Cody VanGosen

Southern New Hampshire University

CS-405 Secure Coding

Professor Brian Enochson

February 23, 2025

**Project Two: Security Policy Presentation**

<https://youtu.be/7ZCLXVEKf9U>

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome to the Green Pace Security Policy Presentation. My name is Cody VanGosen, and I will be presenting our security policies designed to protect Green Pace's development process and systems. In this presentation, we will discuss security principles, coding standards, encryption policies, automation, and risk assessments. |
| **2** | The purpose of this policy is to establish secure coding guidelines for Green Pace developers. It aims to protect our systems using a defense-in-depth strategy and reduce risks from cyber threats like SQL injection, buffer overflows, and unauthorized access. Additionally, this policy ensures compliance with NIST CSF and ISO/IEC 27001 standards while promoting automation and real-time security enforcement |
| **3** | A defense-in-depth strategy is essential for cybersecurity. It integrates multiple layers of protection, including firewalls, encryption, access controls, and automated security testing. This approach ensures that even if one layer is compromised, other safeguards remain in place to protect sensitive data and systems. Green Pace uses a combination of static and dynamic security analysis tools to detect vulnerabilities. Static tools like SonarQube and Clang Static Analyzer identify security flaws in the code before execution. Dynamic testing tools like AFL and Burp Suite expose vulnerabilities during runtime. Automated security testing is integrated within our CI/CD pipeline using Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST). |
| **4** | Validate Input Data – Prevents injection attacks by sanitizing input.  Heed Compiler Warnings – Addresses potential vulnerabilities detected during compilation.  Architect and Design for Security – Ensures security is embedded from the start.  Keep It Simple – Avoids unnecessary complexity that can introduce vulnerabilities.  Default Deny – Restricts access unless explicitly permitted.  Adhere to Least Privilege – Grants minimal access to users and processes.  Sanitize Data Sent to Other Systems – Prevents data leaks and unauthorized manipulation.  Practice Defense-in-Depth – Layers multiple security mechanisms.  Use Effective Quality Assurance Techniques – Implements rigorous security testing.  Adopt a Secure Coding Standard – Ensures best practices for security are followed." |
| **5** | [Insert text.] |
| **6** | Encryption is essential for protecting sensitive data at all stages:   * **Encryption in Flight** – TLS 1.3 secures transmitted data to prevent eavesdropping. * **Encryption at Rest** – AES-256 encrypts stored data, restricting unauthorized access. * **Encryption in Use** – Secure enclave technologies and homomorphic encryption protect data in memory. * Include some slide content as well. |
| **7** | The **Authentication, Authorization, and Accounting (AAA)** framework ensures controlled access:   * **Authentication**: Requires multi-factor authentication (MFA) for user verification. * **Authorization**: Uses role-based access control (RBAC) to limit access. * **Accounting**: Tracks all login attempts and system modifications via audit logs.   Include some slide content as well. |
| **8-13** | Unit tests are critical in maintaining security, ensuring that:   * **SQL Injection is Prevented** – Malicious input does not alter database queries. * **Valid Queries Execute Correctly** – Proper queries return expected results. * **Input Length is Properly Validated** – Protects against buffer overflows. * **Malformed Input is Rejected** – Incorrect SQL syntax does not execute.   Additionally read off some main ideas from each of the slide content. |
| **14** | Automation is integrated into Green Pace’s DevSecOps pipeline to detect and remediate security flaws in real time. Key security tools include:   * **Static Analysis** – SonarQube, CodeQL * **Dynamic Analysis** – OWASP ZAP, Burp Suite * **Software Composition Analysis (SCA)** – Snyk, Black Duck * **Container Security** – Aqua Security, Trivy By automating security scans within CI/CD workflows, vulnerabilities are detected before code is deployed. * Include some slide content as well to further explain the picture. |
| **15** | Review the DevSecOps Pipeline from the slide before. Examine static & dynamic security testing tools. Explain integration into the stages of the DevSecOps Workflow. Additionally, review other main ideas from slide content. |
| **16** | Green Pace prioritizes security risks based on severity, likelihood, and remediation costs. Critical risks are addressed first to mitigate potential data breaches and system vulnerabilities. Security logs, anomaly detection, and automated threat intelligence are used to evaluate ongoing risks |
| **17** | Key recommendations to strengthen Green Pace’s security posture include:   * **Implementing Intrusion Detection Systems (IDS)** and Security Information and Event Management (SIEM) solutions such as Splunk and IBM QRadar. * **Conducting mandatory security training** through OWASP Secure Coding Academy. * **Adopting Software Composition Analysis (SCA)** tools like Snyk and Black Duck to assess supply chain security.   Additional slide content as well. |
| **18** | Review prior slide information included within this slide as well. Summarize key points related to ZTA, enhancing secure coding standards, automated security integration, application throughout comprehensive supply chain, & regular/routine security training as this is a new delivery and requirement. |
| **19** | Touch upon the importance of NIST & OWASP as primary sources of information. Additionally, cover the Netflix source as it was a good example in a prior side of applicable real-world use cases and implementation/software solutions. |