# CS 405 Project Two Script Template

Jaden Williams

SNHU – CS405

6/22/2025

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | My name is Jaden Williams and I am the developer behind the Green Pace security policy. The policy is in place to help secure our data and the software we create. |
| **2** | The security policy ensures that our software design processes and build tools implement a Defense in Depth strategy, providing multiple layers of security to protect our assets. |
| **3** | The threat matrix table was created by analyzing the selected code stack for common coding errors that have historically led to known security vulnerabilities. It outlines potential issues within the stack and assesses the likelihood of a security breach. These vulnerabilities were identified using examples from the Software Engineering Institute wiki at Carnegie Mellon University. Additionally, potential security flaws can be detected during the development and build stages of the CI/CD pipeline through automated security scanning tools. |
| **4** | Ten key principles of software engineering have been chosen to integrate security into every phase of the SDLC, fostering a DevSecOps culture at Green Pace. Notably, the principle of Defense in Depth has been included in this framework. By blending modern standards with foundational, time-tested security practices, we can design and architect systems with security in mind, ensuring a secure environment. |
| **5** | For the Green Pace tech stack, we’ve analyzed the following coding standards to establish a secure development environment. The table outlines each rule, the likelihood of associated security vulnerabilities, and the severity of their potential business impact. The priority column ranks the rules by importance, with higher values indicating greater priority for addressing specific vulnerabilities. |
| **6** | Green Pace’s encryption policies apply to all stages of data handling. Data at rest, such as when stored or idle, must be encrypted. Data in transit, like during transmission or travel, should be protected using HTTPS and SSL/TLS protocols. Additionally, sensitive data—such as credit card numbers and passwords—must remain encrypted while in use, ensuring that it is never left unsecured. |
| **7** | Triple-A policies are designed to track system access and define user permissions. Authentication verifies a user's identity, ensuring they are who they claim to be. Authorization determines the actions a user is permitted to perform through a role-based access system. Accounting monitors system activity, including which resources were accessed, by whom, what commands were executed, and what data was transmitted. |
| **8** | The implementation of unit testing is key to ensuring you have a secure and reliable program. Unit testing involves testing individual components or functions of a software application to ensure they work as intended in isolation. |
| **9** | This unit test ensures that the clear function clears the data set from the collection. This removes the old data and confirms that the size of the collection is now zero. If clear did not work, this could lead to data leakage from protected data remaining in memory – which could lead to it being accessed by threat actors. |
| **10** | This unit test is a negative test to ensure an exception is thrown for an out-of-range index. By intentionally attempting to access an index that is not inbound, we can see how the code reacts, and if it reacts properly. If the code can write to out-of-bounds memory allocations, this could lead to memory being overwritten, or crashes due to the overload. A malicious actor could use this to employ a DoS attack to gain access to the system. |
| **11** | The Automation Summary illustrates the full flow of the DevSecOps pipeline, highlighting where security is integrated at each stage of the process. |
| **12** | DevSecOps integrates security throughout the entire DevOps pipeline, ensuring that security is part of every stage—from planning and coding to testing, deployment, and monitoring. Key tools in this process include CPPCheck, which performs static code analysis to catch potential vulnerabilities early. For front-end compilation, Babel ensures compatibility across environments. And for automated testing, OWASP ZAP helps identify security issues in web applications before deployment. These tools work together to make security an integral part of the development process, not an afterthought. |
| **13** | Today, organizations face security gaps, slow release cycles, and compliance risks. The solution is simple—integrate security at every stage with DevSecOps and automate testing to address these challenges. By acting now, you’ll benefit from faster, more secure releases while ensuring compliance. However, be mindful of the initial setup cost and the time required for teams to adapt. If you delay, you risk increased vulnerabilities, compliance issues, and higher remediation costs down the line. Next steps? Train your teams on DevSecOps, automate security and testing processes, and integrate CI/CD pipelines for continuous feedback and improvement |
| **14** | To stay ahead of evolving threats, it's crucial to update your security policy frequently. Malicious actors are constantly innovating their attack methods, and your policy must adapt to prevent vulnerabilities. Consider bringing in an outside company for annual reviews—a fresh set of eyes, like a White Hat hacking agency, can significantly improve your policy and uncover hidden weaknesses. Finally, implement the policy early and often. The sooner every team member adopts it, the quicker your program becomes and stays secure. |
| **15** | Proactive security is key—embed security early in the development cycle by shifting left. Use continuous monitoring with automated tools to detect and respond to threats in real time. As threats evolve, it's essential to adapt and evolve your security measures to stay ahead. Collaboration across teams—developers, security, and operations—is critical for success. The long-term benefits include reduced risk exposure, faster, more secure software delivery, and better compliance with regulatory standards. |
| **16** | A big thank you to these references for aiding my creation of this presentation. Thank you all for watching. |