Application Security – Verification Standard (ASVS):

What is the ASVS?

- The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.
- The primary aim of the **OWASP Application Security Verification Standard (ASVS) Project** is to normalize the range in the coverage and level of rigor available in the market when it comes to performing Web application security verification using a commercially-workable open standard.
- The standard provides a basis for testing application technical security controls, as well as any technical security controls in the environment, that are relied on to protect against vulnerabilities such as **Cross-Site Scripting (XSS) and SQL injection.**
- This standard can be used to establish a level of confidence in the security of Web applications. The requirements were developed with the following objectives in mind:
  - **Use as a metric** - Provide application developers and application owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications.
  - **Use as guidance** - Provide guidance to security control developers as to what to build into security controls in order to satisfy application security requirements.
  - **Use during procurement** - Provide a basis for specifying application security verification requirements in contracts.

Understanding Application Security Verification Levels:

- There are three levels of security verification in ASVS designed to enhance security of applications with different security needs. They are as follows:

i) ASVS Level 1 Basic

- ASVS Level 1 is relevant for applications that don't deal in sensitive information and are less susceptible to attacks. Every application must adhere to the basic security standards stated in ASVS Level 1. This level outlines security controls that are designed to safeguard against known vulnerabilities. The security measures listed in this level are pen-testable and integration testable. Level 1 is suited for small and medium sized businesses facing no major security risks.

ii) ASVS Level 2 Standard

- Level 2 guidelines are applicable to applications that conduct business-to-business transactions. Following these guidelines will help app developers secure their applications against illegitimate access, injection flaws, validation and authentication errors. ASVS Level 2 Standard ensures that the measures implemented are in line with the vulnerabilities and threats that pose a risk to the application in focus. ASVS Level 2 is recommended by security experts for safeguarding most applications. Testing at this level requires access to source code, documentation, configuration as well as people involved in the development.

iii) ASVS Level 3 Advanced

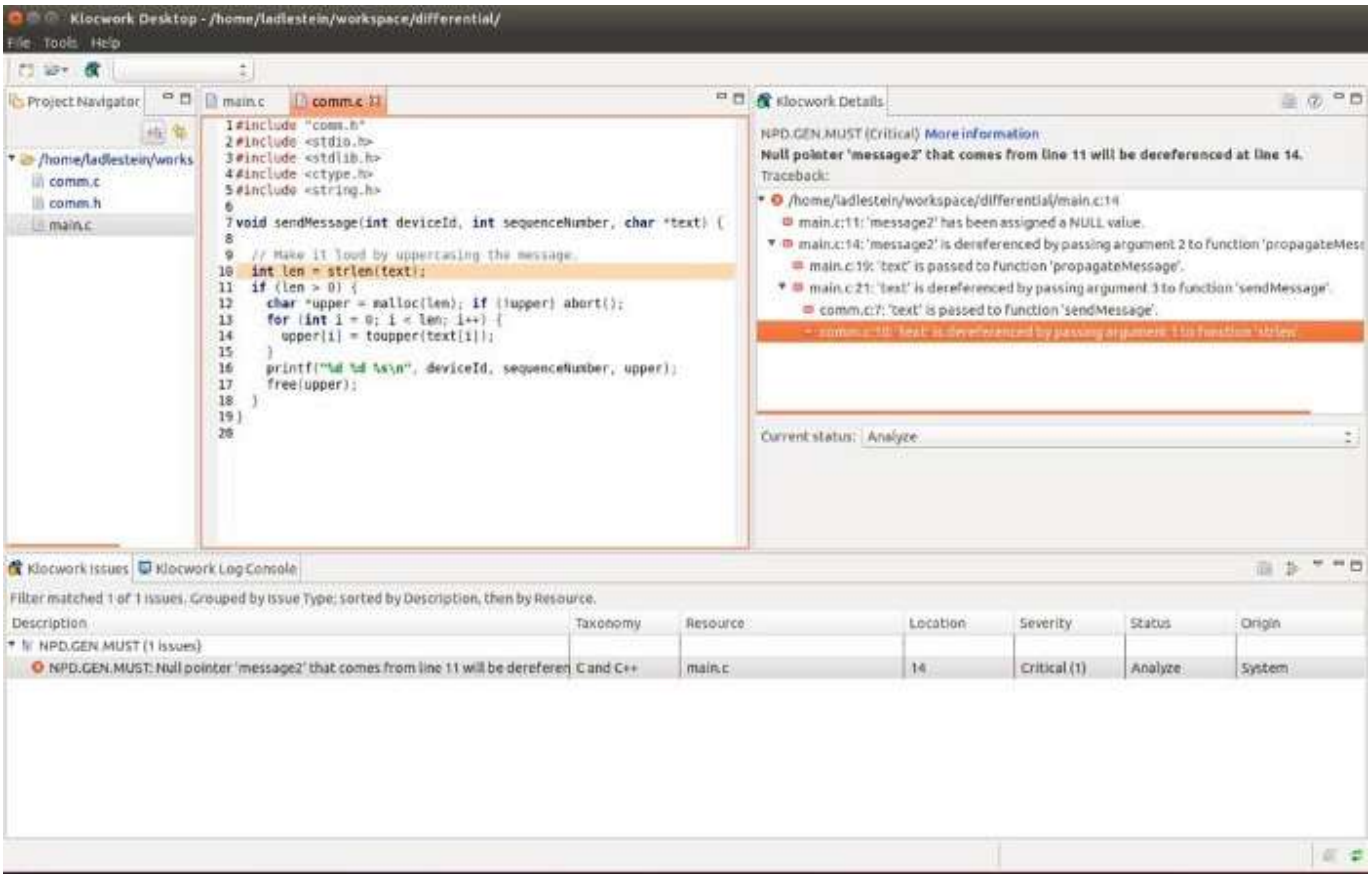
- Applications that deal in highly sensitive information need to adhere to ASVS Level 3 Advanced. Examples of such applications include healthcare, defense, finance, legal document management apps among others. To meet the requirements of level 3, app developers must embed security layers into the application right from the earlier stages. Plus, all the security efforts must be documented and audited.

SAST and DAST:

Static Application Security Testing (SAST) :

White-box testing

- SAST is a form of white-box security testing which has full access to the underlying source code and binary. It will test your program via an inside-out approach. Specialized SAST software such as GitLab, Klockwork or AppThreat will scan your source code automatically either during the coding process, or after you have committed the code to your pipeline.
- For example, for Klockwork's users, once you have established a link between your project and Klockwork Desktop, it allows you to code your program normally using any IDE of your choice as long as it is open in the background. Each time you save the file, Klockwork Destop will update the code automatically and perform a scan on the spot. If it detects any security issues, it will display them on the user interface.



Early Detection

- SAST is typically conducted at the early stage of the system development life cycle, usually during or after the development stage. This allows you to identify any form of security vulnerabilities before going into the testing or quality assurance phase.
- It is a lot easier to fix problems when they are discovered early. Besides, most SAST executions will flag lines of code with the vulnerabilities. This can be extremely useful and serve as pointers to developers when fixing vulnerabilities. It also costs less to maintain and develop the project.

False Positives

- SAST methodology is prone to high number of false positives compared to DAST. As a result, a situation might arise where developers have wasted valuable time and resources fixing imaginary problems in their system. Such a downfall can be costly in the long term if there are hundreds or thousands of false positives.

Language-Dependent

- SAST is language-dependent in terms of analyzing the underlying source code and binary. Most SAST tools specialize only in a few computer languages. You will not be able to find a one-size-fits-all SAST tool for every programming language used in your projects. As a result, scaling and maintaining a project build with different computer languages will be an enormous task.

Does not cover run-time vulnerabilities

- SAST is not capable of detecting any form of run-time vulnerabilities as it only scans the static code and binary. If you have incorrectly configured your system, a SAST tool will not be able to identify run-time issues leading to a false sense of security among the developers.

Dynamic Application Security Testing (DAST)

Black-box testing

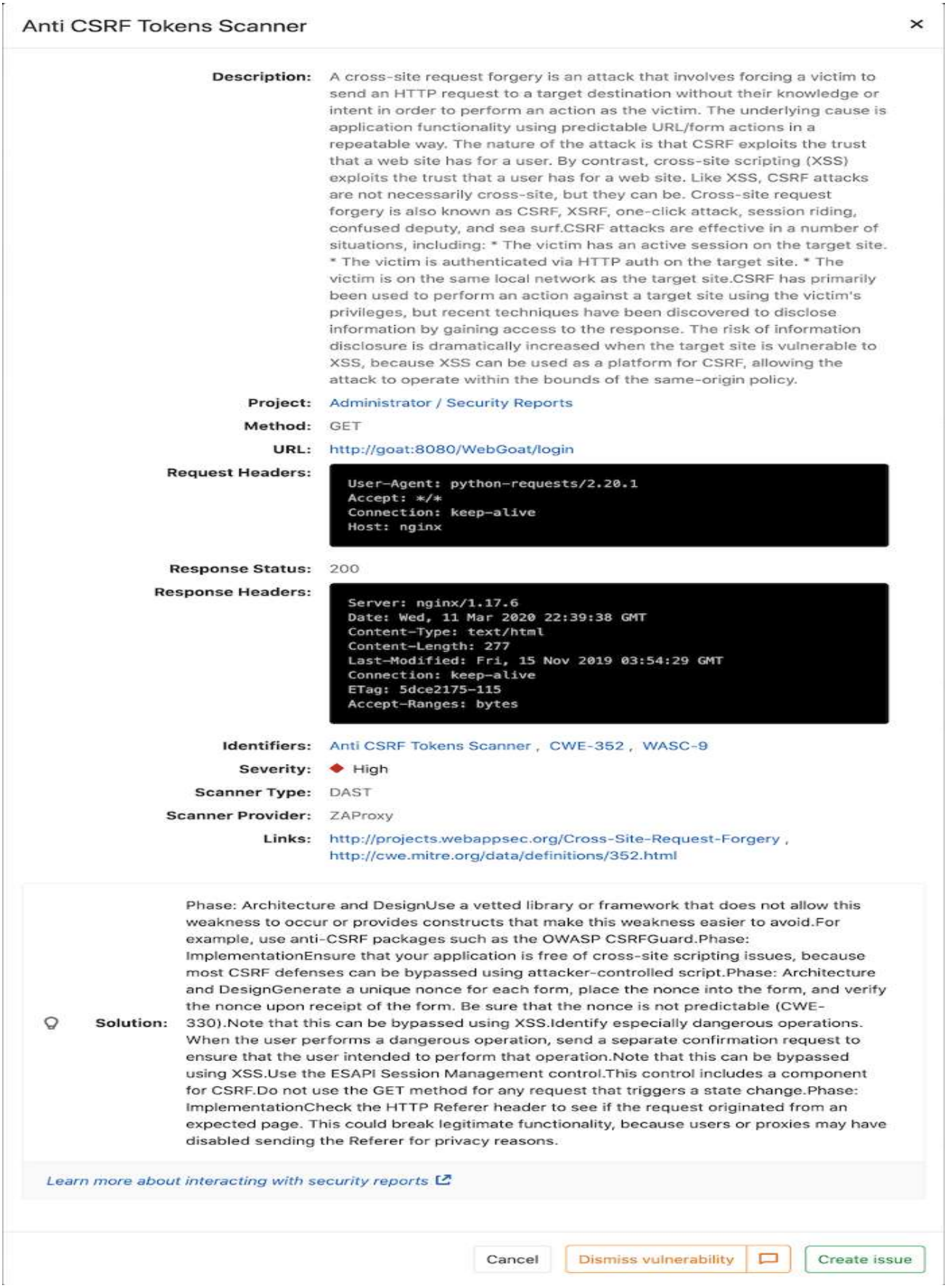
- On the other hand, DAST is termed dynamic because it does not have any access to the underlying static code or binary. Tests are conducted from the outside-in. You can think of it as a hacker trying to test the security vulnerabilities of your system.
- Unlike SAST, it analyzes the running application without any knowledge of the technology used to develop the system. Once it detects any potential vulnerabilities and risks, it logs the issues for the developers.
- The following example illustrates how Gitlab made a comparison on the vulnerabilities between the source and target branches. The information will be displayed on each merge request.

Late Detection

- DAST is usually conducted at the end of the system development life cycle. It typically takes place during the testing phase, right before the user acceptance tests. It serves as the final gateway to secure your application before deploying it out to your users.
- Most of the time, issues that are detected under DAST will not be fixed immediately unless they are deemed critical. This is mainly due to lack of time available before the UAT or deployment phase. So, most of the issues are pushed into the next development cycle.

Language-Independent

- Since tests are conducted directly on running applications, DAST is not tied to any computer languages. It is a lot easier to scale and maintain your test since it is independent of the programming languages used in the development of your system.
- However, most DAST tools only support specific types of applications, like web applications or web services. You might need a few different DAST tools if your project consists of web applications, desktop applications and mobile applications.
- For example, let's have a look at the DAST report produced by Gitlab. It highlights the Anti CSRF Tokens analysis performed on a web application.


Anti CSRF Tokens Scanner
Description:
A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site.CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
Project:
Administrator / Security Reports
Method:
GET
URL:
http://goat:8080/WebGoat/login
Request Headers:
User-Agent: python-requests/2.20.1
Accept: */*
Connection: keep-alive
Host: nginx
Response Status:
200
Response Headers:
Server: nginx/1.17.6
Date: Wed, 11 Mar 2020 22:39:38 GMT
Content-Type: text/html
Content-Length: 277
Last-Modified: Fri, 15 Nov 2019 03:54:29 GMT
Connection: keep-alive
ETag: 5dce2175-115
Accept-Ranges: bytes
Identifiers:
Anti CSRF Tokens Scanner , CWE-352 , WASC-9
Severity:
High
Scanner Type:
DAST
Scanner Provider:
ZAProxy
Links:
http://projects.webappsec.org/Cross-Site-Request-Forgery ,
http://cwe.mitre.org/data/definitions/352.html
Solution:
Phase: Architecture and DesignUse a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.For example, use anti-CSRF packages such as the OWASP CSRFGuard.Phase: ImplementationEnsure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.Phase: Architecture and DesignGenerate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).Note that this can be bypassed using XSS.Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.Note that this can be bypassed using XSS.Use the ESAPI Session Management control.This control includes a component for CSRF.Do not use the GET method for any request that triggers a state change.Phase: ImplementationCheck the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.
Learn more about interacting with security reports
Cancel
Dismiss vulnerability
Create issue

- Behind the hood, it uses an open-source web app scanner called OWASP ZAP (Zed Attack Proxy) for scanning your running application.

Cover run-time vulnerabilities

- One main advantage of DAST over SAST is that it is capable of finding run-time vulnerabilities. This includes configuration issues, authentication issues and system memory issues. You will be able to identify many more issues from the user perspective.

WHAT IS INTERACTIVE APPLICATION SECURITY TESTING (IAST)?

- Application security testing describes the various approaches used by organizations as they attempt to find and eliminate vulnerabilities in their software. Also referred to as AppSec testing and AST, application security testing is the process of testing, analyzing, and reporting on the security level of a software application as it moves through the software development lifecycle (SDLC).

Why is it Important?

- Interactive application security testing (IAST) combines static application security testing (SAST) with dynamic application security testing (DAST) to create a synergistic and self-learning interactive application security testing approach. With IAST, interactive application security testing techniques cover more code, produce better results, and verify a broader range of security rules faster than either SAST or DAST tools working alone.

Continuous vs. a Snapshot in Time

- In Gartner research, 84% of breaches exploit vulnerabilities in the application layer. Because SAST, DAST, and penetration testing only provide a snapshot in time, they can't keep up with today's agile software development lifecycle processes or the ever-evolving threat landscape. This means that development, ops, and security teams are always at least one step behind as they develop, test, and move software into production.

The Power of IAST

- IAST security solutions, on the other hand, deploy agents and sensors that continuously monitor and analyze applications from within as they run. Because they are self-learning, they produce real-time analysis as software is being developed and tested. This makes them ideal for Agile, DevOps, and DevSecOps environments as they enable IT to find and fix security flaws early in the SDLC when they are easiest and cheapest to remediate.
- Imagine if you could identify vulnerabilities at DevOps speed, stopping attacks before they occur, and preventing problems before they can do real damage to your organization. Contrast has broken through legacy barriers with an innovative, automated IAST security solution, Contrast Assess, that infuses software with vulnerability assessment capabilities to identify security flaws automatically and in real time wherever software is running.

Advantages of IAST over SAST and DAST

- When we honestly assess the strengths and weaknesses of SAST vs. DAST vs, IAST, we see that IAST gets far better results across these seven metrics.

➢ **False Positives:** False positives are where ZAP raises alerts for things that are not really vulnerabilities. Representing the single biggest weakness in legacy security tools, false positives occur in over 50% of testing results. This increases the workload on scarce security resources and makes it difficult to identify the most critical flaws. IAST, on the other hand, produces the real-time intelligence and continuous visibility necessary to detect and remediate vulnerabilities with virtually no false positives or false negatives.

➢ **Vulnerability Coverage:** Interactive analysis provides the best of static and dynamic testing. Not only do

interactive testing tools focus on the most common and most risky flaws found in applications, but they also allow for custom rules to personalize the threat coverage for specific enterprises.

- **Code Coverage:** Both static and dynamic testing miss huge portions of most applications. SAST doesn't examine libraries or frameworks, severely limiting vulnerability analysis. DAST can only examine an application's exposed surface. But IAST examines the entire application from the inside – including the libraries and frameworks. So you get far superior coverage over your entire codebase.
- **Scalability:** Static and dynamic tools don't scale well. They typically require experts to set up and run as well as interpret the results. But an application's size and complexity don't affect interactive application security testing, which can handle extremely large applications in stride.
- **Instant Feedback:** Static and dynamic tools get run on a periodic basis, which means the lag time between the mistake and the vulnerability detection could be weeks or even months. IAST provides instant feedback to a developer, within seconds of coding and testing new code. With IAST, developers can be sure they are only checking in "clean" code.
- **No Experts Required:** When you buy something, you just want it to work out of the box. IAST interactive tools eliminate months of configuration, tuning, and customization. As the application is exercised, the application is tested – continuously and automatically.
- **Zero Process Disruption:** Businesses put a premium on time-to-market. Agile and DevOps strategies limit testing time. Because interactive testing operates transparently during normal QA or unit testing, there is no process disruption. IAST integrates smoothly with existing security testing activities.

Why IAST Delivers Better Results

- Here are 8 key reasons why IAST, Contrast Assess, and Contrast OSS deliver superior results:
- These tools have been purpose-built from the ground up to work interactively with developers as they write and test web applications and APIs.
- They fuse together the most effective elements of IAST, SAST, and DAST application security testing approaches with configuration and open-source security analyses, delivering them directly into applications.
- They are a testing solution that can deliver security results as fast as code changes.
- They embed agents to monitor code and report from inside the app.
- Security flaws are automatically identified, both in development and across the SDLC.
- Developers have the tools they need to fix vulnerabilities without security experts.
- They produce real-time results to find and fix security flaws early when they are easiest and cheapest to remediate.
- They can scale across the entire application portfolio.

Key Features

- Contrast Assess and Contrast OSS(Open Source Software) represent a new kind of security designed for the way today's software is created.
- **Extensive vulnerability coverage:** Contrast provides extensive coverage over the most common application security risks, including the OWASP Top Ten.
- **Code-level remediation advice:** Contrast's innovative security trace format pinpoints exactly where a vulnerability appears in the code, and how it works. Contrast "speaks the developer's language," providing remediation guidance that is easy to understand and implement.
- **Third-party code analysis:** Like icebergs, 80% of the code in modern applications is "beneath the surface," lurking in libraries, frameworks, and other components. Applications often have 50 or more of these libraries, comprising millions of lines of potentially vulnerable code.
- **Application inventory:** You can't protect what you can't see. Today's organizations may have hundreds or

thousands of applications, microservices, and APIs – each with multiple instances of different versions installed across development and QA – and they're all constantly changing. Contrast tracks and continuously feeds information about internal and external web services, and their relationships across an application into a unified security inventory and bill of materials that's always up-to-date.

- **Live application architecture:** Contrast automatically generates simple diagrams that illustrate the application's major architectural components. This information helps the developer quickly identify the meaning of a vulnerability and take decisive action.

For all these reasons and more, Contrast Security would like to welcome you to the wonderful world of real-time, automated, self-protecting software.

For a given site (local), conduct a dynamic analysis scan using OWASP ZAP, Check for False positives and create a report.

What is OWASP ZAP?

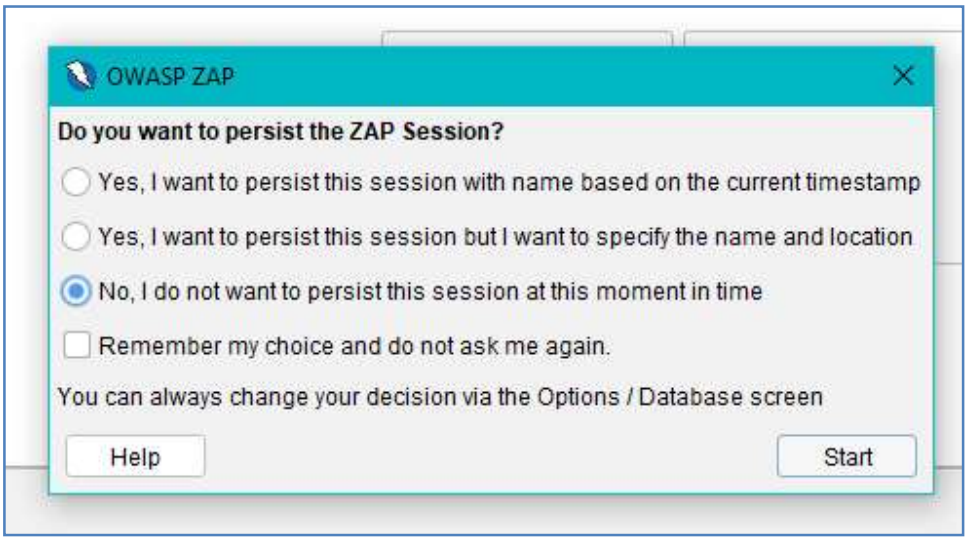
- OWASP (Open Web Application Security Project) is worldwide non-profit organization focused on improving the security of software.
- OWASP ZAP (Zed Attack Proxy) is one of the world's most popular security tool. It's a part of OWASP community that means it's totally free.

Setting up your ZAP Environment

- **JAVA 8+**: In order to install ZAP you need to install JAVA 8+ to your Windows or Linux system. If you use the Mac OS you don't need JAVA as it's already installed. Go to **https://java.com/en/download/** and install it.
- **Installer:** Download ZAP installer according to your OS. **https://github.com/zaproxy/zaproxy/wiki/Downloads**

Starting OWASP ZAP

- After you install the application to the default directory, you can start clicking the OWASP ZAP icon on your Windows desktop. The default install directory;
- **C:\Program Files\OWASP\Zed Attack Proxy\ZAP.exe**
- When you run the app, it asks you whether you want to save the session or not. If you want to reach your website configuration or test results later, you should save the session for later. For now let's keep it default ***"No,I do not want to persist the session"***



What Is the Difference Between Active & Passive Scan?

What is passive scan?

- In terms of penetration test, a passive scan is a harmless test that looks only for the responses and checks them against known vulnerabilities. Passive scan doesn't modify your website data. So it's really safe for the websites that we don't have permission. As you know OWASP number 1 vulnerability in 2018 is still **Injection**. And be aware that you cannot detect even a SQL Injection with passive scan.

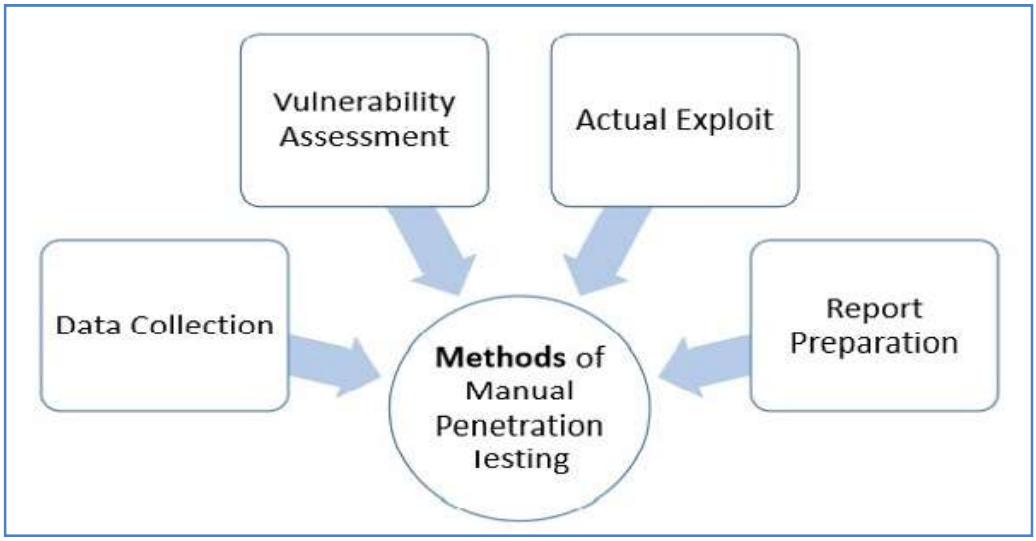
What is active scan?

- Active scan, attacks the website using known techniques to find vulnerabilities. Active scan does modify data and can insert malicious scripts to the website. So when you really test your website against security issues deploy it to a new environment and run the active scan. And only run the active scan for the sites you have permission!

Introduce Manual Security Testing using OWASP Testing Guide. Add Misuse case testing to the framework in addition

What is Manual Penetration Testing?

- Manual penetration testing is the testing that is done by human beings. In such type of testing, vulnerability and risk of a machine is tested by an expert engineer.
- Generally, testing engineers perform the following methods −
  - **Data Collection** − Data collection plays a key role for testing. One can either collect data manually or can use tool services (such as webpage source code analysis technique, etc.) freely available online. These tools help to collect information like table names, DB versions, database, software, hardware, or even about different third party plugins, etc
  - **Vulnerability Assessment** − Once the data is collected, it helps the testers to identify the security weakness and take preventive steps accordingly.
  - **Actual Exploit** − This is a typical method that an expert tester uses to launch an attack on a target system and likewise, reduces the risk of attack.
  - **Report Preparation** − Once the penetration is done, the tester prepares a final report that describes everything about the system. Finally the report is analyzed to take corrective steps to protect the target system.



Types of Manual Penetration Testing

Manual penetration testing is normally categorized in two following ways −

- **Focused Manual Penetration Testing** − It is a much focused method that tests specific vulnerabilities and risks. Automated penetration testing cannot perform this testing; it is done only by human experts who examine specific application vulnerabilities within the given domains.
- **Comprehensive Manual Penetration Testing** − It is through testing of whole systems connected with each other to identify all sorts of risk and vulnerability. However, the function of this testing is more situational, such as investigating whether multiple lower-risk faults can bring more vulnerable attack scenario, etc

Conduct a manual security testing for a local web application or an API using proxy tools like burp suite/paros etc and provide a report. Compare the results of both manual and automated scans.

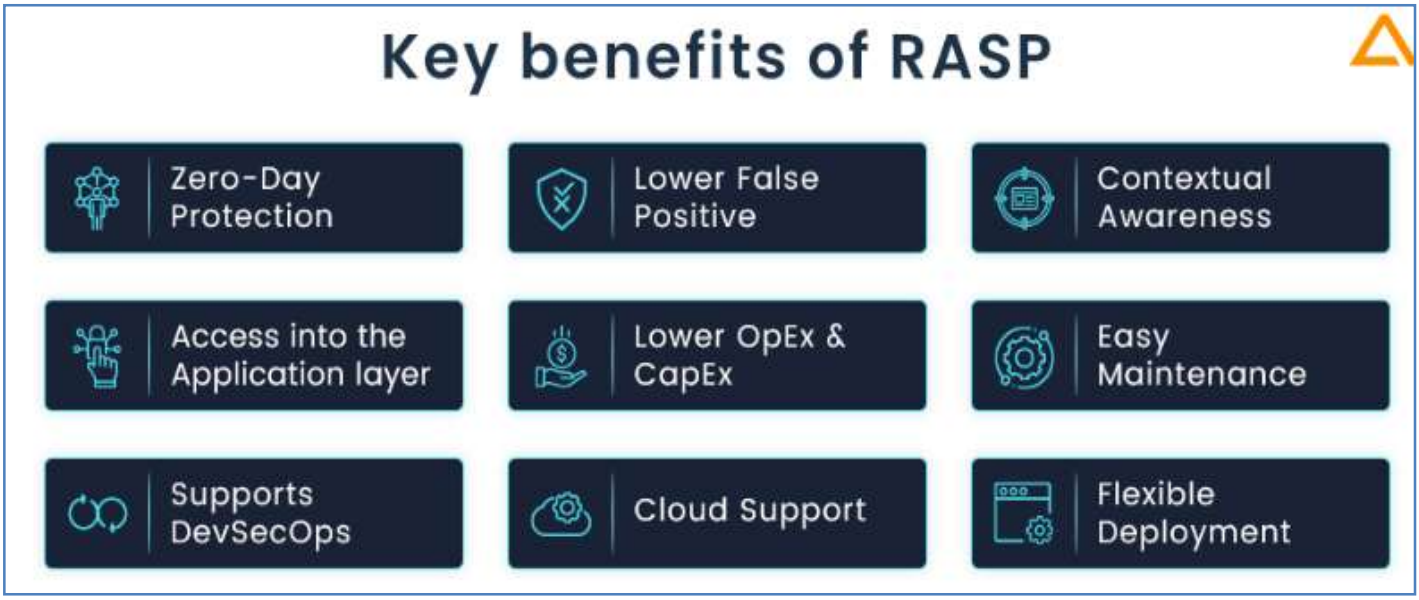
- Burp or Burp Suite is a set of tools used for penetration testing of web applications. It is developed by the company named Portswigger, which is also the alias of its founder Dafydd Stuttard.
- BurpSuite aims to be an all in one set of tools and its capabilities can be enhanced by installing add-ons that are called BApps.
- It is the most popular tool among professional web app security researchers and bug bounty hunters. Its ease of use makes it a more suitable choice over free alternatives like OWASP ZAP.
- Burp Suite is available as a community edition which is free, professional edition that costs **$399/year** and an enterprise edition that costs **$3999/Year**. This article gives a brief introduction to the tools offered by BurpSuite.

Explain Run Time Application Self Protection – Contrast Security or Microfocus Fortify Software can be used as an example.

- Runtime Application Self Protection (RASP) is a security solution designed to provide personalized protection to applications. It takes advantage of insight into an application's internal data and state to enable it to identify threats at runtime that may have otherwise been overlooked by other security solutions.

How RASP Works

- RASP wraps around and protects a particular application, rather than a general network-level or endpoint-level defensive solution. This more targeted deployment location enables RASP to monitor the inputs, outputs, and internal state of the application that it is protecting**. By deploying RASP, developers can identify vulnerabilities within their applications.** Additionally, the RASP solution can block attempts to exploit existing vulnerabilities in deployed applications.
- RASP's focused monitoring makes it capable of detecting a wide range of threats, including zero-day attacks. Since RASP has insight into the internals of an application, it can detect behavioral changes that may have been caused by a novel attack. This enables it to respond to even zero-day attacks based upon how they affect the target application.



- **Contextual Awareness:** When a RASP solution identifies a potential threat, it has additional contextual information about the current state of the application and what data and code is affected. This context can be invaluable for investigating, triaging, and remediating potential vulnerabilities since it indicates where the vulnerability is located in the code and exactly how it can be exploited.
- **Visibility into Application-Layer Attacks:** RASP has deep visibility into the application layer because it is integrated with a particular application. This application-layer visibility, insight, and knowledge can help to detect a wider range of potential attacks and vulnerabilities.
- **Zero-Day Protection:** While RASP can use signatures to identify attacks, it is not limited to signature-based detection. By identifying and responding to anomalous behaviors within the protected application, RASP can detect and block even zero-day attacks.
- **Lower False Positives:** RASP has deep insight into an application's internals, including the ability to see how a potential attack affects the application's execution. This dramatically increases RASP's ability to differentiate true attacks (which have a true negative impact on application performance and security) from false positives (such as SQL injection attempts that are never included in an SQL query). This reduction in false positives decreases load on security teams and enables them to focus on true threats.
- **Lower CapEx and OpEx:** RASP is designed to be easy to deploy yet is able to make a significant difference in an application's vulnerability to attack and rate of false positive alerts. This combination reduces both up-front expenses (CapEx) and the cost of effectively protecting the application (OpEx) compared to manual patching and web application firewalls (WAFs).
- **Easy Maintenance:** RASP works based upon insight into an application, not traffic rules, learning, or blacklists. SOC teams love this reliability and CISOs appreciate the resource savings. Applications become self-protected and remain protected wherever they go.
- **Flexible Deployment:** While RASP is typically based upon HTML standards, it is easy to adapt its API to work with different standards and application architectures. This enables it to protect even non-web applications using standards like XML and RPC.
- **Cloud Support:** RASP is designed to integrate with and be deployed as part of the application that it protects. This enables it to be deployed in any location where the protected applications can run, including in the cloud.
- **DevSecOps Support:** RASP solutions are designed to be integrated into a DevOps continuous integration and deployment (CI/CD) pipeline. This makes RASP easy to deploy and supports DevSecOps operations.

Microfocus Fortify:

- Using Microfocus' Fortify Application Defender, you can analyze and safeguard your applications in real-time against risks and typical cyberattacks. It protects operational applications against zero-day attacks by distinguishing between valid requests and harmful threats in .NET and Java applications. Its end-to-end application protection services encompass every stage of the programming process.
- In complement to line-of-code data, Fortify provides logs transparency. For privacy transparency and regulation, it also enables you to submit attack and record data to a log administrator or SIEM without needing to update the code base.

  Key Features:
  - You will be provided with a multi-layered protection system.
  - It has 32 different protection rule categories to safeguard you from security breaches.
  - With adaptable and fast installation, you may get immediate protection.
  - You can control your security from a single, easy-to-use administration interface.

  Cost:

  You can request a quote through their website.

Contrast Security:

- Contrast Security Is an output App and API Security That Helps Developer Players Utilize Security Vulnerabilities Additional Source by Blocking Threats and Reducing False Positives. Without Needing New Versions, Contrast Security Can Prevent the log4j Issue in Your Production Environments Right Now. Contrast Protect Also Protected the Apps from the Fundamental Issue. This Indicates Contrast Was Defending You From Log Injections.

  Key Features:
  - Unparalleled Accuracy
  - Extensive Forensics
  - Features Software Development and Maintenance Apps

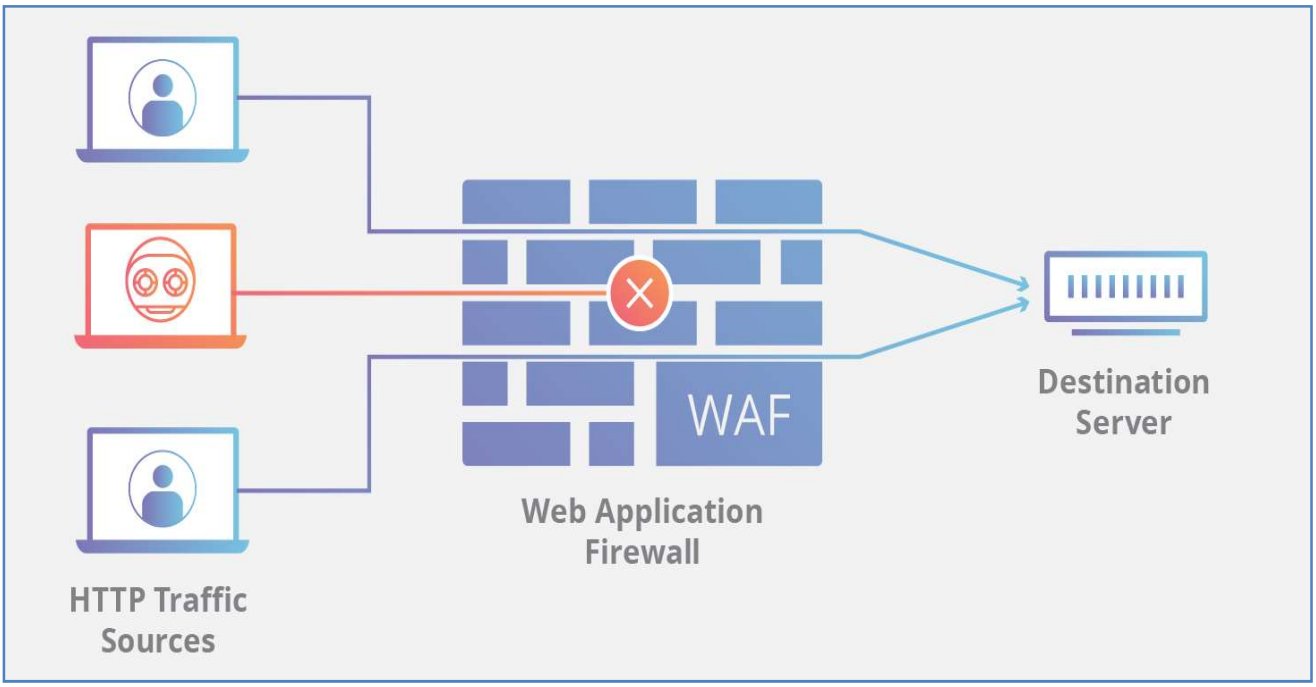
  Cost:

  The pricing starts from $2,800 per year.

Define Web Application Firewall. Demonstrate using a tool.

- A WAF or web application firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others.
- A WAF is a protocol layer 7 defense (in the OSI model), and is not designed to defend against all types of attacks. This method of attack mitigation is usually part of a suite of tools which together create a holistic defense against a range of attack vectors.
- By deploying a WAF in front of a web application, a shield is placed between the web application and the Internet. While a proxy server protects a client machine's identity by using an intermediary, a WAF is a type of reverse-proxy, protecting the server from exposure by having clients pass through the WAF before reaching the server.

- A WAF operates through a set of rules often called policies. These policies aim to protect against vulnerabilities in the application by filtering out malicious traffic. The value of a WAF comes in part from the speed and ease with which policy modification can be implemented, allowing for faster response to varying attack vectors; during a DDoS attack, rate limiting can be quickly implemented by modifying WAF policies.



Types of Web Application Firewalls

There are three primary ways to implement a WAF:

- **Network-based WAF**—usually hardware-based, it is installed locally to minimize latency. However, this is the most expensive type of WAF and necessitates storing and maintaining physical equipment.
- **Host-based WAF**—can be fully integrated into the software of an application. This option is cheaper than network-based WAFs and is more customizable, but it consumes extensive local server resources, is complex to implement, and can be expensive to maintain. The machine used to run a host-based WAF often needs to be hardened and customized, which can take time and be costly.
- **Cloud-based WAF**—an affordable, easily implemented solution, which typically does not require an upfront investment, with users paying a monthly or annual security-as-a-service subscription. A cloud-based WAF can be regularly updated at no extra cost, and without any effort on the part of the user. However, since you rely on a third party to manage your WAF, it is important to ensure cloud-based WAFs have sufficient customization options to match your organization's business rules.

Demonstrate using a tool: AWS WAF

Best For Scalable use for businesses of all sizes as long as they are AWS clients.
Price:

Resource Type	Price
Web ACL	$5.00 per month (prorated hourly)
Rule	$1.00 per month (prorated hourly)
Request	$0.60 per 1 million requests

- The Amazon AWS web application firewall is a robust website security solution. However, AWS WAF is only available to customers who just use the company's Web Services.
- The solution is just an add-on to an existing subscription to cloud services such as the Amazon content delivery network and Application Load Balancer.

Features:

- Agile protection against web attacks
- Improved web traffic visibility
- Ease of deployment and maintenance
- Cost-effective web application protection
- Security integrated with how you develop applications.

Verdict: AWS Amazon Web App Firewall is a highly robust and scalable solution facilitated with countless useful security features that ensures that your website remains safe against different types of cyberattacks.

Elaborate on Standard Operating Procedure for Operations, Secure Provisioning, deployment and decommissioning:

- Imagine you are starting a new job at a car dealership and a customer walks in asking for your latest hatchback. How would you cater to the customer?
- Most likely, your manager must have given you proper training along with a set of documents to study the basics of how to deal with customers. These sets of documents and training guides are often known as **standard operating procedures or SOPs.**
- A standard operating procedure (SOP) document guides new as well as current employees on how to carry out routine tasks and maintain consistency and quality throughout business operations.
- Since SOPs are crucial documents, we decided to uncover everything there's to know about standard operating procedures and provide a tool to create SOPs with ease.

What are Standard Operating Procedures (SOP)? (Definition)

- **A standard operating procedure (SOP) is a step-by-step instructions guide to help an employee in performing specific operations smoothly. The main objective of SOP is to ensure uniform and quality output, while simultaneously reducing miscommunication and ambiguity.**
- SOPs are detail-oriented documents and provide step-by-step instructions as to how employees within an organization must go about completing certain tasks and processes.

Types of Standard Operating Procedures (SOP)

1. Checklists

- A checklist or the to-do list is one of the simplest methods of writing a standard operating procedures (SOP) document. A checklist can be created on an online note-taking app like Bit or can be printed out and handed over to employees.

2. Step-by-Step List
   - Similar to checklists, a step-by-step bullet list works in the same way where you describe a procedure in relevant, easy-to-follow steps.
3. Hierarchical Lists
   - If your procedures are more complex and need additional info, you can create hierarchical checklists or bullet lists. If you are unable to explain a task in a single step and at the same time, don't want to make the SOP lengthy, adding hierarchical steps can be beneficial.

Process Flowchart

- Flowcharts are a wonderful way to represent how a process works visually and help give better context around the workflow.
- A flowchart also shows how one step is related to another, helping employees conceptualize the whole concept and have a better understanding of the work they are doing.

Why do you Need Standard Operating Procedures (SOP)?

- Some of you may be wondering- *If we are already training our employees to do the tasks they are hired to do, why take on this extra work of documenting operating procedures*? We understand your dilemma, which is why we are going to look at some of the reasons *why every* business should create standard operating procedures (SOP) no matter what…
  1. Time-saving
  2. Ensure the safety of employees
  3. Ensures compliance standards are met
  4. Improved communication
  5. Enhanced accountability
  6. Provides consistency
  7. Maintains Organizational Knowledge
  8. Provides a guiding hand
  9. Onboarding and training

Steps for Writing a Standard Operating Procedure (SOP)

- Now you know what a standard operating procedure is and why your organization needs to create one, it's time to actually get down to business and create one. Standard operating procedures require a ton of effort and planning before you can even begin to document your procedures.
- Here are the key steps you need to follow to create a robust standard operating procedure document:

Step1: Generate a list of your business processes

- The first thing you need to do in order to create an SOP is to find out which tasks, processes, or workflows, you need an SOP for. Conduct a survey or ask your employees to fill out a form defining what tasks they do on a regular basis.
- This will form the basis of your list for the standard operating procedure (SOP) document. Once you have gathered a list, you can review it with other managers and look for any repetitions.

Step 2: Start with why

- Once you have your list ready, it's time to note down your objectives. Having a clear answer to why you are creating the SOP document should be your number one priority. Asking yourself questions like "how will this document help the employees?" or "how will the SOP impact our bottom line?" are great starting points.
- For a more granular approach, identify the pain points or challenges your employees face in their day to day and create your SOP around it. This gives you a solid "why" to go through all that hard work of creating an SOP and also improves employee's buy-in in the whole process.

Step 3: Choose a format

- Chances are that your organization already has some SOP documents written for past procedures. You can refer to those documents as templates and guide your current SOP.
- If not, then refer to our "types of SOP documents" section above and decide whether you want to write a list of steps, create a checklist, create workflow diagrams, or a mixture of everything!

Step 4: Identify your audience

- Knowing your audience is key in creating an awesome SOP document. Ask yourself the following questions in order to get an idea about your audience:
  - Are they new employees?
  - What's the size of the audience?
  - What prior knowledge do they have?
  - Does an SOP already exist?
- The more information you have on your audience, the better you can understand their points of view and create an SOP that will be relevant to them.

Step 5: Collaborate with employees

- Standard operating procedures (SOP) are written with the end-user, i.e, the employees in mind. Having employees collaborate with you in this process is a no-brainer.
- You cannot really understand their pain points and challenges unless you talk to your employees and ask for their honest feedback and suggestions. We recommend using collaboration software like bit.ai to bring your entire team inside a common document and collaborate effectively.

Step 6: Get down to writing

- Once you have spoken to your employees and have enough data points to start, immediately move to your document editor and start adding your notes. Once done creating the document, you can go through the document with your employees and management and ask for their feedback and input.
- This is also a great time to specify who would be responsible for updating and maintaining the standard operating procedures and when will you be conducting a periodic review to gauge engagement.

Step 7: Make it interactive

- While SOP documents are text-heavy and boring, they don't have to be. Add screenshots, screen recordings, images, flow charts, videos- anything that's relevant to the step being talked about.
- Media like these can help make your SOP's pop while providing a visual aid to otherwise bland steps. Making your standard operating procedures interactive will boost your engagement levels as employees are surely going to find them more useful and even entertaining!

Step 8: Distribution

- After you are done creating the SOPs, you've come to the most essential part of the process: distributing them to your employees. It's crucial to find a place to store all your standard operating procedures (SOP) and other training material in one place for employees to access as and when they like.
- This is why we recommend using Bit to store all company documents in one place and store company assets like videos, images, PDFs, and more in Bit's content library. You can quickly create a workspace in Bit, invite your employees, and share SOPs and more in a robust and safe environment.

Step 9: Make them "living documents"

- While many organizations view creating SOPs as a one-time process, that's hardly the case. As processes and workflows are often changing and ever-evolving in the hopes of making them more efficient, standard operating procedures quickly become outdated.
- This is why SOPs should be converted to living documents that get reviewed periodically (ideally after every six months) so that they don't get out of sync with the process or workflow they are describing.

Standard Operating Procedures (SOP): Best Practices

Here are some tips to keep in mind while writing your SOP document:

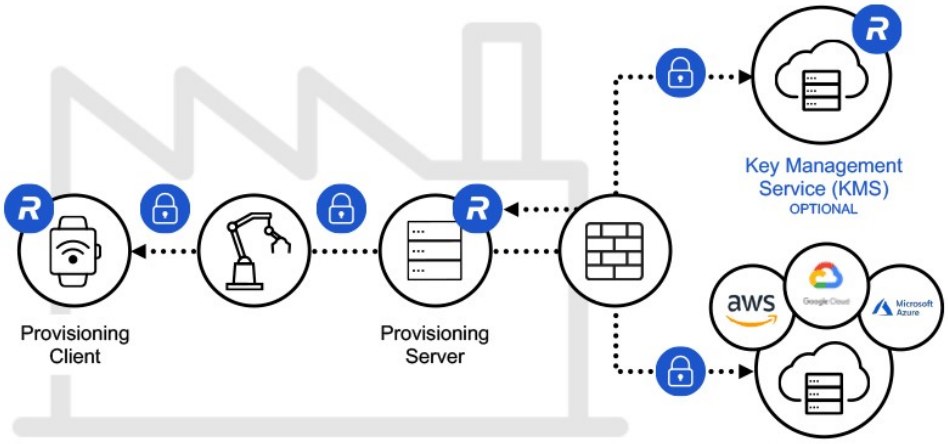
- **Be clear and concise:** Since standard operating procedures are text-heavy, it helps if they are written in simple language for your audience to go through it quickly. Avoid technical jargon, wordiness, and ambiguity, and remember to keep it simple.
- **Make it scannable:** Make your SOP's scannable so that employees can quickly go through them and find what they are looking for. Don't go on and on in a paragraph and make sure the length of every paragraph doesn't exceed 3 lines.
- **Take input:** Take input from your employees and understand their pain points before you begin writing your SOP. What are the areas they need help with? What processes are complex and require a lot of time? Focus on challenges and write an SOP that helps them overcome those challenges.
- **Choose your tool wisely:** While there are many editors on the market, using a collaboration platform like Bit makes sure you have a single place to write, store, share, and track all your SOPs and workplace documents easily.

Secure Device Provisioning.

- Protect your device secrets throughout the manufacturing process. We revolutionize secure provisioning by offering the simplest secure way of injecting secret identity data to a device.

How Secure Device Provisioning works

- Secure Device Provisioning (formerly from Inside Secure) ensures that secrets are not exposed or manipulated when provisioned at manufacturing time, and the innovative code protection and whitebox technology ensures that these secrets remain protected for the rest of the device's lifetime. The solution doesn't require any hardware security resource available on the device, however if available, it can be easily adapted to variety of common hardware and software architectures.
- With the Secure Device Provisioning solution, device makers can also remotely monitor the manufacturing process and even control it, without setting foot at the manufacturing site. It simplifies provisioning and makes it affordable to all device makers. The client environment can be further secured through the application of secure boot, code protection and whitebox technologies.



Features & Benefits

- Cost effective to meet your budgetary needs (light configuration, low maintenance)
- Flexible provisioning to meet your device specific needs (single or two-stage provisioning)
- Over 425 millions devices provisioned successfully
- Over 75 licensed customers, from chipset makers to service providers
- The world's first independent provisioning service

Deployment:

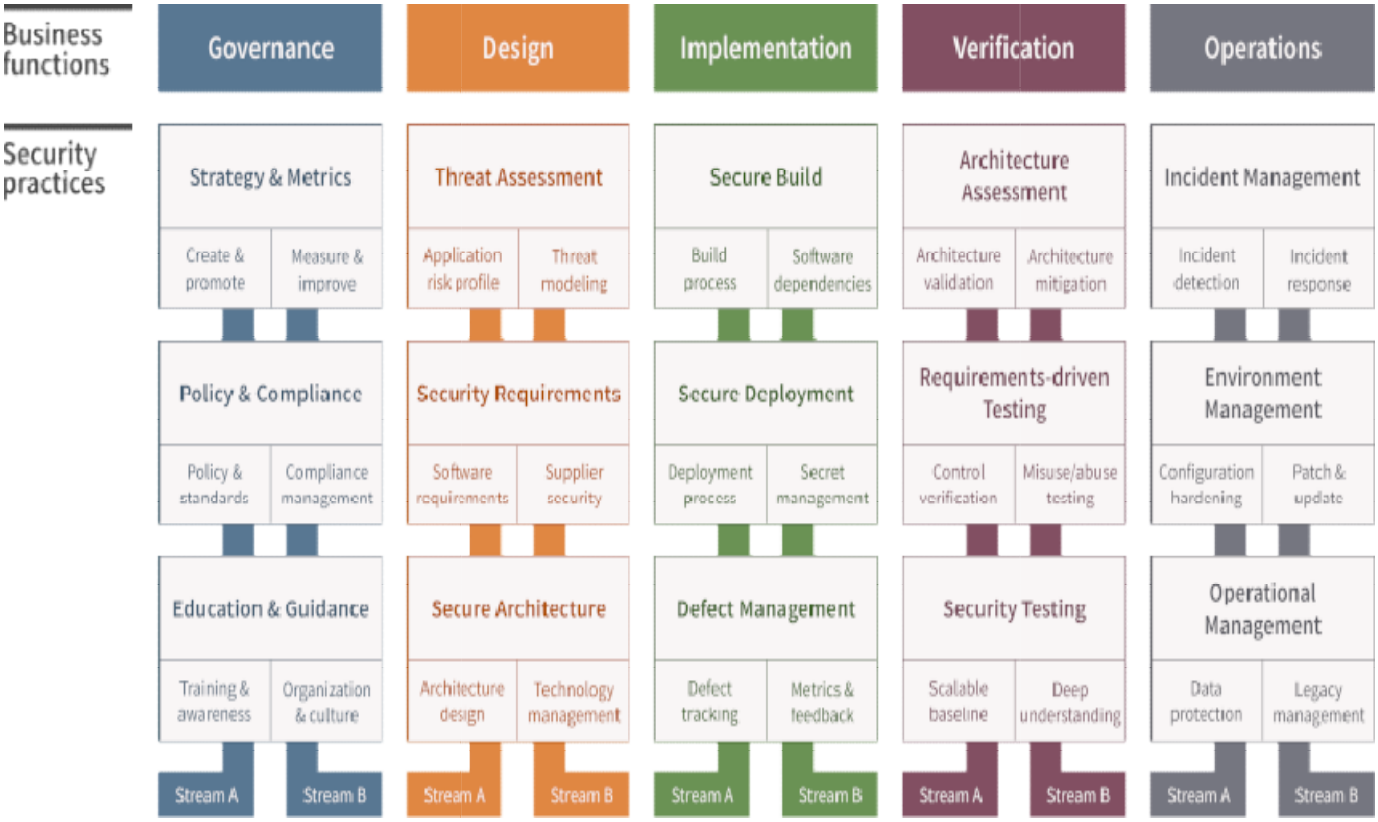
- An automation process workflow must be deployed to send it to the server. From server, the deployed process is assigned to a robot for execution.
- The Deployment tab allows you to deploy the published processes in Automation Studio. It also enables you to decommission the current version, restore the previous version and add a process as a processbot. Published as well as deployed processes are available in this tab. You can view details related to the process such as name of the process, its type, assigned profile, and version and so on.
- The version of the process displayed in the Deployment tab is the version that gets created while publishing the process. If you edit a deployed process, save and publish the deployed process again. The version of the saved and published process post deployment gets incremental by 1. Every version of the saved process that you publish is available for deployment.

Introduce OWASP SAMM – to attain software assurance maturity:

- The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization. SAMM helps you:

  - Evaluate an organization's existing software security practices
  - Build a balanced software security assurance program in well-defined iterations
  - Demonstrate concrete improvements to a security assurance program
  - Define and measure security-related activities throughout an organization

- SAMM provides a model for organisations to assess their current level software assurance readiness and capabilities. It is not expected nor is it required that an organisation should achieve the maximum maturity level in each category. Instead organisations are encouraged to determine the target maturity levels for each Security Practice that best fits the organisation's goals, and adapt the SAMM templates accordingly.
- Structurally SAMM defines five critical business functions:
  - Governance
  - Design
  - Implementation
  - Verification
  - Operations
- Each business function has three security practices, each of which has two streams. A stream is an objective by itself, which can be described through three levels of maturity:
  - Level 0: an implicit starting point with an unfulfilled Security Practice
  - Level 1: an initial understanding and ad hoc implementation of the Security Practice
  - Level 2: a structured realisation with increased efficiency and effectiveness of the Security Practice
  - Level 3: a comprehensive mastery of the Security Practice at the scale that comes with an optimised solution.
- For each level, SAMM defines the objective, a set of activities and describes expected results.
- How is SAMM utilised in practice? With just four (more or less) easy steps:
  1. Assess the organisation's current software security posture
  2. Define the organisation's targets for each Security Practice
  3. Define the implementation roadmap to achieve the set targets
  4. Make it so! Do the necessary work to implement it all.



Define Metrics, Type of Metrics (Operations, Efficiency, Quality etc). Example Application Security Metrics from OWASP.

Metrics:

- Use metrics to measure the progress of the security uplift program. According to OWASP SAMM metrics "evaluate the effectiveness and efficiency of the application security program"[1]. Metrics help to improve the security maturity by measuring the progress on the defined maturity goals, and allow for adjustments to the security program to ensure the maturity goals are met.
- Metrics are used to tell a story to justify an action, such as a security budget, and argue for change[2]. Metrics are a useful tool to present to management to highlight where there may be gaps in the organisation's security posture that require additional resources to address, or to show that resources spent on security are having the desired effect of reducing risk to the organisation.
- To start, examine business processes to see where they can be quantified and measured. Metrics are measured over time, such as quarterly, to show a trend. It is important to know what the desired direction for the metric to move is, up or down, and what activity may have caused significant movement.[3]
- Take note of which Software Development Lifecycle (SDLC) phase has identified the most security issues. It is more expensive to fix security issues later in the development cycle.[4] Aim to identify issues sooner in the SDLC in accordance with maturity goals.

Example metrics

Design - Threat Modelling

- number of threat models or threat modelling activities conducted
- number of findings from threat models

Develop - Code Reviews

- number of code reviews conducted
- number of findings from code reviews

Deploy - Security testing

- number of findings from penetration testing (internal vs vendor)
- number of findings from vulnerability scanning
- number of findings remediated
- time taken to remediate a vulnerability, does it meet an SLA (An organisation may require vulnerabilities to be remediated in a certain time frame in accordance with the vulnerabilities risk rating)

Security team collaboration

- number of secure by default modules created
- number of secure architecture designs created and provided to developers

Security champions

- number of security champions onboarded

Training activities

- number of training activities introduced to developers
- number of developers onboarded to training activities
- number of developers engaged in particular training activities

Example graph using metrics

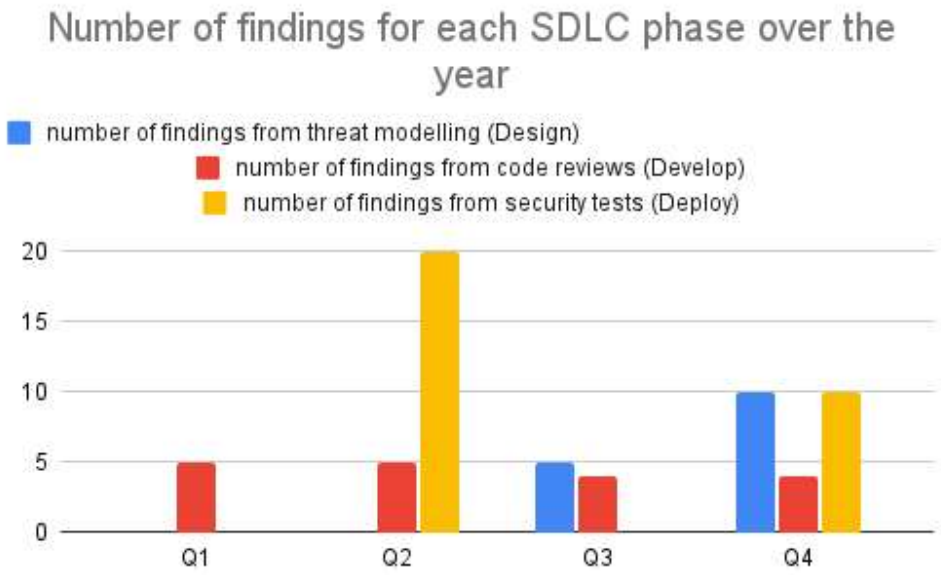


Figure : Example Metrics Graph

- The metric for the number of findings from threat modelling in the Design phase increases in Q3 upon the threat modelling activity implementation for the organisation. This metric increase is expected and desired. The metric further increases in Q4 as the threat modelling activity is scaled out by Security Champions, an expected and desired increase.
- The metric for the number of findings from security tests increases in Q2 as a result of a penetration test. This metric increase is expected. Upon a further penetration test in Q4, the metric has decreased from it's earlier record. This metric change is desired, as it represents a cost saving. The cost of addressing security findings increases at later stages of the development lifecycle.