**Secure Software Development Plan for Greentech Engineering and Services (GES)**

Nicholas Hoats

South University

Course ID: ITS4221

Michael Goyden

5/26/2024

**Secure Software Development Plan for Greentech Engineering and Services (GES)**

As Senior IT Security Consultant at Salus Cybersecurity Services, LLC, I developed a comprehensive safe software development plan for Greentech Engineering and Services (GES). This Plan will incorporate security throughout such software development lifecycle (SDLC), which begins with the initiation of conceptual requirements and specifications, transitions through design and coding phases, and ends on final testing cycles to move smoothly into operational use. Embedding security considerations at the outset of the development process can help ensure that GES yields resilient, secure, and compliant green technology applications and web-based services. The methodologies and best practices needed to realize these objectives are discussed throughout this document, offering specific recommendations for every stage of development. This plan is designed to be proactive about addressing these potential areas of vulnerability in our security posture and also keep GES compliant with industry standards for our crucial role at the nexus of technology and media while promoting a culture of always working toward getting better. This type of all-encompassing strategy doesn’t only mitigate risks presented by PCI-DSS compliance but also strengthen GES’s cybersecurity framework, securing its digital resources and keeping clients happy.

In the first phase of software development, we want to find out and document each security consideration that will be crucial during the application development for GES. All of this occurs in the background, proactively mitigating risks and increasing your security posture from the beginning.

**1. Worksheet for Conceptual-Stage Activities**

The objective of this worksheet is to facilitate the early identification and documentation of security considerations essential to GES’s application development process. By addressing these areas proactively, we can mitigate risks and enhance the security posture from the outset.

**Key Sections to Include in the Worksheet:**

* **Sensitive Information Identification:** Consider the types of sensitive data involved and outline strategies for protecting this data and mitigating exposure risks. This approach aligns with the best practices of categorizing and securing sensitive information as recommended by industry standards (learn.microsoft, 2023).
* ​**Information Flow Mapping:** Implementing a framework for visualizing data flow within the application can help identify potential vulnerabilities. This strategy is supported by secure development methodologies that emphasize understanding and securing the flow of data within software systems​ (learn.microsoft, 2023).
* **Data Access Requirements:**
  + Criteria for data access control specifying who can access data, conditions for access, and access times. Effective data access management practices include the use of role-based access control and least privilege principles (frontegg, 2024) (NIST, 2016).
* **Infrastructure Services Usage:**
  + Catalog of external and internal services the application will utilize, with an emphasis on security implications.
* **Application and Service Dependencies:**
  + Inventory of all critical dependencies with an evaluation of their security postures to ensure secure integration.

**Key Application Security Considerations for the Conceptual Stage:**

* **Privacy by Design:**
  + Incorporate privacy controls from the design phase to ensure ongoing compliance with data protection laws (learn.microsoft, 2023).
* **Threat Modeling:**
  + Early identification and mitigation of potential security threats through threat modeling (learn.microsoft, 2023).
* **Security Requirements as Functional Requirements:**
  + Elevate security requirements to ensure they receive attention equivalent to functional requirements (learn.microsoft, 2023).

**2. Guidelines for Requirements and Specifications Stage**

The guidelines for the requirements and specifications stage are crafted to ensure that GES’s applications address both functional and security needs in a detailed and precise manner. These requirements form the foundation of a rigorous test plan.

**Key Criteria to Include in Requirements and Specifications:**

* **User Roles and Access Control:**
  + Clearly define user roles and specify access rights and restrictions. This practice is fundamental in secure software development to ensure appropriate access control mechanisms are in place (frontegg, 2024) (OWASP, n.d.)​.
* **Use Cases and Security Scenarios:**
  + Document typical user interactions and potential security threats with planned countermeasures. Secure software development frameworks emphasize the importance of understanding and planning for security from the outset, which includes robust threat modeling and defining security scenarios (positive technologies, 2020).
* **Interfaces to Internal and External Systems:**
  + Detailed security measures for all data exchanges with internal and external systems.
* **Concurrency Requirements:**
  + Strategies for maintaining data integrity and security in multi-access environments.
* **Performance, Scalability, and Reliability:**
  + Performance benchmarks that account for scalability and reliability under various operational loads.
* **Security Features:**
  + Essential security features such as encryption, activity logging, and continuous monitoring should be mandatory.
* **Compliance and Standards:**
  + Ensure alignment with industry standards and regulatory requirements throughout the design and implementation process. It's essential to adopt frameworks like SSDF that offer guidance on integrating security practices throughout the software development lifecycle (NIST, 2021)​.

**Further Considerations for Testing and Verification:**

* **Testability of Each Requirement:**
  + Each requirement should be clear, measurable, and testable to facilitate thorough testing.
* **Security Testing Plans:**
  + Comprehensive plans to test all aspects of security, ensuring robustness and compliance.
* **Verification Protocols:**
  + Protocols for verifying compliance with all functional and security specifications.

With the foundation established during the conceptual requirements and specifications stage, the next step is to translate these guidelines into actionable security measures during the application software design and coding phase.

In the design and coding phase, we focus on enhancing cybersecurity measures to address significant risks such as input attacks, object reuse vulnerabilities, and backdoor vulnerabilities.

Input attacks, including SQL Injection, Cross-Site Scripting (XSS), and Command Injection, present significant risks to application security (ElevatEd, 2024). For SQL Injection, best practices include using parameterized queries or prepared statements to separate SQL commands from user input, thereby reducing the risk of malicious code execution. Examples include using PreparedStatement in Java and parameterized queries in .NET and other programming environments (OWASP Cheat Sheet Series, n.d.) (learn.microsoft, 2021). This approach ensures that SQL queries are not only safe from injection but also make the application relatively database independent​.

For preventing XSS, it is crucial to validate and sanitize user input, ensuring that only expected data is processed and potentially dangerous characters are removed or encoded (S., n.d.). Implementing a robust Content Security Policy (CSP) is also effective in specifying valid sources of executable scripts and preventing unauthorized script execution (developer.mozilla, 2024). Secure handling of cookies, such as marking them secure and HTTP-only, can prevent them from being accessed by client-side scripts, reducing the risk of session hijacking​ (developer.mozilla, 2024).

To mitigate object reuse vulnerabilities, essential practices include proper memory management techniques such as secure allocation and deallocation of memory and clearing memory buffers after use. Regular scrubbing of databases and file systems to remove residual data can prevent this data from being accessed by unauthorized processes​ (Kime, 2023)​.

To avoid backdoor vulnerabilities, strict version control, limiting developer access to the production codebase, and conducting extensive code audits to discover illegal or suspicious code paths are all important. Monitoring and managing access during development and deployment are also strongly reliant on the use of secure authentication and logging technologies​ (Kime, 2023)​.

Building on the secure design and coding practices, it is crucial to implement rigorous testing procedures to verify and validate the security and functionality of the software applications developed for GES.

The application testing phase involves verifying and validating the security and functionality of the software applications built for GES. This phase ensures complete traceability by linking each test case directly to its respective requirement.

It is important to verify that an application complies with the specified functional and non-functional criteria once it has been constructed. Because they are verifiable and testable, these criteria serve as the foundation for rigorous testing that verifies the software's compliance (Dixit, 2023).

The testing plan will ensure complete traceability by linking each test case directly to its respective requirement. This approach not only guarantees coverage but also facilitates the identification of any gaps in the testing process. The plan will include:

* A traceability matrix that maps requirements to test scenarios and cases (visure, n.d.).
* Detailed documentation of the test coverage to ensure all aspects of the design and requirements are thoroughly tested (visure, n.d.).
* The steps for updating the traceability matrix when new or modified requirements appear (Sealights, n.d.).

Reviews are an integral step in confirming and validating that the software meets business needs and operates as designed. These reviews are most effective when conducted by the product development team since they can provide insight into the functionality and design of the application (tuleap, n.d.).

Guidelines for conducting reviews will include:

* Regularly scheduled review meetings at the end of each development phase.
* Inclusion of cross-functional teams in the review process to provide diverse perspectives (Dixit, 2023).
* A standard checklist of items to be reviewed, ensuring consistency across all reviews (Saeed, 2024).
* Procedures for documenting review outcomes and tracking the resolution of any issues identified.
* Ensuring that all feedback from reviews is integrated into the development process in a timely manner (Sealights, n.d.).

The objective of testing is to demonstrate compliance with the requirements and identify any deviations from expected results.

The plan will detail various types of testing activities that need to be implemented, including:

* Unit Testing: To verify each component or module on its own for correct behavior (testbytes, 2024).
* Integration Testing: To ensure that integrated components function together as expected (testbytes, 2024).
* System Testing: To validate the complete and integrated software product (testbytes, 2024).
* Security Testing: Concentrating on finding software vulnerabilities in particular (Sealights, n.d.).
* User Acceptance Testing (UAT): Is done with real users to make sure the system satisfies their needs and standards (testbytes, 2024).
* Regression Testing: Is done following software modifications to make that previously designed and tested software continues to function as intended (testbytes, 2024).

For each type of testing, the plan will specify the timing, responsible parties, and methodologies to be used.

Transitioning from the testing phase, we integrate the principles of DevSecOps to ensure continuous security throughout the software development lifecycle, fostering a culture of shared responsibility for security.

DevSecOps is among the most successful methods I've come across. By adding a critical layer of security to the DevOps concepts, this approach makes sure that security is integrated into the development process as a basic component rather than being treated as an afterthought.

The primary concept of DevSecOps is to remove the security team and give this responsibility to everyone else. In other words, instead of a group of outside specialists taking care of all security issues towards the end of a project, they are integrated from the start. There are a few advantages to this approach; the most important one is that it makes projects significantly more secure. Since we detect and fix issues early in the software development process, we lower the possibility of security violations (IBM, n.d.).

Also, it helps achieve shorter time-to-market for software products. Thanks to continuous integration and delivery, it is possible to make the development process more straightforward. I will be able to release them faster and at the pace of today’s market, where time often means everything (microsoft, n.d.).

Cost efficiency is another strong point. It is much more expensive to correct security problems after a product has been released than if any issues are detected at the development stage (IBM, n.d.). It is high time than ridding of the vulnerabilities after deployment. As such, the balance between the costs is unequal. Solving this problem within development costs nothing in comparison to the post-deployment version (elastic, n.d.).

Improved compliance is also a significant advantage. Automated verification of compliance assures that all security policies and regulations will be observed during the software development process. Non-compliance risk decreases, and the amount of manual work maintaining compliance reduces (IBM, n.d.).

I appreciate the great extent to which DevSecOps fosters collaboration and communication. It removes the silos and allows our development, security, and operations teams to pool their expertise to combat threats to our system (elastic, n.d.).

In conclusion, continuous monitoring and feedback mechanisms are beneficial because they allow us to be proactive in identifying and containing security threats. discover and respond to security events as they occur.subnetting and identifying possible attacks and system performance.exposure availability (IBM, n.d.) (microsoft, n.d.).

Development in an Agile environment is done in friendly cycles known as sprints. Through sprints, DevSecOps can be input ensures security is continuously practiced both during the development phase and after deployment. Here is how it will work for Greentech Engineering and Services :

Systems Engineers:

They also have a significant contribution to the role of securing the infrastructure. They accomplish this by automating the provisioning and management of infrastructure through Infrastructure as Code practice, which enforces security configuration persistency. In this way, they should add security testing tools in the CI/CD pipeline that include static, dynamic, and dependency scanning (elastic, n.d.).

Software Developers:

Developers have the most direct control over securing the application. Secure coding practices help prevent such vulnerabilities like SQL injection, cross-site scripting, or buffer overflows and underflow. Peer reviews and pair programming ensure security by catching issues that typically would be overlooked (IBM, n.d.).

System Testers:

Testers guarantee the application’s security via intensive testing. Security testing should be executed in each sprint, and a combination of automated tools and manual probing should be used to eradicate the detour. Penetration testing should be conducted regularly since it imitates legitimate hacking and checks the application’s deal with various scams (microsoft, n.d.).

Securing a strong application is an ongoing project that requires the inclusion of security practices at every step of the Agile development cycle. When planning, we need to identify minimum security requirements and acceptance criteria in sprint planning. In development, adding security checks to automated CI/CD pipeline ensures that vulnerabilities are encountered in the early days. After the application has been created and its first version is ready, the third phase, testing, involves running automated security scans in pipeline and manual assessment. Deploying the application, the company should use secure deployment practices and monitor for security incidents in real-time. Finally, training should always include feedback from security testing and incident response activities to proactively improve security (elastic, n.d.).

To ensure that the security measures implemented are effective and continuously improved, we employ an Application Security Assessment (ASA) blueprint to guide and evaluate our efforts.

**Asa Blueprint Checklist**

Preparation Phase:

* Define assessment objectives and scope
* Identify stakeholders and assemble the assessment team
* Gather relevant documentation and access credentials

Assessment Phase:

* Conduct a threat model and risk assessment
* Perform static and dynamic analysis
* Execute penetration testing and vulnerability scanning
* Review third-party components and dependencies

Analysis Phase:

* Analyze findings and categorize vulnerabilities by severity
* Prioritize remediation efforts based on risk impact
* Prepare a detailed assessment report

Remediation Phase:

* Develop a remediation plan with clear action items
* Implement fixes and retest to verify resolution
* Update documentation and ensure compliance with security policies

Entry and Exit Criteria:

Entry Criteria:

* Completion of initial project setup and requirement definitions
* Availability of necessary resources and documentation
* Agreement on assessment scope and objectives

Exit Criteria:

* All identified vulnerabilities are addressed and verified
* Final assessment report is reviewed and approved by stakeholders
* Continuous monitoring and improvement plan is in place

This document covers the GES secure software development plan, which incorporates security into every stage of the software development process. If GES follows the above guidelines and implements the actions outlined in this plan, they can be confident that they are producing software that is secure, dependable, and compliant to attain an overall greater level of the GES security posture while minimizing risks. As a result, GES can protect their digital assets while remaining focused on providing quality services to their clients in accordance with market demand, while also assuring regulatory compliance by incorporating security from the early design phases through deployment and maintenance.

# References

developer.mozilla. (2024, 3 6). *Content Security Policy (CSP)*. Retrieved from developer.mozilla: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

developer.mozilla. (2024, 5 2). *Using HTTP cookies*. Retrieved from developer.mozilla: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

Dixit, V. (2023, July 10). *What Are Requirements Traceability Matrix: With Examples*. Retrieved from Lambdatest: https://www.lambdatest.com/learning-hub/requirements-traceability-matrix

elastic. (n.d.). *What is DevSecOps?* Retrieved from elastic: https://www.elastic.co/what-is/devsecops

ElevatEd. (2024, April 15). *Avoiding SQL Injection and XSS Attacks: Tips for Secure Web Development*. Retrieved from 98th Percentile: https://www.98thpercentile.com/blog/secure-web-development-sql-injection-and-xss-attacks

frontegg. (2024, 2 7). *Access Control in Security: Methods and Best Practices*. Retrieved from frontegg: https://frontegg.com/guides/access-control-in-security

IBM. (n.d.). *What is DevSecOps?* Retrieved from IBM: https://www.ibm.com/topics/devsecops

Kime, C. (2023, 5 16). *How to Prevent SQL Injection: 5 Key Prevention Methods*. Retrieved from eSecurity Planet: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks/

learn.microsoft. (2021, 9 15). *.NET Framework Data Providers*. Retrieved from learn.microsoft: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/data-providers

learn.microsoft. (2023, September 26). *Secure development best practices on Azure*. Retrieved from learn.microsoft: https://learn.microsoft.com/en-us/azure/security/develop/secure-dev-overview

microsoft. (n.d.). *What is DevSecOps?* Retrieved from microsoft: https://www.microsoft.com/en-us/security/business/security-101/what-is-devsecops

NIST. (2016, 9 2). *Access Control Policy and Implementation Guides*. Retrieved from NIST: https://csrc.nist.gov/Projects/Access-Control-Policy-and-Implementation-Guides

NIST. (2021, 2 25). *Secure Software Development Framework*. Retrieved from CSRC.NIST: https://csrc.nist.gov/Projects/ssdf

OWASP. (n.d.). *Access Control*. Retrieved from OWASP: https://owasp.org/www-community/Access\_Control

OWASP Cheat Sheet Series. (n.d.). *SQL Injection Prevention Cheat Sheet*. Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html#sql-injection-prevention-cheat-sheet

positive technologies. (2020, 2 25). *How to approach secure software development*. Retrieved from positive technologies: https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-approach-secure-software-development/

S., K. (n.d.). *Cross Site Scripting (XSS)*. Retrieved from OWASP: https://owasp.org/www-community/attacks/xss/

Saeed, M. (2024, April 3). *Project Cohesion: Importance of Traceability in Software Testing*. Retrieved from kualitee: https://www.kualitee.com/traceability/importance-traceability-software-testing/

Sealights. (n.d.). *Software Traceability: Keeping Track of Dev and Test Productivity when WFH.* Retrieved from Sealights: https://www.sealights.io/software-quality/software-traceability-keeping-track-of-dev-and-test-productivity/

testbytes. (2024, 2 17). *What is Software Testing Traceability Matrix, Its Types & Significance?* Retrieved from testbytes: https://www.testbytes.net/blog/software-testing-traceability-matrix/

tuleap. (n.d.). *How traceability hits compliance and quality software development*. Retrieved from tuleap: https://www.tuleap.org/software-quality/how-traceability-hits-compliance-and-quality-software-development

visure. (n.d.). *Traceability Matrix Requirements for Software: Ensuring Compliance and Quality Assurance*. Retrieved from visure: https://visuresolutions.com/blog/software/traceability-matrix/