testing: Performance testing is a form of security testing that assesses the system's resilience against performance-related security incidents. By conducting performance testing, organizations can identify potential vulnerabilities or weaknesses that could be exploited by attackers to compromise the system's security.
5	SAMM Security by Operations	Performance Testing contributes to the SAMM Security by Operations domain by ensuring the ongoing security and reliability of software applications during their operational phase. By continuously monitoring and conducting performance testing, organizations can detect any deviations or degradation in performance that may indicate potential security risks or vulnerabilities, enabling them to take corrective actions and maintain the secure and reliable operation of the software system.

11.3.6.2 Assessment – OWASP Top 10

Performance Testing in DevSecOps primarily focuses on evaluating the system's performance and identifying performance-related security risks; however, it indirectly contributes to addressing some OWASP threats by identifying potential vulnerabilities and weaknesses that may impact the security posture of the application. However, it does not directly tackle all the OWASP threats, as some of them require specific security controls and measures beyond performance testing. Therefore, a comprehensive security approach should include other security practices and capabilities alongside Performance Testing to address the full range of OWASP threats effectively.

#	Name	Description
1	Broken Access Control (A01-2021)	Performance testing indirectly addresses this threat by simulating realistic scenarios, including various access levels and user interactions. By doing so, it can help identify potential access control vulnerabilities, such as insufficient privilege validation or missing authorization checks, which may lead to unauthorized access.
2	Cryptographic Failures (A02-2021)	Performance testing does not directly address this threat. Cryptographic vulnerabilities are more related to the implementation of cryptographic algorithms and mechanisms, which are not the primary focus of performance testing.
3	Injection (A03-2021)	Performance testing indirectly addresses injection vulnerabilities by validating and sanitizing input data during the testing process. It helps identify potential weaknesses in input validation mechanisms that might lead to injection attacks.
4	Insecure Design (A04-2021)	Performance testing does not directly address this threat. Insecure design vulnerabilities are more related to architectural flaws and inadequate security considerations during the design phase, which are not the primary focus of performance testing.
5	Security Misconfiguration (A05-2021)	Performance testing indirectly addresses security misconfigurations by evaluating the system under different configurations. It can identify misconfigured security settings that may lead to potential security risks.
6	Vulnerable and Outdated Components (A06-2021)	Performance testing indirectly addresses this threat by assessing the system's performance under different software configurations. It may identify outdated or vulnerable components that need to be updated to mitigate the risk of exploitation.
7	Identification and Authentication Failures (A07-2021)	Performance testing does not directly address this threat. Identification and authentication failures are more related to the design and implementation of authentication mechanisms, which are not the primary focus of performance testing.
8	Software and Data Integrity Failures (A08-2021)	Performance testing indirectly addresses data integrity vulnerabilities by evaluating the system's response under various loads and conditions. It can detect potential data tampering issues that may arise from performance-related bottlenecks.
9	Security Logging and Monitoring Failures (A09-2021)	Performance testing does not directly address this threat. Security logging and monitoring failures are more related to the implementation of logging and monitoring mechanisms, which are not the primary focus of performance testing.
10	Server-Side Request Forgery (A10-2021)	Performance testing indirectly addresses this threat by simulating different requests and interactions with the server-side components. It may help identify potential SSRF vulnerabilities that could be triggered under high loads or stress conditions.

11.3.7 Regression Testing

Regression Testing is a vital security capability within the DevSecOps framework. It involves testing previously developed and functional software components after modifications or updates to ensure that existing security features remain intact. The primary goal of regression testing is to prevent regression

bugs, which are unintended changes or introductions of issues that may occur when new modifications are made to the software.

Regression Testing is a critical security capability within DevSecOps that focuses on testing previously developed software components after modifications or updates. By verifying the integrity of existing security features, preventing the reintroduction of security vulnerabilities, and ensuring the stability of the software, organizations can maintain the security posture of their applications or systems. By integrating regression testing into the development and deployment processes, organizations can proactively identify and address any security-related regressions, providing greater confidence in the security of their software.

Here are key aspects of Regression Testing that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Test Coverage	Regression testing aims to achieve comprehensive test coverage of the software components that have been modified or updated. This ensures that all critical security aspects are thoroughly tested to identify any potential regression bugs or security vulnerabilities that may have been introduced. The testing process may include both manual and automated tests to verify the functionality and security of the software.
2	Security Feature Verification	Regression testing focuses on verifying that the existing security features and controls of the software remain intact after modifications. This includes authentication mechanisms, access controls, encryption algorithms, data validation, and other security-related functionalities. By testing these features, organizations can ensure that any changes made to the software do not compromise its security posture.
3	Security Vulnerability Prevention	Regression testing plays a crucial role in preventing the reintroduction of security vulnerabilities that may have been previously identified and addressed. By retesting the software components, organizations can identify any unintended changes that may have reintroduced security weaknesses or vulnerabilities. This allows them to quickly address and fix these issues before they can be exploited by attackers.
4	Continuous Integration and Testing	Regression testing is integrated into the continuous integration and testing processes within DevSecOps. As new modifications are made to the software, automated regression tests are triggered to verify the stability and security of the system. This ensures that any potential regressions are identified and resolved early in the development cycle, reducing the risk of deploying insecure code.

#	Name	Description
5	Test Environment Reproduction	To perform effective regression testing, it is essential to replicate the production or target environment as closely as possible. By creating a representative test environment, organizations can simulate real-world conditions and evaluate the software's behavior and security under these circumstances. This helps uncover any discrepancies or issues that may arise when the software is deployed in the actual production environment.
6	Test Case Management	Regression testing requires the establishment and maintenance of a comprehensive test case suite. This suite includes test cases specifically designed to validate the security features and functionality of the software components. Test case management ensures that all critical areas of the software are adequately covered and that the test cases are regularly updated to accommodate new modifications or updates.
7	Version Control and Change Management	Regression testing relies on version control and change management practices to track and manage software modifications effectively. By maintaining a detailed history of changes, organizations can identify the specific components that need to undergo regression testing and ensure that all relevant security aspects are thoroughly evaluated. Version control also enables organizations to roll back to previous versions if regressions or security vulnerabilities are discovered.
8	Collaboration and Communication:	Regression testing involves collaboration and effective communication among developers, testers, and security teams. This ensures that everyone is aligned regarding the modifications being made, the expected behavior of the software, and the security requirements. Collaborative efforts help identify potential areas of concern and allow for proactive measures to be taken to address any security-related issues identified during regression testing.

11.3.7.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Regression testing helps ensure that existing security policies and standards are maintained and validated after modifications or updates to software components. Risk Management: Regression testing aids in identifying potential security risks and vulnerabilities introduced through modifications or updates, supporting risk management processes. Compliance Management: Regression testing verifies that the software remains compliant with relevant regulatory requirements, industry standards, and best practices, contributing to compliance management efforts. Security Training and Awareness: Regression testing ensures that the security training and awareness programs are effective by validating that security features remain intact after modifications. Security Metrics and Reporting: Regression testing helps assess the effectiveness of software security practices by providing metrics and reporting on the stability and security of the software. Security Roles and Responsibilities: Regression testing contributes to accountability for software security by verifying that security controls are properly implemented according to assigned roles and responsibilities.

#	Name	Description
2	SAMM Security by Design	Security Requirements: Regression testing validates that security requirements are met by ensuring that security features remain intact after modifications or updates. Secure Architecture: Regression testing helps verify that the secure architecture is maintained by testing the functionality and security of software components. Secure Coding Practices: Regression testing aids in verifying that secure coding practices are adhered to by ensuring the integrity of security features during modifications or updates. Threat Modeling: Regression testing supports threat modeling efforts by identifying any potential security risks or vulnerabilities introduced through modifications or updates. Security Testing: Regression testing contributes to security testing activities during the design phase by validating the stability and security of the software components.
3	SAMM Security by Implementation	Secure Development Training: Regression testing ensures that developers are trained on secure coding practices by validating the integrity of security features during modifications or updates. Secure Architecture: Regression testing supports the implementation of secure architecture practices by verifying the stability and security of the software components. Secure Coding Guidelines: Regression testing helps ensure adherence to secure coding guidelines by verifying the integrity of security features after modifications or updates. Security Requirements: Regression testing contributes to the implementation of security requirements by validating that the security objectives and expectations are met. Security Testing Integration: Regression testing is an integral part of security testing activities during the implementation phase, aiding in identifying and addressing security vulnerabilities. Security Verification: Regression testing verifies the effectiveness of security controls implemented in the software system, contributing to security verification efforts. Security Architecture Review: Regression testing helps identify potential vulnerabilities or weaknesses in the security architecture, supporting security architecture review processes. Security Operations Integration: Regression testing ensures that security considerations are integrated into operational processes, such as incident management and secure deployment practices.
4	SAMM Security by Verification	Security Testing: Regression testing is a form of security testing that helps identify security vulnerabilities in software applications. Code Review: Regression testing complements code review activities by verifying the integrity of security features and the adherence to secure coding practices. Security Architecture Review: Regression testing aids in the security architecture review process by validating the stability and security of the software components. Security Requirements Verification: Regression testing verifies that the security requirements are properly implemented in the software applications. Threat Modeling: Regression testing supports threat modeling efforts by identifying potential security risks introduced through modifications or updates. Secure Deployment Verification: Regression testing ensures that software applications are securely deployed and configured, validating the effectiveness of secure deployment practices.
5	SAMM Security by Operations	Environment Hardening: Regression testing helps ensure the security of the software environment by identifying and addressing any regressions that may compromise the secure configurations, patches, updates, and infrastructure configurations.

#	Name	Description
		Secure Build: Regression testing supports the secure build practices by verifying that modifications or updates do not introduce vulnerabilities during the build process, aligning with secure coding standards and practices. Configuration Management: Regression testing contributes to configuration management by validating that modifications or updates are properly documented, tracked, and controlled, ensuring the integrity and security of software configurations. Vulnerability Management: Regression testing aids in vulnerability management by identifying any regressions that may reintroduce vulnerabilities, allowing organizations to prioritize and address them in a timely manner. Incident Management: Regression testing helps incident management by detecting any regressions that may impact the security and reliability of software systems, supporting incident response, containment, and recovery efforts. Secure Deployment: Regression testing ensures secure deployment practices by verifying that the software is deployed securely and that modifications or updates do not introduce unauthorized access or compromise during the deployment process. Security Testing: Regression testing is an integral part of security testing activities by proactively identifying any regressions that may introduce security vulnerabilities and ensuring that the software meets the desired security objectives. Operational Enablement: Regression testing contributes to operational enablement by providing ongoing support and guidance to ensure the security and reliability of software systems, aligning with operational procedures and incident response capabilities. Policies and Standards: Regression testing contributes to the establishment of policies and standards by ensuring that existing security features and controls are thoroughly tested after modifications or updates, aligning with secure coding practices, vulnerability management, and incident response requirements. Risk Management: Regression testing helps mitigate risks associated with software modifications by verifying the integrity of existing security features and preventing the reintroduction of security vulnerabilities. Compliance Management: Regression testing supports compliance management efforts by validating that security features remain intact and that modifications or updates do not introduce compliance issues. Security Training and Awareness: Regression testing reinforces the importance of security training and awareness by proactively identifying any security-related regressions and providing opportunities for knowledge transfer and skill development. Security Metrics and Reporting: Regression testing contributes to security metrics and reporting by demonstrating the effectiveness of the testing process in identifying and resolving security-related regressions. Security Roles and Responsibilities: Regression testing ensures that security roles and responsibilities are fulfilled by verifying that modifications or updates do not compromise the assigned security responsibilities within the organization. Collaboration and Communication: Regression testing promotes collaboration and effective communication among developers, testers, and security teams to address any security-related issues identified during the testing process.

11.3.7.2 Assessment – OWASP Top 10

Regression Testing alone cannot fully address all OWASP threats; however, it significantly contributes to maintaining the security posture of applications during the DevSecOps process. By combining Regression Testing with other security practices like secure coding, code reviews, and penetration testing, organizations can create a robust security strategy to defend against a wide range of OWASP threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Regression Testing helps verify that existing security features and access controls remain intact after modifications. By thoroughly testing access control mechanisms during regression testing, organizations can prevent unauthorized access and privilege escalation vulnerabilities.
2	Cryptographic Failures (A02-2021)	While Regression Testing may not directly address cryptographic failures, it indirectly helps ensure that cryptographic components and algorithms continue to function as intended after updates. By maintaining the integrity of the application during regression testing, it helps protect against potential attacks that could exploit cryptographic weaknesses.
3	Injection (A03-2021)	Regression Testing plays a critical role in preventing Injection vulnerabilities. By ensuring that user input is properly validated and sanitized, it can identify regressions that could introduce injection flaws, thereby reducing the risk of attackers manipulating the application's execution flow.
4	Insecure Design (A04-2021)	Regression Testing, when performed diligently, can uncover insecure design decisions that might have been introduced during modifications. By identifying and addressing insecure design elements, organizations can prevent vulnerabilities from being introduced into the software.
5	Security Misconfiguration (A05-2021)	Regression Testing helps in detecting configuration-related regressions, ensuring that the software components maintain secure configurations and do not expose sensitive information or unnecessary functionality, reducing the risk of security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Regression Testing can indirectly contribute to identifying vulnerabilities related to third-party components. By retesting components after updates, organizations can verify that vulnerable dependencies are patched or replaced, reducing the risk associated with using outdated or known vulnerable components.
7	Identification and Authentication Failures (A07-2021)	Regression Testing can help identify regressions that might lead to authentication vulnerabilities. By verifying the effectiveness of authentication mechanisms, organizations can prevent attackers from bypassing or manipulating the authentication process.
8	Software and Data Integrity Failures (A08-2021)	Regression Testing ensures that the integrity of software and data is maintained during and after modifications. By detecting regressions that could compromise software and data integrity, organizations can address potential vulnerabilities and protect against unauthorized modifications.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Regression Testing may not directly address logging and monitoring issues, but it can help ensure that the functionality responsible for logging and monitoring is not negatively impacted by updates. By maintaining these critical security features, organizations can enhance their ability to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	Regression Testing helps in validating that server-side components handle requests correctly. By identifying regressions that could lead to SSRF vulnerabilities, organizations can protect against unauthorized access to internal resources and potential data exposure.

11.3.8 Mock Testing

Mock Testing is a valuable security capability within the DevSecOps framework. It involves simulating specific scenarios or inputs to test the system's response and identify potential security vulnerabilities. The primary purpose of mock testing is to uncover security weaknesses by testing how the system handles unexpected or malicious inputs.

Mock Testing is a critical security capability within DevSecOps that focuses on simulating specific scenarios or inputs to test the system's response and identify potential security vulnerabilities. By crafting mock inputs, evaluating the system's behavior, and assessing its ability to handle unexpected or malicious inputs, organizations can proactively identify and mitigate security weaknesses. By integrating mock testing into the development and testing processes, organizations can enhance the security posture of their applications or systems and reduce the risk of exploitation.

Here are key aspects of Mock Testing that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Simulation of Scenarios	Mock testing allows security teams to simulate various scenarios that may occur in real-world usage, including both expected and unexpected inputs. By crafting mock inputs that mimic different types of user behavior or malicious attacks, organizations can evaluate how the system responds and identify any security vulnerabilities that may be exposed.
2	Input Validation	Mock testing emphasizes the importance of input validation by deliberately providing inputs that may be invalid, malformed, or malicious. This helps assess the system's ability to handle such inputs securely. By validating and sanitizing user inputs effectively, organizations can prevent common security issues like injection attacks (e.g., SQL injection, cross-site scripting) and buffer overflows.
3	Boundary Testing	Mock testing involves testing the system's response to inputs that fall within or exceed specified boundaries. By providing inputs at the lower and upper limits of acceptable values, organizations can verify that the system handles edge cases correctly. This helps identify potential security vulnerabilities that may arise from improper input handling, such as integer overflows or underflows.

#	Name	Description
4	Error Handling and Resilience	Mock testing enables organizations to evaluate the system's error handling and resilience capabilities. By intentionally triggering error conditions and exceptions, such as network failures, unexpected file operations, or memory exhaustion, organizations can assess how the system handles these scenarios. Robust error handling ensures that the system fails gracefully without exposing sensitive information or becoming vulnerable to attacks.
5	Integration Testing	Mock testing can be performed in conjunction with integration testing to assess the security of interactions between different system components or services. By simulating the behavior of external dependencies or systems, organizations can evaluate how the system handles integration scenarios and ensure that security measures are properly implemented at the interface boundaries.
6	API Security Testing	In the context of APIs (Application Programming Interfaces), mock testing can be particularly useful for assessing API security. By crafting mock requests and responses, organizations can test the API's authentication mechanisms, access controls, input validation, and response handling. This helps identify potential security weaknesses in the API implementation, preventing unauthorized access or data breaches.
7	Test Automation	Mock testing can be automated to enhance its efficiency and effectiveness. By developing automated test scripts that generate mock inputs and evaluate the system's responses, organizations can conduct comprehensive and repetitive tests without significant manual effort. Test automation ensures that mock testing is integrated into the overall continuous integration and testing processes of DevSecOps, providing timely feedback on security vulnerabilities.
8	Collaboration and Knowledge Sharing	Mock testing requires collaboration and knowledge sharing between security teams, developers, and testers. By sharing insights and best practices, organizations can ensure that mock testing covers relevant security scenarios and addresses potential vulnerabilities effectively. Collaboration helps align security objectives with development efforts and promotes a culture of security awareness across the organization.

11.3.8.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Mock Testing does not directly contribute to the Security by Governance domain in SAMM. While mock testing helps uncover security weaknesses, it is not specifically related to establishing and maintaining a robust security governance framework or defining policies, risk management, compliance management, security training and awareness, security metrics and reporting, or defining security roles and responsibilities.

#	Name	Description
2	SAMM Security by Design	Mock Testing contributes to the SAMM Security by Design domain by supporting the following activity: Security Requirements. Mock testing helps validate the effectiveness of security requirements by simulating different scenarios and inputs to test the system's response and identify potential vulnerabilities related to security requirements.
3	SAMM Security by Implementation	Mock Testing contributes to the SAMM Security by Implementation domain by supporting the following activities: Security Testing Integration and Security Verification. Mock testing enables organizations to include security testing activities in the development process, such as simulating mock inputs to identify vulnerabilities and validate security controls. It also helps in verifying the effectiveness of security controls by evaluating the system's response to mock inputs and ensuring that they meet the desired security objectives.
4	SAMM Security by Verification	Mock Testing contributes to the SAMM Security by Verification domain by supporting the following activities: Security Testing and Code Review. Mock testing helps in conducting security testing by simulating specific scenarios and inputs to identify vulnerabilities. It also aids in code review by assessing the system's response to mock inputs and identifying security flaws or adherence to secure coding practices.
5	SAMM Security by Operations	Mock Testing contributes to the SAMM Security by Operations domain by supporting the following activities: Security Testing and Incident Management. Mock testing helps in conducting security testing to identify vulnerabilities and ensure ongoing security of software systems. It also contributes to incident management by proactively testing the system's response to mock inputs and assessing its resilience in handling unexpected or malicious inputs, reducing the risk of security incidents.

11.3.8.2 Assessment – OWASP Top 10

Mock testing within the DevSecOps framework contributes to addressing various OWASP threats by simulating scenarios, testing input validation, assessing system behavior, identifying misconfigurations, evaluating authentication mechanisms, testing software and data integrity, assessing logging and monitoring practices, and uncovering SSRF vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	Mock testing helps identify potential weaknesses in access controls by simulating different scenarios and inputs. By evaluating the system's response to these inputs, organizations can uncover vulnerabilities related to access control enforcement.
2	Cryptographic Failures (A02-2021)	It does not address this domain directly.

#	Name	Description
3	Injection (A03-2021)	Mock testing emphasizes input validation by providing inputs that may be invalid or malicious. By testing how the system handles these inputs, organizations can identify injection vulnerabilities and ensure that user-supplied data is properly validated and sanitized.
4	Insecure Design (A04-2021)	Mock testing allows organizations to assess the system's behavior and response to various scenarios. By simulating potential security threats during mock testing, organizations can identify insecure design decisions and address them early in the development process.
5	Security Misconfiguration (A05-2021)	Mock testing can help identify misconfigurations by testing the system's behavior under different conditions. By intentionally triggering error conditions and exceptions, organizations can assess how the system handles these situations and uncover potential security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Mock testing can be used to assess the behavior of the system when interacting with third-party components. By simulating scenarios involving these components, organizations can identify vulnerabilities related to their usage and ensure that they are up to date.
7	Identification and Authentication Failures (A07-2021)	Mock testing can evaluate the system's authentication mechanisms by simulating different authentication scenarios. By testing how the system handles authentication inputs and responses, organizations can identify weaknesses in the identification and authentication process.
8	Software and Data Integrity Failures (A08-2021)	Mock testing can help assess the system's ability to handle and protect software components and data. By simulating scenarios involving unauthorized modifications or data tampering, organizations can identify vulnerabilities related to software and data integrity.
9	Security Logging and Monitoring Failures (A09-2021)	Mock testing can evaluate the system's logging and monitoring capabilities by simulating security-related events. By testing how the system collects, analyzes, and retains logs, organizations can identify weaknesses in logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	Mock testing can help identify vulnerabilities related to server-side request forgery by simulating different requests and assessing the system's response. By testing how the system handles these requests, organizations can uncover potential SSRF vulnerabilities and ensure that proper security controls are in place.

11.3.9 Integration Testing

Integration Testing is a crucial security capability within the DevSecOps framework. It involves testing the interaction between different software components to identify integration-related security issues. The primary objective of integration testing is to ensure that different components work together securely and to identify vulnerabilities that may arise due to the integration of different systems.

Integration Testing is a crucial security capability within DevSecOps that focuses on testing the interaction between different software components to identify integration-related security issues. By validating component interactions, ensuring interface security, validating data flow, verifying compatibility, and

addressing other aspects of secure integration, organizations can enhance the security of their systems. By integrating integration testing into the development and deployment processes, organizations can proactively identify and mitigate security vulnerabilities, reducing the risk of security breaches and ensuring the overall integrity and reliability of their software systems.

Here are key aspects of Integration Testing that contributes to the overall security when embedded into DevSecOps:

#	Name	Description
1	Component Interaction	Integration testing focuses on testing the interactions between various software components or modules that make up a larger system. It verifies that these components exchange data and communicate securely, preventing unauthorized access or data breaches that can occur due to insecure integration.
2	Interface Security	Integration testing places significant emphasis on verifying the security of interfaces between components. This includes validating the authentication mechanisms, authorization controls, and data encryption used during data exchange. By ensuring the integrity and confidentiality of data transferred between components, integration testing helps prevent security vulnerabilities that can arise from weak interface security.
3	Data Flow Validation	Integration testing evaluates the flow of data between components to ensure it is handled securely. This involves verifying that data is correctly transformed, validated, and sanitized as it passes between components. By validating the input and output data at each integration point, integration testing helps identify potential security flaws such as injection attacks or data tampering.
4	Compatibility and Interoperability	Integration testing verifies the compatibility and interoperability of different software components. It ensures that components developed by different teams or vendors can work together seamlessly without compromising security. By identifying compatibility issues and potential security vulnerabilities that may arise from conflicting components, integration testing helps maintain the integrity and security of the overall system.
5	Secure Data Exchange	Integration testing includes validating the secure exchange of sensitive data, such as personally identifiable information (PII), financial data, or healthcare records, between components. It ensures that data is encrypted, transmitted securely over networks, and decrypted by the intended recipients. By testing the encryption algorithms, key management practices, and secure transmission protocols, integration testing helps safeguard sensitive data from unauthorized access or interception.
6	Service-Level Agreements (SLAs)	Integration testing validates that the system meets the defined service-level agreements regarding response times, availability, and security. It ensures that integrated components perform within expected parameters and adhere to security requirements. By assessing the compliance of the integrated system with SLAs, integration testing helps identify potential performance bottlenecks and security vulnerabilities that may impact the overall user experience.

#	Name	Description
7	Dependency Management	Integration testing includes assessing the security of dependencies or third-party components used within the system. It verifies that these dependencies are up-to-date, free from known vulnerabilities, and comply with security standards. By identifying and addressing security weaknesses in third-party components, integration testing helps mitigate the risk of exploiting vulnerabilities introduced through external dependencies.
8	Test Environment Security	Integration testing considers the security of the test environment itself, including the servers, networks, and infrastructure used for conducting the tests. It ensures that the test environment is isolated, properly configured, and protected from unauthorized access. By maintaining a secure test environment, integration testing helps prevent potential security breaches during testing that could compromise the integrity of the system.
9	Continuous Integration and Deployment	Integration testing is seamlessly integrated into the continuous integration and deployment pipelines of DevSecOps. It ensures that security testing is performed throughout the development lifecycle, from small-scale integration tests during development iterations to larger-scale integration tests before deployment. By automating integration testing and including it in the overall continuous testing process, DevSecOps teams can detect and address security issues early, reducing the risk of vulnerabilities reaching production environments.

11.3.9.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Integration testing does not directly address the SAMM Security by Governance domain.
2	SAMM Security by Design	Integration testing contributes to the SAMM Security by Design domain by ensuring the secure integration of different software components, validating interface security, data flow, and compatibility. It helps address the following SAMM sub-domains: Security Requirements: Integration testing validates the secure interaction and data exchange between components, addressing security requirements related to integration. Secure Architecture: Integration testing verifies the secure integration of components, ensuring the architecture is resilient to integration-related vulnerabilities. Secure Coding Practices: Integration testing helps identify and prevent integration-related vulnerabilities that can arise from insecure coding practices. Threat Modeling: Integration testing considers the potential security risks and threats associated with component integration, contributing to threat modeling efforts.

#	Name	Description
3	SAMM Security by Implementation	Integration testing contributes to the SAMM Security by Implementation domain by ensuring the secure integration of software components. It helps address the following SAMM sub-domains: Secure Development Training: Integration testing helps validate secure coding practices and reinforces the importance of secure integration during development training. Secure Architecture: Integration testing ensures that the integrated components adhere to secure architectural principles and design patterns. Secure Coding Guidelines: Integration testing helps identify and address integration-related vulnerabilities, aligning with secure coding guidelines. Security Testing Integration: Integration testing is a form of security testing that detects and addresses integration-related security vulnerabilities. Security Verification: Integration testing validates the effectiveness of security controls in integrated software components. Security Architecture Review: Integration testing contributes to evaluating the security architecture by assessing the secure integration of components.
4	SAMM Security by Verification	Integration testing contributes to the SAMM Security by Verification domain by verifying the security of software components during integration. It helps address the following SAMM sub-domains: Security Testing: Integration testing is a form of security testing that ensures the secure integration of components. Code Review: Integration testing helps identify integration-related security flaws and adherence to secure coding practices. Security Architecture Review: Integration testing contributes to assessing the security architecture of integrated software components. Security Requirements Verification: Integration testing validates the secure interaction and adherence to security requirements during integration.
5	SAMM Security by Operations	Detect security vulnerabilities: Integration testing helps ensure the secure operation of software systems by validating the integration and compatibility of components within the operational environment. It helps detect security vulnerabilities that may arise from improper configuration or deployment practices, supporting secure deployment and operations. Validate the effectiveness of security measures: By testing the integration of components in different operational scenarios, it contributes to environment hardening, secure build practices, configuration management, vulnerability management, and incident management, all of which are key components of SAMM's Security by Operations domain. It helps validate the effectiveness of security measures during software operations.

11.3.9.2 Assessment – OWASP Top 10

Integration testing can address several OWASP threats; however, it is not a comprehensive solution. Other security practices and capabilities should be implemented in conjunction with integration testing to provide a robust defense against a wide range of security vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	Integration testing helps ensure that access controls are properly implemented and enforced across different software components. By testing the interactions between components, integration testing can identify access control flaws that may lead to unauthorized access or privilege escalation.
2	Cryptographic Failures (A02-2021)	It does not address this domain directly.
3	Injection (A03-2021)	Integration testing plays a role in preventing injection vulnerabilities by validating input and output data at each integration point. By ensuring that data is correctly transformed, validated, and sanitized, integration testing helps identify potential injection attacks and data tampering.
4	Insecure Design (A04-2021)	Integration testing contributes to addressing insecure design vulnerabilities by focusing on the interactions between different components. By verifying the compatibility and interoperability of components, integration testing helps identify potential security weaknesses in the overall design and architecture of the system.
5	Security Misconfiguration (A05-2021)	Integration testing helps identify misconfigurations that may arise due to the integration of different components. By validating the secure exchange of data, verifying interface security, and assessing the compliance of the integrated system with service-level agreements, integration testing can detect security misconfigurations and improve the overall security posture.
6	Vulnerable and Outdated Components (A06-2021)	Integration testing includes assessing the security of dependencies or third-party components used within the system. By verifying that these components are up-to-date and free from known vulnerabilities, integration testing helps mitigate the risk of using vulnerable or outdated components.
7	Identification and Authentication Failures (A07-2021)	Integration testing can contribute to addressing identification and authentication failures by verifying the secure exchange of authentication information and validating the authentication mechanisms between components. By testing the interactions related to user identification and authentication, integration testing helps ensure the security of the overall system.
8	Software and Data Integrity Failures (A08-2021)	Integration testing helps ensure the integrity of software components, configurations, and data within web applications. By validating the flow of data between components and verifying data transformation processes, integration testing helps detect potential vulnerabilities that may compromise the integrity of the system.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Integration testing can indirectly contribute to addressing security logging and monitoring failures by testing the interactions related to logging and monitoring mechanisms between components. By verifying the collection, analysis, and retention of security-related logs, integration testing helps improve the effectiveness of logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	It does not address this domain directly.

11.3.10 Security Integration Testing

Security Integration Testing is an essential security capability within the DevSecOps framework that focuses on testing the integration of security controls and mechanisms to ensure their effectiveness and interoperability within the system. It plays a vital role in validating the proper functioning of security measures and identifying any gaps or weaknesses in their implementation.

By incorporating robust Security Integration Testing into the DevSecOps approach, organizations can validate the effectiveness of their security controls, identify and mitigate vulnerabilities, and ensure compliance with industry standards and regulations. This capability plays a crucial role in building secure and resilient systems while maintaining a proactive approach to security.

Here are key aspects of the Security Integration Testing capability within DevSecOps:

#	Name	Description
1	Integration of Security Controls	Security Integration Testing involves verifying that the implemented security controls have been effectively integrated into the system. This includes authentication mechanisms, access controls, encryption protocols, and other security measures. The testing ensures that these controls work together seamlessly to provide a robust defense against potential threats.
2	Interoperability Testing	Security controls often interact with other components and systems within the environment. Interoperability testing focuses on verifying that the security controls can effectively communicate and collaborate with other system components without causing conflicts or disruptions. This testing ensures that security measures do not hinder the system's functionality or performance.
3	Effectiveness of Security Controls	Security Integration Testing assesses the effectiveness of security controls in mitigating known vulnerabilities and addressing security requirements. It involves validating that the implemented controls adequately protect against common attack vectors and adhere to industry best practices and security standards. By evaluating their effectiveness, organizations can identify any weaknesses or gaps in the security controls' capabilities and make necessary adjustments or enhancements.

#	Name	Description
4	End-to-End Security Scenarios	Security Integration Testing involves testing security measures in real-world scenarios that simulate potential attack vectors and threat scenarios. It aims to assess the system's resilience against various security threats, such as unauthorized access, injection attacks, cross-site scripting, and data breaches. By testing end-to-end security scenarios, organizations can evaluate the effectiveness of their security controls in mitigating different types of attacks.
5	Compliance and Standards Verification	Security Integration Testing ensures that the implemented security controls comply with relevant regulatory requirements and industry standards. It involves validating that the system meets the necessary security certifications, such as ISO 27001, PCI DSS, HIPAA, or GDPR. By verifying compliance with these standards, organizations demonstrate their commitment to security and ensure adherence to legal and industry-specific security obligations.
6		Security Integration Testing includes conducting vulnerability and threat testing to identify weaknesses or vulnerabilities in the system's security controls. This testing involves techniques such as penetration testing, vulnerability scanning, and security assessment to uncover potential security flaws or misconfigurations. By proactively identifying vulnerabilities, organizations can address them before they can be exploited by attackers.
7	Vulnerability and Threat Testing	Security Control Optimization: Through Security Integration Testing, organizations can optimize their security controls based on the feedback and insights gained during the testing process. This may involve refining configuration settings, updating control parameters, or fine-tuning control implementations to enhance their effectiveness and efficiency. Regular testing and optimization ensure that security controls remain up-to-date and resilient against emerging threats.
8	Test Environment Security	Security Integration Testing also includes ensuring the security of the testing environment itself. This involves implementing appropriate security measures, such as network segmentation, access controls, and encryption, to protect sensitive data and prevent unauthorized access. A secure testing environment helps maintain the integrity of the testing process and ensures that the results accurately reflect the system's security capabilities.

11.3.10.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Security Integration Testing helps validate that the implemented security controls align with the organization's policies and standards for software security. Risk Management: Security Integration Testing identifies and mitigates vulnerabilities, contributing to the organization's risk management efforts in software security. Compliance Management: Security Integration Testing ensures that the implemented security controls comply with regulatory requirements and industry standards, supporting the organization's compliance management efforts.

#	Name	Description
		Security Training and Awareness: Security Integration Testing helps raise awareness among employees and stakeholders about the importance of security controls and practices through testing activities and their outcomes. Security Metrics and Reporting: Security Integration Testing generates metrics and reports on the effectiveness of security controls, providing valuable insights for security governance and decision-making.
2	SAMM Security by Design	Security Requirements: Security Integration Testing validates the integration of security requirements into the system design by testing the effectiveness of security controls in mitigating vulnerabilities associated with design flaws. Secure Architecture: Security Integration Testing verifies the interoperability and effectiveness of security controls within the architecture, ensuring they are designed and implemented securely. Secure Coding Practices: Security Integration Testing validates the effectiveness of secure coding practices by testing the integration and functionality of security controls implemented during the coding phase. Threat Modeling: Security Integration Testing helps validate the effectiveness of security controls in addressing identified threats and risks through testing real-world attack scenarios.
3	SAMM Security by Implementation	Secure Development Training: Security Integration Testing supports secure development training efforts by validating the correct implementation and integration of security controls during the development process. Security Testing Integration: Security Integration Testing ensures that security testing activities, including integration testing, are effectively integrated into the software development life cycle. Security Verification: Security Integration Testing helps validate the effectiveness of security controls implemented during the development phase by testing their functionality and interoperability. Security Operations Integration: Security Integration Testing ensures that security controls implemented during the development phase are compatible with operational processes and contribute to the overall security of the
4	SAMM Security by Verification	Security Testing: Security Integration Testing is a key aspect of security testing, verifying the effectiveness of security controls and identifying vulnerabilities and weaknesses in software applications. Code Review: Security Integration Testing involves code review activities to identify security flaws and vulnerabilities, supporting the verification of software security.
5	SAMM Security by Operations	Environment Hardening: Security Integration Testing validates the effectiveness of environment hardening measures by testing the security controls and configurations implemented in the operational environment. Secure Build: Security Integration Testing verifies the secure build practices implemented during the software development process to ensure the integrity and security of the deployed software.

#	Name	Description
		Configuration Management: Security Integration Testing validates the proper implementation and management of software configurations, ensuring that unauthorized changes or misconfigurations are detected and addressed. Vulnerability Management: Security Integration Testing identifies vulnerabilities through various testing techniques, contributing to the organization's vulnerability management efforts. Incident Management: Security Integration Testing helps identify and mitigate vulnerabilities, reducing the likelihood and impact of security incidents, and supporting incident management processes.

11.3.10.2 Assessment – OWASP Top 10

Security Integration Testing capability directly addresses multiple OWASP threats by validating the integration and effectiveness of security controls, verifying secure configurations, testing for vulnerabilities, and ensuring adherence to secure design principles and cryptographic practices.

#	Name	Description
1	Broken Access Control (A01-2021)	By conducting Security Integration Testing, organizations can verify that the implemented security controls, such as authentication mechanisms and access controls, have been effectively integrated into the system. This testing ensures that access controls are properly enforced and reduces the risk of unauthorized access.
2	Cryptographic Failures (A02-2021)	Security Integration Testing involves assessing the effectiveness of cryptographic algorithms and mechanisms within a web application. By validating the proper implementation of cryptography and identifying any weaknesses or mistakes, organizations can mitigate vulnerabilities related to cryptographic failures.
3	Injection (A03-2021)	Security Integration Testing includes validating input validation and handling mechanisms. By thoroughly testing the application for injection vulnerabilities, organizations can ensure that untrusted data is properly handled, preventing attackers from injecting malicious code or unintended commands.
4	Insecure Design (A04-2021)	Security Integration Testing plays a role in identifying flaws and weaknesses in the overall design and architecture of a web application. By testing security measures in real-world scenarios, organizations can uncover insecure design decisions and address them to mitigate the risk of exploitation.
5	Security Misconfiguration (A05-2021)	Security Integration Testing involves verifying the configuration of web applications and associated components. By conducting comprehensive testing and ensuring secure configurations, organizations can reduce the risk of security misconfigurations that may lead to unauthorized access or exposure of sensitive information.
6	Vulnerable and Outdated Components (A06-2021)	Security Integration Testing includes assessing the security of third-party software components used within the web application. By actively managing software dependencies and regularly testing for vulnerabilities, organizations can identify and address any vulnerable or outdated components that may introduce security risks.

#	Name	Description
7	Identification and Authentication Failures (A07-2021)	Security Integration Testing assesses the effectiveness of identification and authentication mechanisms. By validating the proper implementation of secure authentication mechanisms and testing for vulnerabilities, organizations can reduce the risk of unauthorized access and impersonation attacks.
8	Software and Data Integrity Failures (A08-2021)	Security Integration Testing involves validating the integrity and protection of software components, configurations, and data within web applications. By testing for vulnerabilities that could compromise the integrity of software and data, organizations can enhance the overall security posture of their applications.
9	Security Logging and Monitoring Failures (A09-2021)	Security Integration Testing includes verifying the logging and monitoring capabilities of web applications. By assessing the collection, analysis, and retention of security-related logs and the effectiveness of monitoring mechanisms, organizations can identify and address any failures in logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	Security Integration Testing can help identify vulnerabilities related to Server-Side Request Forgery (SSRF) by validating input validation mechanisms and secure configurations. By conducting thorough testing and implementing appropriate security measures, organizations can mitigate the risk of SSRF attacks.

11.3.11 Dread Testing

DREAD testing is an important security capability within the DevSecOps framework. It involves evaluating the potential impact of identified security vulnerabilities by considering factors such as damage, reproducibility, exploitability, and affected users. The purpose of DREAD testing is to assess and prioritize security issues based on their potential impact, allowing organizations to allocate their resources effectively and address the most critical vulnerabilities first.

DREAD testing is a valuable security capability in DevSecOps that allows organizations to evaluate and prioritize vulnerabilities based on their potential impact. By considering factors such as damage, reproducibility, exploitability, and affected users, organizations can allocate their resources effectively, develop risk mitigation plans, and continuously improve their security posture. This helps to proactively address the most critical vulnerabilities and minimize the overall risk to the organization's systems and data.

#	Name	Description
1	Damage Assessment	DREAD testing considers the potential damage that could result from a security vulnerability. This includes evaluating the impact on confidentiality, integrity, and availability of data or systems. By assessing the potential consequences of a vulnerability, organizations can prioritize their efforts to address those vulnerabilities that pose the greatest risk to the business.

#	Name	Description
2	Reproducibility Analysis	DREAD testing assesses the ease with which a vulnerability can be reproduced. By determining the level of effort required to exploit the vulnerability consistently, organizations can gauge the likelihood of it being exploited in real-world scenarios. Vulnerabilities that are easily reproducible may pose a higher risk and require immediate attention.
3	Exploitability Evaluation	DREAD testing examines the ease with which a vulnerability can be exploited. It considers factors such as the required technical expertise, the availability of exploit tools or techniques, and the likelihood of an attacker successfully leveraging the vulnerability. By understanding the exploitability of a vulnerability, organizations can prioritize remediation efforts accordingly.
4	Affected Users Analysis	DREAD testing takes into account the number and type of users who may be impacted by a vulnerability. This includes evaluating the potential impact on end-users, customers, or other stakeholders. Vulnerabilities that affect a large number of users or critical systems may require immediate attention to minimize the potential harm.
5	Risk Prioritization	DREAD testing helps organizations prioritize security issues based on their potential impact. By assigning a numerical score or rating to each factor (damage, reproducibility, exploitability, and affected users), vulnerabilities can be ranked according to their overall risk level. This prioritization allows organizations to focus their resources on addressing the most critical vulnerabilities first, ensuring efficient allocation of time and effort.
6	Resource Allocation	DREAD testing assists in the effective allocation of resources for vulnerability remediation. By identifying and prioritizing high-risk vulnerabilities, organizations can allocate their security resources, including personnel, tools, and time, to address the most critical issues promptly. This approach ensures that limited resources are utilized effectively and that the highest-risk vulnerabilities are mitigated in a timely manner.
7	Risk Mitigation Planning	DREAD testing provides valuable insights for developing risk mitigation plans. Once vulnerabilities are prioritized based on their potential impact, organizations can create action plans to address each vulnerability systematically. These plans may involve implementing security controls, applying patches or updates, improving coding practices, or conducting additional security testing. By having a structured approach to mitigating risks, organizations can effectively reduce the overall security risk landscape.
8	Continuous Monitoring and Improvement	DREAD testing is an ongoing process within DevSecOps. As new vulnerabilities are discovered or changes are made to the system, DREAD testing should be performed regularly to assess their potential impact and prioritize remediation efforts. This iterative approach ensures that the security posture of the system is continuously monitored and improved, reducing the likelihood of critical vulnerabilities going unnoticed or unaddressed.

11.3.11.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Dread Testing contributes to the Security by Governance domain in SAMM by providing a method to assess and prioritize security vulnerabilities based on their potential impact. This helps in establishing effective decision-making and risk management processes within the security governance framework.
2	SAMM Security by Design	Dread Testing indirectly helps address the SAMM Security by Design domain by enabling organizations to prioritize and address vulnerabilities during the design phase of software development. By considering the potential impact, exploitability, and affected users, organizations can incorporate security controls and considerations into the design to mitigate various OWASP threats.
3	SAMM Security by Implementation	Dread Testing indirectly helps address the SAMM Security by Implementation domain by enabling organizations to identify vulnerabilities early in the development process. By incorporating security testing activities, such as static code analysis and security code reviews, organizations can detect and address vulnerabilities related to OWASP threats, ensuring that secure coding practices and security requirements are implemented.
4	SAMM Security by Verification	Dread Testing directly contributes to the SAMM Security by Verification domain by providing a method to validate and verify the security of software applications. By conducting security testing, code reviews, and threat modeling exercises, organizations can identify vulnerabilities and weaknesses, ensuring the effectiveness of security controls and compliance with security requirements.
5	SAMM Security by Operations	Dread Testing indirectly helps address the SAMM Security by Operations domain by enabling organizations to continuously monitor and improve the security posture of software systems. By conducting regular DREAD testing and security testing activities, organizations can detect and address vulnerabilities, apply patches and updates, and effectively manage security incidents, contributing to the secure and reliable operation of software systems.

11.3.11.2 Assessment – OWASP Top 10

The capability of DREAD testing within the DevSecOps framework directly addresses several OWASP threats by providing a structured approach to assessing and prioritizing vulnerabilities based on their potential impact.

Here's how DREAD testing contributes to addressing each OWASP threat:

#	Name	Description
1	Broken Access Control (A01-2021)	DREAD testing indirectly addresses this threat by helping organizations prioritize vulnerabilities that can lead to unauthorized access. Vulnerabilities related to access controls, if assessed with a high DREAD score due to their potential impact, will receive more attention and resources for remediation.

#	Name	Description
2	Cryptographic Failures (A02-2021)	DREAD testing does not directly address cryptographic failures, but it can be adapted to assess the potential impact of vulnerabilities related to cryptography, such as vulnerabilities in encryption algorithms or key management. These vulnerabilities can be prioritized based on their DREAD scores for remediation.
3	Injection (A03-2021)	DREAD testing indirectly addresses injection vulnerabilities by evaluating the potential damage, reproducibility, and exploitability of such vulnerabilities. High DREAD scores for injection vulnerabilities would signal their criticality and the need for immediate attention.
4	Insecure Design (A04-2021)	DREAD testing contributes to addressing insecure design by prioritizing vulnerabilities stemming from design flaws. Vulnerabilities resulting from insecure design decisions can receive high DREAD scores, prompting organizations to focus on them during remediation.
5	Security Misconfiguration (A05-2021)	DREAD testing indirectly addresses security misconfiguration by evaluating the potential impact and exploitability of misconfigurations. Misconfigurations with high DREAD scores indicate a higher risk level, leading to quicker remediation efforts.
6	Vulnerable and Outdated Components (A06-2021)	DREAD testing can be used to assess the potential impact of vulnerabilities arising from the use of outdated or vulnerable components. Components with high DREAD scores should be promptly addressed to reduce the risk associated with them.
7	Identification and Authentication Failures (A07-2021)	DREAD testing does not directly address identification and authentication failures. However, it can indirectly contribute by prioritizing vulnerabilities that could lead to authentication bypass or unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	DREAD testing helps organizations evaluate vulnerabilities that may compromise software and data integrity. Vulnerabilities affecting integrity can receive high DREAD scores, indicating the need for immediate mitigation.
9	Security Logging and Monitoring Failures (A09-2021)	DREAD testing indirectly addresses this threat by prioritizing vulnerabilities that may affect security logging and monitoring capabilities. Vulnerabilities impacting these areas can be assigned higher DREAD scores to ensure timely remediation.
10	Server-Side Request Forgery (A10-2021)	DREAD testing indirectly addresses SSRF by assessing the potential impact, exploitability, and reproducibility of vulnerabilities related to SSRF. High DREAD scores for SSRF vulnerabilities would signal their criticality and the need for immediate attention.

11.3.12 Software License Validation

Software License Validation is a critical security capability in DevSecOps that focuses on ensuring compliance with software licensing requirements and identifying any license-related risks or violations. By validating software licenses, organizations can prevent legal and security issues that may arise from the use of unlicensed or improperly licensed software components.

By integrating Software License Validation into the DevSecOps workflow, organizations can effectively manage software licensing compliance and mitigate associated risks. This capability not only helps ensure

legal and regulatory compliance but also contributes to maintaining the security and integrity of the software development process.

Here are some key aspects of Software License Validation within the DevSecOps framework:

#	Name	Description
1	License Compliance	Software License Validation involves verifying that the software components used in the development process comply with the licensing terms and conditions specified by the software vendors or the open-source licenses being utilized. It ensures that the organization is legally authorized to use the software and helps prevent potential legal consequences and penalties.
2	License Risk Identification	This capability helps identify any license-related risks or violations associated with the software being used. It includes detecting and flagging instances where the organization may be using software components that have expired licenses, using licenses that are incompatible with the intended use, or violating any license restrictions. By identifying these risks, organizations can take corrective actions to mitigate them and avoid potential legal and security consequences.
3	Compliance Monitoring	Software License Validation also involves continuous monitoring of software licenses throughout the development lifecycle. It ensures that any changes or updates to the software components, such as new releases or patches, are assessed for license compliance. By actively monitoring licenses, organizations can proactively address any licensing issues and maintain compliance with the applicable licensing requirements.
4	Risk Mitigation and Remediation	In the event of license-related risks or violations, Software License Validation enables organizations to take appropriate measures for risk mitigation and remediation. This may involve actions such as acquiring valid licenses, replacing or removing unlicensed software components, or seeking legal advice to resolve licensing issues. By promptly addressing these concerns, organizations can prevent potential legal disputes, security vulnerabilities, or reputational damage.
5	Tooling and Automation	To streamline the Software License Validation process, organizations can leverage various tools and automation techniques. These tools can scan software components, including third-party libraries and open-source software, to identify their associated licenses and check for any violations or risks. Automation helps in reducing manual efforts and ensuring consistent license validation across different software projects.

11.3.12.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Software License Validation contributes to the establishment and implementation of policies and standards related to software licensing compliance, ensuring that organizations have clear expectations and requirements for software security.

#	Name	Description
		Risk Management: Software License Validation helps identify and mitigate risks associated with software licensing, ensuring that organizations have appropriate risk management processes in place to address licensing-related risks. Compliance Management: Software License Validation ensures compliance with software licensing requirements, supporting organizations in meeting regulatory requirements and industry standards related to software security. Security Training and Awareness: Software License Validation raises awareness among employees and stakeholders about the importance of software licensing compliance and its impact on security, contributing to the overall security training and awareness efforts within the organization. Security Metrics and Reporting: Software License Validation involves monitoring and reporting on software license compliance, providing key security metrics related to software licensing that can be used to assess and improve governance practices. Security Roles and Responsibilities: Software License Validation helps define and assign roles and responsibilities related to software licensing compliance, ensuring accountability for maintaining the security and integrity of software development processes.
2	SAMM Security by Design	Security Requirements: Software License Validation, although not directly related to addressing individual OWASP threats, contributes to the overall security requirements by ensuring compliance with licensing terms and conditions, which indirectly helps mitigate vulnerabilities associated with OWASP threats. Secure Architecture: Software License Validation, as a security capability, helps establish a strong foundation for secure architecture by verifying that the software components used in the development process comply with licensing requirements, minimizing potential vulnerabilities and risks. Secure Coding Practices: While Software License Validation does not directly address secure coding practices, it complements the security by design domain by ensuring that software components used in development are properly licensed, allowing developers to focus on secure coding practices without violating licensing restrictions. Threat Modeling: Software License Validation does not directly contribute to threat modeling activities but supports the overall security by design efforts by ensuring compliance with licensing requirements and minimizing licensing-related risks. Security Testing: Software License Validation is not directly involved in security testing activities, but it helps create a solid foundation for security testing by ensuring that software components used in testing are properly licensed and compliant with licensing terms and conditions.
3	SAMM Security by Implementation	Secure Development Training: Software License Validation, although not directly related to development training, contributes to the overall security training efforts by raising awareness about software licensing compliance and its impact on secure development practices.

#	Name	Description
		Secure Architecture: Software License Validation, as a security capability, helps establish a secure architecture by ensuring that the software components used in development are properly licensed, minimizing potential vulnerabilities and risks. Secure Coding Guidelines: Software License Validation, while not directly related to secure coding guidelines, complements secure coding practices by verifying that the software components used in development are compliant with licensing requirements, allowing developers to focus on secure coding without violating licensing restrictions. Security Requirements: Software License Validation contributes to implementing security requirements by ensuring compliance with licensing terms and conditions, which indirectly helps address security objectives related to software security. Security Testing Integration: Software License Validation is not directly involved in security testing activities but supports the integration of security testing by ensuring that software components used in testing are properly licensed and compliant with licensing terms and conditions. Security Verification: Software License Validation helps establish the foundation for security verification by ensuring compliance with licensing requirements, allowing organizations to focus on verifying the effectiveness of security controls implemented in the software system. Security Architecture Review: Software License Validation does not directly contribute to security architecture review activities but supports the overall security by implementation efforts by ensuring compliance with licensing requirements and minimizing licensing-related risks. Security Operations Integration: Software License Validation helps integrate security considerations into operational processes by ensuring compliance with licensing requirements and minimizing the risk of security incidents related to unlicensed or improperly licensed software components.
4	SAMM Security by Verification	Policies and Standards: Software License Validation contributes to this domain by ensuring compliance with software licensing policies and standards. It verifies that the software components used in the development process comply with the licensing terms and conditions specified by software vendors or open-source licenses. Risk Management: Software License Validation helps identify and mitigate license-related risks or violations. By validating software licenses, organizations can assess the risk associated with using unlicensed or improperly licensed software components. Compliance Management: Software License Validation ensures compliance with software licensing requirements. It helps organizations monitor and manage license compliance throughout the development lifecycle, contributing to compliance management activities. Security Testing: The Security by Verification domain involves conducting security testing to identify vulnerabilities in software applications. Software License Validation, although not directly related to security testing, contributes to the overall security verification process by ensuring compliance with licensing requirements, which indirectly enhances the security posture of the software.

#	Name	Description
		Code Review: While Software License Validation is not directly related to code review, it can indirectly contribute to code review practices by identifying potential license-related issues in the software components used during development.
5	SAMM Security by Operations	Environment Hardening: Software License Validation does not directly address environment hardening in the Security by Operations domain. Secure Build: Software License Validation does not directly address secure build practices, as its focus is on software licensing compliance rather than the actual build process. Configuration Management: Software License Validation does not directly address configuration management activities within the Security by Operations domain. Vulnerability Management: Software License Validation does not directly address vulnerability management practices. It focuses on ensuring software licensing compliance rather than vulnerability identification and mitigation. Incident Management: Software License Validation does not directly address incident management activities within the Security by Operations domain. Secure Deployment: Software License Validation does not directly address secure deployment practices, as its primary focus is on software licensing compliance. Security Testing: Software License Validation does not directly address security testing activities within the Security by Operations domain. Operational Enablement: Software License Validation does not directly address operational enablement practices, as it primarily focuses on software licensing compliance.

11.3.12.2 Assessment – OWASP Top 10

Software License Validation does not directly address the OWASP threats mentioned above. It primarily focuses on ensuring compliance with software licensing requirements and identifying license-related risks or violations. While it indirectly contributes to the mitigation of some threats, such as Vulnerable and Outdated Components, it is important to implement additional security measures to address the specific vulnerabilities associated with each threat.

#	Name	Description
1	Broken Access Control (A01-2021)	Software License Validation does not directly address the Broken Access Control threat. While it focuses on ensuring compliance with software licensing requirements, it does not specifically address flaws in the enforcement of access controls within a web application.

#	Name	Description
2	Cryptographic Failures (A02-2021)	Software License Validation does not directly address the Cryptographic Failures threat. It focuses on validating software licenses and ensuring compliance, but it does not specifically deal with cryptographic vulnerabilities or weaknesses in the implementation of cryptographic algorithms.
3	Injection (A03-2021)	Software License Validation does not directly address the Injection threat. It is unrelated to the handling of untrusted data and the prevention of injection attacks in web applications.
4	Insecure Design (A04-2021)	Software License Validation does not directly address the Insecure Design threat. It focuses on license compliance and risk identification related to software licensing, rather than flaws in the overall design and architecture of web applications.
5	Security Misconfiguration (A05-2021)	Software License Validation does not directly address the Security Misconfiguration threat. It focuses on validating software licenses and ensuring compliance, but it does not specifically deal with insecure configurations or misconfigurations of web applications.
6	Vulnerable and Outdated Components (A06-2021)	Software License Validation indirectly addresses the Vulnerable and Outdated Components threat. By validating software licenses, organizations can identify outdated or vulnerable components and take appropriate actions to mitigate the risks associated with them. However, it is important to note that Software License Validation alone is not sufficient to address all aspects of this threat. Additional measures, such as actively managing software dependencies and applying security patches, are necessary.
7	Identification and Authentication Failures (A07-2021)	Software License Validation does not directly address the Identification and Authentication Failures threat. It is unrelated to weaknesses in the mechanisms used for user identification and authentication in web applications.
8	Software and Data Integrity Failures (A08-2021)	Software License Validation does not directly address the Software and Data Integrity Failures threat. It focuses on validating software licenses and ensuring compliance, but it does not specifically deal with vulnerabilities related to the integrity of software components, configurations, or data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	Software License Validation does not directly address the Security Logging and Monitoring Failures threat. It is unrelated to deficiencies in logging and monitoring practices within web applications.
10	Server-Side Request Forgery (A10-2021)	Software License Validation does not directly address the Server-Side Request Forgery threat. It focuses on software license compliance and risk identification, rather than vulnerabilities related to making unintended requests to internal or external resources on the server-side.

11.3.13 Libraries Assessment

Libraries Assessment is an important security capability in the DevSecOps framework. It involves assessing the third-party libraries used in an application for security vulnerabilities and known issues. By conducting a thorough assessment of these libraries, organizations can identify and address potential vulnerabilities that could pose security risks to the application.

By integrating Libraries Assessment into the DevSecOps practices, organizations can proactively identify and address security vulnerabilities in third-party libraries. This capability helps protect the application from potential risks and enhances the overall security posture of the software development lifecycle.

Here are some key aspects of Libraries Assessment within the DevSecOps context:

#	Name	Description
1	Library Identification	The first step in Libraries Assessment is identifying and documenting all the third-party libraries used in the application. This includes libraries and frameworks that are directly integrated into the codebase, as well as those indirectly included through other dependencies. Maintaining an up-to-date inventory of libraries is crucial for effective security management.
2	Vulnerability Scanning	Once the libraries are identified, vulnerability scanning tools and techniques can be applied to assess them for known security vulnerabilities. These tools leverage databases of known vulnerabilities, such as the National Vulnerability Database (NVD), to check if any libraries have known security issues. The scanning process may involve analyzing library versions, configurations, and dependencies to identify potential vulnerabilities.
3	Patch and Upgrade Analysis	Libraries Assessment also involves analyzing the available patches and updates for the identified libraries. This step helps determine if any security vulnerabilities have been addressed in newer versions of the libraries. Organizations can then assess the impact of upgrading to the latest secure version and make informed decisions on patching or upgrading vulnerable libraries.
4	Security Risk Prioritization	Libraries Assessment helps in prioritizing security risks based on the severity and impact of identified vulnerabilities. By categorizing vulnerabilities and their potential consequences, organizations can allocate resources effectively and focus on mitigating high-priority risks that pose the most significant security threats.
5	Remediation and Mitigation	Once vulnerabilities are identified, appropriate remediation steps can be taken to mitigate the risks. This may involve applying patches, updating libraries to secure versions, or seeking alternative libraries that do not have the identified vulnerabilities. The remediation process should be integrated into the overall development workflow to ensure timely fixes.
6	Continuous Monitoring	Libraries Assessment is an ongoing process that should be incorporated into the DevSecOps pipeline. As new vulnerabilities are discovered and patches are released, continuous monitoring helps ensure that libraries remain secure and up to date. Regular scanning and assessment of libraries help organizations maintain a strong security posture and reduce the risk of exploitation through known vulnerabilities.

11.3.13.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Libraries Assessment contributes to the Security by Governance domain by enabling organizations to assess the security of third-party libraries used in software development. This helps organizations establish and enforce policies and standards related to secure coding practices, vulnerability management, and data protection (Policies and Standards activity). By identifying and addressing potential vulnerabilities in third-party libraries, Libraries Assessment supports risk management activities by reducing the risk associated with insecure software components (Risk Management activity). The continuous monitoring of libraries and the integration of Libraries Assessment into the DevSecOps pipeline provide visibility into the status of software security. This supports security metrics and reporting, helping organizations assess and report on the effectiveness of their software security practices (Security Metrics and Reporting activity). Libraries Assessment helps organizations ensure compliance with relevant regulatory requirements and industry standards by addressing security vulnerabilities in third-party libraries (Compliance Management activity). The integration of Libraries Assessment into the DevSecOps practices includes security training and awareness for developers and stakeholders. It promotes a culture of security throughout the organization, contributing to security training and awareness activities (Security Training and Awareness activity). Libraries Assessment also helps define security roles and responsibilities by identifying the individuals responsible for assessing and addressing library vulnerabilities. It supports establishing accountability for software security and ensuring that individuals have the necessary knowledge and skills to fulfill their roles effectively (Security Roles and Responsibilities activity).
2	SAMM Security by Design	Libraries Assessment indirectly contributes to the Security by Design domain by providing a mechanism to assess and address security vulnerabilities in third-party libraries used in software development. This aligns with the Secure Architecture activity, as it helps organizations select libraries and frameworks that align with secure design principles. The identification and analysis of library vulnerabilities through Libraries Assessment contribute to secure coding practices by identifying potential vulnerabilities and helping developers prevent common coding mistakes and vulnerabilities associated with OWASP threats. Libraries Assessment supports the threat modeling activity by identifying potential security risks associated with the use of third-party libraries and guiding the implementation of appropriate security controls. By integrating Libraries Assessment into the DevSecOps pipeline, organizations can include security testing activities in the design phase. This helps identify and address vulnerabilities related to OWASP threats, ensuring that the design is resilient to potential attacks.
3	SAMM Security by Implementation	Libraries Assessment contributes to the Secure Development Training activity by providing insights into the security risks and vulnerabilities associated with third-party libraries. This information can be incorporated into secure development training programs to educate developers on the potential risks and best practices for integrating secure libraries.

#	Name	Description
		By analyzing library vulnerabilities and applying appropriate remediation steps, Libraries Assessment helps ensure that secure architecture practices are implemented. It supports the establishment and adoption of secure coding guidelines and standards by promoting the use of secure versions of libraries and addressing security vulnerabilities. Libraries Assessment aligns with the Security Requirements activity by helping organizations identify security requirements related to third-party libraries. This information can be used to define and document security requirements and ensure their implementation during the development process. The integration of security testing, including Libraries Assessment, into the software development life cycle supports security testing integration and security verification activities. It helps identify and address security vulnerabilities throughout the development process, ensuring the implementation of effective security controls. By conducting security architecture reviews and evaluating the security of third-party libraries, Libraries Assessment contributes to security architecture review activities. It helps assess the security controls and identify potential vulnerabilities or weaknesses in the software system's design and architecture. The integration of Libraries Assessment into the DevSecOps pipeline supports the integration of security considerations into operational processes. By addressing library vulnerabilities, it helps ensure the secure deployment and configuration of software components.
4	SAMM Security by Verification	Libraries Assessment in DevSecOps contributes to the Security by Verification domain of SAMM. By assessing third-party libraries for security vulnerabilities and known issues, organizations can validate the security of their software applications and mitigate potential risks (SAMM Security by Verification: Security Testing, Code Review). Vulnerability scanning in Libraries Assessment helps identify and address security vulnerabilities in third-party libraries, contributing to the verification of software security (SAMM Security by Verification: Security Testing). Patch and upgrade analysis in Libraries Assessment enables organizations to verify if security vulnerabilities have been addressed in newer versions of libraries, ensuring the verification of software security (SAMM Security by Verification: Security Testing, Security Architecture Review). Security risk prioritization in Libraries Assessment allows organizations to prioritize the mitigation of high-priority vulnerabilities, contributing to the verification of software security (SAMM Security by Verification: Security Testing). Remediation and mitigation steps taken based on identified vulnerabilities in Libraries Assessment help verify and address security weaknesses in software applications (SAMM Security by Verification: Security Testing, Security Architecture Review). Continuous monitoring in Libraries Assessment, by regularly scanning and assessing libraries for vulnerabilities, contributes to the ongoing verification of software security (SAMM Security by Verification: Security Testing, Security Architecture Review, Security Requirements Verification, Secure Deployment Verification).
5	SAMM Security by Operations	The secure build practices in SAMM's Operations domain align with the secure build practices promoted in DevSecOps, including secure coding standards, secure build tools, and secure software development frameworks. These practices contribute to the secure operation of software systems (SAMM Security by Operations: Secure Build).

#	Name	Description
		Configuration management practices in SAMM's Operations domain, such as version control and change management, help ensure the integrity and security of software systems. These practices align with DevSecOps principles, contributing to the secure operation of software (SAMM Security by Operations: Configuration Management). Vulnerability management practices in SAMM's Operations domain, including vulnerability scanning and assessment, align with DevSecOps approaches to identify and address vulnerabilities in software systems. These practices contribute to the secure operation of software (SAMM Security by Operations: Vulnerability Management, Security Testing). Incident management practices in SAMM's Operations domain, such as incident detection, response, and recovery, align with DevSecOps incident management practices. By effectively handling security incidents, organizations can maintain the secure operation of software systems (SAMM Security by Operations: Incident Management). Secure deployment practices in SAMM's Operations domain, including secure software distribution and installation procedures, align with DevSecOps principles of secure deployment. These practices contribute to the secure operation of software (SAMM Security by Operations: Secure Deployment). Security testing activities recommended in SAMM's Operations domain, such as static code analysis and dynamic application security testing, align with DevSecOps practices of integrating security testing throughout the software development life cycle. These activities contribute to the secure operation of software systems (SAMM Security by Operations: Security Testing). Operational enablement practices in SAMM's Operations domain, such as security training for operational staff and documentation of operational procedures, align with DevSecOps principles of providing operational support and guidance. These practices contribute to the secure operation of software systems (SAMM Security by Operations: Operational Enablement).

11.3.13.2 Assessment – OWASP Top 10

Libraries Assessment does not directly address all OWASP threats; however, it significantly contributes to mitigating several threats by identifying and addressing vulnerabilities in third-party libraries used in the application. It helps organizations maintain a strong security posture and reduce the risk of exploitation through known vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	Libraries Assessment helps identify and address security vulnerabilities in third-party libraries that may affect access controls within a web application. By assessing the libraries for known issues, organizations can ensure that proper access controls are implemented and enforced, reducing the risk of unauthorized access.

#	Name	Description
2	Cryptographic Failures (A02-2021)	Libraries Assessment involves assessing the cryptographic libraries used in an application for vulnerabilities and weaknesses. By identifying and addressing these issues, organizations can ensure the proper implementation of cryptographic algorithms and mechanisms, reducing the risk of cryptographic failures.
3	Injection (A03-2021)	Libraries Assessment helps identify and assess libraries that might have vulnerabilities related to injection attacks. By analyzing library versions, configurations, and dependencies, organizations can identify potential weaknesses and take appropriate remediation steps to mitigate the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	Libraries Assessment contributes to addressing insecure design vulnerabilities by assessing the security posture of third-party libraries used in the application. By identifying and addressing vulnerabilities in these libraries, organizations can reduce the risk of insecure design decisions that could be exploited by attackers.
5	Security Misconfiguration (A05-2021)	Libraries Assessment helps identify misconfigurations and insecure settings in third-party libraries. By assessing the libraries for secure configurations and ensuring that the latest patches and updates are applied, organizations can reduce the risk of security misconfigurations within the application.
6	Vulnerable and Outdated Components (A06-2021)	Libraries Assessment directly addresses the threat of using vulnerable and outdated software components. By identifying and assessing the versions of third-party libraries used in the application, organizations can take appropriate actions such as applying patches, upgrading to secure versions, or seeking alternative libraries to mitigate the risk of using known vulnerable components.
7	Identification and Authentication Failures (A07-2021)	Libraries Assessment indirectly contributes to addressing identification and authentication failures by ensuring that the third-party libraries used in the application do not have vulnerabilities that could impact authentication mechanisms. By addressing vulnerabilities in these libraries, organizations can reduce the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	Libraries Assessment helps ensure the integrity of software components by identifying and addressing vulnerabilities in third-party libraries. By regularly assessing the libraries for security vulnerabilities, organizations can reduce the risk of unauthorized modifications or tampering of software and data within the application.
9	Security Logging and Monitoring Failures (A09-2021)	Libraries Assessment indirectly contributes to addressing security logging and monitoring failures by ensuring that the third-party libraries used in the application do not have vulnerabilities that could impact logging and monitoring capabilities. By addressing vulnerabilities in these libraries, organizations can improve the overall effectiveness of their logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	Libraries Assessment can indirectly contribute to addressing server-side request forgery vulnerabilities by assessing the libraries used in the application for vulnerabilities that could be exploited to bypass security controls and make unintended requests. By addressing these vulnerabilities, organizations can reduce the risk of server-side request forgery attacks.

11.3.14 Library Dependencies

Library Dependencies is a crucial security capability in the DevSecOps framework. It involves managing and monitoring the dependencies used in an application to ensure that they are up-to-date and do not introduce security vulnerabilities. By effectively managing dependencies, organizations can prevent the use of outdated or vulnerable components that can be exploited by attackers.

By incorporating Library Dependencies management into the DevSecOps practices, organizations can effectively mitigate the risks associated with outdated or vulnerable dependencies. This capability helps ensure that the application's software supply chain remains secure and reduces the likelihood of successful attacks targeting known vulnerabilities in the dependencies.

Here are key aspects of Library Dependencies within the DevSecOps context:

#	Name	Description
1	Dependency Management	The first step in managing library dependencies is establishing a robust process to track and manage the libraries used in the application. This includes maintaining an inventory of dependencies, their versions, and their sources. Organizations can utilize package managers or dependency management tools to simplify this process.
2	Version Control and Updates	Keeping dependencies up-to-date is crucial for security. Developers and DevOps teams need to regularly monitor for new versions or updates of dependencies and assess their impact on the application. This involves staying informed about security advisories and release notes from the library developers to identify any security-related updates or patches.
3	Dependency Analysis	Conducting a thorough analysis of dependencies is essential to identify potential security vulnerabilities. This analysis may involve utilizing vulnerability databases, security scanners, or static code analysis tools to check for known security issues in the dependency versions being used. By conducting this analysis, organizations can determine if any dependencies pose security risks and take appropriate actions.
4	Risk Assessment and Prioritization	Once vulnerabilities are identified, a risk assessment is performed to evaluate the potential impact and severity of each vulnerability. This assessment helps prioritize the remediation efforts based on the criticality of the dependencies and their associated vulnerabilities. High-risk dependencies that are more likely to be exploited or have severe consequences should be addressed with higher priority.
5	Patching and Remediation	When vulnerabilities are discovered in dependencies, it is crucial to apply patches or take necessary remediation steps. This may involve updating to the latest secure version of the dependency, patching the vulnerability manually, or seeking alternative dependencies that do not have the identified vulnerabilities. Automation and integration with the DevSecOps pipeline can help ensure timely patching and remediation.
6	Continuous Monitoring	Library Dependencies management is an ongoing process that requires continuous monitoring. As new vulnerabilities are discovered or updates are released, it is essential to regularly assess the dependencies and ensure that they remain secure. Continuous monitoring helps detect and address any new security issues that may arise in the dependencies used in the application.

11.3.14.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Library Dependencies contribute to the Security by Governance domain by enabling organizations to establish policies and standards for managing dependencies. By effectively managing library dependencies, organizations can ensure compliance with secure coding practices, vulnerability management, and incident response policies and standards. Library Dependencies also support risk management by helping organizations identify and assess risks associated with outdated or vulnerable dependencies. By keeping dependencies up-to-date and monitoring for security updates, organizations can mitigate the risk of exploitation and potential security incidents. Compliance management is enhanced through Library Dependencies management, as organizations can ensure compliance with relevant regulatory requirements and industry standards by addressing vulnerabilities in dependencies. Security training and awareness are supported by Library Dependencies management, as it enables organizations to educate employees and stakeholders about the importance of managing dependencies securely and the risks associated with outdated or vulnerable components. Library Dependencies contribute to security metrics and reporting by providing visibility into the status of software supply chain security. By monitoring dependencies and maintaining an inventory of their versions and sources, organizations can measure key security metrics and report on the effectiveness of dependency management practices. Regarding security roles and responsibilities, Library Dependencies management helps establish accountability for software security by assigning responsibilities for managing and monitoring dependencies effectively.
2	SAMM Security by Design	Library Dependencies contribute to the Security by Design domain by supporting the implementation of secure architecture practices. By managing and monitoring dependencies, organizations can ensure that secure architectural patterns and principles are followed, minimizing potential vulnerabilities and addressing OWASP threats related to insecure design and cryptographic failures. Library Dependencies management enables organizations to incorporate secure coding practices by regularly updating dependencies to versions that address known security issues. This helps mitigate OWASP threats such as injection, insecure design, and security misconfigurations. Dependency analysis is an important aspect of Library Dependencies management, which helps organizations conduct threat modeling exercises during the design phase. By analyzing dependencies for known security vulnerabilities, organizations can identify potential threats and prioritize security controls accordingly. Security testing activities in the design phase, such as static code analysis and security code reviews, are supported by Library Dependencies management. By ensuring that dependencies are secure and free from vulnerabilities, organizations can verify the effectiveness of security controls and countermeasures during the design phase.

#	Name	Description
3	SAMM Security by Implementation	Library Dependencies contribute to the Implementation domain by enabling organizations to provide secure development training on managing dependencies effectively. By educating developers and development teams on the importance of dependency management, organizations can integrate secure coding practices into the implementation phase. Secure architecture practices, including threat modeling and secure component selection, are supported by Library Dependencies management. By analyzing the security of dependencies, organizations can make informed decisions during implementation, minimizing the risk of vulnerabilities associated with insecure components. Secure coding guidelines can include recommendations on managing dependencies securely, such as avoiding outdated or vulnerable components. Library Dependencies management helps organizations adhere to these guidelines and implement secure coding practices during the implementation phase. Security requirements play a crucial role in the implementation phase, and managing dependencies effectively contributes to meeting security requirements. By ensuring that dependencies are up-to-date and free from known vulnerabilities, organizations can meet the security objectives defined in the requirements. Security testing integration is enhanced by Library Dependencies management, as it helps organizations identify vulnerabilities related to dependencies and address them during the implementation phase. Security verification activities, such as security code reviews and vulnerability assessments, benefit from effective Library Dependencies management. By ensuring that dependencies are secure and free from vulnerabilities, organizations can verify the security controls implemented during the implementation phase.
4	SAMM Security by Verification	Security Testing: DevSecOps' capability of managing library dependencies helps address the SAMM Security by Verification domain's activity of security testing. By effectively managing and monitoring dependencies, organizations can prevent the use of outdated or vulnerable components, reducing the likelihood of security vulnerabilities. This capability enables organizations to conduct activities such as penetration testing, vulnerability scanning, and code review to identify and mitigate security flaws in software applications. Code Review: DevSecOps' capability of managing library dependencies indirectly contributes to the SAMM Security by Verification domain's activity of code review. By keeping dependencies up-to-date and addressing known vulnerabilities, organizations can enhance the overall security posture of the codebase. This reduces the likelihood of security flaws being introduced during development and improves adherence to secure coding practices. Security Architecture Review: DevSecOps' capability of managing library dependencies indirectly supports the SAMM Security by Verification domain's activity of security architecture review. By conducting dependency analysis and vulnerability assessments, organizations can identify potential security risks associated with the dependencies used in the software system. This information contributes to the overall security architecture review process and helps ensure the presence of proper security controls and mechanisms.

#	Name	Description
		Security Requirements Verification: DevSecOps' capability of managing library dependencies indirectly assists in the SAMM Security by Verification domain's activity of security requirements verification. By incorporating dependency analysis and vulnerability assessments, organizations can validate that the security requirements defined for the software system adequately address the risks associated with the dependencies. This verification helps ensure that the implemented security controls align with the intended security objectives. Threat Modeling: DevSecOps' capability of managing library dependencies indirectly supports the SAMM Security by Verification domain's activity of threat modeling. By conducting dependency analysis and vulnerability assessments, organizations can identify potential threats and risks associated with the dependencies used in the software system. This information contributes to the overall threat modeling process and helps prioritize security controls and countermeasures. Secure Deployment Verification: DevSecOps' capability of managing library dependencies indirectly contributes to the SAMM Security by Verification domain's activity of secure deployment verification. By ensuring that dependencies are up-to-date and free from known vulnerabilities, organizations can reduce the risk of introducing security weaknesses during the deployment process. This verification ensures that the software system is securely deployed and configured in the production environment.
5	SAMM Security by Operations	Library Dependencies: Managing and monitoring dependencies helps establish secure operational processes and practices. It contributes to the "Environment Hardening" activity by ensuring that dependencies are securely configured and maintained. Dependency Management: Tracking and managing dependencies aligns with the "Configuration Management" activity. It helps organizations maintain the integrity of software systems and detect unauthorized changes or misconfigurations. Version Control and Updates: Keeping dependencies up-to-date supports the "Configuration Management" activity. It ensures that software systems are maintained with secure configurations, including applying patches and updates. Dependency Analysis: Analyzing dependencies contributes to the "Vulnerability Management" activity. It helps identify vulnerabilities and prioritize them for remediation. Risk Assessment and Prioritization: Performing risk assessments for dependencies helps organizations prioritize vulnerabilities and allocate resources accordingly, supporting the "Vulnerability Management" activity. Patching and Remediation: Applying patches and taking remediation steps for vulnerabilities in dependencies aligns with the "Vulnerability Management" activity. It helps organizations address identified vulnerabilities and reduce the risk of exploitation. Continuous Monitoring: Continuous monitoring of dependencies supports the "Vulnerability Management" and "Incident Management" activities. It helps identify and respond to security incidents promptly, ensuring ongoing security and reliability of software systems.

11.3.14.2 Assessment – OWASP Top 10

The capability of Library Dependencies in the DevSecOps framework plays a significant role in addressing several OWASP threats by ensuring the use of secure, up-to-date, and properly validated libraries within web applications.

#	Name	Description
1	Broken Access Control (A01-2021)	By managing and monitoring library dependencies, organizations can prevent the use of outdated or vulnerable components that could introduce access control flaws. Keeping dependencies up-to-date helps ensure proper enforcement of access controls and reduces the risk of unauthorized access or privilege escalation.
2	Cryptographic Failures (A02-2021)	Library Dependencies management can include verifying and updating cryptographic libraries used in the application. By staying informed about security advisories and using secure versions of cryptographic components, organizations can mitigate vulnerabilities related to cryptographic weaknesses and ensure the integrity and confidentiality of data.
3	Injection (A03-2021)	Through dependency analysis, organizations can identify and address vulnerabilities in libraries that may introduce injection flaws. By using vulnerability databases, security scanners, or static code analysis tools, organizations can detect and mitigate injection vulnerabilities caused by insecure or poorly validated library dependencies.
4	Insecure Design (A04-2021)	The capability of managing library dependencies can contribute to addressing insecure design vulnerabilities by promoting the use of secure and well-maintained libraries. By incorporating secure design principles and selecting libraries with good security track records, organizations can reduce the risk of insecure design decisions associated with library usage.
5	Security Misconfiguration (A05-2021)	Library Dependencies management helps prevent security misconfigurations related to libraries by ensuring that the correct and secure versions of dependencies are used. By following secure configuration practices and maintaining an inventory of dependencies, organizations can minimize the risk of misconfigurations that may expose sensitive information or create potential entry points for attackers.
6	Vulnerable and Outdated Components (A06-2021)	The capability of managing library dependencies directly addresses the vulnerabilities associated with using outdated or known vulnerable software components. By regularly monitoring for updates, applying patches, and replacing dependencies with secure alternatives, organizations can mitigate the risk of exploiting known vulnerabilities in the components used by the application.
7	Identification and Authentication Failures (A07-2021)	Library Dependencies management indirectly contributes to addressing identification and authentication failures by ensuring that the libraries used for authentication mechanisms are secure and up-to-date. By using libraries with strong authentication features and following secure coding practices, organizations can reduce the risk of authentication vulnerabilities that may lead to unauthorized access or impersonation.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	By managing library dependencies and using secure versions of libraries, organizations can reduce the risk of software and data integrity failures. Libraries with integrity checks and secure coding practices can help protect against unauthorized modifications or tampering of software components, configurations, and data.
9	Security Logging and Monitoring Failures (A09-2021)	Although Library Dependencies management does not directly address logging and monitoring failures, using secure and well-maintained libraries can contribute to the overall security logging and monitoring capabilities of the application. Libraries with built-in logging and monitoring features can assist in the detection and response to security incidents.
10	Server-Side Request Forgery (A10-2021)	Library Dependencies management can indirectly contribute to mitigating Server-Side Request Forgery vulnerabilities by using secure libraries that properly handle external requests. Libraries with security features and proper input validation can help prevent the manipulation of requests to internal or external resources and protect against SSRF attacks.

11.3.15 Library Deprecation

Library Deprecation is an important security capability within the DevSecOps framework. It involves identifying and replacing deprecated libraries used in an application to mitigate security risks. By ensuring that the application uses up-to-date and supported libraries, organizations can reduce the risk of security vulnerabilities associated with deprecated versions.

By incorporating Library Deprecation into the DevSecOps practices, organizations can effectively manage the security risks associated with deprecated libraries. This capability helps ensure that the application remains protected against known vulnerabilities and reduces the likelihood of successful attacks targeting outdated or unsupported libraries.

Here are key aspects of Library Deprecation within the DevSecOps context:

#	Name	Description
1	Identification of Deprecated Libraries	The first step in the library deprecation process is identifying libraries that are deprecated or no longer actively maintained by their developers. Deprecated libraries may pose security risks as they no longer receive updates or patches for newly discovered vulnerabilities. Organizations can refer to official documentation, release notes, or online resources to identify deprecated libraries.
2	Risk Assessment	Once deprecated libraries are identified, a risk assessment is performed to evaluate the potential security impact. This assessment may involve considering the severity of known vulnerabilities, the level of community support, and the availability of alternatives or migration paths. Libraries with high-risk vulnerabilities or lack of support may require immediate attention.

#	Name	Description
3	Migration Plan	After assessing the risks associated with deprecated libraries, a migration plan is developed to replace them with up-to-date and supported alternatives. The plan should outline the steps, resources, and timeline required for the migration process. It may involve finding alternative libraries, modifying the application code to accommodate the changes, and conducting thorough testing to ensure compatibility and functionality.
4	Testing and Validation	It is crucial to thoroughly test the application after replacing deprecated libraries to ensure that it remains secure and functions as expected. This includes regression testing to verify that the new libraries do not introduce any new security vulnerabilities or break existing functionality. Automated testing frameworks and tools can assist in streamlining the testing process.
5	Continuous Monitoring and Updates	Library deprecation is an ongoing process that requires continuous monitoring of the libraries used in the application. Regularly checking for deprecation notices and updates from library developers helps ensure that the application remains up-to-date and secure. Continuous monitoring can be integrated into the DevSecOps pipeline to provide timely alerts and facilitate the replacement of deprecated libraries.
6	Documentation and Knowledge Sharing	Maintaining proper documentation of deprecated libraries and their replacements is essential for knowledge sharing within the development and operations teams. This documentation helps in maintaining awareness of deprecated libraries and facilitates future updates or migrations. It also assists in onboarding new team members and ensures consistent practices across the organization.

11.3.15.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Library deprecation helps organizations adhere to secure coding practices by replacing deprecated libraries, thereby supporting the establishment of policies and standards related to secure coding practices. Risk Management: Library deprecation helps organizations assess and mitigate the risks associated with deprecated libraries, reducing the potential security impact and aligning with risk management processes. Compliance Management: Library deprecation ensures that organizations remain compliant with relevant regulatory requirements and industry standards by replacing deprecated libraries that may introduce vulnerabilities or non-compliance. Security Training and Awareness: Library deprecation contributes to security training and awareness efforts by educating developers and stakeholders about the risks associated with deprecated libraries and promoting a culture of security. Security Metrics and Reporting: Library deprecation supports the measurement of security metrics by reducing the number of deprecated libraries, which can be tracked as a metric to assess the effectiveness of software security practices. Security Roles and Responsibilities: Library deprecation helps define and assign security roles and responsibilities by ensuring that individuals are aware of the risks associated with deprecated libraries and their responsibility to replace them with up-to-date alternatives.

#	Name	Description
2	SAMM Security by Design	Security Requirements: Library deprecation supports the inclusion of security requirements by addressing the specific security concerns associated with deprecated libraries and ensuring their replacement with secure alternatives. Secure Architecture: Library deprecation helps establish a secure architecture by replacing deprecated libraries that may introduce vulnerabilities or compromise the overall security of the system. Secure Coding Practices: Library deprecation promotes secure coding practices by replacing deprecated libraries and preventing the use of outdated or vulnerable code in software applications. Threat Modeling: Library deprecation contributes to threat modeling exercises by identifying potential threats and risks associated with deprecated libraries and ensuring their mitigation through replacement. Security Testing: Library deprecation supports security testing activities by replacing deprecated libraries and validating that the new libraries do not introduce security vulnerabilities or compromise the software's functionality.
3	SAMM Security by Implementation	Secure Development Training: Library deprecation reinforces secure development training by educating developers about the risks of using deprecated libraries and the importance of replacing them with up-to-date and supported alternatives. Secure Architecture: Library deprecation helps implement a secure architecture by replacing deprecated libraries and ensuring that the new libraries align with secure architectural patterns and principles. Secure Coding Guidelines: Library deprecation supports the adoption of secure coding guidelines by replacing deprecated libraries and promoting the use of secure coding practices when selecting and integrating new libraries. Security Requirements: Library deprecation aligns with security requirements by replacing deprecated libraries that may violate the specified security objectives and expectations. Security Testing Integration: Library deprecation facilitates the integration of security testing activities by ensuring that the software is free from vulnerabilities associated with deprecated libraries before entering the testing phase.
4	SAMM Security by Verification	Contribution: Library deprecation supports security verification efforts by ensuring that the software system no longer relies on deprecated libraries and meets the desired security objectives. Security Architecture Review: Library deprecation assists in security architecture reviews by replacing deprecated libraries and validating the effectiveness of the new libraries in addressing potential vulnerabilities or weaknesses. Security Operations Integration: Library deprecation contributes to security operations integration by ensuring that the replacement of deprecated libraries is supported and maintained throughout the operational life of the software system.
5	SAMM Security by Operations	It does not address this domain directly.

11.3.15.2 Assessment – OWASP Top 10

Library Deprecation within the DevSecOps framework may not directly address all OWASP threats; however, it indirectly contributes to mitigating several vulnerabilities by ensuring the use of up-to-date and supported libraries, reducing the reliance on deprecated or insecure functions, and minimizing the risk of using outdated or vulnerable components.

#	Name	Description
1	Broken Access Control (A01-2021)	While Library Deprecation does not directly address this threat, using up-to-date and supported libraries can indirectly contribute to mitigating Broken Access Control vulnerabilities. By keeping libraries current, organizations reduce the risk of using deprecated or insecure functions that could potentially lead to access control flaws.
2	Cryptographic Failures (A02-2021)	Library Deprecation directly addresses this threat by ensuring that deprecated or vulnerable cryptographic libraries are replaced with up-to-date and secure alternatives. By identifying and replacing deprecated cryptographic libraries, organizations can minimize the risk of implementing weak encryption algorithms, improper usage, or key management issues.
3	Injection (A03-2021)	Library Deprecation does not directly address Injection vulnerabilities. However, using up-to-date libraries can indirectly contribute to mitigating Injection vulnerabilities by reducing the risk of using outdated libraries that may have known vulnerabilities, including those related to input validation and sanitization.
4	Insecure Design (A04-2021)	Library Deprecation does not directly address Insecure Design vulnerabilities. However, by ensuring that libraries used in the application are up-to-date and supported, organizations can indirectly reduce the risk of relying on insecure or outdated design patterns or architectural decisions that could contribute to security weaknesses.
5	Security Misconfiguration (A05-2021)	Library Deprecation does not directly address Security Misconfiguration vulnerabilities. However, by replacing deprecated libraries, organizations can indirectly reduce the risk of misconfigurations associated with outdated or improperly configured libraries, frameworks, or components.
6	Vulnerable and Outdated Components (A06-2021)	Library Deprecation directly addresses this threat. By identifying and replacing deprecated libraries, organizations reduce the use of outdated or known vulnerable components, such as libraries, frameworks, or plugins. This helps minimize the risk of exploitation through known vulnerabilities in these components.
7	Identification and Authentication Failures (A07-2021)	Library Deprecation does not directly address Identification and Authentication Failures vulnerabilities. However, using up-to-date libraries can indirectly contribute to stronger authentication mechanisms by reducing the risk of relying on deprecated or insecure authentication-related functions or libraries.
8	Software and Data Integrity Failures (A08-2021)	Library Deprecation does not directly address Software and Data Integrity Failures vulnerabilities. However, by ensuring that libraries are up-to-date and supported, organizations can indirectly reduce the risk of integrity failures associated with outdated or vulnerable libraries or components.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Library Deprecation does not directly address Security Logging and Monitoring Failures vulnerabilities. However, using up-to-date libraries can indirectly contribute to improved logging and monitoring capabilities by reducing the risk of relying on deprecated or insecure logging-related functions or libraries.
10	Server-Side Request Forgery (A10-2021)	Library Deprecation does not directly address Server-Side Request Forgery vulnerabilities. However, by replacing deprecated libraries, organizations can indirectly reduce the risk of relying on insecure or outdated functions that may be susceptible to SSRF attacks.

11.3.16 Architecture Design Conformance

Architecture Design Conformance is a crucial security capability within the DevSecOps framework. It involves verifying that the system architecture conforms to security best practices and design principles. By integrating security considerations into the system's design, organizations can reduce the likelihood of security flaws and vulnerabilities.

By incorporating Architecture Design Conformance into the DevSecOps practices, organizations can establish a strong foundation for secure system development and deployment. This capability helps ensure that security considerations are integrated from the early stages of design, reducing the likelihood of security flaws and vulnerabilities. It also promotes a proactive and systematic approach to security, allowing organizations to address potential risks in a structured and comprehensive manner.

Here are key aspects of Architecture Design Conformance within the DevSecOps context:

#	Name	Description
1	Security Design Principles	The first step in Architecture Design Conformance is establishing security design principles that guide the development and implementation of the system architecture. These principles define the desired security objectives and requirements, such as confidentiality, integrity, availability, and resilience. They serve as a foundation for evaluating and verifying the architecture's adherence to security best practices.
2	Security Architecture Review	A comprehensive review of the system architecture is conducted to assess its alignment with the established security design principles. This review involves examining architectural diagrams, component interactions, data flows, and access controls. It aims to identify any potential security gaps or weaknesses in the design and ensures that appropriate security controls are implemented at each layer of the architecture.
3	Threat Modeling	Threat modeling is performed to identify potential threats and attack vectors that the system may face. It involves analyzing the system's components, data flows, and interactions to understand the potential security risks and vulnerabilities. By considering potential threats and their impacts, organizations can make informed decisions about security controls and design choices to mitigate those risks effectively.

#	Name	Description
4	Security Controls Integration	During the Architecture Design Conformance process, security controls are integrated into the system architecture. This includes incorporating authentication mechanisms, access controls, encryption, logging and monitoring, and other relevant security measures. The goal is to ensure that the architecture provides adequate protection against common security threats and aligns with industry standards and best practices.
5	Compliance and Regulatory Requirements	Architecture Design Conformance also includes verifying compliance with applicable regulatory requirements, industry standards, and internal policies. This may involve ensuring adherence to privacy regulations, data protection laws, industry-specific security frameworks, or specific organizational security guidelines. Compliance checks help ensure that the architecture meets legal and regulatory obligations related to security and privacy.
6	Documentation and Communication	It is essential to document the design decisions, security controls, and rationale behind the architecture to facilitate understanding, collaboration, and knowledge sharing among team members. Clear documentation helps communicate the security design choices, promotes consistency, and provides a reference for future audits, reviews, or updates. Regular communication and collaboration among architects, developers, and security teams are crucial to ensure alignment and address any security-related concerns.

11.3.16.1 Assessment – SAMM

Architecture Design Conformance primarily focuses on design aspects within DevSecOps; however, it contributes to various SAMM security domains by promoting security principles and practices from the early design phase, thereby indirectly addressing multiple OWASP threats associated with design and architecture weaknesses.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Architecture Design Conformance contributes by verifying that the system architecture adheres to security policies and standards. It ensures that the design complies with established security principles and requirements, aligning with policies related to secure design practices.

#	Name	Description
2	SAMM Security by Design	Security Requirements: Architecture Design Conformance aligns with this activity by ensuring that the security design principles and requirements are integrated into the system architecture. It establishes the foundation for addressing specific security concerns, contributing to the creation of secure designs. Secure Architecture: Architecture Design Conformance promotes secure architectural practices by reviewing the system architecture for security gaps and weaknesses. It helps in creating a robust and secure architectural foundation, which is essential for addressing various OWASP threats related to insecure design and architecture. Secure Coding Practices: While Architecture Design Conformance primarily focuses on design, it indirectly contributes to secure coding practices by emphasizing the importance of integrating security from the early stages. This ensures that the architecture provides a secure foundation for subsequent coding efforts.
3	SAMM Security by Implementation	Secure Development Training: Architecture Design Conformance encourages the integration of security training into the design phase. This ensures that architects and developers have the necessary knowledge and skills to implement secure architectural designs. Secure Coding Guidelines: Although the primary focus is on design, Architecture Design Conformance indirectly supports this activity by promoting the use of secure design principles, which influence secure coding guidelines during the development phase. Security Requirements: This activity is supported as Architecture Design Conformance ensures that security requirements are integrated into the system's architectural design.
4	SAMM Security by Verification	Code Review: While Architecture Design Conformance primarily focuses on design reviews, it indirectly supports code reviews by ensuring that architectural designs are free from security gaps and vulnerabilities. Security Architecture Review: This activity is directly supported, as Architecture Design Conformance involves reviewing the system architecture for security aspects, aligning with the need for architectural security reviews. Threat Modeling: Architecture Design Conformance contributes to threat modeling activities by identifying potential threats and risks during the design phase, which can inform threat modeling efforts in later stages.
5	SAMM Security by Operations	Architecture Design Conformance indirectly supports this activity by ensuring that architectural designs consider security controls, encryption mechanisms, and other security measures from the early stages, which is critical for secure deployment.

11.3.16.2 Assessment – OWASP Top 10

Architecture Design Conformance within DevSecOps directly addresses various OWASP threats by incorporating security considerations, best practices, and controls into the system architecture. It helps organizations establish a strong foundation for secure development and deployment, reducing the likelihood of security flaws and vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	Architecture Design Conformance ensures that proper access controls are integrated into the system architecture. This includes defining security design principles and verifying that the architecture enforces proper access restrictions, reducing the risk of unauthorized access.
2	Cryptographic Failures (A02-2021)	Architecture Design Conformance involves reviewing the system architecture to ensure the correct implementation of cryptographic mechanisms. By integrating secure cryptographic practices into the architecture design, organizations can mitigate vulnerabilities related to weak algorithms, improper usage, and key management issues.
3	Injection (A03-2021)	Architecture Design Conformance emphasizes proper input validation and secure coding practices, which help prevent injection vulnerabilities. By considering potential injection threats during architecture design, organizations can reduce the risk of injection attacks by ensuring untrusted data is handled properly.
4	Insecure Design (A04-2021)	Architecture Design Conformance aims to eliminate insecure design decisions by adopting secure design principles from the early stages of development. By considering security requirements and potential threats during architecture design, organizations can mitigate vulnerabilities stemming from poor architectural decisions.
5	Security Misconfiguration (A05-2021)	Architecture Design Conformance ensures that systems, frameworks, servers, and web applications are properly configured to adhere to secure practices. By incorporating secure configuration practices into the architecture design, organizations can mitigate security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Architecture Design Conformance promotes active management of software dependencies and ensuring that only up-to-date and secure components are used in the system architecture. By considering the risks associated with vulnerable and outdated components, organizations can reduce the likelihood of exploiting known vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	Architecture Design Conformance involves implementing secure authentication mechanisms and considering authentication requirements during architecture design. By incorporating strong identification and authentication mechanisms into the architecture, organizations can reduce the risk of unauthorized access and impersonation.
8	Software and Data Integrity Failures (A08-2021)	Architecture Design Conformance includes implementing secure coding techniques and practices that ensure the integrity of software components, configurations, and data. By considering integrity requirements during architecture design, organizations can mitigate vulnerabilities related to unauthorized modifications or tampering.
9	Security Logging and Monitoring Failures (A09-2021)	Architecture Design Conformance promotes the inclusion of comprehensive logging and effective monitoring mechanisms in the system architecture. By considering logging and monitoring requirements during architecture design, organizations can enhance their ability to detect and respond to security incidents.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Architecture Design Conformance includes secure input validation and secure configurations to prevent server-side request forgery vulnerabilities. By considering the risks associated with SSRF during architecture design, organizations can reduce the risk of unauthorized requests to internal or external resources.

11.3.17 Automated Threat Modeling

Automated Threat Modeling is a critical security capability within the DevSecOps framework. It involves the use of automated tools and processes to identify and analyze potential threats and vulnerabilities in a system. By proactively assessing security risks, organizations can make informed decisions to mitigate those risks and strengthen their security controls.

By leveraging Automated Threat Modeling within DevSecOps practices, organizations can proactively identify and mitigate potential security risks in their systems. It helps in understanding the system's security posture, prioritizing security efforts, and making informed decisions to strengthen security controls. By automating the threat modeling process, organizations can save time, increase efficiency, and improve the overall security of their systems.

Here are key aspects of Automated Threat Modeling within the DevSecOps context:

#	Name	Description
1	System Understanding	The first step in Automated Threat Modeling is gaining a comprehensive understanding of the system's architecture, components, and functionalities. This includes examining the system's design, data flows, interactions, and external dependencies. Understanding the system's context is essential for accurately identifying potential threats and vulnerabilities.
2	Threat Identification	Automated tools and techniques are employed to identify potential threats that the system may face. These tools leverage security knowledge bases, threat intelligence, and predefined attack patterns to identify known and emerging threats. The tools analyze the system's design and configuration to identify potential entry points, weak points, and vulnerabilities that could be exploited by attackers.
3	Vulnerability Analysis	Once potential threats are identified, the Automated Threat Modeling process includes analyzing the system for vulnerabilities that could be exploited by those threats. This involves scanning the system for common security weaknesses such as insecure configurations, improper input handling, insufficient authentication and authorization mechanisms, and other known vulnerabilities. The tools may use techniques like static code analysis, dynamic testing, and vulnerability databases to identify vulnerabilities.
4	Risk Assessment	Automated Threat Modeling also involves assessing the potential impact and likelihood of each identified threat. This step helps prioritize threats based on their severity and potential impact on the system's security. Risk assessment considers factors such as the sensitivity of data, potential business impact, regulatory requirements, and the likelihood of the threat being exploited. By prioritizing threats, organizations can allocate their resources effectively to mitigate the most critical risks.

#	Name	Description
5	Mitigation Recommendations	Based on the identified threats and vulnerabilities, Automated Threat Modeling provides recommendations for mitigating the risks. These recommendations may include specific security controls, architectural changes, code modifications, or configuration adjustments. The goal is to provide actionable guidance to developers and security teams for addressing the identified risks in an efficient and timely manner.
6	Integration with DevSecOps Pipeline	Automated Threat Modeling is integrated into the DevSecOps pipeline to ensure continuous monitoring and assessment of the system's security. It is often implemented as part of the build and deployment process, allowing organizations to identify and address security issues early in the development lifecycle. Automated Threat Modeling can be triggered automatically during the development process or integrated with other security testing tools and processes.

11.3.17.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Automated Threat Modeling contributes to this domain by providing insights into potential threats and vulnerabilities, which can inform the development and implementation of security policies and standards. Risk Management: Automated Threat Modeling helps identify and assess security risks, enabling organizations to make informed decisions and develop risk management strategies. Compliance Management: Automated Threat Modeling assists in identifying and addressing potential compliance issues by identifying security gaps and vulnerabilities. Security Training and Awareness: Automated Threat Modeling can inform the content and focus of security training programs, helping to raise awareness and educate employees and stakeholders about potential threats and vulnerabilities. Security Metrics and Reporting: Automated Threat Modeling provides data and analysis that can contribute to security metrics and reporting, offering insights into the effectiveness of security controls and the overall security posture. Security Roles and Responsibilities: Automated Threat Modeling helps identify the roles and responsibilities related to addressing specific threats and vulnerabilities, supporting the establishment of clear security roles within the organization.
2	SAMM Security by Design	Security Requirements: Automated Threat Modeling can help identify potential threats and vulnerabilities, which can inform the definition and incorporation of specific security requirements into the design phase. Secure Architecture: Automated Threat Modeling supports the identification of potential vulnerabilities and the design of secure architectural patterns, contributing to the creation of a robust and secure architecture. Secure Coding Practices: Automated Threat Modeling can provide insights into potential vulnerabilities, which can guide the establishment and adoption of secure coding practices during the design phase.

#	Name	Description
		Threat Modeling: Automated Threat Modeling is an integral part of SAMM's Security by Design domain, as it enables organizations to identify and address potential threats and risks during the design phase. Security Testing: Automated Threat Modeling can inform the selection and implementation of security testing techniques, such as static code analysis and security code reviews, to detect and address vulnerabilities in the design phase.
3	SAMM Security by Implementation	Secure Development Training: Automated Threat Modeling can be incorporated into secure development training programs, providing real-world examples and insights into potential vulnerabilities. Secure Architecture: Automated Threat Modeling contributes to the establishment of secure architecture practices, such as threat modeling and secure design patterns, supporting the implementation of security measures during development. Secure Coding Guidelines: Automated Threat Modeling helps identify potential vulnerabilities, which can inform the development of secure coding guidelines and standards. Security Requirements: Automated Threat Modeling supports the implementation of security requirements by identifying potential threats and vulnerabilities that need to be addressed. Security Testing Integration: Automated Threat Modeling can inform the integration of security testing activities, such as static code analysis and dynamic application security testing, into the development process. Security Verification: Automated Threat Modeling contributes to security verification by identifying potential vulnerabilities and verifying the effectiveness of security controls implemented in the software system. Security Architecture Review: Automated Threat Modeling can provide insights into potential weaknesses in the security architecture, supporting security architecture reviews and recommendations for improvement. Security Operations Integration: Automated Threat Modeling can inform the integration of security considerations into operational processes, such as incident management and vulnerability management.
4	SAMM Security by Verification	Threat Identification: Automated Threat Modeling helps in identifying potential threats and vulnerabilities in software systems. This contributes to the SAMM Security by Verification activity of "Threat Modeling." Vulnerability Analysis: Automated Threat Modeling includes analyzing the system for vulnerabilities that could be exploited by threats. This aligns with the SAMM Security by Verification activity of "Vulnerability Analysis." Risk Assessment: Automated Threat Modeling assesses the potential impact and likelihood of identified threats, helping prioritize them based on severity. This aligns with the SAMM Security by Verification activity of "Risk Assessment." Mitigation Recommendations: Automated Threat Modeling provides recommendations for mitigating identified risks and vulnerabilities. This supports the SAMM Security by Verification activity of "Mitigation Recommendations." Integration with DevSecOps Pipeline: Automated Threat Modeling can be integrated into the DevSecOps pipeline, ensuring continuous monitoring and assessment of the system's security. This aligns with the SAMM Security by Verification activity of "Integration with Development Process."

#	Name	Description
5	SAMM Security by Operations	Environment Hardening Activity: Automated Threat Modeling assists in identifying potential security weaknesses and vulnerabilities in the software environment, enabling organizations to strengthen security configurations. Vulnerability Management Activity: Automated Threat Modeling helps in identifying vulnerabilities in software applications, supporting the vulnerability management process. Incident Management Activity: Automated Threat Modeling contributes to incident management by providing insights into potential security incidents and helping prioritize incident response efforts. Security Testing Activity: Automated Threat Modeling integrates with security testing activities, such as penetration testing and vulnerability scanning, to identify and address security vulnerabilities in software systems. Secure Deployment Activity: Automated Threat Modeling helps ensure secure deployment by identifying potential security risks and providing recommendations for secure software distribution and installation procedures. Indirectly contributes to the Environment Hardening, Secure Build, Configuration Management, Vulnerability Management, Incident Management, Secure Deployment, Security Testing, and Operational Enablement sub-domains. It helps in identifying vulnerabilities, ensuring secure configurations and deployment, and supporting incident management and operational procedures.

11.3.17.2 Assessment – OWASP Top 10

Automated Threat Modeling within the DevSecOps framework helps address various OWASP threats by proactively identifying and analyzing potential vulnerabilities and providing recommendations for their mitigation. It enhances system understanding, threat identification, vulnerability analysis, risk assessment, mitigation recommendations, and integration with the DevSecOps pipeline to strengthen the overall security posture of applications.

#	Name	Description
1	Broken Access Control (A01-2021)	Automated Threat Modeling helps identify potential access control vulnerabilities by analyzing the system's design, data flows, and external dependencies. It helps in proactively assessing security risks related to access controls and making informed decisions to mitigate those risks.
2	Cryptographic Failures (A02-2021)	Automated Threat Modeling includes vulnerability analysis, which can identify cryptographic weaknesses such as weak algorithms, improper usage, or key management issues. By analyzing the system's design and configurations, the process can identify vulnerabilities related to cryptography and provide recommendations for their mitigation.
3	Injection (A03-2021)	Automated Threat Modeling involves threat identification and vulnerability analysis, which can identify injection vulnerabilities. By analyzing the system for improper handling of untrusted data, the process can help detect potential injection points and recommend mitigations to prevent injection attacks.

#	Name	Description
4	Insecure Design (A04-2021)	Automated Threat Modeling contributes to addressing insecure design vulnerabilities by analyzing the system's architecture and design. It helps in identifying flaws and weaknesses in the overall design and provides recommendations for secure design principles, architectural changes, and security controls to mitigate these vulnerabilities.
5	Security Misconfiguration (A05-2021)	Automated Threat Modeling helps identify security misconfigurations by analyzing the system's configurations and settings. By scanning for insecure configurations and providing recommendations for secure practices, the process can assist in mitigating security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Automated Threat Modeling contributes to addressing vulnerabilities related to vulnerable and outdated components by analyzing the system's dependencies. It helps in identifying insecure versions of third-party software components and provides recommendations for managing software dependencies, staying updated with security patches, and following secure coding practices.
7	Identification and Authentication Failures (A07-2021)	Automated Threat Modeling helps identify vulnerabilities related to identification and authentication mechanisms by analyzing the system's design and configurations. It can detect weaknesses in authentication mechanisms and provide recommendations for implementing secure authentication practices, enforcing strong password policies, and addressing potential vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	Automated Threat Modeling contributes to addressing vulnerabilities related to software and data integrity by analyzing the system's components and configurations. It helps in identifying potential weaknesses that could lead to unauthorized modifications, data tampering, or other integrity-related issues. The process provides recommendations for secure practices and coding techniques to mitigate these vulnerabilities.
9	Security Logging and Monitoring Failures (A09-2021)	Automated Threat Modeling helps in addressing vulnerabilities related to security logging and monitoring by analyzing the system's logging and monitoring capabilities. It can identify deficiencies in log collection, analysis, and retention, and recommend improvements to ensure effective logging and real-time monitoring mechanisms.
10	Server-Side Request Forgery (A10-2021)	Automated Threat Modeling contributes to addressing Server-Side Request Forgery (SSRF) vulnerabilities by analyzing the system's design and configurations. It can help identify potential SSRF vulnerabilities and provide recommendations for input validation, secure configurations, and regular security testing to prevent unauthorized requests.

11.3.18 Secure Coding Practices

Secure Coding Practices is a crucial security capability within the DevSecOps framework. It involves following coding practices and guidelines that prioritize security to minimize the introduction of vulnerabilities during the development process. By incorporating secure coding practices from the outset, organizations can significantly reduce the likelihood of common security vulnerabilities and enhance the overall security of their software.

By following secure coding practices within the DevSecOps approach, organizations can significantly reduce the likelihood of introducing vulnerabilities during the development process. It enables developers to write more secure code from the outset, minimizing the potential for common security vulnerabilities. Incorporating secure coding practices as an integral part of the software development lifecycle helps build a strong foundation of security within the application and contributes to the overall security posture of the organization.

Here are key aspects of Secure Coding Practices within the DevSecOps context:

#	Name	Description
1	Knowledge of Security Best Practices	Developers and software engineers need to have a solid understanding of security best practices and guidelines. This includes knowledge of secure coding principles, secure coding standards, secure coding frameworks, and industry-recognized secure coding practices such as the OWASP Top Ten. By having this knowledge, developers can write code that is resilient against common security vulnerabilities.
2	Input Validation and Output Sanitization	Secure coding practices emphasize the importance of input validation and output sanitization. Input validation involves validating and sanitizing all user inputs to prevent common attack vectors such as SQL injection, cross-site scripting (XSS), and command injection. Output sanitization ensures that any data being sent to users or external systems is properly encoded and sanitized to prevent unintended execution of malicious code.
3	Authentication and Authorization	Secure coding practices focus on implementing strong and secure authentication and authorization mechanisms. This includes using secure password storage techniques such as hashing and salting, implementing multi-factor authentication where appropriate, and properly managing user sessions to prevent session-related attacks. Authorization controls should be implemented to ensure that only authorized users have access to specific resources and functionalities.
4	Secure Communication	Secure coding practices emphasize the use of secure communication protocols, such as HTTPS, to protect sensitive data during transit. Developers should ensure that sensitive information, such as passwords, credit card details, and personal data, is always transmitted over encrypted channels. Additionally, the proper implementation of SSL/TLS certificates and adherence to cipher suite configurations are essential for secure communication.
5	Error and Exception Handling	Secure coding practices include robust error and exception handling mechanisms. Proper error handling helps prevent the disclosure of sensitive information and can provide attackers with insights into the system's vulnerabilities. Developers should implement appropriate error messages that do not reveal implementation details or provide attackers with useful information for exploiting the system.

#	Name	Description
6	Secure Configuration Management	Secure coding practices also involve managing the configuration of the application securely. This includes securely storing and managing sensitive configuration data, such as database credentials, API keys, and encryption keys. Developers should ensure that sensitive configuration information is not hardcoded within the codebase and is stored securely, such as using encrypted configuration files or secure vaults.
7	Regular Code Reviews and Testing	Secure coding practices emphasize the importance of regular code reviews and testing. Peer code reviews help identify security vulnerabilities and coding mistakes, while testing, such as unit testing and integration testing, ensures that the code functions as expected and does not introduce security flaws. Automated security testing tools can be used to identify common security vulnerabilities and provide feedback to developers during the development process.
8	Security Training and Awareness	Secure coding practices require a culture of security awareness and continuous learning within the development team. Regular security training sessions and workshops can help developers stay up to date with the latest security threats, vulnerabilities, and countermeasures. By fostering a security-conscious mindset, developers can proactively incorporate security considerations into their coding practices.

11.3.18.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Secure coding practices contribute to the development and implementation of policies and standards related to software security. By following secure coding practices, organizations can define secure coding standards and guidelines within their policies and standards. Risk Management: Secure coding practices help mitigate risks associated with software vulnerabilities by reducing the likelihood of introducing security weaknesses during the development process. By following secure coding practices, organizations can address risks related to common security vulnerabilities. Compliance Management: Secure coding practices support compliance with regulatory requirements and industry standards by incorporating security controls and practices that align with these compliance frameworks. Adhering to secure coding practices helps organizations meet the necessary security requirements. Security Training and Awareness: Secure coding practices require ongoing security training and awareness programs to educate developers and software engineers about secure coding principles. By promoting a culture of security awareness, organizations can ensure that secure coding practices are understood and followed by the development team. Security Metrics and Reporting: Secure coding practices contribute to security metrics and reporting by reducing the number of security vulnerabilities introduced during the development process. By adhering to secure coding practices, organizations can demonstrate improved security metrics and report on the effectiveness of their secure coding initiatives.

#	Name	Description
		Security Roles and Responsibilities: Secure coding practices help define and assign security roles and responsibilities within the organization. By incorporating secure coding practices into job roles and responsibilities, organizations ensure that individuals have the necessary knowledge and skills to fulfill their security-related tasks effectively.
2	SAMM Security by Design	Security Requirements: Secure coding practices contribute to addressing security requirements by ensuring that developers follow secure coding principles and guidelines. By incorporating secure coding practices, organizations can design software that meets the defined security requirements. Secure Architecture: Secure coding practices support the creation of a secure architecture by minimizing vulnerabilities and addressing potential security risks. By following secure coding practices, organizations can implement secure design patterns and principles that contribute to a secure architecture. Secure Coding Practices: Secure coding practices are a core aspect of the SAMM Security by Design domain. By incorporating secure coding practices, organizations can minimize the introduction of vulnerabilities during the design phase of software development, addressing various OWASP threats. Threat Modeling: Secure coding practices contribute to threat modeling by considering potential security risks and vulnerabilities associated with the code. By following secure coding practices, organizations can proactively identify and mitigate potential threats.
3	SAMM Security by Implementation	Secure Development Training: DevSecOps promotes secure coding practices by providing training to developers, enabling them to implement secure coding practices during the implementation phase. Secure Architecture: Secure coding practices contribute to building a secure architecture during the implementation phase by adhering to secure design patterns and principles. Secure Coding Guidelines: Secure coding practices involve establishing and adopting secure coding guidelines, which provide developers with recommendations for writing secure code. Security Requirements: Secure coding practices align with the need to implement security requirements during the implementation phase, ensuring that security controls are implemented. Security Testing Integration: Secure coding practices encourage the integration of security testing activities, such as static code analysis and security code reviews, during the implementation phase. Security Verification: Secure coding practices contribute to security verification by ensuring that the implemented security controls effectively address identified vulnerabilities. Security Architecture Review: Secure coding practices help establish a secure architecture during the implementation phase, aligning with the need for security architecture reviews. Security Operations Integration: Secure coding practices support the integration of security considerations into operational processes, ensuring that security measures are implemented and maintained throughout the software system's operational life.

#	Name	Description
4	SAMM Security by Verification	Security Testing: Secure coding practices involve conducting security testing activities, such as penetration testing and vulnerability scanning, to identify vulnerabilities in software applications. Code Review: Secure coding practices include code reviews, which help identify security flaws and ensure adherence to secure coding practices. Security Architecture Review: Secure coding practices contribute to the security architecture review process by ensuring that proper security controls are implemented. Security Requirements Verification: Secure coding practices help verify that security requirements are properly implemented during the verification phase. Threat Modeling: Secure coding practices align with the need for threat modeling exercises, which help identify and prioritize security threats. Secure Deployment Verification: Secure coding practices ensure that software applications are securely deployed and configured, addressing the need for deployment verification.
5	SAMM Security by Operations	Policies and Standards: Secure coding practices contribute to the development and implementation of policies and standards related to software security. By incorporating secure coding principles and guidelines, organizations can define expectations and requirements for secure coding practices within their policies and standards. Risk Management: Secure coding practices help mitigate risks associated with software vulnerabilities by reducing the likelihood of introducing security weaknesses during the development process. By following secure coding practices, organizations can address risks related to common security vulnerabilities. Compliance Management: Secure coding practices support compliance with regulatory requirements and industry standards by incorporating security controls and practices that align with these compliance frameworks. Adhering to secure coding practices helps organizations meet the necessary security requirements. Security Training and Awareness: Secure coding practices require ongoing security training and awareness programs to educate developers and software engineers about secure coding principles. By promoting a culture of security awareness, organizations can ensure that secure coding practices are understood and followed by the development team. Security Metrics and Reporting: Secure coding practices contribute to security metrics and reporting by reducing the number of security vulnerabilities introduced during the development process. By adhering to secure coding practices, organizations can demonstrate improved security metrics and report on the effectiveness of their secure coding initiatives. Security Roles and Responsibilities: Secure coding practices help define and assign security roles and responsibilities within the organization. By incorporating secure coding practices into job roles and responsibilities, organizations ensure that individuals have the necessary knowledge and skills to fulfill their security-related tasks effectively.

11.3.18.2 Assessment – OWASP Top 10

The Secure Coding Practices capability directly addresses multiple OWASP threats by incorporating secure coding principles, input validation, secure configurations, proper authentication and authorization mechanisms, secure communication practices, error handling, secure configuration management, regular code reviews and testing, security training and awareness, and other secure coding practices.

#	Name	Description
1	Broken Access Control (A01-2021)	By following secure coding practices, developers can implement proper access controls and enforce them consistently throughout the application. This helps mitigate the risk of unauthorized access and privilege escalation, addressing the Broken Access Control vulnerability.
2	Cryptographic Failures (A02-2021)	Secure coding practices emphasize the correct implementation and usage of cryptographic algorithms and mechanisms. By following recommended cryptographic practices, developers can avoid weaknesses and mistakes in cryptography, reducing the risk of cryptographic failures.
3	Injection (A03-2021)	Secure coding practices focus on input validation and output sanitization. By implementing proper input validation and using secure coding techniques, developers can prevent injection attacks, such as SQL injection and command injection, which fall under the Injection vulnerability category.
4	Insecure Design (A04-2021)	Secure coding practices include considerations for secure design principles. By adopting secure design practices and addressing security considerations during the early stages of development, developers can mitigate the risk of insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Secure coding practices involve managing the configuration of the application securely. By following secure configuration practices and avoiding insecure settings, developers can reduce the risk of security misconfigurations, which can lead to vulnerabilities and unauthorized access.
6	Vulnerable and Outdated Components (A06-2021)	Secure coding practices emphasize the importance of actively managing software dependencies and following secure coding practices. By staying updated with security patches and avoiding the use of outdated or known vulnerable components, developers can reduce the risk of vulnerabilities associated with vulnerable and outdated components.
7	Identification and Authentication Failures (A07-2021)	Secure coding practices focus on implementing secure authentication mechanisms and enforcing strong password policies. By following secure coding practices related to identification and authentication, developers can mitigate the risk of authentication failures and unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Secure coding practices include implementing secure coding techniques and following secure practices. By ensuring the integrity and protection of software components, configurations, and data, developers can mitigate the risk of software and data integrity failures.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	Secure coding practices emphasize the importance of logging and monitoring mechanisms. By implementing comprehensive logging, effective log analysis, and real-time monitoring, developers can address security logging and monitoring failures and improve incident detection and response capabilities.
10	Server-Side Request Forgery (A10-2021)	Secure coding practices focus on input validation and secure configurations. By implementing robust input validation and avoiding vulnerabilities that can be exploited for Server-Side Request Forgery, developers can mitigate the risk associated with this vulnerability.

11.3.19 Security Documentation and Policies

Security Documentation and Policies are an important security capability within the DevSecOps framework. It involves developing and maintaining documentation and policies that outline security practices and procedures within an organization. This ensures that security practices are well-documented, communicated effectively, and consistently followed across teams and projects.

By developing and maintaining comprehensive security documentation and policies, organizations can establish a strong foundation for consistent and effective security practices within their DevSecOps approach. Documentation ensures that security standards, procedures, and best practices are clearly defined and accessible to all stakeholders. It promotes awareness, consistency, and accountability in security practices, leading to better risk management and improved overall security posture.

Here are key aspects of Security Documentation and Policies within the DevSecOps context:

#	Name	Description
1	Security Policies	Security policies provide a framework for defining the organization's approach to security. These policies outline the rules, guidelines, and standards that govern security practices. They cover areas such as data protection, access controls, incident response, vulnerability management, and secure coding practices. Security policies serve as a reference for employees, developers, and other stakeholders to understand their responsibilities and obligations regarding security.
2	Security Standards and Guidelines	Security documentation includes detailed standards and guidelines that specify best practices and requirements for implementing security controls. These standards and guidelines may cover areas such as network security, system hardening, encryption, identity and access management, secure software development, and secure configuration management. By providing clear guidance, organizations can ensure consistent implementation of security measures across projects and systems.
3	Security Procedures	Security procedures outline step-by-step instructions for carrying out specific security-related tasks or processes. These procedures can include incident response procedures, vulnerability management procedures, security incident handling procedures, secure deployment procedures, and security testing procedures. Having well-defined procedures ensures that security-related tasks are executed consistently and efficiently, reducing the risk of errors or omissions.

#	Name	Description
4	Security Awareness and Training Materials	Security documentation includes materials that promote security awareness and provide training resources for employees and developers. This may include security awareness presentations, training modules, online resources, and interactive exercises. By making these materials available, organizations can educate their personnel about security threats, best practices, and the importance of adhering to security policies and procedures.
5	Security Risk Assessments and Reports	Security documentation should include the results of security risk assessments and reports. These assessments evaluate the potential risks and vulnerabilities associated with systems, applications, or processes. They help identify areas of concern and recommend appropriate security controls and mitigations. Risk assessment reports provide valuable insights into the organization's security posture and guide decision-making on security investments and priorities.
6	Incident Response Plans	Security documentation should include incident response plans that outline the procedures and responsibilities for responding to security incidents. These plans define the steps to be taken in the event of a security breach, including incident detection, containment, eradication, and recovery. Incident response plans ensure that a coordinated and timely response is executed, minimizing the impact of security incidents and facilitating the restoration of normal operations.
7	Security Audit and Compliance Documentation	Organizations often need to demonstrate compliance with industry standards and regulations. Security documentation should include audit reports, compliance assessments, and documentation of control implementation. These documents provide evidence of adherence to security standards and regulatory requirements, ensuring that the organization meets its legal and compliance obligations
8	Security Training and Communication Plans	Security documentation should include plans for ongoing security training and communication efforts. This may include scheduling security awareness programs, conducting regular security training sessions, and implementing internal communication channels to share security updates, alerts, and reminders. These plans ensure that security practices are communicated effectively and that employees and developers stay informed about the latest security threats and countermeasures.

11.3.19.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Security Documentation and Policies contribute to the Security by Governance domain by establishing and maintaining comprehensive security documentation and policies that outline security practices and procedures. This helps in defining security policies (sub-domain) and security standards and guidelines (sub-domain) within the organization. Security Risk Assessments and Reports contribute to the Risk Management sub-domain by evaluating potential risks associated with software development and deployment. This helps in identifying and assessing risks (sub-domain) and implementing risk mitigation strategies (sub-domain).

#	Name	Description
		Security Audit and Compliance Documentation contribute to the Compliance Management sub-domain by ensuring compliance with regulatory requirements, industry standards, and best practices related to software security. This helps in compliance management (sub-domain) and alignment with security frameworks and guidelines (sub-domain). Security Training and Communication Plans contribute to the Security Training and Awareness sub-domain by educating and raising awareness among employees and stakeholders about software security. This helps in providing security training and awareness (sub-domain) throughout the organization.
2	SAMM Security by Design	Security Documentation and Policies, specifically Security Requirements, contribute to addressing the Insecure Design sub-domain by incorporating security requirements into the design phase of software development. Secure Architecture practices contribute to addressing Insecure Design, Cryptographic Failures, Security Misconfiguration, and other OWASP threats by promoting the use of secure architectural patterns, threat modeling, and secure design principles. Secure Coding Practices contribute to addressing Injection, Insecure Design, Security Misconfiguration, and other OWASP threats by adhering to secure coding guidelines and preventing common coding mistakes and vulnerabilities. Threat Modeling contributes to addressing various OWASP threats by identifying and mitigating potential security risks through the analysis of the application's architecture, data flows, and attack vectors. Security Testing, including static code analysis and security code reviews, contributes to addressing OWASP threats by detecting and addressing vulnerabilities in the design phase of software development.
3	SAMM Security by Implementation	Security Documentation and Policies, including Secure Coding Guidelines and Security Requirements, contribute to implementing secure development practices. Secure Architecture practices, Secure Coding Guidelines, and Security Testing Integration contribute to building secure software systems and addressing potential vulnerabilities associated with OWASP threats. Security Testing, Security Verification, and Security Architecture Review contribute to verifying the effectiveness of security controls and mitigating vulnerabilities in software applications.
4	SAMM Security by Verification	Security Testing, including penetration testing, vulnerability scanning, code review, and security testing automation, contributes to verifying the security of software applications and detecting vulnerabilities. Code Review, Security Architecture Review, and Security Requirements Verification contribute to verifying the adherence to secure coding practices, the effectiveness of security controls, and the implementation of security requirements. Threat Modeling contributes to identifying and prioritizing security threats and risks, which aids in the verification of security measures.
5	SAMM Security by Operations	Environment Hardening contributes to securing the software environment and addressing potential vulnerabilities. Secure Build practices, Configuration Management, and Vulnerability Management contribute to maintaining the integrity and security of software systems during operations. Incident Management, Secure Deployment, and Security Testing contribute to the ongoing security and reliability of software systems. Operational Enablement activities, such as security training, operational procedures documentation, and incident response capabilities, support the secure operation of software systems.

11.3.19.2 Assessment – OWASP Top 10

Security Documentation and Policies contribute to addressing many OWASP threats; however, it's important to note that it's just one aspect of a comprehensive DevSecOps approach. Other capabilities like secure coding practices, vulnerability scanning, continuous security testing, and incident response are also crucial to effectively address OWASP threats and enhance the overall security posture of web applications.

#	Name	Description
1	Broken Access Control (A01-2021)	Security Policies: By documenting and communicating access control rules, organizations can ensure that proper access controls are defined and enforced consistently, mitigating the risk of unauthorized access and privilege escalation. Security Standards and Guidelines: Documenting access control best practices and requirements helps organizations implement robust access control mechanisms and prevent access control bypass vulnerabilities. Security Procedures: Clearly defined procedures for access control enforcement ensure that access rights are consistently validated, reducing the risk of improper access to sensitive resources.
2	Cryptographic Failures (A02-2021)	Security Policies: Establishing cryptographic policies helps define secure cryptographic practices, ensuring proper algorithm selection, key management, and secure implementation. Security Standards and Guidelines: Documenting cryptographic best practices guides developers in correctly implementing cryptographic algorithms and mechanisms, reducing the risk of vulnerabilities such as weak encryption or improper usage. Security Risk Assessments and Reports: Including cryptographic risk assessments helps identify weaknesses in cryptographic implementations and recommend appropriate mitigations to address cryptographic vulnerabilities.
3	Injection (A03-2021)	Security Policies: Defining secure coding practices and input validation guidelines in policies helps prevent injection vulnerabilities by ensuring that untrusted data is properly handled and sanitized. Security Standards and Guidelines: Documenting secure coding practices, including input validation techniques, helps developers write code that is resilient to injection attacks. Security Procedures: Outlining secure coding and testing procedures ensures that developers follow best practices for input validation, parameterization, and prepared statements, reducing the risk of injection vulnerabilities.
4	Insecure Design (A04-2021)	Security Policies: Including security considerations in design policies ensures that architectural decisions take into account potential threats and risks, minimizing the likelihood of insecure design vulnerabilities. Security Standards and Guidelines: Providing design guidelines with security in mind helps developers make informed decisions to build secure architectures, reducing the risk of vulnerabilities resulting from insecure design choices.

#	Name	Description
5	Security Misconfiguration (A05-2021)	Security Policies: Establishing secure configuration policies helps ensure that systems, frameworks, and web applications are correctly configured to adhere to secure practices and industry standards, reducing the risk of misconfigurations. Security Standards and Guidelines: Documenting secure configuration settings and providing configuration hardening guidelines helps organizations avoid common misconfiguration pitfalls that can lead to security vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Security Policies: Establishing policies for managing software dependencies and conducting regular security assessments helps organizations address vulnerabilities associated with using outdated or known vulnerable components. Security Standards and Guidelines: Providing guidelines for evaluating the security of third-party components and enforcing their regular updates helps mitigate the risk of vulnerabilities stemming from using outdated or vulnerable software.
7	Identification and Authentication Failures (A07-2021)	Security Policies: Defining strong identification and authentication policies helps organizations enforce secure authentication mechanisms, password policies, and session management practices, reducing the risk of unauthorized access. Security Standards and Guidelines: Documenting secure authentication protocols, password hashing algorithms, and session management best practices helps developers implement robust identification and authentication mechanisms.
8	Software and Data Integrity Failures (A08-2021)	Security Policies: Establishing policies for ensuring the integrity and protection of software components, configurations, and data helps prevent unauthorized modifications, tampering, or destruction. Security Standards and Guidelines: Documenting secure coding practices, secure deployment procedures, and data integrity controls helps developers build applications that can detect and prevent software and data integrity failures.
9	Security Logging and Monitoring Failures (A09-2021)	Security Procedures: Security documentation outlines logging and monitoring procedures, ensuring comprehensive logging and real-time monitoring capabilities to detect and respond to security incidents, including SSRF attacks.
10	Server-Side Request Forgery (A10-2021)	Security Policies and Security Procedures: Security documentation should address secure configurations and input validation practices to prevent SSRF vulnerabilities and protect against unintended requests.

11.3.20 Security Incident Response Planning

Security Incident Response Planning is a critical security capability within the DevSecOps framework. It involves developing plans and procedures to respond effectively to security incidents. By having a structured and coordinated approach to handling security incidents, organizations can minimize their impact, contain and mitigate the damage, and facilitate a speedy recovery.

By implementing a robust Security Incident Response Planning capability within DevSecOps, organizations can effectively respond to security incidents, minimize their impact, and learn from each incident to improve

their overall security posture. The goal is to have a proactive and well-prepared approach that reduces the time to detect and respond to incidents, enhances collaboration and communication, and enables efficient recovery and continuous improvement

Here are the key aspects of Security Incident Response Planning within the DevSecOps context:

#	Name	Description
1	Incident Response Team	Establishing an incident response team is essential. This team consists of individuals from various disciplines, such as IT, security, legal, and communications. They are responsible for managing and coordinating the organization's response to security incidents. The team members should have defined roles and responsibilities, ensuring efficient decision-making, communication, and execution during incident response.
2	Incident Response Plan	Developing an incident response plan is crucial for a well-coordinated response. This plan outlines the steps to be followed when a security incident occurs, including the identification, containment, eradication, recovery, and lessons learned. The plan should be documented, regularly updated, and readily accessible to the incident response team and other relevant stakeholders.
3	Incident Classification and Severity Levels	Defining an incident classification framework helps in categorizing and prioritizing incidents based on their severity and impact. This allows the incident response team to allocate resources appropriately and respond in a timely manner. Severity levels can range from low to critical, indicating the urgency and level of response required for each incident.
4	Incident Detection and Reporting	Establishing mechanisms for timely incident detection and reporting is essential. This can include security monitoring tools, intrusion detection systems, log analysis, and user reporting. The incident response plan should define how incidents are reported, who should be notified, and the channels through which incident information is communicated.
5	Escalation and Communication	Clear lines of communication and escalation procedures should be defined in the incident response plan. This ensures that incidents are escalated to appropriate personnel or teams based on their severity and impact. Regular communication updates should be provided to stakeholders, including senior management, IT teams, legal teams, and affected parties, to keep them informed about the incident and the response progress.
6	Incident Investigation and Analysis	Incident response involves investigating the root cause of the incident and conducting a thorough analysis to understand the scope, impact, and potential vulnerabilities that led to the incident. This helps in identifying necessary remediation actions and implementing measures to prevent similar incidents in the future.
7	Containment and Mitigation	Prompt containment of the incident is crucial to prevent further damage and minimize the impact on systems and data. This involves isolating affected systems, blocking access, and implementing temporary mitigations to limit the incident's spread. The incident response plan should include predefined containment strategies based on the nature of the incident and the affected assets.

#	Name	Description
8	Evidence Collection and Preservation	During incident response, it is important to collect and preserve evidence related to the incident. This may include network logs, system logs, memory dumps, and other relevant artifacts. Proper evidence handling ensures that the incident can be investigated thoroughly and supports any potential legal actions or forensic analysis.
9	Recovery and Restoration	Once the incident is contained and mitigated, the focus shifts to recovery and restoration. This involves restoring affected systems and data to a known good state and ensuring that normal operations can resume. Backups, disaster recovery plans, and system restoration procedures play a crucial role in this phase.
10	Lessons Learned and Continuous Improvement	After each incident, conducting a post-incident analysis is important to identify areas for improvement. This includes analyzing the incident response process, identifying any gaps or weaknesses, and implementing corrective actions. Lessons learned from incidents should be documented and shared within the organization to enhance future incident response capabilities.

11.3.20.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Security incident response planning contributes to the establishment and implementation of policies and standards related to incident response. Risk Management: Security incident response planning helps in managing the risks associated with security incidents and supports risk assessment and mitigation. Compliance Management: Security incident response planning helps organizations comply with regulatory requirements by defining incident response processes aligned with relevant standards and regulations. Security Training and Awareness: Security incident response planning includes training and awareness programs to educate employees and stakeholders about incident response and their roles in the process. Security Metrics and Reporting: Security incident response planning establishes mechanisms for measuring and reporting incident-related metrics, which contributes to security metrics and reporting activities. Security Roles and Responsibilities: Security incident response planning defines roles and responsibilities for incident response team members, ensuring accountability and clear security roles within the organization.
2	SAMM Security by Design	Security Requirements: Security incident response planning contributes to the design phase by incorporating incident response requirements into the overall software security requirements. Secure Architecture: Security incident response planning ensures that incident response considerations are integrated into the secure architecture of the software system.

#	Name	Description
		Secure Coding Practices: Security incident response planning promotes secure coding practices, which contribute to addressing vulnerabilities and preventing incidents during the design phase. Threat Modeling: Security incident response planning supports threat modeling exercises during the design phase, helping identify potential security risks and aligning incident response strategies accordingly. Security Testing: Security incident response planning includes security testing activities during the design phase, which aids in assessing the effectiveness of incident response measures.
3	SAMM Security by Implementation	Secure Development Training: Security incident response planning emphasizes the importance of training developers and development teams on incident response practices, contributing to secure development training. Secure Architecture: Security incident response planning ensures that incident response considerations are incorporated into the secure architecture and component selection during implementation. Secure Coding Guidelines: Security incident response planning promotes adherence to secure coding guidelines, which helps prevent vulnerabilities and aligns with secure coding practices during implementation. Security Requirements: Security incident response planning ensures that security requirements, including incident response requirements, are implemented during the development process. Security Testing Integration: Security incident response planning encourages the integration of security testing activities, including incident response testing, during the implementation phase. Security Verification: Security incident response planning supports security verification techniques, such as security code reviews and vulnerability assessments, contributing to security verification practices. Security Architecture Review: Security incident response planning promotes security architecture reviews, which help assess the effectiveness of incident response measures implemented during implementation. Security Operations Integration: Security incident response planning integrates incident management practices into operational processes, contributing to security operations integration.
4	SAMM Security by Verification	Security Testing: Security incident response planning includes security testing activities, such as penetration testing and vulnerability scanning, which contribute to security testing practices during verification. Code Review: Security incident response planning encourages code reviews to identify security flaws, contributing to code review practices during verification. Security Architecture Review: Security incident response planning promotes security architecture reviews, which aid in evaluating the security architecture and aligning with security architecture review practices.

#	Name	Description
		Security Requirements Verification: Security incident response planning ensures that security requirements, including incident response requirements, are properly verified during the verification phase. Threat Modeling: Security incident response planning supports threat modeling exercises, which contribute to identifying and mitigating potential security threats during the verification phase. Secure Deployment Verification: Security incident response planning includes verification of secure deployment practices, which aligns with secure deployment verification activities.
5	SAMM Security by Operations	Security Incident Response Planning - Incident Response Team: The DevSecOps capability of Security Incident Response Planning contributes to the SAMM Security by Operations domain by establishing an incident response team. This team consists of individuals from various disciplines, such as IT, security, legal, and communications, who are responsible for managing and coordinating the organization's response to security incidents. This helps address the SAMM domain's activity of Incident Management by ensuring there is a dedicated team with defined roles and responsibilities to handle security incidents effectively. Security Incident Response Planning - Incident Response Plan: The DevSecOps capability of Security Incident Response Planning also contributes to the SAMM Security by Operations domain through the development of an incident response plan. This plan outlines the steps to be followed when a security incident occurs, including identification, containment, eradication, recovery, and lessons learned. By having a well-documented and regularly updated incident response plan, organizations address the SAMM domain's activity of Incident Management by having a structured approach to handle security incidents. Security Incident Response Planning - Incident Classification and Severity Levels: Defining an incident classification framework is another contribution of the DevSecOps capability of Security Incident Response Planning to the SAMM Security by Operations domain. This framework helps categorize and prioritize incidents based on their severity and impact. By doing so, organizations can allocate resources appropriately and respond in a timely manner, addressing the SAMM domain's activity of Incident Management. Security Incident Response Planning - Incident Detection and Reporting: The capability of Security Incident Response Planning also helps address the SAMM Security by Operations domain through the establishment of mechanisms for timely incident detection and reporting. This includes the use of security monitoring tools, intrusion detection systems, log analysis, and user reporting. By having these mechanisms in place, organizations can detect and report incidents promptly, contributing to the SAMM domain's activity of Incident Management. Security Incident Response Planning - Escalation and Communication: Clear lines of communication and escalation procedures are essential in incident response. The DevSecOps capability of Security Incident Response Planning addresses the SAMM Security by Operations domain by defining these lines of communication and escalation procedures. This ensures that incidents are escalated to appropriate personnel or teams based on their severity and impact, enabling effective communication during incident response.

#	Name	Description
		Security Incident Response Planning - Incident Investigation and Analysis: Incident response involves investigating the root cause of incidents and conducting thorough analysis. The DevSecOps capability of Security Incident Response Planning contributes to the SAMM Security by Operations domain by emphasizing incident investigation and analysis. By conducting these activities, organizations address the SAMM domain's activity of Incident Management by identifying vulnerabilities and implementing necessary remediation actions. Security Incident Response Planning - Containment and Mitigation: The DevSecOps capability of Security Incident Response Planning also helps address the SAMM Security by Operations domain through prompt incident containment and mitigation. This involves isolating affected systems, blocking access, and implementing temporary mitigations to limit the incident's spread. By having predefined containment strategies, organizations address the SAMM domain's activity of Incident Management. Security Incident Response Planning - Evidence Collection and Preservation: Proper evidence collection and preservation are crucial during incident response. The capability of Security Incident Response Planning in DevSecOps helps address the SAMM Security by Operations domain by emphasizing these activities. By collecting and preserving evidence related to incidents, organizations can support forensic analysis and potential legal actions, aligning with the SAMM domain's activity of Incident Management. Security Incident Response Planning - Recovery and Restoration: The DevSecOps capability of Security Incident Response Planning contributes to the SAMM Security by Operations domain by focusing on incident recovery and restoration. This involves restoring affected systems and data to a known good state and ensuring that normal operations can resume.

11.3.20.2 Assessment – OWASP Top 10

The Security Incident Response Planning capability within DevSecOps directly addresses several OWASP threats by providing a structured and coordinated approach to detecting, responding to, and mitigating security incidents. It helps organizations identify vulnerabilities, implement necessary improvements, and enhance the overall security posture of their web applications.

#	Name	Description
1	Broken Access Control (A01-2021)	A well-defined incident response plan helps in detecting and responding to unauthorized access attempts or privilege escalation incidents, minimizing their impact and preventing exposure of sensitive data.
2	Cryptographic Failures (A02-2021)	Incident investigation and analysis within the incident response plan can identify cryptographic weaknesses or mistakes in the implementation of cryptographic algorithms. By addressing these issues, organizations can improve the security and integrity of their cryptographic practices.

#	Name	Description
3	Injection (A03-2021)	Incident response planning can help organizations detect and respond to injection attacks. By analyzing the impact and scope of the incident, organizations can identify the vulnerabilities that allowed the injection and implement necessary remediation actions to prevent future occurrences.
4	Insecure Design (A04-2021)	Incident response planning can identify and address flaws in the design and architecture of web applications. By conducting post-incident analysis, organizations can identify insecure design decisions and implement corrective actions to enhance the overall security of their applications.
5	Security Misconfiguration (A05-2021)	Incident response planning involves investigating and analyzing security incidents, which can uncover security misconfigurations that led to the incident. By addressing these misconfigurations and updating configuration practices, organizations can reduce the risk of future security misconfiguration vulnerabilities.
6	Vulnerable and Outdated Components (A06-2021)	Incident response planning can help identify incidents related to vulnerable or outdated components within web applications. By promptly responding to such incidents, organizations can take necessary actions to update or replace the vulnerable components, reducing the risk of exploitation.
7	Identification and Authentication Failures (A07-2021)	Incident response planning can detect and respond to incidents related to identification and authentication failures. By analyzing these incidents, organizations can identify weaknesses in authentication mechanisms and implement necessary improvements to prevent unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Incident response planning includes investigating incidents related to software and data integrity failures. By identifying vulnerabilities that allowed unauthorized modification or destruction of software or data, organizations can implement measures to ensure integrity and protect against similar incidents in the future.
9	Security Logging and Monitoring Failures (A09-2021)	Incident response planning can address deficiencies in logging and monitoring practices. By improving logging capabilities and implementing real-time monitoring mechanisms, organizations can enhance their ability to detect and respond to security incidents.
10	Server-Side Request Forgery (A10-2021)	Incident response planning can help organizations detect and respond to Server-Side Request Forgery (SSRF) incidents. By analyzing the incident, organizations can identify the vulnerabilities that allowed SSRF attacks and implement necessary measures to prevent their occurrence in the future.

11.3.21 Secure Secrets Management

Secure Secrets Management is an important security capability within the DevSecOps framework. It involves safely storing and managing sensitive information, such as passwords, API keys, cryptographic keys, and other credentials. The goal is to protect critical credentials from unauthorized access and ensure that they are securely used by applications and systems, reducing the risk of unauthorized system access or data breaches.

By implementing a robust Secure Secrets Management capability within DevSecOps, organizations can effectively protect sensitive information, mitigate the risk of unauthorized access, and strengthen the overall security posture of their applications and systems.

Here are the key aspects of Secure Secrets Management within the DevSecOps context:

#	Name	Description
1	Centralized Secrets Storage	Utilizing a centralized and secure storage solution for storing secrets is essential. This can be a dedicated secrets management system or a secure password vault. Centralization ensures that secrets are not scattered across multiple systems or repositories, making it easier to manage and control access.
2	Encryption and Access Controls	Secrets should be encrypted both at rest and in transit to protect them from unauthorized access. Encryption ensures that even if the secrets are compromised, they remain unusable without the corresponding decryption keys. Access controls should be implemented to restrict access to secrets based on the principle of least privilege. Only authorized individuals or systems should have access to specific secrets.
3	Secure Secret Transmission	When secrets are transmitted between systems or users, it is important to use secure channels such as encrypted connections (e.g., TLS/SSL) or secure protocols. This prevents eavesdropping or interception of the secrets during transmission.
4	Secret Rotation	Regularly rotating secrets, such as passwords and cryptographic keys, is a good security practice. This helps mitigate the risk of compromised secrets being used maliciously. Automated mechanisms should be in place to facilitate the secure rotation of secrets without causing disruptions to applications or systems that use them.
5	Secret Lifecycle Management	Secrets have a lifecycle, including creation, usage, expiration, and revocation. It is important to establish processes and policies for managing the entire lifecycle of secrets. This includes procedures for securely generating secrets, tracking their usage, monitoring their validity, and securely revoking or retiring them when no longer needed.
6	Audit and Logging	Logging and auditing mechanisms should be implemented to track and monitor access to secrets. This allows organizations to detect and investigate any suspicious or unauthorized access attempts. Detailed logs should capture information such as who accessed the secret, when it was accessed, and the purpose of access.
7	Integration with Deployment Pipelines	Secrets should be seamlessly integrated into the DevSecOps deployment pipelines. This ensures that applications and systems can securely access the required secrets during runtime without exposing them in configuration files or source code repositories. Secure integration can be achieved through environment variables, secrets injection mechanisms, or integration with secrets management tools.
8	Secure Development Practices	Secure coding practices should be followed to minimize the exposure of secrets within the codebase. This includes avoiding hardcoding of secrets, using secure coding libraries and frameworks, and implementing secure coding guidelines that emphasize the proper handling and protection of sensitive information.

#	Name	Description
9	Regular Security Assessments	Periodic security assessments, such as vulnerability scanning and penetration testing, should be conducted to identify any weaknesses or vulnerabilities in the secrets management process. This helps ensure that the secrets storage and access controls remain secure and effective.

11.3.21.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Secure Secrets Management contributes to the establishment of policies and standards related to the secure storage and management of sensitive information. It helps define expectations and requirements for protecting critical credentials and ensures compliance with security governance policies.
2	SAMM Security by Design	Security Requirements: Secure Secrets Management addresses the security requirement of protecting sensitive information by implementing secure storage and access controls for secrets. Secure Architecture: Secure Secrets Management contributes to the overall secure architecture by providing a secure mechanism for storing and accessing secrets, minimizing the risk of unauthorized access and misuse.
3	SAMM Security by Implementation	Secure Development Training: Secure Secrets Management supports secure development training by promoting secure coding practices related to the proper handling and protection of sensitive information, such as secrets. Security Requirements: Secure Secrets Management helps implement security requirements related to the secure storage and management of secrets. Security Testing Integration: Secure Secrets Management facilitates security testing by providing mechanisms to securely access secrets during testing, ensuring that the security of the secrets management process is verified.
4	SAMM Security by Verification	Security Testing: Secure Secrets Management can be included in security testing activities to validate the effectiveness of the secrets management controls and identify any vulnerabilities or weaknesses. Code Review: Secure Secrets Management practices can be reviewed to ensure adherence to secure coding practices and identify any potential security flaws or vulnerabilities.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening: Secure Secrets Management supports environment hardening by providing secure storage and access controls for secrets, reducing the risk of unauthorized access to critical credentials. Configuration Management: Secure Secrets Management contributes to configuration management by securely managing and tracking secrets as part of the overall configuration management process. Vulnerability Management: Secure Secrets Management helps mitigate vulnerabilities related to unauthorized access to secrets, reducing the risk of exploitation and supporting vulnerability management efforts. Incident Management: Secure Secrets Management can be integrated into the incident management process to address incidents related to compromised secrets and unauthorized access.

11.3.21.2 Assessment – OWASP Top 10

Secrets Management can contribute to addressing these OWASP threats; however, it is not a complete solution on its own. It should be implemented as part of a comprehensive security strategy that considers other security controls and practices.

#	Name	Description
1	Broken Access Control (A01-2021)	By implementing secure secrets management, organizations can ensure that access to sensitive information, such as cryptographic keys and credentials, is properly controlled and restricted. This reduces the risk of unauthorized access to critical resources, mitigating the Broken Access Control vulnerability.
2	Cryptographic Failures (A02-2021)	Secure secrets management involves implementing encryption for secrets both at rest and in transit. This ensures that cryptographic algorithms and mechanisms are correctly applied, reducing the risk of vulnerabilities and weaknesses that could lead to Cryptographic Failures.
3	Injection (A03-2021)	It does not address this domain directly.
4	Insecure Design (A04-2021)	Secure secrets management should be considered during the design phase of applications. By incorporating proper access controls and secure storage mechanisms, organizations can address potential Insecure Design vulnerabilities related to the improper handling of sensitive information.
5	Security Misconfiguration (A05-2021)	Secure secrets management includes ensuring proper configurations for secrets storage, access controls, and transmission. By following secure configuration practices and avoiding insecure defaults, organizations can mitigate Security Misconfiguration vulnerabilities related to the incorrect setup or exposure of sensitive information.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	Secure secrets management involves regularly assessing and updating secrets, such as passwords and cryptographic keys. By actively managing secrets and avoiding the use of outdated or vulnerable components, organizations can reduce the risk of using insecure dependencies and address Vulnerable and Outdated Components vulnerabilities.
7	Identification and Authentication Failures (A07-2021)	Secure secrets management plays a role in authentication mechanisms by securely storing and managing credentials used for user authentication. By implementing proper secrets management, organizations can reduce the risk of Identification and Authentication Failures, such as weak or compromised credentials leading to unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Secure secrets management ensures the integrity of secrets by implementing secure processes for secret generation, rotation, and storage. By protecting the confidentiality and integrity of sensitive information, organizations can mitigate Software and Data Integrity Failures that could result from unauthorized modification or tampering of secrets.
9	Security Logging and Monitoring Failures (A09-2021)	Secure secrets management includes logging and monitoring access to secrets. By implementing comprehensive logging and monitoring mechanisms, organizations can detect and respond to any unauthorized or suspicious access attempts, addressing Security Logging and Monitoring Failures related to inadequate logging and monitoring practices.
10	Server-Side Request Forgery (A10-2021)	It does not address this domain directly.

11.3.22 Security Testing Methodologies

Security Testing Methodologies is a crucial security capability within the DevSecOps framework. It involves the application of systematic and comprehensive testing methodologies to identify and address security vulnerabilities in software applications and systems. The goal is to ensure that security testing is conducted consistently and covers all relevant areas, reducing the risk of overlooking vulnerabilities that could be exploited by attackers.

By incorporating robust Security Testing Methodologies into the DevSecOps approach, organizations can proactively identify and address security vulnerabilities throughout the software development lifecycle. This helps improve the overall security posture of applications and systems, reducing the risk of successful attacks and ensuring the confidentiality, integrity, and availability of critical data and resources.

Here are the key aspects of Security Testing Methodologies within the DevSecOps context:

#	Name	Description
1	Threat Modeling	Prior to conducting security testing, it is important to perform threat modeling. This involves identifying potential threats, understanding their impact and likelihood, and prioritizing them based on risk. Threat modeling helps guide the security testing efforts by focusing on the most critical areas and potential attack vectors.

#	Name	Description
2	Security Test Planning	A well-defined test plan should be created to outline the objectives, scope, and approach of the security testing. The plan should identify the testing techniques, tools, and resources to be used. It should also specify the expected outcomes and success criteria for each test, ensuring that the testing process is consistent and measurable.
3	Vulnerability Scanning	Automated tools are used to scan applications and systems for known vulnerabilities. This helps identify common security weaknesses such as outdated software versions, misconfigurations, and insecure coding practices.
4	Penetration Testing	Controlled simulated attacks are conducted to identify vulnerabilities and weaknesses that may not be discovered by automated scanning tools. Penetration testing involves attempting to exploit vulnerabilities to gain unauthorized access or perform malicious activities, helping to validate the effectiveness of security controls and response mechanisms.
5	Security Code Review	Manual or automated examination of source code is performed to identify potential security vulnerabilities, such as insecure coding practices, insufficient input validation, and insecure data storage.
6	Security Architecture Review	The security architecture of the system is analyzed to ensure that it aligns with industry best practices and design principles. This review helps identify any security gaps or weaknesses in the overall system architecture.
7	Security Testing of APIs and Interfaces	Testing the security of application programming interfaces (APIs) and interfaces with external systems is essential, as they can be potential entry points for attackers. This includes testing for authentication, authorization, input validation, and data integrity.
8	Test Automation	Automation plays a vital role in security testing within the DevSecOps approach. It helps ensure the repeatability and scalability of tests and enables the integration of security testing into the continuous integration and continuous deployment (CI/CD) pipelines. Automated security testing tools can be used to perform scanning, vulnerability assessments, and other security tests on a regular basis.
9	Continuous Monitoring	Security testing should not be a one-time activity but rather a continuous process. Continuous monitoring helps detect and respond to security vulnerabilities and incidents in real-time. This can be achieved through the use of security monitoring tools, intrusion detection systems, and log analysis.
10	Reporting and Remediation	After conducting security tests, it is crucial to document and report the findings in a clear and actionable manner. The reports should include detailed descriptions of vulnerabilities, their potential impact, and recommendations for remediation. The development team can then prioritize and address the identified vulnerabilities based on their severity.
11	Collaboration and Communication	Effective collaboration and communication between security teams, development teams, and operations teams are essential. Security findings and recommendations should be communicated promptly and transparently, fostering a collaborative culture that enables timely resolution of security issues.

11.3.22.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: DevSecOps capability of Security Testing Methodologies contributes to the establishment of policies and standards by providing comprehensive security testing methodologies to identify and address vulnerabilities (Contribution: Does Not Apply Directly). Risk Management: The Security Testing Methodologies capability helps address risk management in SAMM's Security by Governance domain by identifying and assessing risks through threat modeling and prioritizing them based on risk (Contribution: Risk Management). Compliance Management: Security Testing Methodologies support compliance management by scanning applications and systems for known vulnerabilities, ensuring compliance with relevant regulatory requirements and industry standards (Contribution: Compliance Management). Security Training and Awareness: The capability of Security Testing Methodologies indirectly contributes to security training and awareness by promoting a culture of security through collaboration and communication between security teams, development teams, and operations teams (Contribution: Does Not Apply Directly). Security Metrics and Reporting: The Security Testing Methodologies capability includes reporting and remediation, which involves documenting and reporting findings in a clear and actionable manner, supporting security metrics and reporting (Contribution: Security Metrics and Reporting). Security Roles and Responsibilities: The Security Testing Methodologies capability indirectly contributes to security roles and responsibilities by fostering collaboration and communication between teams and ensuring the necessary skills and knowledge are available for conducting security testing (Contribution: Does Not Apply Directly).
2	SAMM Security by Design	Security Requirements: Security Testing Methodologies help address security requirements by incorporating threat modeling and security testing techniques to identify and mitigate potential security risks and vulnerabilities (Contribution: Security Requirements). Secure Architecture: Security Testing Methodologies indirectly contribute to secure architecture by identifying vulnerabilities and weaknesses through security testing and helping to establish secure architectural patterns and principles (Contribution: Secure Architecture). Secure Coding Practices: The capability of Security Testing Methodologies supports secure coding practices by identifying potential security vulnerabilities and providing guidance on secure coding practices during security testing (Contribution: Secure Coding Practices). Threat Modeling: The Security Testing Methodologies capability includes threat modeling as a key aspect, which aligns with the SAMM Security by Design activity of conducting threat modeling exercises to identify and mitigate potential security risks (Contribution: Threat Modeling).

#	Name	Description
		Security Testing: The capability of Security Testing Methodologies directly contributes to security testing in the SAMM Security by Design domain by providing a systematic and comprehensive approach to security testing, including static code analysis, dynamic application security testing, and security code reviews (Contribution: Security Testing).
3	SAMM Security by Implementation	Security Testing Methodologies contribute directly to this domain. They help address activities such as Secure Development Testing, Secure Deployment Testing, and Secure Operations Testing by providing systematic and comprehensive testing methodologies to identify and address security vulnerabilities in software applications and systems.
4	SAMM Security by Verification	Security Testing Methodologies also contribute directly to this domain. They help address activities such as Security Testing, Code Review, and Security Architecture Review by providing testing techniques, including vulnerability scanning, penetration testing, security code review, and security architecture review.
5	SAMM Security by Operations	Security Testing Methodologies contribute directly to this domain. They help address activities such as Secure Operations Testing and Operational Enablement by providing ongoing security testing and monitoring capabilities. Additionally, DevSecOps capabilities such as Continuous Monitoring, Reporting and Remediation, and Collaboration and Communication contribute indirectly to this domain by ensuring continuous monitoring of security vulnerabilities, effective reporting and remediation of security findings, and fostering collaboration and communication between security teams, development teams, and operations teams.

11.3.22.2 Assessment – OWASP Top 10

The capability of Secure Secrets Management within DevSecOps addresses several OWASP threats by providing a robust and secure approach to handling sensitive information, cryptographic keys, and access controls. Proper implementation of this capability strengthens the overall security posture of applications and systems by reducing the risk of unauthorized access, data breaches, and other security.

#	Name	Description
1	Broken Access Control (A01-2021)	Contribution: Secure Secrets Management helps in enforcing proper access controls to sensitive information such as API keys, passwords, and cryptographic keys. By ensuring that access to secrets is restricted based on the principle of least privilege, it reduces the risk of unauthorized access and privilege escalation, thus mitigating the Broken Access Control threat.

#	Name	Description
2	Cryptographic Failures (A02-2021)	Contribution: Secure Secrets Management ensures the proper management and protection of cryptographic keys and other sensitive information used in encryption. By following best practices for storing and rotating cryptographic keys, it helps prevent cryptographic failures, such as weak key management and improper usage, which are common sources of vulnerabilities.
3	Injection (A03-2021)	Contribution: Properly implementing Secure Secrets Management can prevent injection attacks that exploit hardcoded or unencrypted sensitive information. For example, if passwords or API keys are securely stored in a centralized vault, there is a reduced risk of attackers injecting malicious data to manipulate the application's execution flow.
4	Insecure Design (A04-2021)	Contribution: Secure Secrets Management is an essential part of the secure design principles. By having a centralized and secure secrets storage system and proper access controls, it addresses the design weaknesses that could lead to insecure handling of sensitive information.
5	Security Misconfiguration (A05-2021)	Contribution: Secure Secrets Management involves defining and enforcing the proper configuration of access controls and encryption for sensitive data. By eliminating misconfigurations related to secrets storage, it helps prevent potential entry points for attackers, thus mitigating the Security Misconfiguration threat.
6	Vulnerable and Outdated Components (A06-2021)	Contribution: Secure Secrets Management is relevant to this threat as it ensures that sensitive credentials, such as API keys, are securely managed within the application, reducing the risk of exploiting vulnerabilities in third-party components due to exposed or hardcoded credentials.
7	Identification and Authentication Failures (A07-2021)	Contribution: Secure Secrets Management plays a crucial role in proper authentication and identification mechanisms by securely storing and managing user credentials, tokens, and other authentication-related secrets. By ensuring the secure handling of these secrets, it helps mitigate the risk of identification and authentication failures.
8	Software and Data Integrity Failures (A08-2021)	Contribution: Secure Secrets Management is vital for maintaining the integrity of software and data within web applications. By securely managing cryptographic keys and secrets, it prevents unauthorized modification, manipulation, or destruction of sensitive data, addressing the Software and Data Integrity Failures threat.
9	Security Logging and Monitoring Failures (A09-2021)	Contribution: While Secure Secrets Management itself may not directly address this threat, it indirectly contributes to effective logging and monitoring practices. If unauthorized access attempts to sensitive secrets are logged and monitored, security incidents related to secrets management can be detected and mitigated.
10	Server-Side Request Forgery (A10-2021)	Contribution: Secure Secrets Management can mitigate this threat by ensuring that sensitive information (e.g., internal server URLs) is not exposed or accessible externally. By securely handling secrets, it reduces the risk of attackers manipulating the application to perform unauthorized requests to internal resources.

11.3.23 Vulnerability Management

Vulnerability Management is a critical security capability within the DevSecOps framework. It focuses on identifying, assessing, and mitigating vulnerabilities in the system to minimize the risk of exploitation. By actively monitoring for new vulnerabilities, prioritizing their remediation, and applying patches or fixes, organizations can effectively manage their security risks and maintain a robust security posture.

By incorporating robust Vulnerability Management practices within the DevSecOps approach, organizations can effectively identify and mitigate security vulnerabilities in their systems. This proactive approach minimizes the risk of exploitation, enhances the overall security posture, and helps protect critical assets and data from potential threats.

Here are the key aspects of Vulnerability Management within DevSecOps:

#	Name	Description
1	Vulnerability Detection	The first step in Vulnerability Management is the continuous detection of vulnerabilities within the system. This involves using automated vulnerability scanning tools, security testing techniques, and threat intelligence feeds to identify vulnerabilities in software components, applications, infrastructure, and configurations. The scanning process may include network scanning, web application scanning, and code scanning.
2	Vulnerability Assessment	Once vulnerabilities are detected, they need to be assessed to determine their severity, impact, and exploitability. This involves analyzing the vulnerabilities based on standardized vulnerability scoring systems such as the Common Vulnerability Scoring System (CVSS). Vulnerability assessment provides organizations with a prioritized list of vulnerabilities that need to be addressed based on their potential impact on the system.
3	Risk Prioritization	Vulnerability Management includes a risk prioritization process to determine which vulnerabilities should be addressed first. This process takes into account factors such as the severity of the vulnerability, the potential impact on the system, the availability of public exploits, and the value of the asset or data at risk. By prioritizing vulnerabilities, organizations can allocate their resources effectively and address the most critical security risks first.
4	Patch Management	Once vulnerabilities are prioritized, organizations need to apply patches or fixes to remediate the identified vulnerabilities. This involves closely monitoring vendor security advisories, software updates, and patches released by the software providers. Patch management processes ensure that systems and software are kept up-to-date with the latest security patches to close known vulnerabilities.
5	Vulnerability Remediation	Remediation involves taking action to address identified vulnerabilities. This may include applying patches, implementing configuration changes, updating software versions, or reconfiguring systems to eliminate the identified vulnerabilities. It is important to have well-defined processes and procedures for vulnerability remediation, including testing and validation of patches before deployment to avoid any unintended disruptions or conflicts with existing systems.

#	Name	Description
6	Vulnerability Tracking and Reporting	Vulnerability Management also includes tracking and reporting capabilities to keep an ongoing record of identified vulnerabilities, their remediation status, and any associated risks. This helps organizations maintain visibility into their security posture, track the effectiveness of their vulnerability management efforts, and generate reports for compliance audits or management reviews.
7	Continuous Monitoring	Vulnerability Management is an ongoing process that requires continuous monitoring of the system for new vulnerabilities and emerging threats. This includes staying updated with security bulletins, subscribing to vulnerability feeds, and participating in vulnerability disclosure programs. Continuous monitoring helps ensure that organizations can promptly identify and address newly discovered vulnerabilities to maintain a robust security posture.

11.3.23.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Vulnerability Management: The capability contributes to defining and implementing policies and standards related to vulnerability management, aligning with the activity of establishing policies and standards in security governance. Risk Management: Vulnerability Management supports risk management processes by identifying and assessing risks associated with software development, aligning with the activity of risk management in security governance. Compliance Management: The capability aids in ensuring compliance with regulatory requirements and security frameworks, aligning with the activity of compliance management. Security Training and Awareness: Vulnerability Management contributes to educating employees and stakeholders about software security, aligning with the activity of security training and awareness in security governance. Security Metrics and Reporting: The capability provides tracking and reporting features to assess the effectiveness of vulnerability management efforts, supporting the activity of security metrics and reporting in security governance.

#	Name	Description
2	SAMM Security by Design	Security Requirements: DevSecOps contributes to addressing this sub-domain by incorporating security requirements into the design phase of software development. By identifying and addressing specific security concerns, organizations can proactively mitigate vulnerabilities associated with OWASP threats such as Broken Access Control, Injection, Insecure Design, and others. Secure Architecture: DevSecOps promotes the use of secure architectural patterns and principles during the design phase. Designing a robust and secure architecture helps address various OWASP threats such as Insecure Design, Cryptographic Failures, Security Misconfiguration, and others by minimizing potential vulnerabilities and creating a strong foundation for security controls. Secure Coding Practices: DevSecOps encourages the use of secure coding practices during the design phase. By adhering to secure coding guidelines, organizations can reduce the likelihood of OWASP threats such as Injection, Insecure Design, Security Misconfiguration, and others by preventing common coding mistakes and vulnerabilities. Threat Modeling: DevSecOps suggests conducting threat modeling exercises during the design phase to identify and mitigate potential security risks. Threat modeling helps organizations assess the impact of OWASP threats and enables them to implement appropriate security controls and countermeasures. Security Testing: DevSecOps encourages the inclusion of security testing activities in the design phase. By incorporating security testing techniques, such as static code analysis and security code reviews, organizations can detect and address vulnerabilities related to OWASP threats, ensuring that the design is resilient to potential attacks.
3	SAMM Security by Implementation	Vulnerability Detection: DevSecOps' vulnerability management capability contributes to identifying vulnerabilities within the system, addressing the activity of identifying security weaknesses during software development and deployment. Vulnerability Assessment: The capability aids in assessing the severity and impact of vulnerabilities, aligning with the activity of evaluating the severity of identified vulnerabilities. Risk Prioritization: Vulnerability Management enables risk prioritization by considering factors such as vulnerability severity and potential impact on the system, aligning with the activity of prioritizing vulnerabilities based on their potential impact. Patch Management: The capability assists in applying patches or fixes to address vulnerabilities, supporting the activity of implementing security patches. Vulnerability Remediation: By providing processes and procedures for vulnerability remediation, the capability aligns with the activity of taking action to address identified vulnerabilities. Vulnerability Tracking and Reporting: The capability offers tracking and reporting features to maintain visibility into vulnerabilities and their remediation status, supporting the activity of tracking vulnerability remediation efforts and generating reports. Continuous Monitoring: DevSecOps' vulnerability management capability enables continuous monitoring for new vulnerabilities and emerging threats, aligning with the activity of staying updated with security bulletins and vulnerability feeds.

#	Name	Description
4	SAMM Security by Verification	Security Testing: Vulnerability Management, as a part of DevSecOps, includes security testing activities such as vulnerability scanning and penetration testing, contributing to the verification of software security. Code Review: The capability assists in reviewing the source code to identify security flaws, aligning with the activity of code review during security verification. Security Architecture Review: Vulnerability Management can contribute to assessing the security architecture of software applications, aligning with the activity of reviewing the security architecture during verification. Security Requirements Verification: The capability supports the verification of security requirements by providing visibility into vulnerabilities and potential risks, aligning with the activity of verifying security requirements. Threat Modeling: DevSecOps' vulnerability management capability aids in conducting threat modeling exercises, aligning with the activity of analyzing potential threats during security verification. Secure Deployment Verification: The capability can contribute to ensuring secure deployment and configuration of software applications, aligning with the activity of verifying secure deployment.
5	SAMM Security by Operations	Vulnerability Management: The capability supports a robust vulnerability management process, including vulnerability scanning, assessment, and remediation, contributing to secure operations. Incident Management: By addressing vulnerabilities and promptly addressing security incidents, vulnerability management helps minimize the impact of security incidents, aligning with the activity of incident management. Secure Deployment: The capability aids in secure software deployment practices, ensuring that software is deployed securely and controlled during operations. Security Testing: Vulnerability Management, as part of DevSecOps, includes security testing activities to identify and address vulnerabilities, aligning with the activity of security testing during operations.

11.3.23.2 Assessment – OWASP Top 10

Vulnerability Management is a crucial capability within DevSecOps that contributes to addressing various OWASP threats by proactively identifying, assessing, and mitigating vulnerabilities in web applications, infrastructure, and configurations. It helps organizations maintain a robust security posture, protecting critical assets and data from potential threats.

#	Name	Description
1	Broken Access Control (A01-2021)	Vulnerability Detection: Continuous scanning and testing can identify access control weaknesses and misconfigurations that could lead to broken access controls. Vulnerability Assessment: Assessing vulnerabilities helps identify access control flaws that may allow unauthorized access or privilege escalation. Risk Prioritization: Prioritizing vulnerabilities based on their impact and severity can ensure that broken access control issues are addressed promptly. Patch Management: Applying patches and fixes helps remediate vulnerabilities that could lead to broken access controls. Vulnerability Remediation: Taking action to address identified vulnerabilities ensures that access controls are properly enforced. Vulnerability Tracking and Reporting: Tracking and reporting vulnerabilities provides visibility into broken access control issues and their remediation status.
2	Cryptographic Failures (A02-2021)	Vulnerability Detection: Scanning and testing can identify cryptographic weaknesses, such as the use of weak algorithms or insecure cryptographic configurations. Vulnerability Assessment: Assessing cryptographic vulnerabilities helps determine the impact and severity of these weaknesses. Risk Prioritization: Prioritizing cryptographic vulnerabilities ensures that they are addressed in a timely manner to prevent exploitation. Patch Management: Applying patches or updates to cryptographic libraries and components can mitigate known vulnerabilities. Vulnerability Remediation: Implementing secure cryptographic practices and configurations helps remediate cryptographic failures. Vulnerability Tracking and Reporting: Tracking and reporting cryptographic vulnerabilities helps monitor the effectiveness of remediation efforts.
3	Injection (A03-2021)	Vulnerability Detection: Automated scanning and manual testing can identify injection vulnerabilities by detecting improper handling of untrusted data. Vulnerability Assessment: Assessing injection vulnerabilities helps determine their impact and exploitability. Risk Prioritization: Prioritizing injection vulnerabilities based on their severity and potential impact ensures they are addressed promptly. Patch Management: Applying patches or updates to frameworks or libraries helps remediate injection vulnerabilities. Vulnerability Remediation: Implementing secure coding practices, input validation, and parameterization helps mitigate injection vulnerabilities. Vulnerability Tracking and Reporting: Tracking and reporting injection vulnerabilities helps monitor their remediation progress and effectiveness.

#	Name	Description
4	Insecure Design (A04-2021)	Vulnerability Detection: Scanning and testing can identify insecure design decisions that may lead to vulnerabilities. Vulnerability Assessment: Assessing insecure design vulnerabilities helps determine their impact and potential for exploitation. Risk Prioritization: Prioritizing insecure design vulnerabilities ensures that they are addressed during the development process. Patch Management: Applying updates or making design changes helps remediate insecure design vulnerabilities. Vulnerability Remediation: Adopting secure design principles and implementing appropriate security controls mitigate insecure design vulnerabilities. Vulnerability Tracking and Reporting: Tracking and reporting insecure design vulnerabilities helps monitor their remediation progress.
5	Security Misconfiguration (A05-2021)	Vulnerability Detection: Scanning and testing can identify security misconfigurations in web applications and associated components. Vulnerability Assessment: Assessing security misconfigurations helps determine their impact and the potential for unauthorized access. Risk Prioritization: Prioritizing security misconfigurations ensures that they are addressed promptly to avoid exposure of sensitive information. Patch Management: Applying updates or making configuration changes helps remediate security misconfigurations. Vulnerability Remediation: Configuring systems and web applications according to secure practices helps mitigate security misconfigurations. Vulnerability Tracking and Reporting: Tracking and reporting security misconfigurations helps monitor their remediation progress.
6	Vulnerable and Outdated Components (A06-2021)	Vulnerability Management is instrumental in detecting vulnerabilities arising from the use of outdated or known vulnerable software components. By promptly patching or replacing these components, organizations can reduce the risk of exploitation through these dependencies.
7	Identification and Authentication Failures (A07-2021)	Vulnerability Management contributes by identifying weaknesses in authentication mechanisms, which are crucial for preventing unauthorized access. By prioritizing the resolution of these vulnerabilities, organizations can ensure stronger identification and authentication controls.
8	Software and Data Integrity Failures (A08-2021)	Vulnerability Management helps in identifying vulnerabilities that can compromise the integrity of software and data in web applications. By fixing these weaknesses, organizations can ensure that the application's components and data remain unaltered and secure.
9	Security Logging and Monitoring Failures (A09-2021)	Vulnerability Management supports improved logging and monitoring capabilities, which are essential for detecting and responding to security incidents promptly. By addressing these issues, organizations can better protect their applications against attacks and breaches.

#	Name	Description
10	Server-Side Request Forgery (A10-2021)	Vulnerability Management aids in detecting and addressing vulnerabilities that could lead to server-side request forgery. By fixing these issues, organizations can prevent attackers from manipulating the application to make unauthorized requests.

11.3.24 Security Auditing and Compliance

Security Auditing and Compliance is a crucial security capability within the DevSecOps framework. It involves conducting regular audits and assessments to ensure compliance with security standards, regulations, and best practices. By automating these audits, organizations can identify potential security gaps, assess their security posture, and ensure adherence to industry-specific requirements.

By incorporating Security Auditing and Compliance practices within DevSecOps, organizations can maintain a strong security posture, meet regulatory requirements, and effectively manage security risks. Automating audits and assessments allows for continuous monitoring, reduces manual effort, and ensures that security controls are consistently evaluated and maintained. This proactive approach helps organizations identify and address security gaps, improve their overall security posture, and build trust with customers and stakeholders.

Here are the key aspects of Security Auditing and Compliance within DevSecOps:

#	Name	Description
1	Compliance Frameworks	Security Auditing and Compliance involve following established compliance frameworks and standards relevant to the industry and regulatory requirements. These frameworks may include industry-specific standards such as PCI DSS (Payment Card Industry Data Security Standard), HIPAA (Health Insurance Portability and Accountability Act), GDPR (General Data Protection Regulation), ISO 27001, NIST (National Institute of Standards and Technology), and others. Adhering to these frameworks helps organizations meet legal and regulatory obligations, protect sensitive data, and maintain trust with customers and partners.
2	Automated Auditing	To efficiently manage security audits, organizations leverage automation tools and technologies that enable the automated collection of security-related data and perform audits against predefined security controls. These tools can scan systems, networks, and applications for compliance violations, vulnerabilities, misconfigurations, and policy deviations. Automated audits save time and resources, provide consistent results, and help organizations identify non-compliant areas.
3	Compliance Assessments	Security Auditing and Compliance include conducting comprehensive assessments to evaluate the organization's adherence to security standards and regulations. These assessments may involve reviewing security policies, processes, and controls, as well as conducting technical evaluations of the systems and applications. Compliance assessments help identify areas of non-compliance, vulnerabilities, and gaps in security controls, allowing organizations to prioritize remediation efforts.

#	Name	Description
4	Gap Analysis	Following compliance assessments, a gap analysis is performed to identify the discrepancies between the current security posture and the desired state of compliance. This analysis helps organizations understand the extent of non-compliance and determine the necessary actions to address identified gaps. It may involve reviewing existing security controls, policies, procedures, and documentation, as well as assessing the effectiveness of security measures in place.
5	Remediation and Continuous Improvement	Based on the findings from compliance assessments and gap analysis, organizations undertake remediation activities to address non-compliance issues and improve their security posture. This may involve implementing additional security controls, updating policies and procedures, providing employee training, and conducting regular vulnerability scans and security testing. Continuous improvement efforts help organizations maintain compliance, mitigate risks, and adapt to evolving security threats and regulatory requirements.
6	Reporting and Documentation	Security Auditing and Compliance require comprehensive reporting and documentation to demonstrate adherence to security standards and regulations. Organizations should maintain detailed records of audit findings, remediation actions taken, and evidence of compliance. These reports can be used for internal purposes, such as management reviews and decision-making, as well as for external audits and regulatory compliance requirements.
7	Integration with DevOps Processes	To ensure security auditing and compliance are seamlessly integrated into the DevSecOps approach, organizations embed security controls, audits, and compliance checks throughout the software development lifecycle. This includes incorporating security requirements into the development process, performing code reviews, automating security testing, and integrating security checks into the CI/CD pipeline. By integrating security measures from the early stages of development, organizations can identify and address security issues proactively.

11.3.24.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: DevSecOps incorporates security auditing and compliance practices, ensuring adherence to established policies and standards. Risk Management: DevSecOps integrates risk management processes, conducting risk assessments and implementing risk mitigation strategies. Compliance Management: DevSecOps automates compliance assessments, assists with regulatory reporting, and ensures alignment with security frameworks. Security Training and Awareness: DevSecOps promotes security training programs and awareness campaigns to educate employees and stakeholders. Security Metrics and Reporting: DevSecOps provides automated auditing and reporting mechanisms, enabling visibility into the security status. Security Roles and Responsibilities: DevSecOps defines and assigns security roles, ensuring accountability and knowledge among individuals.
2	SAMM Security by Design	Security Auditing and Compliance: Security Auditing and Compliance capability within DevSecOps indirectly contributes to the Secure Architecture activity of the SAMM Security by Design domain. By automating audits and assessments, organizations can identify security gaps and vulnerabilities in the software architecture, which helps address the Insecure Design threat. Security Testing: The Security Testing capability within DevSecOps directly contributes to the Threat Modeling activity of the SAMM Security by Design domain. By conducting security testing activities such as penetration testing and vulnerability scanning, organizations can identify potential threats and risks in the software design, helping to mitigate vulnerabilities associated with OWASP threats.

#	Name	Description
3	SAMM Security by Implementation	Security Auditing and Compliance: Security Auditing and Compliance capability within DevSecOps contributes to the Secure Deployment Verification activity of the SAMM Security by Implementation domain. By ensuring compliance with security standards and best practices, organizations can verify that secure deployment practices are followed. Security Testing: The Security Testing capability within DevSecOps contributes to the Security Verification and Security Operations Integration activities of the SAMM Security by Implementation domain. By conducting security testing activities, organizations can verify the effectiveness of security controls implemented during the software development process and ensure that security measures are integrated into operational processes. Security Verification: The Security Verification capability within DevSecOps directly contributes to the Security Verification activity of the SAMM Security by Implementation domain. By verifying the implementation of security controls and conducting security code reviews, organizations can ensure that the software system meets the desired security objectives. Security Operations Integration: The Security Operations Integration capability within DevSecOps contributes to the Security Operations Integration activity of the SAMM Security by Implementation domain. By integrating security considerations into operational processes, such as incident management and secure deployment practices, organizations can ensure that security measures are supported and maintained throughout the operational life of the software system.
4	SAMM Security by Verification	Security Auditing and Compliance: Security Auditing and Compliance capability within DevSecOps contributes to the Security Requirements Verification activity of the SAMM Security by Verification domain. By ensuring compliance with security requirements, organizations can verify that security objectives are properly defined and implemented in software applications. Security Testing: The Security Testing capability within DevSecOps directly contributes to the Security Testing and Code Review activities of the SAMM Security by Verification domain. By conducting security testing activities and code reviews, organizations can identify security vulnerabilities and weaknesses in software applications.
5	SAMM Security by Operations	Security Auditing and Compliance: Security Auditing and Compliance capability within DevSecOps contributes to the Vulnerability Management and Incident Management activities of the SAMM Security by Operations domain. By conducting regular audits, vulnerability scanning, and incident management processes, organizations can identify and address security vulnerabilities and effectively respond to security incidents. Security Operations Integration: The Security Operations Integration capability within DevSecOps contributes to all activities of the SAMM Security by Operations domain. By integrating security considerations into operational processes, organizations can ensure the ongoing security and reliability of software systems. This includes environment hardening, secure build practices, configuration management, vulnerability management, incident management, and secure deployment practices.

11.3.24.2 Assessment – OWASP Top 10

The Security Auditing and Compliance capability within DevSecOps directly addresses multiple OWASP threats by automating audits, assessing access controls, evaluating cryptographic practices, validating input handling, reviewing design decisions, addressing misconfigurations, managing software dependencies, enhancing authentication mechanisms, ensuring data integrity, implementing logging and monitoring, and preventing server-side request forgery vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	By conducting regular security audits and assessments, organizations can identify potential flaws in the enforcement of access controls within their web applications. This helps ensure that proper access controls are implemented and enforced, reducing the risk of unauthorized access and privilege escalation.
2	Cryptographic Failures (A02-2021)	Security Auditing and Compliance involves assessing the implementation of cryptographic algorithms and mechanisms within web applications. Through automated audits, organizations can identify weaknesses and mistakes in cryptographic practices, such as the use of weak algorithms or improper key management. By addressing these vulnerabilities, the security and integrity of the application's cryptographic functions can be improved.
3	Injection (A03-2021)	Security Auditing and Compliance practices include comprehensive assessments to evaluate the adherence of web applications to secure coding practices and input validation techniques. By identifying vulnerabilities related to improper handling of untrusted data, organizations can take corrective actions to prevent injection attacks and the unauthorized execution of malicious code.
4	Insecure Design (A04-2021)	Security Auditing and Compliance efforts involve reviewing the overall design and architecture of web applications. By identifying and addressing insecure design decisions, organizations can mitigate the risk of vulnerabilities that can be exploited by attackers. This includes considering security from the early stages of development and adopting secure design principles.
5	Security Misconfiguration (A05-2021)	Security Auditing and Compliance practices help identify and rectify insecure configurations within web applications. By automating audits and assessments, organizations can detect and address misconfigurations that may expose sensitive information or create entry points for attackers. Regular auditing ensures that configurations remain secure and aligned with industry standards.
6	Vulnerable and Outdated Components (A06-2021)	Security Auditing and Compliance efforts include assessing the usage of third-party software components within web applications. By identifying outdated or known vulnerable components, organizations can take appropriate actions, such as updating or replacing them, to reduce the risk of exploitation and enhance the overall security of the application.
7	Identification and Authentication Failures (A07-2021)	Security Auditing and Compliance practices involve evaluating the identification and authentication mechanisms of web applications. By conducting comprehensive assessments, organizations can identify weaknesses in authentication processes and take steps to strengthen them. This helps prevent unauthorized access and impersonation attacks

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	Security Auditing and Compliance efforts include reviewing the integrity protection measures of web applications. By ensuring the proper implementation of secure coding techniques and performing regular vulnerability testing, organizations can reduce the risk of unauthorized modifications, data tampering, and integrity-related issues.
9	Security Logging and Monitoring Failures (A09-2021)	Security Auditing and Compliance practices emphasize the importance of effective logging and monitoring capabilities within web applications. By implementing comprehensive logging, analysis, and real-time monitoring mechanisms, organizations can detect and respond to security incidents in a timely manner, reducing the impact of potential breaches.
10	Server-Side Request Forgery (A10-2021)	Security Auditing and Compliance efforts involve assessing the robustness of input validation and configurations within web applications. By identifying and addressing vulnerabilities related to server-side request forgery, organizations can prevent attackers from tricking the application into making unintended requests to internal or external resources.

11.3.25 Secure Deployment Pipeline

The Secure Deployment Pipeline is a critical security capability within the DevSecOps framework. It focuses on ensuring that the entire deployment pipeline is secure and free from vulnerabilities. This capability involves implementing secure practices and measures at each stage of the deployment process to protect the integrity and security of the software being deployed.

By integrating a Secure Deployment Pipeline into the DevSecOps approach, organizations can ensure that software is deployed securely, reducing the risk of vulnerabilities, unauthorized code changes, and potential security breaches. Implementing code signing, secure artifact management, secure configuration management, and incorporating automated security controls throughout the pipeline helps maintain the integrity and trustworthiness of the deployed software. Additionally, security monitoring and logging provide visibility into the deployment process, enabling prompt detection and response to any security incidents or breaches. Overall, the Secure Deployment Pipeline capability strengthens the security posture of the organization and enhances the resilience of the deployed software.

Here are key aspects of the Secure Deployment Pipeline within DevSecOps:

#	Name	Description
1	Code Signing	Code signing is an essential practice in securing the deployment pipeline. It involves digitally signing the software artifacts to verify their authenticity and integrity. Code signing helps prevent tampering with the code during transit or deployment, ensuring that only trusted and verified code is executed. This helps mitigate the risk of deploying malicious or compromised code.

#	Name	Description
2	Secure Artifact Management	Secure artifact management is crucial for maintaining the integrity and security of the software artifacts throughout the deployment pipeline. It involves securely storing and managing the artifacts, such as compiled binaries, container images, and configuration files, in a controlled and protected environment. Secure artifact management includes implementing access controls, encryption, and integrity checks to prevent unauthorized access or modifications to the artifacts.
3	Secure Configuration Management	Secure configuration management ensures that the deployment pipeline and the underlying infrastructure are configured securely. This involves implementing security best practices and hardening measures to minimize vulnerabilities and potential attack vectors. Secure configuration management includes practices such as using secure default configurations, disabling unnecessary services or features, and applying patches and updates promptly to address known vulnerabilities.
4	Continuous Integration and Continuous Deployment (CI/CD) Security	The secure deployment pipeline integrates security practices into the CI/CD process. This includes incorporating security checks, such as static code analysis and vulnerability scanning, into the build and deployment stages. These checks help identify security flaws, vulnerabilities, and configuration errors early in the pipeline, allowing for prompt remediation before code is deployed to production.
5	Secure Release Management	Secure release management ensures that the deployment process follows secure practices when promoting software from one environment to another. It involves conducting rigorous testing, such as functional and security testing, before deploying code to production. Secure release management also includes implementing change management processes to track and control code releases, minimizing the risk of deploying untested or unauthorized changes.
6	Automated Security Controls	To enhance the security of the deployment pipeline, automated security controls are implemented. These controls include automated vulnerability scanning, security testing, and compliance checks at various stages of the pipeline. Automated security controls help identify security issues and vulnerabilities in real-time, allowing for quick remediation and reducing the window of exposure to potential attacks.
7	Security Monitoring and Logging	Security monitoring and logging are essential components of the secure deployment pipeline. It involves implementing mechanisms to monitor and log relevant security events and activities throughout the deployment process. This includes capturing and analyzing logs, performing security incident detection and response, and conducting regular audits to identify and investigate any suspicious or unauthorized activities.

11.3.25.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: DevSecOps capability of Secure Deployment Pipeline contributes to the development and implementation of policies and standards for software security. It ensures that secure practices and measures are implemented at each stage of the deployment process, aligning with the organization's expectations and requirements for software security. Risk Management: The Secure Deployment Pipeline capability helps organizations assess and manage risks associated with software deployment. By integrating security practices into the deployment pipeline, vulnerabilities and potential security breaches can be mitigated, reducing the overall risk to the organization. Compliance Management: The Secure Deployment Pipeline incorporates security controls and measures that align with relevant regulatory requirements, industry standards, and best practices. It helps organizations ensure compliance with security frameworks and guidelines during the deployment process. Security Training and Awareness: DevSecOps promotes a culture of security through secure deployment practices. The Secure Deployment Pipeline capability provides training programs and resources to developers and operational staff, raising awareness about software security and promoting secure practices. Security Metrics and Reporting: The Secure Deployment Pipeline capability includes security monitoring and logging, which contribute to the establishment of security metrics and reporting mechanisms. It provides visibility into the deployment process, enabling organizations to assess the effectiveness of software security practices and support decision-making. Security Roles and Responsibilities: The Secure Deployment Pipeline capability involves clearly defining and assigning security roles and responsibilities within the deployment process. It ensures that individuals have the necessary knowledge and skills to fulfill their roles effectively and contributes to establishing accountability for software security.
2	SAMM Security by Design	Security Requirements: The Secure Deployment Pipeline capability ensures that security requirements are considered and implemented throughout the design phase of software development. By incorporating secure practices into the deployment pipeline, organizations can address security concerns associated with specific OWASP threats, such as Broken Access Control, Injection, and Insecure Design. Secure Architecture: The Secure Deployment Pipeline capability promotes the use of secure architectural patterns and principles. It helps organizations design robust and secure architectures that address OWASP threats like Insecure Design, Cryptographic Failures, and Security Misconfiguration. Secure Coding Practices: The Secure Deployment Pipeline capability includes secure coding practices, which contribute to addressing OWASP threats such as Injection, Insecure Design, and Security Misconfiguration. By following secure coding guidelines, organizations can reduce the likelihood of introducing vulnerabilities during the design phase. Threat Modeling: While not directly addressed by the Secure Deployment Pipeline capability, organizations can conduct threat modeling exercises during the design phase to identify and mitigate potential security risks associated with OWASP threats. Threat modeling helps assess the impact of specific threats and enables the implementation of appropriate security controls and countermeasures.

#	Name	Description
		Security Testing: The Secure Deployment Pipeline capability encourages the integration of security testing activities during the design phase. By incorporating security testing techniques like static code analysis and security code reviews, organizations can detect and address vulnerabilities related to OWASP threats early on in the development process.
3	SAMM Security by Implementation	Secure Development Training: The Secure Deployment Pipeline capability aligns with the need for secure development training. By integrating security practices into the deployment process, it promotes secure coding practices, secure design principles, and common software vulnerability awareness among developers and development teams. Secure Architecture: The Secure Deployment Pipeline capability contributes to the implementation of secure architecture practices. It helps organizations adopt secure design patterns, conduct threat modeling, and make secure component selections, thereby reducing potential vulnerabilities during software development. Secure Coding Guidelines: The Secure Deployment Pipeline capability supports the establishment and adoption of secure coding guidelines and standards. It helps organizations provide developers with specific recommendations and best practices for writing secure code, addressing common vulnerabilities in areas like input validation, output encoding, access controls, and error handling. Security Requirements: The Secure Deployment Pipeline capability ensures that security requirements are documented and implemented as part of the software development process. It aligns with SAMM's emphasis.
4	SAMM Security by Verification	Security Testing: The Secure Deployment Pipeline incorporates security testing practices, such as penetration testing, vulnerability scanning, and code review, to identify vulnerabilities and weaknesses in software applications. Code Review: The Secure Deployment Pipeline includes code review as part of the deployment process, helping to identify security flaws and adherence to secure coding practices. Security Architecture Review: The Secure Deployment Pipeline ensures that proper security controls and mechanisms are in place by conducting security architecture reviews during the deployment process. Security Requirements Verification: The Secure Deployment Pipeline verifies that security requirements are properly defined, documented, and implemented in software applications during the deployment process. Threat Modeling: The Secure Deployment Pipeline can incorporate threat modeling exercises to identify and prioritize security threats during the deployment process. Secure Deployment Verification: The Secure Deployment Pipeline ensures that software applications are securely deployed and configured in the production environment, verifying the implementation of security controls, encryption mechanisms, and access controls.
5	SAMM Security by Operations	Environment Hardening: The Secure Deployment Pipeline includes practices to harden the software environment by implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations. Secure Build: The Secure Deployment Pipeline promotes secure build practices, including secure coding standards, secure build tools, and secure software development frameworks. Configuration Management: The Secure Deployment Pipeline advocates for effective configuration management practices, such as version control, change management, and documentation of software configurations.

#	Name	Description
		Vulnerability Management: The Secure Deployment Pipeline incorporates vulnerability management processes, including vulnerability scanning and assessment, to identify and prioritize vulnerabilities. Incident Management: The Secure Deployment Pipeline supports incident management by providing security monitoring and logging capabilities, enabling prompt detection and response to security incidents. Secure Deployment: The Secure Deployment Pipeline ensures that software is deployed in a secure and controlled manner, following secure software distribution, installation procedures, and configuration of software components. Security Testing: The Secure Deployment Pipeline includes security testing activities, such as static code analysis, dynamic application security testing (DAST), and security code reviews, to identify and address security vulnerabilities during deployment. Operational Enablement: The Secure Deployment Pipeline supports operational enablement by providing operational support, guidance, and documentation to ensure ongoing security and reliability of software systems.

11.3.25.2 Assessment – OWASP Top 10

The Secure Deployment Pipeline capability enhances the security posture of the organization by addressing multiple OWASP threats and ensuring that the deployment process is secure and free from vulnerabilities.

#	Name	Description
1	Broken Access Control (A01-2021)	The Secure Deployment Pipeline ensures that proper access controls are implemented and enforced throughout the deployment process. This helps prevent unauthorized access to resources and protects against privilege escalation.
2	Cryptographic Failures (A02-2021)	The Secure Deployment Pipeline incorporates code signing practices, which involve digitally signing software artifacts to verify their authenticity and integrity. This helps mitigate the risk of deploying software with weak or improperly implemented cryptographic algorithms.
3	Injection (A03-2021)	The Secure Deployment Pipeline includes security checks such as static code analysis and vulnerability scanning. These checks help identify and mitigate injection vulnerabilities by detecting improper handling of untrusted data.
4	Insecure Design (A04-2021)	The Secure Deployment Pipeline emphasizes the importance of secure design principles and implementing appropriate security controls. By considering security from the early stages of development, it helps mitigate insecure design decisions that could lead to vulnerabilities.
5	Security Misconfiguration (A05-2021)	The Secure Deployment Pipeline promotes secure configuration management practices. This ensures that the deployment pipeline and underlying infrastructure are properly configured, minimizing the risk of security misconfigurations.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	The Secure Deployment Pipeline includes automated vulnerability scanning and compliance checks. By continuously monitoring for vulnerabilities in software components, it helps identify and address outdated or known vulnerable components before they are deployed.
7	Identification and Authentication Failures (A07-2021)	The Secure Deployment Pipeline incorporates secure authentication mechanisms and enforces strong password policies. This helps mitigate vulnerabilities related to identification and authentication, reducing the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	The Secure Deployment Pipeline promotes the integrity of software components, configurations, and data. By implementing secure coding techniques, it reduces the risk of unauthorized modifications or data tampering.
9	Security Logging and Monitoring Failures (A09-2021)	The Secure Deployment Pipeline includes security monitoring and logging mechanisms. This helps address vulnerabilities related to the lack of effective logging and monitoring, enabling timely detection and response to security incidents.
10	Server-Side Request Forgery (A10-2021)	The Secure Deployment Pipeline incorporates robust input validation and secure configurations, which help mitigate vulnerabilities associated with server-side request forgery. By preventing unintended requests to internal or external resources, it reduces the risk of unauthorized access and potential exploitation.

11.3.26 Security Monitoring of Third-Party Components

Security Monitoring of Third-Party Components is a crucial security capability within the DevSecOps framework. It focuses on monitoring and assessing the security of the third-party components used in the system to ensure they do not introduce vulnerabilities or pose security risks.

By incorporating the Security Monitoring of Third-Party Components capability into the DevSecOps approach, organizations can proactively identify and mitigate potential security risks associated with the use of third-party components. Continuously monitoring for vulnerabilities, managing patches and updates, conducting security assessments, and maintaining strong vendor relationships help ensure the security and integrity of the overall system. By addressing potential security risks in third-party components, organizations can enhance their resilience against external threats and reduce the likelihood of security breaches or vulnerabilities stemming from these components.

Here are key aspects of the Security Monitoring of Third-Party Components capability within DevSecOps:

#	Name	Description
1	Third-Party Component Inventory	The first step in security monitoring is to establish an inventory of all the third-party components used in the system. This includes libraries, frameworks, plugins, modules, and other software components obtained from external sources. Maintaining a comprehensive inventory helps in tracking and monitoring the security of these components effectively.

#	Name	Description
2	Vulnerability Monitoring	Security monitoring involves continuously monitoring for vulnerabilities and security issues in the third-party components. This includes subscribing to vulnerability databases and security advisories specific to the components being used. By staying up-to-date with the latest vulnerabilities, organizations can assess the impact on their systems and take necessary actions to mitigate the risks.
3	Patch and Update Management	It is essential to actively manage patches and updates for third-party components. This involves regularly checking for available patches or updates and promptly applying them to address known vulnerabilities. Timely patch management helps prevent exploitation of vulnerabilities and ensures the security of the system.
4	Security Assessment and Testing	In addition to monitoring for known vulnerabilities, conducting security assessments and testing of third-party components is crucial. This can include activities such as penetration testing, code review, and vulnerability scanning specifically targeting the third-party components. By assessing the security of these components, organizations can identify any potential weaknesses or security gaps.
5	Vendor Relationship Management	Building and maintaining strong relationships with third-party component vendors is essential for effective security monitoring. Establishing communication channels with vendors allows organizations to receive timely notifications about security updates, vulnerabilities, and any other security-related information. Proactive engagement with vendors helps in better understanding the security posture of the components and enables prompt response to emerging security issues.
6	Risk Mitigation Strategies	Based on the identified vulnerabilities and security risks associated with third-party components, organizations should develop and implement risk mitigation strategies. This may involve mitigating controls, such as implementing additional security measures around the components, considering alternative components, or even developing contingency plans in case vulnerabilities cannot be addressed in a timely manner.
7	Incident Response and Remediation	Security monitoring of third-party components also includes incident response and remediation processes. If a security incident occurs or a vulnerability is discovered, organizations should have a well-defined process to respond, investigate, and remediate the issue. This may involve communication with the vendor, applying temporary fixes, or finding alternative solutions until a permanent fix becomes available.

11.3.26.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: DevSecOps contributes to the Security by Governance domain by implementing and enforcing security policies and standards related to the monitoring and management of third-party components. This ensures that the organization has guidelines in place for assessing the security of these components and mitigating associated risks.

#	Name	Description
		Risk Management: DevSecOps helps address risk management in the Security by Governance domain by incorporating processes and practices to identify and mitigate risks associated with third-party components. This includes continuously monitoring for vulnerabilities and assessing the impact of these risks on the overall system. Compliance Management: DevSecOps contributes to compliance management by ensuring that third-party components used in the system comply with relevant regulatory requirements and industry standards. This involves monitoring the security posture of these components, conducting assessments, and aligning with security frameworks and guidelines. Security Training and Awareness: DevSecOps supports security training and awareness initiatives by providing training programs and resources to educate employees and stakeholders about the risks and security considerations related to third-party components. Security Metrics and Reporting: DevSecOps helps establish security metrics and reporting mechanisms to assess the effectiveness of security practices related to third-party components. By monitoring and reporting on key security metrics, organizations can demonstrate compliance and make informed decisions regarding the security of these components. Security Roles and Responsibilities: DevSecOps contributes to the Security by Governance domain by clearly defining and assigning security roles and responsibilities within the organization. This ensures that individuals have the necessary knowledge and skills to effectively monitor and manage the security of third-party components.
2	SAMM Security by Design	Security Requirements: DevSecOps helps address security requirements in the SAMM Security by Design domain by incorporating security considerations related to third-party components into the design phase of software development. This ensures that security requirements specific to these components are identified and addressed. Secure Architecture: DevSecOps contributes to secure architecture practices by considering the security implications of third-party components during the design phase. This includes selecting secure components, applying secure design patterns, and integrating these components into the overall system architecture. Secure Coding Practices: While secure coding practices are more directly applicable to in-house development, DevSecOps can still promote secure coding practices for the integration and customization of third-party components. This helps mitigate vulnerabilities and weaknesses associated with the use of these components. Threat Modeling: DevSecOps can incorporate threat modeling exercises that specifically focus on the security risks and threats associated with third-party components. This helps identify potential vulnerabilities and inform the design decisions related to these components. Security Testing: DevSecOps supports security testing activities during the design phase, including testing of third-party components. By incorporating security testing techniques, organizations can detect and address vulnerabilities and weaknesses introduced by these components.
3	SAMM Security by Implementation	Third-Party Component Inventory: DevSecOps contributes to establishing and maintaining an inventory of third-party components, addressing the sub-domain of "Component Inventory Management." Vulnerability Monitoring: DevSecOps enables continuous monitoring for vulnerabilities in third-party components, addressing the sub-domain of "Vulnerability Monitoring and Response."

#	Name	Description
		Patch and Update Management: DevSecOps facilitates active management of patches and updates for third-party components, addressing the sub-domain of "Component Patch Management." Security Assessment and Testing: DevSecOps supports conducting security assessments and testing of third-party components, addressing the sub-domain of "Component Testing." Vendor Relationship Management: DevSecOps helps build and maintain strong vendor relationships for effective security monitoring, addressing the sub-domain of "Supplier Security Assurance." Risk Mitigation Strategies: DevSecOps contributes to the development and implementation of risk mitigation strategies for third-party components, addressing the sub-domain of "Component Risk Management." Incident Response and Remediation: DevSecOps assists in incident response and remediation processes related to third-party components, addressing the sub-domain of "Component Incident Management."
4	SAMM Security by Verification	Security Testing: DevSecOps supports various security testing activities, such as penetration testing, vulnerability scanning, and code review, addressing the sub-domain of "Application Security Testing." Code Review: DevSecOps facilitates code review processes to identify security flaws, addressing the sub-domain of "Secure Code Review." Security Architecture Review: DevSecOps contributes to security architecture reviews, ensuring the presence of proper security controls, addressing the sub-domain of "Security Architecture Review." Security Requirements Verification: DevSecOps helps verify the implementation of security requirements in software applications, addressing the sub-domain of "Requirements Management." Threat Modeling: DevSecOps supports conducting threat modeling exercises to identify and mitigate potential security risks, addressing the sub-domain of "Threat Assessment." Secure Deployment Verification: DevSecOps assists in verifying secure deployment practices, ensuring proper implementation of security controls during deployment, addressing the sub-domain of "Deployment Verification."
5	SAMM Security by Operations	Environment Hardening: DevSecOps helps implement secure configurations and infrastructure, contributing to the sub-domain of "System Hardening." Secure Build: DevSecOps promotes secure coding practices and secure build tools, addressing the sub-domain of "Build Hardening." Configuration Management: DevSecOps facilitates effective configuration management practices, ensuring integrity and minimizing unauthorized changes, addressing the sub-domain of "Configuration Management." Vulnerability Management: DevSecOps supports vulnerability scanning and assessment to identify and address vulnerabilities, addressing the sub-domain of "Vulnerability Management." Incident Management: DevSecOps contributes to the incident management process, including detection, response, containment, and recovery, addressing the sub-domain of "Incident Management." Secure Deployment: DevSecOps assists in secure software distribution and installation procedures, addressing the sub-domain of "Secure Deployment." Security Testing: DevSecOps enables security testing activities throughout the software development life cycle, addressing the sub-domain of "Operational Testing."

#	Name	Description
		Operational Enablement: DevSecOps promotes operational support and guidance for ongoing security and reliability, addressing the sub-domain of "Operational Enablement."

11.3.26.2 Assessment – OWASP Top 10

The "Security Monitoring of Third-Party Components" capability contributes to addressing various OWASP Top 10 threats by proactively monitoring, identifying, and mitigating security risks in third-party components used within the application. It ensures that these components do not introduce vulnerabilities or weaknesses that could be exploited by attackers, thereby enhancing the overall security posture of the application.

#	Name	Description
1	Broken Access Control (A01-2021)	While the Security Monitoring of Third-Party Components capability does not directly address this threat, it indirectly contributes to mitigating broken access control vulnerabilities. By monitoring the security of third-party components, organizations can identify and address vulnerabilities or weaknesses that could potentially lead to broken access control.
2	Cryptographic Failures (A02-2021)	The Security Monitoring of Third-Party Components capability helps mitigate cryptographic failures by ensuring that third-party components related to cryptography, such as encryption libraries or cryptographic algorithms, are monitored for security vulnerabilities. By regularly assessing these components, organizations can identify and address cryptographic weaknesses or mistakes in their implementation, reducing the risk of cryptographic failures.
3	Injection (A03-2021)	The Security Monitoring of Third-Party Components capability indirectly contributes to mitigating injection vulnerabilities. By monitoring and assessing third-party components, organizations can identify any vulnerable components that may be susceptible to injection attacks. This allows them to take necessary actions, such as applying patches or using alternative components, to reduce the risk of injection vulnerabilities within the system.
4	Insecure Design (A04-2021)	The Security Monitoring of Third-Party Components capability does not directly address insecure design vulnerabilities. However, it indirectly contributes to mitigating insecure design by monitoring the security of third-party components used in the system. If any third-party components are found to have insecure design flaws, organizations can take remedial actions, such as seeking alternative components or implementing additional security measures, to mitigate the risk.
5	Security Misconfiguration (A05-2021)	The Security Monitoring of Third-Party Components capability indirectly helps address security misconfiguration vulnerabilities. By monitoring the security of third-party components, organizations can identify any misconfigurations or insecure settings within these components that could lead to security misconfigurations in the overall system. By promptly addressing these issues, organizations can reduce the risk of security misconfigurations and enhance the overall security posture.

#	Name	Description
6	Vulnerable and Outdated Components (A06-2021)	The Security Monitoring of Third-Party Components capability directly addresses the threat of vulnerable and outdated components. By continuously monitoring the security of third-party components, organizations can identify any known vulnerabilities associated with these components. This allows them to take proactive measures, such as applying patches or updates, to mitigate the risk of using vulnerable or outdated components within the system.
7	Identification and Authentication Failures (A07-2021)	The Security Monitoring of Third-Party Components capability indirectly contributes to addressing identification and authentication failures. By monitoring the security of third-party components, organizations can identify any weaknesses or vulnerabilities that could impact the identification and authentication mechanisms of the system. By addressing these issues, organizations can reduce the risk of identification and authentication failures and enhance the overall security of their applications.
8	Software and Data Integrity Failures (A08-2021)	The Security Monitoring of Third-Party Components capability indirectly helps address software and data integrity failures. By monitoring the security of third-party components, organizations can identify any vulnerabilities or weaknesses that could compromise the integrity of software components, configurations, or data. By addressing these issues, organizations can mitigate the risk of software and data integrity failures and maintain the integrity of their applications.
9	Security Logging and Monitoring Failures (A09-2021)	This capability directly addresses the threat of security logging and monitoring failures. By ensuring that third-party components are effectively logged and monitored, organizations can detect potential security incidents and respond promptly.
10	Server-Side Request Forgery (A10-2021)	Monitoring third-party components can help detect any vulnerabilities that could lead to SSRF attacks. By addressing such vulnerabilities in these components, organizations can reduce the risk of SSRF exploits.

11.3.27 Secure API Design and Management

Secure API Design and Management is a critical security capability within the DevSecOps framework. It focuses on designing and managing APIs (Application Programming Interfaces) with robust security measures in place. This capability ensures that APIs are developed and maintained in a secure manner, protecting sensitive data and preventing unauthorized access.

By incorporating the Secure API Design and Management capability into the DevSecOps approach, organizations can ensure that their APIs are developed and managed with security as a priority. This helps protect sensitive data, prevent unauthorized access, and maintain the integrity and availability of the API. By following secure design principles, implementing robust authentication and encryption mechanisms, and conducting regular security testing, organizations can mitigate potential risks and vulnerabilities associated with API usage.

Here are key aspects of the Secure API Design and Management capability within DevSecOps:

#	Name	Description
1	Input Validation and Sanitization	APIs should include robust input validation and sanitization mechanisms to prevent common security vulnerabilities, such as injection attacks (e.g., SQL injection, XSS). Input validation ensures that the data provided by clients adheres to expected formats and ranges, while sanitization removes or neutralizes potentially malicious content from user-supplied data.
2	Secure API Design	Secure API design involves incorporating security considerations throughout the API development lifecycle. This includes implementing secure authentication mechanisms, such as OAuth or API keys, to verify the identity of clients accessing the API. Additionally, authorization mechanisms, such as role-based access control (RBAC), should be employed to control access to API resources based on user permissions. Secure communication protocols, such as HTTPS, should be used to encrypt data transmitted between the client and the API.
3	Data Protection and Encryption	Sensitive data transmitted through APIs should be protected using encryption. This can be achieved through mechanisms such as Transport Layer Security (TLS) encryption, which ensures that data is securely transmitted over the network. Additionally, encryption techniques, such as encrypting data at rest or using encryption algorithms like AES, can be applied to safeguard data stored within the API or its associated databases.
4	Rate Limiting and Throttling	APIs should implement rate limiting and throttling mechanisms to prevent abuse or excessive usage. Rate limiting controls the number of requests that a client can make within a specific time frame, while throttling limits the speed at which requests are processed. These measures protect the API from distributed denial-of-service (DDoS) attacks and ensure fair usage by clients.
5	API Monitoring and Logging	Implementing robust monitoring and logging capabilities for APIs is essential for detecting and responding to security incidents. Monitoring can include real-time monitoring of API traffic, tracking usage patterns, and identifying potential anomalies or suspicious activities. Logging API events and activities helps in forensic analysis, incident investigation, and compliance audits.
6	API Versioning and Lifecycle Management	Proper versioning and lifecycle management of APIs are important from a security perspective. APIs should follow semantic versioning principles to ensure backward compatibility while introducing security enhancements or bug fixes. Deprecated or insecure API versions should be phased out, and clients should be encouraged to migrate to newer, more secure versions.
7	API Security Testing	Regular security testing of APIs is crucial to identify vulnerabilities and weaknesses. This can involve various testing techniques, such as penetration testing, vulnerability scanning, and fuzz testing, to uncover potential security flaws. Security testing should be performed throughout the development lifecycle, including during the design phase, implementation, and ongoing maintenance.
8	API Documentation and Education	Clear and comprehensive API documentation is essential for developers and users to understand the security features, requirements, and best practices associated with the API. Additionally, providing education and training to developers on secure API design and usage helps foster a security-conscious development culture.

11.3.27.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Secure API Design and Management contributes to this sub-domain by establishing policies and standards for secure API development, including authentication mechanisms, authorization controls, and communication protocols. Risk Management: Secure API Design and Management helps mitigate security risks associated with APIs by implementing robust authentication, encryption, and input validation mechanisms. Compliance Management: Secure API Design and Management ensures compliance with security standards and regulations by incorporating secure design principles and encryption techniques into API development. Security Training and Awareness: Secure API Design and Management includes educating developers and users on secure API design and usage, promoting a culture of security within the organization. Security Metrics and Reporting: Secure API Design and Management contributes to security metrics and reporting by monitoring API traffic, tracking usage patterns, and logging API events and activities for analysis and compliance audits. Security Roles and Responsibilities: Secure API Design and Management involves defining security roles and responsibilities for API development and management, ensuring individuals have the necessary knowledge and skills to fulfill their roles effectively.
2	SAMM Security by Design	Security Requirements: Secure API Design and Management addresses security requirements by incorporating secure authentication, authorization, and encryption mechanisms into API design. Secure Architecture: Secure API Design and Management promotes the use of secure architectural patterns and encryption techniques to minimize vulnerabilities and address potential OWASP threats. Secure Coding Practices: Secure API Design and Management includes guidelines for secure coding practices, preventing common vulnerabilities associated with OWASP threats such as injection attacks and insecure design. Threat Modeling: Secure API Design and Management encourages threat modeling exercises to identify and mitigate potential security risks, aligning with the objectives of SAMM's Security by Design domain. Security Testing: Secure API Design and Management incorporates security testing techniques such as penetration testing and vulnerability scanning to uncover and address vulnerabilities related to OWASP threats.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training: Secure API Design and Management provides training on secure API design and management practices, ensuring developers have the necessary knowledge and skills to build secure APIs. Secure Architecture: Secure API Design and Management promotes the use of secure architecture practices, such as threat modeling and secure component selection, aligning with the objectives of SAMM's Implementation domain. Secure Coding Guidelines: Secure API Design and Management includes secure coding guidelines and best practices, helping developers write secure code and address vulnerabilities during implementation. Security Requirements: Secure API Design and Management ensures that security requirements are considered and implemented during API development, aligning with SAMM's emphasis on security requirements. Security Testing Integration: Secure API Design and Management encourages the integration of security testing activities, such as static code analysis and security code reviews, into the implementation phase. Security Verification: Secure API Design and Management contributes to security verification by ensuring that security controls implemented in APIs are effective and aligned with security requirements. Security Architecture Review: Secure API Design and Management supports security architecture reviews to evaluate the security of API design and implementation, a key component of SAMM's Implementation domain. Security Operations Integration: Secure API Design and Management emphasizes integrating security considerations into operational processes, supporting incident management, vulnerability management, and secure deployment practices.
4	SAMM Security by Verification	Security Testing: The Secure API Design and Management capability includes security testing techniques such as penetration testing, vulnerability scanning, and fuzz testing. These security testing activities help identify vulnerabilities and weaknesses in APIs, ensuring their security and mitigating potential risks (SAMM Security by Verification, Security Testing).
5	SAMM Security by Operations	Secure Deployment: The Secure API Design and Management capability promotes secure deployment practices for APIs. This includes secure software distribution, secure installation procedures, and secure configuration of API components. By implementing these practices, organizations can ensure that APIs are deployed in a secure and controlled manner, supporting the secure operations of software systems (SAMM Security by Operations, Secure Deployment). Security Testing: The Secure API Design and Management capability integrates security testing activities, such as static code analysis and security code reviews. By conducting regular security testing, organizations can identify and address vulnerabilities in APIs, supporting their secure operation (SAMM Security by Operations, Security Testing).

11.3.27.2 Assessment – OWASP Top 10

The Secure API Design and Management capability within DevSecOps directly addresses multiple OWASP threats by incorporating secure design principles, implementing proper access controls, input validation and sanitization, encryption, monitoring and logging, API versioning and lifecycle management, security testing, and education on secure API design and usage.

#	Name	Description
1	Broken Access Control (A01-2021)	Secure API Design: Implementing secure authentication mechanisms, such as OAuth or API keys, verifies the identity of clients accessing the API, preventing unauthorized access. Authorization Mechanisms: Role-based access control (RBAC) controls access to API resources based on user permissions, ensuring proper access control and preventing privilege escalation. API Versioning and Lifecycle Management: Phasing out deprecated or insecure API versions and encouraging clients to migrate to newer, more secure versions helps maintain proper access control.
2	Cryptographic Failures (A02-2021)	Secure API Design: The use of secure communication protocols, such as HTTPS, ensures the encryption of data transmitted between the client and the API, protecting against cryptographic vulnerabilities.
3	Injection (A03-2021)	Input Validation and Sanitization: Robust input validation and sanitization mechanisms prevent common injection attacks, such as SQL injection and XSS, ensuring that untrusted data cannot manipulate the application's execution flow or interact with data stores.
4	Insecure Design (A04-2021)	Secure API Design: Incorporating secure design principles from the early stages of API development helps prevent insecure architectural decisions and ensures security considerations are properly addressed.
5	Security Misconfiguration (A05-2021)	API Monitoring and Logging: Implementing robust monitoring and logging capabilities for APIs helps identify security misconfigurations, such as insecure settings or exposed sensitive information, enabling organizations to take corrective actions.
6	Vulnerable and Outdated Components (A06-2021)	API Versioning and Lifecycle Management: Regularly updating software dependencies and actively managing third-party components minimize the risk of using outdated or known vulnerable software components within the API.
7	Identification and Authentication Failures (A07-2021)	Secure API Design: Implementing secure authentication mechanisms, enforcing strong password policies, and regularly testing for vulnerabilities help address identification and authentication failures, reducing the risk of unauthorized access.
8	Software and Data Integrity Failures (A08-2021)	Secure API Design: By following secure practices, implementing secure coding techniques, and regularly testing for vulnerabilities, organizations can protect against unauthorized modifications, data tampering, and integrity-related issues.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	API Monitoring and Logging: Implementing comprehensive logging, effective log analysis, and real-time monitoring mechanisms address the lack of effective logging and monitoring practices, enabling timely detection and response to security incidents.
10	Server-Side Request Forgery (A10-2021)	Secure API Design: Robust input validation and secure configurations help prevent server-side request forgery (SSRF) vulnerabilities, ensuring that the API cannot be tricked into making unintended requests to internal or external resources.

11.3.28 Secure Network Architecture

Secure Network Architecture is a vital security capability within the DevSecOps framework. It involves designing and implementing a robust and resilient network infrastructure that incorporates various security controls to protect against unauthorized access, data breaches, and malicious activities.

By incorporating the Secure Network Architecture capability into the DevSecOps approach, organizations can establish a resilient and secure network infrastructure. This helps protect sensitive data, prevent unauthorized access, and detect and respond to network-based threats effectively. By implementing network segmentation, firewalls, intrusion detection/prevention systems, and secure remote access controls, organizations can establish a strong network defense and reduce the risk of network-related security incidents.

Here are key aspects of the Secure Network Architecture capability within DevSecOps:

#	Name	Description
1	Network Segmentation	Network segmentation involves dividing the network into separate segments or zones to isolate sensitive systems and data from the rest of the network. By implementing logical or physical barriers, such as firewalls or VLANs (Virtual Local Area Networks), organizations can restrict unauthorized access and limit the potential impact of a security breach.
2	Firewalls and Intrusion Detection/Prevention Systems (IDS/IPS)	Firewalls act as a first line of defense by monitoring and controlling network traffic based on predefined security policies. They inspect incoming and outgoing traffic, blocking unauthorized access and malicious activities. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) help detect and prevent network-based attacks by analyzing network traffic for known attack signatures and abnormal behavior.
3	Secure Wireless Networks	Secure wireless network design involves implementing strong encryption protocols, such as WPA2 or WPA3, to protect wireless communications. Wireless network access should be carefully managed, ensuring that only authorized devices and users can connect. Regular vulnerability assessments and security updates for wireless access points are essential to maintain the security of wireless networks.
4	Denial-of-Service (DoS) Mitigation	Implementing DoS mitigation techniques helps protect the network infrastructure from overwhelming traffic and service disruptions. This includes deploying DoS protection solutions or services that detect and mitigate volumetric, application layer, or distributed DoS attacks.

#	Name	Description
5	Network Access Controls	Strong access controls are crucial for network security. This involves implementing role-based access control (RBAC) to restrict network access based on user roles and responsibilities. Access control lists (ACLs) can be used to define granular permissions for network resources, limiting access to only authorized users or systems.
6	Secure Remote Access:	With the rise of remote work and cloud-based environments, secure remote access is crucial. This includes implementing secure virtual private networks (VPNs) for encrypted communication between remote users and the corporate network. Two-factor authentication (2FA) and strong password policies further enhance the security of remote access.
7	Network Monitoring and Logging	Continuous network monitoring and logging provide visibility into network activities and help detect suspicious behavior or anomalies. Network monitoring tools can analyze traffic patterns, identify potential security incidents, and generate alerts for further investigation. Logging network events and activities aids in forensic analysis, incident response, and compliance audits.
8	Secure DNS and DHCP Management	Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP) are critical network services that require proper security measures. Implementing secure DNS practices, such as DNSSEC (DNS Security Extensions) and DNS filtering, helps prevent DNS-related attacks. Secure DHCP management ensures that only authorized devices receive valid IP configurations, mitigating the risk of unauthorized network access.
9	Network Resilience and Redundancy	Building network resilience involves implementing redundant network links, failover mechanisms, and backup systems. Redundancy helps ensure uninterrupted network connectivity and minimizes the impact of network failures or disruptions. Regular testing and maintenance of network resilience measures are essential to ensure their effectiveness during critical situations.
10	Compliance and Security Standards	A secure network architecture should align with industry-specific security standards and compliance regulations. This includes adhering to frameworks such as the Payment Card Industry Data Security Standard (PCI DSS) or the Health Insurance Portability and Accountability Act (HIPAA). Regular security assessments and audits help verify compliance and identify areas for improvement.

11.3.28.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: DevSecOps helps organizations develop and implement policies and standards related to software security, such as secure coding practices, vulnerability management, incident response, and data protection. This ensures that security expectations and requirements are defined and followed throughout the software development and operations processes. Risk Management: DevSecOps integrates risk management processes into software development and operations, allowing organizations to identify and assess risks associated with software security. By implementing risk mitigation strategies, organizations

#	Name	Description
		can address identified risks and minimize their impact on software systems. Compliance Management: DevSecOps supports organizations in ensuring compliance with relevant regulatory requirements, industry standards, and best practices related to software security. It enables organizations to assess compliance, report regulatory requirements, and align with security frameworks and guidelines. Security Training and Awareness: DevSecOps emphasizes the importance of security training and awareness programs for employees and stakeholders. It provides resources and programs to educate individuals about software security, promoting a culture of security throughout the organization. Security Metrics and Reporting: DevSecOps helps organizations define and measure key security metrics to assess the effectiveness of software security practices. It also establishes reporting mechanisms to provide visibility into the status of software security, enabling informed decision-making at various levels of the organization. Security Roles and Responsibilities: DevSecOps assists organizations in clearly defining and assigning security roles and responsibilities. It ensures that individuals have the necessary knowledge and skills to fulfill their roles effectively, promoting accountability for software security.
2	SAMM Security by Design	Security Requirements: DevSecOps supports the integration of security requirements into the design phase of software development. It helps identify and address specific security concerns, mitigating vulnerabilities associated with OWASP threats such as Broken Access Control, Injection, Insecure Design, and others. Secure Architecture: DevSecOps encourages the adoption of secure architectural patterns and principles during the design phase. By designing a robust and secure architecture, organizations address OWASP threats like Insecure Design, Cryptographic Failures, Security Misconfiguration, and others, reducing potential vulnerabilities and establishing a strong foundation for security controls. Secure Coding Practices: DevSecOps promotes the use of secure coding practices during the design phase. By adhering to secure coding guidelines, organizations reduce the likelihood of OWASP threats such as Injection, Insecure Design, Security Misconfiguration, and others by preventing common coding mistakes and vulnerabilities. Threat Modeling: DevSecOps suggests conducting threat modeling exercises during the design phase to identify and mitigate potential security risks. It helps organizations assess the impact of OWASP threats and implement appropriate security controls and countermeasures. Security Testing: DevSecOps encourages the inclusion of security testing activities in the design phase. By incorporating techniques like static code analysis and security code reviews, organizations can detect and address vulnerabilities related to OWASP threats, ensuring the design's resilience to potential attacks.
3	SAMM Security by Implementation	Secure Development Training: DevSecOps promotes the adoption of secure coding practices and provides training programs to developers and development teams. This helps organizations address the activity of "Secure Development Training" in SAMM's Implementation domain. Secure Architecture: DevSecOps emphasizes the use of secure architectural patterns and principles, such as threat modeling and secure design patterns. By incorporating secure architecture

#	Name	Description
		practices, organizations can address the activity of "Secure Architecture" in SAMM's Implementation domain. Secure Coding Guidelines: DevSecOps advocates for the establishment and adoption of secure coding guidelines and standards. This aligns with the activity of "Secure Coding Guidelines" in SAMM's Implementation domain. Security Requirements: DevSecOps emphasizes the need to define and document security requirements as part of the software development process. This aligns with the activity of "Security Requirements" in SAMM's Implementation domain. Security Testing Integration: DevSecOps encourages the integration of security testing activities throughout the software development life cycle, such as static code analysis and security code reviews. This aligns with the activity of "Security Testing Integration" in SAMM's Implementation domain. Security Verification: DevSecOps promotes the use of security verification techniques, such as security code reviews and vulnerability assessments, to validate the effectiveness of security controls. This aligns with the activity of "Security Verification" in SAMM's Implementation domain. Security Architecture Review: DevSecOps recommends conducting security architecture reviews to evaluate the security of the software system's design and architecture. This aligns with the activity of "Security Architecture Review" in SAMM's Implementation domain. Security Operations Integration: DevSecOps emphasizes the integration of security considerations into operational processes, such as incident management and vulnerability management. This aligns with the activity of "Security Operations Integration" in SAMM's Implementation domain.
4	SAMM Security by Verification	Secure Network Architecture: The capability of designing and implementing a secure network architecture helps address the SAMM Security by Verification domain by ensuring that the network infrastructure is resilient against security vulnerabilities and threats. It contributes to the validation and verification of the security controls implemented within the network architecture. Security Testing: The capability of conducting security testing, such as penetration testing and vulnerability scanning, directly addresses the SAMM Security by Verification domain. It helps validate the effectiveness of security controls, identify vulnerabilities, and verify the security posture of software applications. Code Review: Code review, which involves reviewing the source code for security flaws, aligns with the SAMM Security by Verification domain. It contributes to the verification of security requirements and adherence to secure coding practices, mitigating vulnerabilities within the software. Security Architecture Review: Conducting a security architecture review helps validate and verify the security architecture of software applications. It aligns with the SAMM Security by Verification domain by assessing the effectiveness of security controls and mechanisms implemented in the architecture. Security Requirements Verification: Verifying security requirements against industry standards, regulatory requirements, and best practices directly addresses the SAMM Security by Verification domain. It ensures that security requirements are properly defined, documented, and implemented in software applications. Threat Modeling: Performing threat modeling exercises aligns with the SAMM Security by Verification domain. It contributes to the identification and prioritization of security threats, enabling organizations to implement appropriate security controls and countermeasures.

#	Name	Description
		Secure Deployment Verification: Verifying secure deployment practices, such as secure configurations, encryption mechanisms, and access controls, directly addresses the SAMM Security by Verification domain. It ensures that software applications are securely deployed and configured in the production environment.
5	SAMM Security by Operations	Secure Network Architecture: The capability of designing and implementing a secure network architecture helps address the SAMM Security by Operations domain by ensuring the secure operation of software systems. It contributes to the ongoing security and reliability of the network infrastructure supporting the software applications. Environment Hardening: Environment hardening, which involves implementing secure configurations and infrastructure settings, aligns with the SAMM Security by Operations domain. It contributes to the secure operation of software systems by protecting them from known vulnerabilities and security weaknesses. Configuration Management: Effective configuration management practices directly address the SAMM Security by Operations domain. They contribute to maintaining the integrity of software systems and ensuring that unauthorized changes or misconfigurations are detected and remediated. Vulnerability Management: Implementing a robust vulnerability management process aligns with the SAMM Security by Operations domain. It helps organizations identify and prioritize vulnerabilities, enabling them to address them promptly and reduce the risk of exploitation. Incident Management: Having a well-defined incident management process directly addresses the SAMM Security by Operations domain. It enables organizations to detect, respond to, contain, and recover from security incidents, minimizing their impact and restoring normal operations promptly. Secure Deployment: Secure deployment practices, including secure software distribution, installation procedures, and configuration, contribute to the SAMM Security by Operations domain. They ensure that software is deployed in a secure and controlled manner, reducing the risk of unauthorized access or compromise. Security Testing: Integrating security testing activities throughout the software development life cycle aligns with the SAMM Security by Operations domain. It helps organizations identify and address security vulnerabilities, ensuring the ongoing security of software systems.

11.3.28.2 Assessment – OWASP Top 10

Secure Network Architecture does not directly address all OWASP threats; however, it provides a foundational security capability within the DevSecOps framework. By incorporating this capability, organizations can establish a resilient and secure network infrastructure that contributes to mitigating various OWASP threats and enhancing the overall security posture of web applications.

#	Name	Description
1	Broken Access Control (A01-2021)	Secure Network Architecture helps enforce access controls and restrict unauthorized access to sensitive resources by implementing network segmentation, firewalls, and secure remote access controls.

#	Name	Description
2	Cryptographic Failures (A02-2021)	While Secure Network Architecture does not directly address cryptographic failures, it provides a foundation for implementing secure network communications. By utilizing secure virtual private networks (VPNs) and strong encryption protocols for wireless networks, organizations can enhance the overall security of cryptographic operations.
3	Injection (A03-2021)	Secure Network Architecture contributes to the prevention of injection attacks by implementing network-level intrusion detection/prevention systems (IDS/IPS). These systems can analyze network traffic for known attack signatures and abnormal behavior, helping to detect and prevent injection attacks.
4	Insecure Design (A04-2021)	Secure Network Architecture does not directly address insecure design vulnerabilities. However, it can contribute indirectly by providing network-level security controls that help protect against the exploitation of insecure design decisions, such as implementing firewalls to control and monitor network traffic.
5	Security Misconfiguration (A05-2021)	Secure Network Architecture directly addresses security misconfigurations by implementing secure configurations for network devices, such as firewalls and access control lists (ACLs). It also promotes network monitoring and logging, which aids in identifying and rectifying security misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	Secure Network Architecture does not directly address vulnerabilities arising from outdated or vulnerable software components. However, by implementing network-level security controls, such as firewalls and IDS/IPS, organizations can reduce the risk of exploitation of vulnerabilities present in outdated components.
7	Identification and Authentication Failures (A07-2021)	While Secure Network Architecture does not directly address identification and authentication failures, it can contribute indirectly by providing secure remote access controls, such as secure VPNs. These controls help ensure that remote users authenticate and establish secure connections before accessing network resources.
8	Software and Data Integrity Failures (A08-2021)	Secure Network Architecture contributes to addressing software and data integrity failures by implementing network monitoring and logging capabilities. These capabilities help detect and respond to unauthorized modifications, manipulation, or destruction of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	Secure Network Architecture directly addresses security logging and monitoring failures by promoting continuous network monitoring and logging. This capability provides visibility into network activities and aids in the detection of suspicious behavior or security incidents.
10	Server-Side Request Forgery (A10-2021)	Secure Network Architecture helps mitigate server-side request forgery (SSRF) vulnerabilities by implementing network-level security controls, such as firewalls and IDS/IPS. These controls can detect and block unauthorized requests made by attackers from within the network.

11.3.29 Security Training and Awareness Programs

Security Training and Awareness Programs are a critical security capability within the DevSecOps framework. This capability focuses on providing education and awareness to employees about security

best practices, policies, and potential threats. By investing in security training and awareness, organizations can empower their workforce to become active participants in maintaining a strong security posture and mitigating security risks.

By incorporating robust Security Training and Awareness Programs into the DevSecOps approach, organizations can create a culture of security awareness and responsibility. Employees become active participants in identifying and mitigating security risks, strengthening the overall security posture of the organization.

Here are key aspects of the Security Training and Awareness Programs capability within DevSecOps:

#	Name	Description
1	Security Awareness Training	This involves conducting regular training sessions to educate employees about various security topics, including password hygiene, phishing attacks, social engineering, physical security, and data protection. The training sessions aim to enhance employees' understanding of security risks and equip them with the knowledge needed to make informed security decisions.
2	Policy and Procedure Communication	Clear communication of security policies, procedures, and guidelines is crucial for creating a security-conscious culture. Organizations should develop comprehensive security policies and ensure that employees are aware of and understand their roles and responsibilities in maintaining security. Regular updates and reminders about security policies help reinforce good security practices throughout the organization.
3	Secure Development Training	DevSecOps emphasizes integrating security into the software development lifecycle. Providing secure development training to developers and other stakeholders involved in the software development process is essential. This training focuses on secure coding practices, secure configuration management, secure code reviews, and other security-related aspects specific to the development process.
4	Phishing and Social Engineering Awareness	Phishing attacks and social engineering are common methods used by attackers to exploit human vulnerabilities. Training programs should educate employees about the signs of phishing emails, suspicious links, and social engineering techniques. This helps employees recognize and respond appropriately to potential threats, reducing the risk of falling victim to phishing attacks.
5	Incident Reporting and Response Training	Effective incident reporting and response are crucial for timely detection and mitigation of security incidents. Training programs should provide guidance on how to identify and report security incidents promptly. Employees should also be educated on the steps to take in the event of a security incident to minimize its impact and ensure proper incident response procedures are followed.
6	Security Metrics and Reporting	Training programs can include education on security metrics and reporting to help employees understand the importance of measuring and reporting security-related data. This encourages a data-driven approach to security and enables organizations to track their security posture, identify trends, and make informed decisions to improve security measures.

#	Name	Description
7	Continuous Education and Updates	The security landscape is constantly evolving, with new threats emerging regularly. Training programs should be ongoing to keep employees up to date with the latest security trends, technologies, and best practices. This can include providing resources such as newsletters, security blogs, webinars, and access to online training platforms to encourage continuous learning and professional development in the field of security.
8	Gamification and Simulations	To enhance engagement and knowledge retention, security training programs can incorporate gamification elements and simulations. Gamification can include quizzes, challenges, or interactive exercises that reinforce security concepts and encourage healthy competition among employees. Simulations can simulate real-world security scenarios, allowing employees to practice their response to security incidents in a safe and controlled environment.
9	Security Champions Program	Establishing a security champions program can further enhance the effectiveness of security training and awareness efforts. This program identifies and trains employees who have a keen interest in security to act as advocates and mentors within their respective teams. They can help disseminate security knowledge, provide guidance, and promote security best practices among their peers.
10	Metrics and Evaluation	To measure the effectiveness of the training and awareness programs, organizations should establish metrics to track employee engagement, knowledge improvement, incident response effectiveness, and overall security posture. Regular evaluation and feedback from employees can help identify areas for improvement and shape future training initiatives.

11.3.29.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: Security training and awareness programs contribute to the establishment and communication of security policies and standards. They educate employees about secure coding practices, vulnerability management, incident response, and data protection. Risk Management: Security training and awareness programs help raise employees' awareness of security risks, enabling them to better understand and assess risks associated with software development and deployment. Compliance Management: Training programs can educate employees about regulatory requirements and industry standards related to software security, ensuring compliance and promoting adherence to security frameworks and guidelines. Security Training and Awareness: This capability directly addresses the SAMM Security by Governance domain, as it emphasizes the importance of educating and raising awareness among employees and stakeholders about software security.

#	Name	Description
		Security Metrics and Reporting: Training programs can include education on security metrics and reporting, enabling employees to understand the importance of measuring and reporting security-related data. This contributes to the organization's overall security governance and decision-making processes. Security Roles and Responsibilities: Security training and awareness programs help clarify security roles and responsibilities within the organization, ensuring that individuals have the necessary knowledge and skills to fulfill their roles effectively.
2	SAMM Security by Design	Security Training and Awareness: Security Training and Awareness Programs contribute to incorporating security controls and considerations into the design phase of software development. By educating developers and stakeholders about secure coding practices, secure architecture, and threat modeling, it helps address security requirements and minimize vulnerabilities associated with OWASP threats.
3	SAMM Security by Implementation	Secure Development Training: Security training and awareness programs provide developers and development teams with the necessary training on secure coding practices, secure design principles, and common software vulnerabilities. This contributes to implementing secure coding guidelines and practices during the development phase. Secure Architecture: Training programs can educate developers about secure architectural patterns and principles, enabling them to design software systems that minimize potential vulnerabilities and address security threats. Secure Coding Practices: Security training and awareness programs promote secure coding practices, helping developers prevent common vulnerabilities and address issues related to OWASP threats, such as injection, insecure design, and security misconfiguration. Threat Modeling: Training programs can educate developers on conducting threat modeling exercises during the design phase, enabling them to identify and mitigate potential security risks associated with OWASP threats. Security Testing: Training programs can include education on security testing techniques, such as static code analysis and security code reviews, which help identify and address vulnerabilities related to OWASP threats during the implementation phase.
4	SAMM Security by Verification	Security Testing: Security training and awareness programs contribute to the implementation of various security testing activities, such as penetration testing, vulnerability scanning, code review, and security testing automation. These activities aim to verify the security of software applications. Code Review: Training programs can educate developers on conducting manual code reviews and using automated code analysis tools to identify security flaws and vulnerabilities in software applications. Security Architecture Review: Training programs can provide guidance on conducting security architecture reviews to evaluate the security of software systems' design and architecture.

#	Name	Description
		Security Requirements Verification: Training programs can educate stakeholders on verifying that security requirements are properly defined, documented, and implemented in software applications. Threat Modeling: Training programs can educate stakeholders on conducting threat modeling exercises to identify and prioritize security threats, including those related to OWASP threats. Secure Deployment Verification: Training programs can educate stakeholders on verifying the secure deployment and configuration of software applications to ensure that security controls and measures are properly implemented.
5	SAMM Security by Operations	Policies and Standards: Security Training and Awareness Programs contribute to the development and implementation of policies and standards related to software security. These programs educate employees about secure coding practices, incident response procedures, and data protection guidelines, helping organizations establish and communicate their security expectations. Risk Management: Security Training and Awareness Programs enhance risk management efforts by educating employees about security risks and potential threats. By raising awareness and providing training on security best practices, these programs help employees understand their role in mitigating risks and contribute to the overall risk management process. Compliance Management: Security Training and Awareness Programs support compliance management by educating employees about relevant regulatory requirements, industry standards, and best practices. These programs ensure that employees are aware of their responsibilities in maintaining compliance and help organizations meet their compliance obligations. Security Training and Awareness: This capability directly addresses the need for security training and awareness within the Security by Operations domain. By providing training programs and awareness campaigns, organizations can educate employees about security risks, promote a culture of security, and enhance security awareness throughout the organization. Security Metrics and Reporting: Security Training and Awareness Programs can educate employees about the importance of security metrics and reporting. By understanding the significance of measuring and reporting security-related data, employees can contribute to the organization's efforts to track security metrics, identify trends, and make informed decisions to improve security measures. Security Roles and Responsibilities: Security Training and Awareness Programs help clarify and communicate security roles and responsibilities within the organization. By providing training on security-related topics and expectations, these programs ensure that individuals have the necessary knowledge and skills to fulfill their security roles effectively.

11.3.29.2 Assessment – OWASP Top 10

Security Training and Awareness Programs within DevSecOps contribute to addressing various OWASP threats by educating employees, raising awareness about vulnerabilities, promoting secure practices, and fostering a culture of security awareness and responsibility.

#	Name	Description
1	Broken Access Control (A01-2021)	Security Training and Awareness Programs can educate employees about the importance of proper access controls and the risks associated with broken access control vulnerabilities. This helps them understand the significance of enforcing access controls correctly, mitigating the threat.
2	Cryptographic Failures (A02-2021)	Security Training and Awareness Programs can provide employees with knowledge about cryptographic practices, including the proper implementation and usage of cryptographic algorithms. By educating employees about cryptographic failures and best practices, organizations can reduce the likelihood of vulnerabilities arising from incorrect cryptography implementation.
3	Injection (A03-2021)	Security Training and Awareness Programs can educate employees about the risks of injection attacks and the importance of input validation and secure coding practices. By raising awareness about the potential consequences of injection vulnerabilities, employees can be more vigilant in validating and sanitizing user input, reducing the risk of injection attacks.
4	Insecure Design (A04-2021)	Security Training and Awareness Programs can emphasize the significance of secure design principles during the development process. By educating employees about secure design practices and the potential consequences of insecure design decisions, organizations can promote the adoption of secure design principles, mitigating insecure design vulnerabilities.
5	Security Misconfiguration (A05-2021)	Security Training and Awareness Programs can educate employees about the risks associated with security misconfigurations and the importance of following secure configuration practices. By providing training on secure configurations, organizations can reduce the likelihood of misconfigurations that may lead to vulnerabilities and improve the overall security posture.
6	Vulnerable and Outdated Components (A06-2021)	Security Training and Awareness Programs can educate employees about the risks of using vulnerable or outdated software components. By raising awareness about the importance of managing software dependencies, regularly updating components, and following secure coding practices, organizations can minimize the usage of vulnerable and outdated components, reducing the associated risks.
7	Identification and Authentication Failures (A07-2021)	Security Training and Awareness Programs can provide employees with knowledge about secure authentication mechanisms and the risks of identification and authentication failures. By educating employees about strong password policies, multi-factor authentication, and secure authentication practices, organizations can enhance the overall security of their authentication systems.

#	Name	Description
8	Software and Data Integrity Failures (A08-2021)	Security Training and Awareness Programs can raise awareness among employees about the importance of ensuring software and data integrity. By providing training on secure coding techniques, secure configuration management, and regular vulnerability testing, organizations can reduce the risk of unauthorized modifications and maintain data integrity.
9	Security Logging and Monitoring Failures (A09-2021)	Security Training and Awareness Programs can educate employees about the significance of effective logging and monitoring practices. By raising awareness about the importance of collecting and analyzing security-related logs and implementing real-time monitoring mechanisms, organizations can improve their logging and monitoring capabilities and detect security incidents more effectively.
10	Server-Side Request Forgery (A10-2021)	Security Training and Awareness Programs can educate employees about the risks of server-side request forgery (SSRF) vulnerabilities and the best practices for input validation and secure configurations. By providing employees with knowledge about SSRF and its potential consequences, organizations can promote secure coding practices and reduce the risk of SSRF attacks.

11.4 SOP (Standard Operating Procedures)

The Standard Operating Procedures are implemented as a concrete strategic measure to support the Software Assurance Maturity Model) Operations domain.

11.4.1 Introduction

The Standard Operating Procedures are the last possibility to contain security requirements tasks for addressing security concerns. The Standard Operating Procedures represent the “By Process or Manual tasks”, and they are after the “Security By Design” and the “Security By Automation” approaches.
Since the SOPs approach for addressing security, this method is not as reliable because they are entirely manual. Therefore, it is recommended to SOPs only when the Solution architecture and the DevOps and DevSecOps have not catered for a security concern and must be addressed with the last of the contingencies methods. Considering this, then the SOPs can help enhance the overall security in IT.
It's important to note that while some SOP categories directly address the OWASP Top 10 vulnerabilities, others contribute indirectly to overall security and risk management. As an IT organization, it is required to implement a holistic approach that combines multiple SOP categories to effectively address the full range of OWASP vulnerabilities and ensure comprehensive IT security.

#	Name	Description
1	IT Service Management (ITSM) SOPs	ITSM SOPs primarily focus on managing IT services and processes according to industry best practices like ITIL. ITSM practices indirectly contribute to overall security by ensuring incident management, change management, and problem management.
2	IT Security SOPs	IT Security SOPs are specifically designed to address security-related procedures. These SOPs cover areas such as access control, vulnerability management, patch management, security incident response, user account management, and security awareness training.
3	IT Infrastructure SOPs	IT Infrastructure SOPs focus on managing and maintaining IT infrastructure components. They play a crucial role in ensuring the overall security and resilience of the infrastructure. Procedures related to configuration management, network administration, backup and recovery, and capacity planning.
4	Software Development SOPs	Software Development SOPs cover the software development lifecycle and include procedures for requirements gathering, design, coding, testing, deployment, and maintenance. They may also include measures such as version control, code review, bug tracking, and release management.
5	IT Asset Management SOPs	IT Asset Management SOPs are primarily concerned with the lifecycle management of IT assets, including procurement, inventory management, software license compliance, hardware disposal, and asset tracking. They support overall security and compliance.
6	IT Disaster Recovery (DR) and Business Continuity (BC) SOPs	IT DR and BC SOPs provide guidelines for handling IT disasters and ensuring business continuity. These SOPs contribute to the overall resilience of IT systems and operations.
7	IT Governance and Compliance SOPs	IT Governance and Compliance SOPs focus on ensuring compliance with regulations and internal governance frameworks. They contribute to a comprehensive security posture by addressing risk management, compliance audits, policy enforcement, and data privacy.

11.4.2 IT Service Management (ITSM) SOPs

IT Service Management (ITSM) SOPs may not directly address the specific technical vulnerabilities outlined in the OWASP Top 10. However, they play a crucial role in ensuring the overall management and governance of IT services, which indirectly contributes to improving security posture and mitigating risks.
IT Service Management (ITSM) SOPs, which focus on managing IT services and processes according to industry best practices like ITIL, may not directly address the specific OWASP threats mentioned. However, they play a crucial role in ensuring the overall management and governance of IT services, which indirectly contributes to improving security posture and mitigating risks.
While ITSM SOPs may not directly provide detailed instructions for addressing each OWASP threat individually, they contribute to establishing a structured and well-governed IT environment. Organizations

should complement ITSM SOPs with specific security-focused SOPs or guidelines that directly address the OWASP threats to ensure comprehensive security management.

11.4.2.1 Assessment – SAMM

Standard Operating Procedures (IT Service Management SOPs) can contribute to addressing the SAMM security domains and sub-domains as follows:

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: ITSM SOPs can include policies and standards that define secure coding practices, vulnerability management, incident response, and data protection, thereby addressing this sub-domain. Risk Management: ITSM SOPs can provide guidelines on risk management processes and risk assessments, contributing to addressing this sub-domain. Compliance Management: ITSM SOPs can incorporate activities like compliance assessments and alignment with security frameworks and guidelines, assisting in addressing this sub-domain. Security Training and Awareness: ITSM SOPs can include training programs and awareness campaigns to promote a culture of security, supporting this sub-domain. Security Metrics and Reporting: ITSM SOPs can define security metrics and reporting mechanisms to assess the effectiveness of software security practices, addressing this sub-domain. Security Roles and Responsibilities: ITSM SOPs can outline security roles and responsibilities, establishing accountability and ensuring individuals have the necessary knowledge and skills, contributing to this sub-domain.
2	SAMM Security by Design	Security Requirements: ITSM SOPs can emphasize the inclusion of security requirements into the design phase, addressing this sub-domain. Secure Architecture: ITSM SOPs can promote the adoption of secure architectural patterns and principles, helping address this sub-domain. Secure Coding Practices: While ITSM SOPs may not provide specific coding guidelines, organizations can complement them with secure coding practices to address this sub-domain. Threat Modeling: ITSM SOPs can suggest conducting threat modeling exercises during the design phase, contributing to this sub-domain. Security Testing: ITSM SOPs can encourage the inclusion of security testing activities in the design phase, indirectly addressing this sub-domain.

#	Name	Description
3	SAMM Security by Implementation	Secure Development Training: ITSM SOPs can include training programs on secure coding practices, contributing to this sub-domain. Secure Architecture: ITSM SOPs can promote secure architectural practices, addressing this sub-domain. Secure Coding Guidelines: While ITSM SOPs may not provide specific coding guidelines, organizations can complement them with secure coding guidelines to address this sub-domain. Security Requirements: ITSM SOPs can emphasize the importance of defining and documenting security requirements, indirectly contributing to this sub-domain. Security Testing Integration: ITSM SOPs can encourage the integration of security testing activities, indirectly supporting this sub-domain. Security Verification: ITSM SOPs can promote security verification techniques, addressing this sub-domain. Security Architecture Review: While ITSM SOPs may not provide specific guidelines, they can contribute to security architecture review by establishing the importance of security controls and mechanisms. Security Operations Integration: ITSM SOPs can promote the integration of security considerations into operational processes, indirectly supporting this sub-domain.
4	SAMM Security by Verification	Security Testing: ITSM SOPs can encourage security testing activities, which are directly related to this sub-domain. Code Review: ITSM SOPs can emphasize the importance of code reviews, indirectly contributing to this sub-domain. Security Architecture Review: While ITSM SOPs may not provide specific guidelines, they can indirectly contribute to security architecture reviews by promoting secure design practices. Security Requirements Verification: ITSM SOPs can support the verification of security requirements by emphasizing their importance during the development process. Threat Modeling: ITSM SOPs can suggest conducting threat modeling exercises, indirectly contributing to this sub-domain. Secure Deployment Verification: ITSM SOPs can indirectly address this sub-domain by emphasizing the importance of secure deployment practices.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening: SOPs can provide guidelines for implementing secure configurations, applying patches and updates, and configuring secure networks and infrastructure. Secure Build: SOPs can outline secure coding standards, secure build tools, and secure software development frameworks, which help ensure that software is developed with security in mind. Configuration Management: SOPs can establish processes for version control, change management, and documentation of software configurations, ensuring the integrity and control of software systems. Vulnerability Management: SOPs can define the process for conducting vulnerability scanning, vulnerability assessment, and penetration testing, which helps identify and prioritize vulnerabilities for timely remediation. Incident Management: SOPs can provide guidelines for incident detection, response, containment, and recovery, facilitating effective handling of security incidents. Secure Deployment: SOPs can outline secure software distribution, installation procedures, and configuration practices, ensuring secure and controlled deployment of software components. Security Testing: SOPs can incorporate security testing activities, such as static code analysis, dynamic application security testing, and security code reviews, to identify and address vulnerabilities. Operational Enablement: SOPs can support operational staff by providing security training, documenting operational procedures, and establishing incident response capabilities.

11.4.2.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	Change Management: ITSM SOPs for can include procedures for managing changes in the IT environment, including software updates and patches. This can help address vulnerabilities related to insecure design, misconfigurations, and outdated components, indirectly addressing OWASP threats such as Broken Access Control (A01-2021).
2	Cryptographic Failures (A02-2021)	Change Management: ITSM SOPs for can include procedures for managing changes in the IT environment, including software updates and patches. This can help address vulnerabilities related to insecure design, misconfigurations, and outdated components, indirectly addressing OWASP threats such as Cryptographic Failures (A02-2021).
3	Injection (A03-2021)	Incident Management: ITSM SOPs can outline procedures for handling security incidents, including timely response and resolution. As incident management can involve identifying and remediating vulnerabilities that may lead to these types of attacks, this can indirectly address threats like such as Injection (A03-2021).

#	Name	Description
4	Insecure Design (A04-2021)	Problem Management: ITSM SOPs can include processes for identifying and resolving underlying problems that contribute to security vulnerabilities. This can indirectly address vulnerabilities. As problem management aims to identify and eliminate root causes of recurring incidents, including those caused by insecure design decisions this can address indirectly the Insecure Design (A04-2021).
5	Security Misconfiguration (A05-2021)	Incident Management: ITSM SOPs can outline procedures for handling security incidents, including timely response and resolution. As incident management can involve identifying and remediating vulnerabilities that may lead to these types of attacks, this can indirectly address threats like such as Security Misconfiguration (A05-2021).
6	Vulnerable and Outdated Components (A06-2021)	Change Management: ITSM SOPs for can include procedures for managing changes in the IT environment, including software updates and patches. This can help address vulnerabilities related to insecure design, misconfigurations, and outdated components, indirectly addressing OWASP threats such as Vulnerable and Outdated Components (A06-2021)
7	Identification and Authentication Failures (A07-2021)	Service Level Management: ITSM SOPs can establish procedures for defining and monitoring service levels, including security-related metrics and requirements. By ensuring that appropriate authentication mechanisms and password policies are in place can his can indirectly address the Identification and Authentication Failures (A07-2021).
8	Software and Data Integrity Failures (A08-2021)	Service Continuity: ITSM SOPs can provide guidelines for business continuity planning and disaster recovery, ensuring the availability and integrity of critical services and data. By establishing measures to protect against data manipulation or destruction can indirectly address threats like Software and Data Integrity Failures (A08-2021) Vendor Management: ITSM SOPs can include guidelines for evaluating and managing third-party vendors, including their security practices. By emphasizing the importance of assessing the security of the components used in the IT environment can indirectly address the Vulnerable and Outdated Components (A06-2021)

11.4.3 IT Security SOPs

While specific SOPs may not be mentioned for each OWASP category, the overall coverage of IT Security SOPs aligns with the goal of mitigating OWASP vulnerabilities.

These SOPs indirectly or directly address various OWASP threats by implementing security controls, best practices, and procedures to ensure the security of web applications.

11.4.3.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: SOPs can define and implement policies and standards related to software security, such as secure coding practices, vulnerability management, incident response, and data protection. Risk Management: SOPs can outline risk management processes and procedures, including risk assessments and mitigation strategies, to address risks associated with software development and deployment. Compliance Management: SOPs can ensure compliance with regulatory requirements, industry standards, and best practices by establishing processes for compliance assessments, reporting, and alignment with security frameworks. Security Training and Awareness: SOPs can provide guidelines and resources for conducting security training programs and awareness campaigns to educate employees and stakeholders about software security. Security Metrics and Reporting: SOPs can define security metrics and reporting mechanisms to assess the effectiveness of software security practices and support decision-making at various levels of the organization. Security Roles and Responsibilities: SOPs can clearly define security roles and responsibilities within the organization, ensuring accountability for software security and enabling individuals to fulfill their roles effectively.
2	SAMM Security by Design	Security Requirements: SOPs can include procedures for identifying and incorporating security requirements into the design phase, addressing specific security concerns and mitigating OWASP threats. Secure Architecture: SOPs can promote the adoption of secure architectural patterns and principles, helping address OWASP threats by minimizing vulnerabilities and creating a strong foundation for security controls. Secure Coding Practices: SOPs can enforce secure coding practices during the design phase, reducing the likelihood of OWASP threats by preventing common coding mistakes and vulnerabilities. Threat Modeling: SOPs can guide the process of conducting threat modeling exercises, allowing organizations to identify and mitigate potential security risks associated with OWASP threats. Security Testing: SOPs can incorporate security testing activities, such as static code analysis and security code reviews, into the design phase to detect and address vulnerabilities related to OWASP threats.
3	SAMM Security by Implementation	Secure Development Training: SOPs can provide guidelines for delivering secure development training programs, ensuring that developers have the necessary knowledge and skills to build secure software. Secure Architecture: SOPs can outline practices like threat modeling and secure design patterns to be integrated into the software development process, enhancing security and addressing vulnerabilities.

#	Name	Description
		Secure Coding Guidelines: SOPs can establish coding standards and best practices to guide developers in writing secure code, reducing the occurrence of OWASP threats. Security Requirements: SOPs can define procedures for documenting and implementing security requirements, serving as a basis for designing and implementing security controls. Security Testing Integration: SOPs can describe the integration of security testing activities throughout the software development life cycle, helping identify and address vulnerabilities associated with OWASP threats. Security Verification: SOPs can outline processes for verifying the effectiveness of security controls implemented in the software system, ensuring the software meets the desired security objectives. Security Architecture Review: SOPs can guide the process of conducting security architecture reviews to evaluate the design and implementation of security controls. Security Operations Integration: SOPs can facilitate the integration of security considerations into operational processes, such as incident management and secure deployment practices.
4	SAMM Security by Verification	Security Testing: Standard Operating Procedures can include guidelines for conducting security testing activities such as penetration testing, vulnerability scanning, code review, and security testing automation. These procedures help validate and verify the security of software applications. Code Review: Standard Operating Procedures can include guidelines for conducting manual code reviews and using automated code analysis tools to identify security flaws and adherence to secure coding practices. Security Architecture Review: Standard Operating Procedures can outline the process for assessing the security architecture of software applications, including reviewing the design and implementation of security features and ensuring compliance with security standards and best practices. Security Requirements Verification: Standard Operating Procedures can provide guidelines for verifying that security requirements are properly defined, documented, and implemented in software applications. This includes reviewing security requirements against industry standards, regulatory requirements, and best practices. Threat Modeling: Standard Operating Procedures can outline the process for conducting threat modeling exercises to identify potential threats and risks to software applications. This helps organizations prioritize security threats and implement appropriate security controls. Secure Deployment Verification: Standard Operating Procedures can include guidelines for verifying that software applications are securely deployed and configured in the production environment. This includes verifying the implementation of security controls, encryption mechanisms, access controls, and other security measures during deployment.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening: IT Security SOPs can include procedures for environment hardening, such as implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations. These procedures align with the Environment Hardening component in SAMM's Operations domain, which focuses on protecting software systems from known vulnerabilities and security weaknesses. Secure Build: IT Security SOPs can incorporate secure build practices, including secure coding standards, secure build tools, and secure software development frameworks. By following these practices, organizations can ensure that the software is developed with security in mind, addressing the Secure Build component in SAMM's Operations domain. Configuration Management: SOPs for configuration management, such as version control, change management, and documentation of software configurations, align with the Configuration Management component in SAMM's Operations domain. These procedures help organizations maintain the integrity of software systems and detect unauthorized changes or misconfigurations. Vulnerability Management: IT Security SOPs can outline a vulnerability management process, including vulnerability scanning, vulnerability assessment, and penetration testing. By following these procedures, organizations can identify and prioritize vulnerabilities, addressing the Vulnerability Management component in SAMM's Operations domain. Incident Management: SOPs for incident management, covering incident detection, response, containment, and recovery, contribute to the Incident Management component in SAMM's Operations domain. These procedures help organizations handle security incidents effectively and minimize their impact. Secure Deployment: IT Security SOPs can provide guidance on secure deployment practices, including secure software distribution, installation procedures, and configuration of software components. These practices align with the Secure Deployment component in SAMM's Operations domain, ensuring that software is deployed securely and reducing the risk of unauthorized access or compromise. Security Testing: Incorporating security testing activities, such as static code analysis, dynamic application security testing (DAST), and security code reviews, into IT Security SOPs helps organizations address the Security Testing component in SAMM's Operations domain. These activities enable organizations to identify and address security vulnerabilities before software systems are deployed. Operational Enablement: IT Security SOPs can include procedures for operational support and guidance, such as security training for operational staff, documentation of operational procedures, and incident response capabilities. These procedures contribute to the Operational Enablement component in SAMM's Operations domain, ensuring ongoing security and reliability of software systems.

11.4.3.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	IT Security SOPs refers to Access control procedures within the IT Security SOPs directly address this threat by implementing proper access controls, verifying and enforcing authorization mechanisms, and preventing unauthorized access.
2	Cryptographic Failures (A02-2021)	IT Security SOPs refer to Cryptographic practices, such as encryption and secure key management, are typically covered in IT Security SOPs. By following recommended cryptographic practices, organizations can address the vulnerabilities mentioned in this category.
3	Injection (A03-2021)	IT Security SOPs refers to SOPs related to input validation, secure coding practices, and regular vulnerability testing can help address injection vulnerabilities by ensuring that untrusted data is properly handled and sanitized to prevent unintended code execution.
4	Insecure Design (A04-2021)	While the specific SOPs might not be explicitly mentioned, addressing insecure design vulnerabilities requires adopting secure design principles, considering security from the early stages of development, and implementing appropriate security controls. These practices are typically covered in IT Security SOPs.
5	Security Misconfiguration (A05-2021)	IT Security SOPs refers to secure configuration practices, regular software updates, and security assessments can help address security misconfigurations by ensuring that systems, servers, frameworks, and web applications are properly configured to adhere to secure practices and industry standards.
6	Vulnerable and Outdated Components (A06-2021)	IT Security SOPs refers to to vulnerability management and patch management can help address this threat by actively managing software dependencies, staying updated with security patches, and following secure coding practices.
7	Identification and Authentication Failures (A07-2021)	IT Security SOPs refers to user account management, authentication mechanisms, and password policies can help address identification and authentication failures by implementing secure authentication mechanisms, enforcing strong password policies, and regularly testing for vulnerabilities.
8	Software and Data Integrity Failures (A08-2021)	IT Security SOPs refers to secure coding techniques, secure practices, and regular vulnerability testing can help address software and data integrity failures by ensuring the integrity and protection of software components, configurations, and data within web applications.
9	Security Logging and Monitoring Failures (A09-2021)	IT Security SOPs refers to security incident response, log management, and real-time monitoring mechanisms can help address security logging and monitoring failures by implementing comprehensive logging, effective log analysis, and proper monitoring mechanisms.
10	Server-Side Request Forgery (A10-2021)	IT Security SOPs refers to input validation, secure configurations, and regular security testing can help address server-side request forgery vulnerabilities by implementing robust input validation, secure configurations, and preventing unintended requests to internal or external resources.

11.4.4 IT Infrastructure SOPs

The provided Standard Operating Procedures (SOPs) categories partially address some of the OWASP threats.

While IT Infrastructure SOPs play a role in overall security and resilience, they may not directly address the specific vulnerabilities outlined in the OWASP Top 10. To comprehensively address the OWASP threats, it is important to have additional SOPs and practices specifically tailored to web application development and security.

While IT Infrastructure SOPs may not directly address the OWASP Top 10 vulnerabilities, they play a crucial role in ensuring the overall security and resilience of the infrastructure.

Procedures related to configuration management, network administration, backup and recovery, and capacity planning indirectly contribute to mitigating some OWASP vulnerabilities.

11.4.4.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: IT Infrastructure SOPs indirectly contribute to defining and implementing policies and standards for software security, including areas such as secure coding practices, vulnerability management, incident response, and data protection. Risk Management: IT Infrastructure SOPs play a role in risk management by addressing areas such as configuration management and backup and recovery procedures, indirectly helping to identify and mitigate risks associated with software development and deployment. Compliance Management: While IT Infrastructure SOPs may not directly address compliance management, they contribute to establishing robust operational processes and practices that support compliance with relevant regulatory requirements, industry standards, and best practices related to software security. Security Training and Awareness: IT Infrastructure SOPs indirectly support security training and awareness efforts by establishing procedures and practices that promote a culture of security throughout the organization. Security Metrics and Reporting: IT Infrastructure SOPs indirectly contribute to security metrics and reporting by establishing procedures related to configuration management, network administration, and capacity planning, which can provide visibility into the status of software security. Security Roles and Responsibilities: IT Infrastructure SOPs play a role in defining and assigning security roles and responsibilities within the organization by establishing procedures and practices related to network administration and configuration management.

#	Name	Description
2	SAMM Security by Design	Security Requirements: IT Infrastructure SOPs do not directly address security requirements in the context of software development and design, as they primarily focus on infrastructure management procedures. Secure Architecture: IT Infrastructure SOPs do not directly address secure architecture practices in software design, as they primarily focus on infrastructure management procedures. Secure Coding Practices: IT Infrastructure SOPs do not directly address secure coding practices in software design, as they primarily focus on infrastructure management procedures. Threat Modeling: IT Infrastructure SOPs do not directly address threat modeling practices in software design, as they primarily focus on infrastructure management procedures. Security Testing: IT Infrastructure SOPs do not directly address security testing practices in software design, as they primarily focus on infrastructure management procedures.
3	SAMM Security by Implementation	Secure Development Training: IT Infrastructure SOPs indirectly contribute to secure development training by establishing procedures and practices related to network administration, configuration management, and capacity planning, which can enhance the overall security knowledge and skills of development teams. Secure Architecture: IT Infrastructure SOPs do not directly address secure architecture practices in software implementation, as they primarily focus on infrastructure management procedures. Secure Coding Guidelines: IT Infrastructure SOPs do not directly address secure coding guidelines in software implementation, as they primarily focus on infrastructure management procedures. Security Requirements: IT Infrastructure SOPs do not directly address security requirements in software implementation, as they primarily focus on infrastructure management procedures. Security Testing Integration: IT Infrastructure SOPs indirectly contribute to security testing integration by establishing procedures and practices related to network administration, configuration management, and capacity planning, which can facilitate the integration of security testing activities throughout the software development life cycle. Security Verification: IT Infrastructure SOPs indirectly contribute to security verification by establishing procedures and practices related to network administration, configuration management, and capacity planning, which can support the verification of security controls implemented in the software system. Security Architecture Review: IT Infrastructure SOPs do not directly address security architecture review in software implementation, as they primarily focus on infrastructure management procedures. Security Operations Integration: IT Infrastructure SOPs indirectly contribute to security operations integration by establishing procedures and practices related to network administration, configuration management, and capacity planning, which can support the integration of security considerations into operational processes.

#	Name	Description
4	SAMM Security by Verification	Security Testing: The IT Infrastructure SOPs indirectly contribute to security testing by ensuring proper configuration management, network administration, backup and recovery, and capacity planning. These activities help create a more secure and resilient infrastructure, which in turn supports effective security testing to identify vulnerabilities and weaknesses in software applications.
5	SAMM Security by Operations	Environment Hardening: The IT Infrastructure SOPs contribute to environment hardening by providing procedures for secure configuration management. These SOPs ensure that the software environment is properly configured, patches and updates are applied, and secure network and infrastructure configurations are in place, minimizing the risk of known vulnerabilities and security weaknesses. Configuration Management: The IT Infrastructure SOPs directly contribute to configuration management by providing procedures for managing software configurations. These SOPs cover areas such as version control, change management, and documentation of configurations, enabling organizations to maintain the integrity of software systems and detect unauthorized changes or misconfigurations. Vulnerability Management: The IT Infrastructure SOPs indirectly contribute to vulnerability management by ensuring proper backup and recovery procedures. In the event of a security incident or vulnerability discovery, the SOPs help organizations recover from the incident and restore normal operations promptly. Incident Management: The IT Infrastructure SOPs indirectly contribute to incident management by providing procedures for backup and recovery. In the event of a security incident, the SOPs help organizations respond, contain, and recover from the incident, minimizing its impact on software systems. Secure Deployment: The IT Infrastructure SOPs indirectly contribute to secure deployment by providing procedures for backup and recovery. These SOPs help organizations ensure that software is deployed securely and that the deployment process includes appropriate security measures to prevent unauthorized access or compromise. Security Testing: The IT Infrastructure SOPs indirectly contribute to security testing by ensuring proper configuration management. By maintaining a secure and well-managed infrastructure, organizations can conduct effective security testing activities, such as static code analysis, dynamic application security testing, and security code reviews, to identify and address vulnerabilities in software systems.

11.4.4.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	IT Infrastructure SOPs indirectly contribute to mitigating this vulnerability by ensuring proper configuration management and access control mechanisms for the infrastructure components. However, the specific implementation details related to web application access controls are not addressed directly.
2	Cryptographic Failures (A02-2021)	IT Infrastructure SOPs may indirectly contribute to addressing this vulnerability by managing cryptographic components at the infrastructure level. However, the SOPs need to specifically address the implementation of cryptographic practices within web applications to fully mitigate this vulnerability.
3	Injection (A03-2021)	It does not address this domain directly.
4	Insecure Design (A04-2021)	It does not address this domain directly.
5	Security Misconfiguration (A05-2021)	IT Infrastructure SOPs indirectly contribute to mitigating security misconfiguration vulnerabilities by ensuring secure configurations for infrastructure components. However, the SOPs need to address web application-specific configurations to fully mitigate this vulnerability.
6	Vulnerable and Outdated Components (A06-2021)	IT Infrastructure SOPs may indirectly contribute to addressing this vulnerability by managing software dependencies and staying updated with security patches at the infrastructure level. However, the SOPs should also consider the management of web application-specific components.
7	Identification and Authentication Failures (A07-2021)	It does not address this domain directly.
8	Software and Data Integrity Failures (A08-2021)	It does not address this domain directly.
9	Security Logging and Monitoring Failures (A09-2021)	IT Infrastructure SOPs may indirectly contribute to addressing this vulnerability by ensuring comprehensive logging and monitoring mechanisms for infrastructure components. However, specific logging and monitoring practices for web applications should be addressed at the application layer.
10	Server-Side Request Forgery (A10-2021)	It does not address this domain directly.

11.4.5 Software Development SOPs

The Software Development SOPs can address the majority of the OWASP Top 10 vulnerabilities by incorporating secure development practices, secure coding techniques, proper access controls, secure configurations, regular security testing, and adherence to industry best practices.

However, it is essential to note that the effectiveness of these SOPs depends on their implementation, adherence, and continuous improvement as per the specific context and requirements of the organization.

The Software Development SOPs align with addressing the OWASP Top 10 vulnerabilities by covering the software development lifecycle and include procedures for requirements gathering, design, coding, testing,

deployment, and maintenance. They may also include measures such as version control, code review, bug tracking, and release management, which directly contribute to mitigating OWASP vulnerabilities.
It's important to note that while SOP are manual documentation and procedures, some of these can be automated by using IDE plugins. These plugins and tools can be helpful, but they are not considred a substitute for secure coding practices, proper design, and thorough security testing. They should be used in conjunction with a comprehensive security approach and customized to the specific requirements and technologies used in your application.

11.4.5.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: The Software Development SOPs contribute to this sub-domain by incorporating secure coding practices, secure configurations, and regular security testing, which align with defining and implementing policies and standards for software security. Risk Management: The Software Development SOPs help address this sub-domain by promoting the identification and mitigation of risks associated with software development through activities such as secure coding techniques, regular security testing, and adherence to industry best practices. Compliance Management: The Software Development SOPs indirectly contribute to compliance management by incorporating security practices aligned with regulatory requirements, industry standards, and best practices, ensuring that the software development process meets the necessary compliance criteria. Security Training and Awareness: The Software Development SOPs can support this sub-domain by including training programs and resources that educate employees and stakeholders about secure coding practices and software security, fostering a culture of security within the organization. Security Metrics and Reporting: While the Software Development SOPs may not directly address this sub-domain, organizations can incorporate metrics and reporting mechanisms into their SOPs to assess the effectiveness of software security practices and provide visibility into the status of security measures. Security Roles and Responsibilities: The Software Development SOPs can contribute to this sub-domain by defining security roles and responsibilities within the organization, ensuring that individuals involved in software development have the necessary knowledge and skills to fulfill their security-related roles effectively.
2	SAMM Security by Design	Security Requirements: The Software Development SOPs can help address this sub-domain by including procedures for gathering security requirements and incorporating them into the design phase of software development, ensuring that security concerns are identified and addressed proactively. Secure Architecture: The Software Development SOPs align with this sub-domain by promoting the adoption of secure architectural patterns and principles, which contribute to designing robust and secure software systems that mitigate vulnerabilities associated with OWASP threats.

#	Name	Description
		Secure Coding Practices: The Software Development SOPs directly contribute to this sub-domain by incorporating secure coding practices, ensuring that developers adhere to secure coding guidelines and mitigate vulnerabilities related to OWASP threats during the design phase. Threat Modeling: While the Software Development SOPs may not explicitly cover threat modeling exercises, organizations can include guidelines or references to threat modeling practices in their SOPs to encourage the identification and mitigation of potential security risks during the design phase. Security Testing: The Software Development SOPs align with this sub-domain by promoting the inclusion of security testing activities, such as static code analysis and security code reviews, during the design phase to detect and address vulnerabilities related to OWASP threats.
3	SAMM Security by Implementation	Secure Development Training: The Software Development SOPs contribute to this sub-domain by including training programs on secure coding practices, secure design principles, and common software vulnerabilities, ensuring that developers have the necessary knowledge and skills to implement secure software measures during the development process. Secure Architecture: The Software Development SOPs align with this sub-domain by promoting the use of secure architecture practices, such as threat modeling and secure design patterns, during the implementation phase, contributing to the integration of security measures into the software development life cycle. Secure Coding Guidelines: The Software Development SOPs directly contribute to this sub-domain by establishing and enforcing secure coding guidelines and standards, guiding developers in writing secure code and reducing the likelihood of introducing vulnerabilities during implementation. Security Requirements: The Software Development SOPs align with this sub-domain by emphasizing the need to define and document security requirements, which serve as a basis for implementing security controls and mitigating vulnerabilities during software implementation. Security Testing Integration: The Software Development SOPs directly contribute to this sub-domain by promoting the integration of security testing activities, such as static code analysis and dynamic application security testing, into the implementation phase to identify and address security vulnerabilities. Security Verification: The Software Development SOPs align with this sub-domain by advocating for security verification techniques.
4	SAMM Security by Verification	SOPs contribute to Security Testing by providing procedures for conducting various types of security testing, such as penetration testing, vulnerability scanning, code review, and security testing automation. SOPs support Code Review by providing guidelines and procedures for reviewing the source code of software applications to identify security flaws and ensure adherence to secure coding practices.

#	Name	Description
		SOPs aid in Security Architecture Review by providing procedures for assessing the security architecture of software applications, reviewing design and implementation of security features, and ensuring compliance with security standards and best practices. SOPs assist in Security Requirements Verification by providing procedures for reviewing and verifying that security requirements are properly defined, documented, and implemented in software applications. SOPs support Threat Modeling by providing procedures for conducting threat modeling exercises to identify potential threats and risks to software applications. SOPs aid in Secure Deployment Verification by providing procedures for ensuring secure deployment and configuration of software applications in the production environment, including the implementation of security controls, encryption mechanisms, and access controls.
5	SAMM Security by Operations	SOPs contribute to Environment Hardening by providing procedures for implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations. SOPs aid in Secure Build by providing guidelines and procedures for implementing secure coding standards, secure build tools, and secure software development frameworks. SOPs support Configuration Management by providing procedures for effective version control, change management, and documentation of software configurations. SOPs assist in Vulnerability Management by providing procedures for conducting vulnerability scanning, vulnerability assessment, and penetration testing to identify and prioritize vulnerabilities. SOPs contribute to Incident Management by providing procedures for incident detection, response, containment, and recovery activities. SOPs aid in Secure Deployment by providing procedures for secure software distribution, installation procedures, and secure configuration of software components. SOPs support Security Testing by providing procedures for conducting regular security testing activities, such as static code analysis, dynamic application security testing (DAST), and security code reviews. SOPs contribute to Operational Enablement by providing operational support and guidance, including security training for operational staff, documentation of operational procedures, and establishing incident response capabilities.

11.4.5.2 Assessment – OWASP Top 10

#	Name	Description
1	Broken Access Control (A01-2021)	The Software Development SOPs can address the Broken Access Control vulnerability by including procedures for implementing and enforcing proper access controls within web applications. This can involve role-based access control (RBAC), authentication mechanisms, authorization checks, and session management techniques to prevent unauthorized access and privilege escalation. Example as IDE Plugin: Authorization and Role-Based Access Control (RBAC) libraries: Plugins like ASP.NET Identity, IdentityServer, or SimpleAuthorization can help implement proper access controls and enforce authorization rules.
2	Cryptographic Failures (A02-2021)	The Software Development SOPs can address Cryptographic Failures by providing guidelines and best practices for implementing secure cryptographic algorithms and mechanisms within web applications. This includes proper key management, encryption/decryption, secure storage of cryptographic materials, and protection against common cryptographic weaknesses like weak algorithms or improper usage. Example as IDE Plugin: CryptoHelper: Plugins like Bouncy Castle or System.Security.Cryptography provide cryptographic libraries and helper classes to ensure secure implementation of encryption, hashing, and digital signatures.
3	Injection (A03-2021)	The Software Development SOPs can address Injection vulnerabilities by emphasizing secure coding practices, input validation, and parameterization of user-supplied data. These SOPs can guide developers on how to prevent common injection attacks such as SQL injection, OS command injection, and LDAP injection by sanitizing user inputs, using prepared statements, or employing ORM frameworks. Example as IDE Plugin: Static Code Analysis Tools: Visual Studio integrates with static code analysis tools like ReSharper, SonarLint, or Fortify that can detect potential injection vulnerabilities and suggest secure coding practices.
4	Insecure Design (A04-2021)	The Software Development SOPs can address Insecure Design vulnerabilities by promoting secure design principles and considerations from the early stages of development. This involves threat modeling, risk assessment, secure architecture design, input/output validation, secure configuration management, and incorporating security controls to mitigate potential vulnerabilities. Example as IDE Plugin: Threat Modeling Tools: Tools like Microsoft Threat Modeling Tool can assist in identifying and addressing design vulnerabilities by facilitating threat modeling exercises and providing guidance on secure design decisions.

#	Name	Description
5	Security Misconfiguration (A05-2021)	The Software Development SOPs can address Security Misconfiguration vulnerabilities by emphasizing secure configuration practices for web applications and associated components. This includes properly configuring server settings, application frameworks, libraries, databases, and other components to adhere to secure industry standards. The SOPs can also cover regular updates and patch management to address known security issues. Example as IDE Plugin: Configuration Management Tools: Visual Studio has built-in configuration management features that can help manage and enforce secure configurations for web applications and associated components.
6	Vulnerable and Outdated Components (A06-2021)	The Software Development SOPs can address Vulnerable and Outdated Components vulnerabilities by providing guidelines on managing software dependencies, libraries, frameworks, and other third-party components. This includes staying updated with security patches, conducting regular vulnerability assessments, and employing secure coding practices to mitigate the risks associated with using outdated or known vulnerable components. Example as IDE Plugin: NuGet Package Manager: Visual Studio's NuGet Package Manager allows you to manage dependencies and update packages to their latest secure versions. Additionally, tools like OWASP Dependency Check or WhiteSource Bolt can scan for vulnerable dependencies.
7	Identification and Authentication Failures (A07-2021)	The Software Development SOPs can address Identification and Authentication Failures vulnerabilities by specifying secure mechanisms for user identification and authentication. This includes implementing strong password policies, multi-factor authentication (MFA), session management, secure credential storage, and protection against common authentication attacks like brute-force attacks or credential stuffing. Example as IDE Plugin: Authentication Libraries: Plugins like IdentityServer, Microsoft Authentication Libraries (MSAL), or OpenID Connect can simplify the implementation of secure authentication mechanisms, including multi-factor authentication (MFA) and secure credential storage.
8	Software and Data Integrity Failures (A08-2021)	The Software Development SOPs can address Software and Data Integrity Failures vulnerabilities by promoting secure coding techniques, secure software development practices, and integrity checks for software components, configurations, and data. This includes implementing mechanisms for data validation, input/output integrity checks, secure storage, and protection against unauthorized modifications or tampering. Example as IDE Plugin: Code Integrity Tools: Visual Studio's code analysis features and plugins like SonarQube or Veracode can help identify potential code integrity issues, including unauthorized modifications or tampering.

#	Name	Description
9	Security Logging and Monitoring Failures (A09-2021)	The Software Development SOPs can address Security Logging and Monitoring Failures vulnerabilities by emphasizing the importance of effective logging and monitoring practices within web applications. This includes defining logging requirements, implementing comprehensive logging mechanisms, conducting log analysis, and setting up real-time monitoring for detecting and responding to security incidents. Example as IDE Plugin: Logging Frameworks: Plugins like Serilog, NLog, or log4net can assist in implementing comprehensive logging mechanisms, including security-related logs. These frameworks offer flexibility and integration options with various log analysis tools.
10	Server-Side Request Forgery (A10-2021)	The Software Development SOPs can address Server-Side Request Forgery (SSRF) vulnerabilities by providing guidelines for input validation, secure configurations, and regular security testing. This includes validating and sanitizing user inputs, implementing whitelist-based URL validation, employing secure network configurations, and conducting penetration testing to identify and mitigate SSRF risks. Example as IDE Plugin: HTTP Request Validation Libraries: Plugins like ASP.NET Request Validation, OWASP AntiSamy, or Microsoft Security Code Analysis can help prevent server-side request forgery by validating and sanitizing user-supplied inputs.

11.4.6 IT Asset Management SOPs

The Standard Operating Procedures (SOPs) categories mentioned, namely IT Asset Management SOPs, do not directly address the specific OWASP Top 10 vulnerabilities. These SOPs primarily focus on the lifecycle management of IT assets, including procurement, inventory management, software license compliance, hardware disposal, and asset tracking. While these procedures indirectly support overall security and compliance, they are not specifically designed to address the OWASP vulnerabilities.

11.4.6.1 Assessment – SAMM

IT Asset Management SOPs primarily contribute to the Security by Governance domain but also have indirect contributions to other SAMM domains by ensuring that IT assets are properly managed, configured, and used securely throughout their lifecycle.

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: IT Asset Management SOPs contribute by defining and enforcing policies and standards related to IT asset management. These policies ensure that assets are acquired, used, and retired following established guidelines, which includes aspects like security and compliance. Risk Management: Effective IT Asset Management plays a role in risk management by ensuring that all assets are properly identified, managed, and secured. This reduces the risk of unauthorized access, data breaches, and non-compliance with regulations. Compliance Management: IT Asset Management SOPs help in compliance management by ensuring that all IT assets, especially software licenses, are tracked and managed in a way that ensures compliance with relevant regulatory requirements and licensing agreements.
2	SAMM Security by Design	Security Requirements: IT Asset Management contributes by ensuring that security requirements are considered during the procurement and deployment of assets. For example, when acquiring new software, compliance with security requirements can be a criterion for selection. Secure Architecture: IT Asset Management SOPs help in secure architecture by ensuring that assets are introduced into the organization's ecosystem in a way that aligns with secure architectural principles. This includes considerations like data protection and access controls.
3	SAMM Security by Implementation	Secure Development Training: While IT Asset Management primarily deals with asset lifecycle management, it indirectly supports secure development training by ensuring that assets used in development (e.g., software tools) are properly managed and secure. Secure Architecture: IT Asset Management supports secure architecture during the implementation phase by ensuring that assets are configured and used in a secure manner as part of the development and deployment process. Security Testing Integration: IT Asset Management ensures that security testing tools and resources are available and managed effectively as assets, which indirectly supports security testing integration.
4	SAMM Security by Verification	Security Testing: IT Asset Management SOPs contribute by ensuring that security testing tools and resources are properly managed and accessible for use during verification activities.

#	Name	Description
5	SAMM Security by Operations	Environment Hardening: IT Asset Management supports environment hardening by ensuring that assets are configured and maintained securely. For example, ensuring that servers and network devices are hardened according to best practices. Secure Build: IT Asset Management ensures that assets related to software development and build processes are managed effectively, indirectly contributing to secure build practices. Configuration Management: IT Asset Management contributes to configuration management by ensuring that all assets are properly documented, tracked, and managed, including changes and updates. Security Testing: IT Asset Management indirectly supports security testing by ensuring that the necessary tools and resources for security testing are available and managed effectively.

11.4.6.2 Assessment – OWASP Top 10

Effective IT asset management practices can contribute to maintaining a secure environment by ensuring that software and hardware components are up-to-date, properly configured, and free from known vulnerabilities. Properly managing software licenses can also help in keeping software components patched and up-to-date, reducing the risk of using vulnerable or outdated components.
While the mentioned SOPs may not directly address the OWASP Top 10 vulnerabilities, they play a crucial role in maintaining a secure and well-managed IT infrastructure, which is an important aspect of overall security. Organizations should adopt a holistic approach that combines SOPs, secure coding practices, secure design principles, and other security measures to effectively address the OWASP vulnerabilities and enhance the security posture of their applications.

#	Name	Description
1	Broken Access Control (A01-2021)	IT Asset Management SOPs indirectly contribute to addressing this threat by ensuring that access controls are correctly applied to IT assets. By maintaining an inventory of assets and their access permissions, organizations can reduce the risk of unauthorized access and privilege escalation, which are key aspects of Broken Access Control.
2	Cryptographic Failures (A02-2021)	While IT Asset Management SOPs do not directly address cryptographic failures, they indirectly support cryptographic best practices by ensuring that assets related to cryptography, such as encryption software and hardware security modules, are properly tracked and managed. This contributes to the overall security posture of the organization.

#	Name	Description
3	Injection (A03-2021)	IT Asset Management SOPs do not directly address injection vulnerabilities. However, they indirectly support secure coding practices by ensuring that software development assets, such as code analysis tools and secure coding guidelines, are available and properly managed. Developers can use these assets to write more secure code and mitigate injection vulnerabilities.
4	Insecure Design (A04-2021)	IT Asset Management SOPs indirectly contribute to addressing insecure design vulnerabilities by ensuring that assets related to design tools and resources are properly managed. This can include software used for architectural design and threat modeling. These assets enable organizations to consider security during the design phase and reduce the risk of insecure design.
5	Security Misconfiguration (A05-2021)	IT Asset Management SOPs do not directly address security misconfigurations. However, they indirectly support secure configuration management by ensuring that assets like configuration management tools and documentation are available and properly managed. These assets assist organizations in maintaining secure configurations and reducing the risk of misconfigurations.
6	Vulnerable and Outdated Components (A06-2021)	IT Asset Management SOPs indirectly contribute to addressing this threat by tracking and managing software assets. This includes monitoring and managing third-party libraries and software components for updates and vulnerabilities. While not directly addressing the threat, this practice helps organizations keep their software components up to date and reduce the risk of vulnerabilities associated with outdated components.
7	Identification and Authentication Failures (A07-2021)	IT Asset Management SOPs do not directly address identification and authentication failures. However, they indirectly support secure authentication mechanisms by ensuring that assets related to authentication systems and tools are available and properly managed. These assets can assist in implementing secure authentication practices.
8	Software and Data Integrity Failures (A08-2021)	IT Asset Management SOPs do not directly address integrity failures. Nevertheless, they indirectly support data integrity by tracking and managing assets related to data protection and integrity, such as encryption tools and data backup solutions. Proper management of these assets contributes to data integrity.
9	Security Logging and Monitoring Failures (A09-2021)	IT Asset Management SOPs do not directly address security logging and monitoring failures. However, they indirectly support logging and monitoring by ensuring that assets related to security information and event management (SIEM) systems and log analysis tools are available and properly managed. These assets enable organizations to enhance their logging and monitoring capabilities.
10	Server-Side Request Forgery (A10-2021)	IT Asset Management SOPs do not directly address server-side request forgery. However, they indirectly support security by ensuring that assets related to security testing tools, including vulnerability scanners, are available and properly managed. These assets can assist in identifying and mitigating server-side request forgery vulnerabilities during security testing.

11.4.7 IT Disaster Recovery (DR) and Business Continuity (BC) SOPs

IT DR and BC SOPs primarily focus on mitigating and recovering from IT disasters and ensuring the continuity of business operations. While these SOPs contribute to the overall resilience of IT systems and operations, they do not specifically address the OWASP Top 10 vulnerabilities.

The OWASP Top 10 vulnerabilities, on the other hand, are specific security threats that affect web applications. These vulnerabilities encompass a range of risks and require targeted mitigation strategies.

11.4.7.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: IT Asset Management SOPs contribute to the establishment and implementation of policies and standards related to software asset management, ensuring compliance with software license requirements and tracking of IT assets (Governance - Policies and Standards). Risk Management: IT Asset Management SOPs help mitigate risks associated with software asset management, such as non-compliance with software license agreements or potential security vulnerabilities resulting from unmanaged assets (Governance - Risk Management). Compliance Management: IT Asset Management SOPs support compliance with regulatory requirements related to software asset management, ensuring proper documentation and tracking of software licenses (Governance - Compliance Management). Security Training and Awareness: IT Asset Management SOPs may include security training and awareness components for employees involved in the management and procurement of IT assets, promoting a culture of security within the organization (Governance - Security Training and Awareness). Security Metrics and Reporting: IT Asset Management SOPs may define metrics and reporting mechanisms for tracking and assessing the compliance and security aspects of software asset management processes (Governance - Security Metrics and Reporting). Security Roles and Responsibilities: IT Asset Management SOPs can help define and assign security roles and responsibilities related to software asset management, ensuring accountability and appropriate knowledge and skills for individuals involved (Governance - Security Roles and Responsibilities).
2	SAMM Security by Design	Security Requirements: IT Asset Management SOPs indirectly contribute to the identification and consideration of security requirements related to software asset management, ensuring secure procurement, inventory management, and data protection (Design - Security Requirements).

#	Name	Description
		Secure Architecture: While IT Asset Management SOPs do not directly address secure architecture, they indirectly support the implementation of security controls and risk management measures associated with software asset management (Design - Secure Architecture). Secure Coding Practices: IT Asset Management SOPs do not directly address secure coding practices but may indirectly contribute to the establishment of procurement processes that consider secure software development practices (Design - Secure Coding Practices). Threat Modeling: IT Asset Management SOPs do not directly address threat modeling activities but may indirectly contribute to risk assessment and management practices associated with software asset management (Design - Threat Modeling). Security Testing: IT Asset Management SOPs do not directly address security testing activities but may indirectly contribute to the identification and management of security vulnerabilities associated with software assets (Design - Security Testing).
3	SAMM Security by Implementation	Secure Development Training: SOPs can provide guidelines for secure coding practices and training programs, equipping developers with the necessary knowledge and skills to build secure software. Secure Architecture: SOPs can guide the use of secure architecture practices, such as threat modeling and secure component selection, ensuring security considerations are integrated from the early stages of development. Secure Coding Guidelines: SOPs can establish coding standards and best practices, reducing the likelihood of introducing vulnerabilities and addressing OWASP threats through proper coding techniques. Security Requirements: SOPs can define processes for documenting and implementing security requirements, which serve as a basis for designing and implementing security controls. Security Testing Integration: SOPs can outline processes for integrating security testing activities, such as static code analysis and dynamic application security testing, throughout the software development life cycle. Security Verification: SOPs can guide security verification techniques, such as security code reviews and vulnerability assessments, to validate the effectiveness of implemented security controls. Security Architecture Review: SOPs can facilitate security architecture reviews to evaluate the design and implementation of security controls in software systems. Security Operations Integration: SOPs can support the integration of security considerations into operational processes, such as incident management and secure deployment practices.
4	SAMM Security by Verification	Security Testing: Implementing SOPs for security testing helps address the SAMM domain's activity of conducting various types of security testing to identify vulnerabilities and weaknesses in software applications.

#	Name	Description
		Code Review: SOPs for code review contribute to the SAMM domain's activity of reviewing the source code of software applications to identify security flaws and adherence to secure coding practices. Security Architecture Review: Having SOPs for security architecture review supports the SAMM domain's activity of assessing the security architecture of software applications to ensure proper security controls and mechanisms are in place. Security Requirements Verification: SOPs for verifying security requirements contribute to the SAMM domain's activity of verifying that security requirements are properly defined, documented, and implemented in software applications. Threat Modeling: Implementing SOPs for threat modeling exercises helps address the SAMM domain's activity of conducting threat modeling exercises to identify potential threats and risks to software applications. Secure Deployment Verification: SOPs for verifying secure deployment contribute to the SAMM domain's activity of ensuring that software applications are securely deployed and configured in the production environment.
5	SAMM Security by Operations	Environment Hardening: Implementing SOPs for environment hardening supports the SAMM domain's activity of hardening the software environment by implementing secure configurations, applying patches and updates, and utilizing secure network and infrastructure configurations. Secure Build: SOPs for secure build practices contribute to the SAMM domain's activity of promoting the use of secure build practices, including secure coding standards, secure build tools, and secure software development frameworks. Configuration Management: Having SOPs for effective configuration management addresses the SAMM domain's activity of advocating for effective configuration management practices, such as version control, change management, and documentation of software configurations. Vulnerability Management: SOPs for vulnerability management contribute to the SAMM domain's activity of emphasizing the importance of implementing a robust vulnerability management process, including vulnerability scanning, vulnerability assessment, and penetration testing. Incident Management: Implementing SOPs for incident management supports the SAMM domain's activity of recognizing the need for a well-defined incident management process to handle security incidents effectively. Secure Deployment: SOPs for secure deployment practices contribute to the SAMM domain's activity of promoting secure deployment practices, including secure software distribution, secure installation procedures, and secure configuration of software components. Security Testing: Integrating SOPs for security testing throughout the software development life cycle addresses the SAMM domain's activity of encouraging the integration of security testing activities.

#	Name	Description
		Operational Enablement: Having SOPs for providing operational support and guidance supports the SAMM domain's activity of emphasizing the importance of providing operational support and guidance to ensure the ongoing security and reliability of software systems.

11.4.7.2 Assessment – OWASP Top 10

IT DR and BC SOPs indirectly support the recovery and continuity efforts in the event of a security incident related to the OWASP Top 10; however, they do not directly address the specific vulnerabilities outlined by OWASP.

11.4.8 IT Governance and Compliance SOPs

It's important to note that while the IT Governance and Compliance SOPs indirectly contribute to a comprehensive security posture, they may not specifically address the individual vulnerabilities outlined in the OWASP Top 10. To address the OWASP threats more directly, additional specific security measures, such as secure coding practices, regular vulnerability assessments, and secure configuration management, should be implemented.

11.4.8.1 Assessment – SAMM

#	Name	Description
1	SAMM Security by Governance	Policies and Standards: SOPs can help in developing and implementing policies and standards for software security, including secure coding practices, vulnerability management, incident response, and data protection. Risk Management: SOPs can define risk management processes and provide guidance on conducting risk assessments and implementing risk mitigation strategies. Compliance Management: SOPs can help ensure compliance with regulatory requirements, industry standards, and best practices by establishing compliance assessment procedures and reporting mechanisms. Security Training and Awareness: SOPs can outline training programs and awareness campaigns to educate employees and stakeholders about software security and promote a culture of security. Security Metrics and Reporting: SOPs can define the key security metrics to be measured and establish reporting mechanisms to provide visibility into the status of software security.

#	Name	Description
		Security Roles and Responsibilities: SOPs can clearly define and assign security roles and responsibilities within the organization and ensure individuals have the necessary knowledge and skills to fulfill their roles effectively.
2	SAMM Security by Design	Security Requirements: SOPs can guide the process of defining and incorporating security requirements into the design phase of software development, addressing specific security concerns related to OWASP threats. Secure Architecture: SOPs can promote the use of secure architectural patterns and principles to address OWASP threats by minimizing vulnerabilities and establishing a strong foundation for security controls. Secure Coding Practices: SOPs can provide guidelines for secure coding practices, reducing the likelihood of OWASP threats by preventing common coding mistakes and vulnerabilities. Threat Modeling: SOPs can outline the process of conducting threat modeling exercises during the design phase to identify and mitigate potential security risks related to OWASP threats. Security Testing: SOPs can encourage the inclusion of security testing activities in the design phase, such as static code analysis and security code reviews, to detect and address vulnerabilities related to OWASP threats.
3	SAMM Security by Implementation	Secure Development Training: SOPs can define training programs on secure coding practices and common software vulnerabilities, ensuring developers have the necessary knowledge and skills to build secure software. Secure Architecture: SOPs can guide the implementation of secure architecture practices, including threat modeling, secure design patterns, and secure component selection, to address potential vulnerabilities and risks. Secure Coding Guidelines: SOPs can establish and promote secure coding guidelines and standards, providing developers with recommendations for writing secure code and addressing OWASP threats. Security Requirements: SOPs can ensure the implementation of security requirements by defining and documenting them as part of the software development process. Security Testing Integration: SOPs can facilitate the integration of security testing activities throughout the software development life cycle, helping identify and address security vulnerabilities related to OWASP threats. Security Verification: SOPs can guide the implementation of security verification techniques, such as code reviews and vulnerability assessments, to validate the effectiveness of security controls.
4	SAMM Security by Verification	Security Testing: SOPs can define the process of conducting various types of security testing, including penetration testing, vulnerability scanning, and code review, to identify and address security vulnerabilities. Code Review: SOPs can outline the procedures for reviewing the source code to identify security flaws and ensure adherence to secure coding practices.

#	Name	Description
		Security Architecture Review: SOPs can guide the review of the security architecture of software applications to ensure proper security controls and mechanisms are in place. Security Requirements Verification: SOPs can provide guidance on verifying that security requirements are properly defined, documented, and implemented in software applications. Threat Modeling: SOPs can define the process of conducting threat modeling exercises to identify potential threats and prioritize security threats related to OWASP threats. Secure Deployment Verification: SOPs can outline the procedures for ensuring secure deployment and configuration of software applications.
5	SAMM Security by Operations	Environment Hardening SOPs can contribute to Environment Hardening activities by providing guidelines and procedures for implementing secure configurations, applying patches and updates, and configuring secure network and infrastructure settings. Secure Build SOPs can support Secure Build practices by defining secure coding standards, secure build tools, and secure software development frameworks. Configuration Management SOPs can assist in effective Configuration Management by providing guidelines and processes for version control, change management, and documentation of software configurations. Vulnerability Management SOPs can contribute to Vulnerability Management activities by establishing procedures for vulnerability scanning, assessment, and remediation. Incident Management SOPs can support Incident Management by defining incident detection, response, containment, and recovery procedures. Secure Deployment SOPs can guide the process of secure software deployment, including secure distribution, installation procedures, and configuration. Security Testing SOPs can assist in conducting regular security testing activities such as static code analysis, dynamic application security testing (DAST), and security code reviews. Operational Enablement SOPs can provide guidance and procedures for operational support, including security training, documentation of operational procedures, and incident response capabilities.

11.4.8.2 Assessment – OWASP Top 10

IT Governance and Compliance SOPs focus on ensuring compliance with regulations and internal governance frameworks. They don't directly address the OWASP Top 10 vulnerabilities; however, they contribute to a comprehensive security posture by addressing risk management, compliance audits, policy enforcement, and data privacy.

#	Name	Description
11	Broken Access Control (A01-2021)	It does not address this domain directly.
12	Cryptographic Failures (A02-2021)	It does not address this domain directly.
13	Injection (A03-2021)	It does not address this domain directly.
14	Insecure Design (A04-2021)	It does not address this domain directly.
15	Security Misconfiguration (A05-2021)	It does not address this domain directly.
16	Vulnerable and Outdated Components (A06-2021)	It does not address this domain directly.
17	Identification and Authentication Failures (A07-2021)	It does not address this domain directly.
18	Software and Data Integrity Failures (A08-2021)	It does not address this domain directly.
19	Security Logging and Monitoring Failures (A09-2021)	It does not address this domain directly.
20	Server-Side Request Forgery (A10-2021)	It does not address this domain directly.

12 Appendix

N.A.

--End of the document--