SNHU CS-405 Project Two: Security Policy Presentation

Tanner Holbrook

Southern New Hampshire University

https://www.youtube.com/watch?v=OPZZxH2LDc8

December 7, 2023

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome to the green pace security policy presentation! My name is Tanner Holbrook, and I have been asked to present implementation guidelines. |
| **2** | Welcome to the defense in-depth overview, highlighting the layered security strategy to prevent security threats. We start with a strong perimeter defense that filters incoming threats. Moving inwards on the diagram, we have user authentication, endpoint security, and data encryption to add critical layers of protection. In addition, application security will measure and continuously monitor our defenses. Lastly, in case of an incident, our response plan will ensure that quick action is taken, leading to a speedy recovery. This multi-layered approach will strengthen our resilience against cyber threats. |
| **3** | The threat matrix shown below summarizes key security risks. The table categorizes internal and external threats based on the likelihood of occurring and their priority level. Each cell will break down the individual threat risk, which will be expanded upon later in this presentation. Automated tools can help identify vulnerabilities by systematically scanning the codebase and highlighting issues like possible SQL injection, buffer overflow, and other potential security flaws. By integrating automation into the development pipeline, these tools can catch and address problems early in development, ensuring a proactive defense and saving money in the long run. |
| **4** | This slide presents the ten core development principles and how they align with specific coding standards to ensure a cohesive and secure approach. For instance, the principle of "Validate Input Data " corresponds to the "Preventing SQL Injection" coding standard because SQL injection can lead to risks involving data vulnerabilities. However, this can be prevented with input sanitization and validation, ensuring data is not malicious or incorrect. Implementing these principles and coding standards will ensure we remain compliant and secure across all our projects, not just one application. |
| **5** | This slide shows an overview of the coding standards. They are efficiently managed by evaluating their severity, exploitation likelihood, and remediation cost. We can ensure practical remediation efforts by prioritizing the implementation of likely threats. In addition, the standards are categorized by priority level, allowing us to focus first on resources that need immediate attention while maintaining a systematic approach. This strategy ensures a proactive and streamlined response to any potential vulnerabilities. |
| **6** | This slide shows our encryption strategies for data at rest, in flight, and in use. Starting with encryption in rest, this defense guarantees the security of stored data, ensuring data stays confidential and protected during extended storage periods. Next, at-flight encryption changes our focus to securing data during transmission and preventing unauthorized access or use from unwanted parties. In addition, encryption in use will safeguard actively processed data to ensure we remain confidential across the data lifecycle. Implementing these strategies together helps create a layered defense to collectively form a resilient and comprehensive shield to secure data at different pipeline stages. |
| **7** | Welcome to the triple-A slide, where we explore the foundational principles of authentication, authorization, and accounting. Authentication will be a gateway by verifying users and system identities while enforcing specific access controls to prevent unauthorized entry. In addition, authorization will specify these access levels through a role-based permission system, supporting the principle of least privilege. Lastly, the accounting part will track user and system activities, ensuring accountability, early detection, and compliance with regulatory requirements. These frameworks form a secure network to fortify defenses against unauthorized access, increasing our security and response capability. |
| **8** | This slide shows how unit testing can be used to detect or find vulnerabilities, specifically when it comes to adding a vector to a collection. For instance, ensuring that adding a vector to a collection functions as intended is fundamental to ensuring security. This will ensure the accuracy and integrity of the data within the system, which is necessary for increasing security. Also, unit testing will help verify that adding the vector is done securely, mitigating the risk of injection attacks. This further highlights the importance of input validation and secure coding principles. |
| **9** | This slide again shows how unit testing can be used to find vulnerabilities, specifically when resizing a collection. For instance, verifying that a collection can be appropriately resized is important regarding resource and efficient memory management. This is because validating that the memory can be resized and freed up will prevent any potential vulnerabilities relating to resource exhaustion. For instance, if an attacker can manipulate the system to exhaust resources, it can lead to denial-of-service vulnerabilities, and unit tests can help prevent these risks. |
| **10** | This unit test shows how an index out-of-bounds error or vulnerability can be tested. Unit tests guarding against out-of-bounds index calls are crucial for preventing buffer overflows. Proper index boundary validation is essential when it comes to mitigating this risk. In addition, ensuring that index calls are within bounds is critical for maintaining data integrity and confidentiality. Unintended out-of-bounds access can expose sensitive data or give hackers a vulnerability they can exploit. |
| **11** | This slide shows how unit testing for length errors is crucial in any project. For instance, detecting a length error early in the project will reduce the likelihood of introducing vulnerabilities later in the project, like buffer overflow. In addition, validating that collections are not reserved too large will contribute to ensuring the program is stable and reliable, eliminating any unwanted crashes or bugs. |
| **12** | In the assess and plan phase, security automation will happen by integrating code analysis tools directly into the IDE to detect and eliminate vulnerabilities. The design and build phase will utilize dynamic analysis tools to identify and address runtime errors or bugs. Next, the verification and test phase will incorporate security checks to prevent the unwanted advancement of any insecure code, leading to a more secure program. After this, the transition and health check phase will have automated security testing to ensure the deployed code adheres to security standards. The monitor and detect phase will have self-protection tools that continuously scan the application, promoting our ability to detect security threats much sooner. Our automated response mechanism will activate in the response phase, addressing security incidents promptly and smoothly. Finally, in the last phase, ongoing security checks and automated testing to keep the program updated will contribute to the overall stability and security of the system. |
| **13** | Embracing a streamlined DevSecOps approach, our tools slide highlights the integration of numerous tools useful during the software development lifecycle. Essential tools such as Jora, CppCheck, SonarQube, Jenkins, Fortify, Terraform, Vault, and Splunk enhance automation, reducing vulnerabilities and ensuring industry compliance. Jira will help with planning, while Jenkins will ensure the building and testing phases function correctly. CppCheck and SonarQube can be used for dynamic code analysis. Whereas Fortify will reinforce our defense against potential vulnerabilities, tools like Terraform, Vault, and Splunk will add additional layers to infrastructure, management, and log security. Using these tools in our toolbox will significantly mitigate any vulnerabilities and reduce the possibility of an attack, allowing us to increase our security and streamline the development process dramatically. |
| **14** | This risk management slide will explore the risk of waiting and the benefits of acting early. Waiting does pose some significant risks, like the potential for expensive and time-consuming updates, failure to protect sensitive information, and hurting the company's reputation and public image. On the flip side, early actions present a bunch of advantages. Acting early will enable us to proactively mitigate or possibly prevent any potential damage, add layers of defense to reduce overall risk, decrease liability, and enhance our reputation and public image. By acting early, we position ourselves to find and eliminate potential problems before they can negatively exploit the program. |
| **15** | This slide will showcase additional recommendations if we want to take security further. First, using tools like Veracode during the development pipeline will significantly mitigate risks of potential security breaches by detecting vulnerabilities early. This proactive approach will ensure our codebase follows secure coding standards, providing a solid defense against threats. After this, we can improve our incident response plan by learning from real-world examples, like the recent Experian data breach. Leveraging these experiences can improve our incident response plan, enhancing its overall effectiveness and ensuring we're well-prepared to respond to any potential problems. These measures will contribute to a well-rounded security strategy, protecting our users and their data. |
| **16** | The ISO/IEC 27001 standard should be adopted to address any future problems or concerns relating to security or sensitive information management. Embrace the zero-trust security model by enabling multi-factor authentication for enhanced access controls. In addition, refining our incident response based on real-world incidents will improve adaptability and minimize the damage done in the case of a breach. Lastly, continuous employee education and training will keep them informed on the latest security threats, ensuring we stay vigilant, informed, and up-to-date with current security standards. These practices will ensure we safeguard sensitive information to the best of our abilities. |