# CS 405 Project Two Script Template

Complete this template by replacing the bracketed text with the relevant information.

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Good morning, I'm Jeffrey Carlson. Today, I'm here to present Green Pace's Security Policy Guide Template. |
| **2** | Secure on-premises servers with restricted access, enforce privacy agreements for Supplier cloud services, implement deny-by-default measures and defense layers in the absence of a private cloud, engage in external pentests, use trusted encryption methods, enable Firewall and PAM, monitor changes, integrate vulnerability scans and patch management in CI/CD, and prioritize team training on layered defense compliance protocols. |
| **3** | 1. Keep it Simple (ALL) 2. Validate Input Data (STD 001, 002, 003, 004) 3. Heed Compiler Warnings (STD 002, 003, 010) 4. Use Effective Quality Assurance Techniques (STD 001, 002, 003, 005, 006, 007, 008, 010) 5. Adopt a Secure Coding Standard (STD 005, 006, 007, 008, 009, 010) 6. Architect for Security Policies (STD 002, 004, 005, 006, 007, 008, 009, 010) 7. Default Deny (STD 004) 8. Sanitize Data Sent to Other Systems (STD 003, 004) 9. Practice Defense in Depth (STD 004) 10. Adhere to Principle of Least Privilege (STD 004, 009, 010) |
| **4** | 1. Data Type 2. String Correctness 3. Expressions 4. Memory Protection 5. Data Value 6. Secure Object Oriented Programming 7. SQL Injection Prevention 8. Assertions 9. API Vulnerability Assessment 10. Exceptions |
| **5** | Rest: Employ full-disk encryption at the server level and database encryption with MySQL Server, along with a backup strategy to secure data at rest.  Flight: Utilize updated, secure libraries, implement Public Key infrastructure for end-to-end protection on message content or attachments, use Managed File Transfer or SSH with link expiration, password access, and leverage Data Leak Prevention mechanisms integrated into cloud services.  Use: Utilize identity management mechanisms to verify user roles and identity, enable conditional access to tool functionalities based on user roles and other parameters. Apply IRM digital rights management for persistent protection of documentation. |
| **6** | Authentication: Refers to verifying a user's identity by employing valid user credentials such as a username and password. Implement a secure authentication process utilizing a local protected database or an external AWS server, while prioritizing well-established and trusted protocols.  Authorization: Following successful authentication, employ authorization to define the user's permitted access to resources and functionalities, governing their actions.  Accounting: Involves tracking and logging user activities during login/logout and resource usage, including user uptime and configured parameters. |
| **7** | We are conducting assessments for vulnerabilities related to out-of-bounds occurrences (STD-001 Data Value, STD-005 Memory Protection).  Our primary testing framework, Google Test, enables us to create autonomous and reproducible tests, making it our preferred choice.  Adhering to the Google Test Primer, we will implement the following guideline: "ASSERT\_\* versions trigger fatal failures upon failure, leading to the termination of the current function."  Initial test: Validate the addition of 5 values to the collection. |
| **8** | In Test 2, ASSERT\_TRUE(collection->max\_size() >= 10) validates whether the maximum size is greater than or equal to specific values. |
| **9** | Test 3 involves ASSERT\_TRUE(collection->size() > initialSize), enabling us to confirm whether the container's size has expanded after a resize. Conversely, this approach can verify if a resize operation has reduced the collection size. |
| **10** | Test 4 uses ASSERT\_TRUE(collection->size() == 0) to validate whether the resizing operation has successfully reduced the collection size to zero. |
| **11** | Test 5, which utilizes ASSERT\_TRUE(collection->size() == 0), aims to confirm that the collection has indeed been resized to zero after the clear operation. Similarly, ASSERT\_TRUE(collection->size() == 0) can be used to validate that the collection has been cleared from the beginning to the end. |
| **12** | Automate processes during Build by integrating tasks like compilation and static code checks into a CI/CD pipeline, which streamlines team collaboration and leverages technologies such as Docker for containerization, GitLab for version control, and Jenkins for continuous integration.  Implement automated security and regression tests in the SecOps phase, employing virtualized container deployment. Embed security tests, including static and dynamic application security testing, within the SDLC to detect vulnerabilities in real-time.  Utilize tools like OWASP Dependency-Check to assess code dependency vulnerabilities. Key tools include ClangTidy, Cppchecker, Parasoft, Coverity, Jenkins, GitLab, Docker for various stages of the development lifecycle. |
| **13** | Fixing bugs during implementation costs significantly more than identifying and rectifying them during the design phase according to IBM. By implementing secure coding practices in the design and pre-production stages through automated tests and threat documentation, we can prevent costlier issues in production, which require defense in depth strategies and proactive security measures. |
| **14** | Discuss Pentesting's ability to uncover and address cyber-security threats in commonplace malware and APIs. Detail Cloud Security's "defense in depth" approach across layers including Critical Assets, Data, App, Endpoint, Host, Network, Perimeter, Cloud, and Physical Security.  Adopt diverse tools for Cloud Security, opt for least privilege security controls for network and host security, and stay updated with current security publications. |