# CS 405 Project Two Script Template

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

Chad Nadeau

**https://youtu.be/HhDHnTsWyKY**

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome to the security policy presentation for Green Pace, my name is Chad Nadeau and today we are going to talk about the introduction of my security policy. |
| **2** | Security policies are crucial for safeguarding organizational assets, ensuring regulatory compliance, and mitigating security risks. They establish clear guidelines and responsibilities to prevent breaches, promote awareness, and maintain operational resilience in evolving threat environments. The diagram below shows the different layers of Defense in Depth and how everything interacts. |
| **3** | Threat matrices are used to prioritize risks based on their likelihood and impact, enabling organizations to allocate resources effectively and address the most critical threats first.  In the upper left we have likely – This describes the high probability threats that require immediate attention.  In the upper right we have priority – This describes the high relevancy of a threat.  In the lower left we have low priority – This describes the low relevancy of a threat.  Last, in the bottom right we have unlikely – This describes the low importance of threats and require minimal attention. |
| **4** | Here we have the 10 principles and what coding standards they align with;  Validate Input Data   * Exclude user input from format strings.   Heed Compiler Warnings   * Detect and handle standard library errors * Close files when they are no longer needed.   Architect and Design for Security Policies   * Never hard code sensitive information * Allocate sufficient memory for an object * Declare objects shared between threads with appropriate storage durations   Keep It Simple   * Guarantee that storage for strings has sufficient space for character data and the null terminator * Free dynamically allocated memory when no longer needed   Default Deny   * Do not form or use out-of-bounds pointers or array subscripts   Adhere to the Principle of Least Privilege   * Ensure that unsigned integer operations do not wrap   Sanitize Data Sent to Other Systems   * Detect and handle standard library errors   Practice Defense in Depth   * Free dynamically allocated memory when no longer needed   Use Effective Quality Assurance Techniques   * Detect and handle standard library errors   Adopt a Secure Coding Standard   * All coding standards apply. |
| **5** | I prioritized the standards by what directly impacts Green Pace’s security the most. Ensuring sensitive information is not hard-coded and avoiding out-of-bounds operations are foundational to preventing exploits. Memory management (freeing memory, closing files, and ensuring proper allocation) was ranked next, as it's crucial for program stability and performance. Error detection and handling came after, as strong handling ensures a program can recover from failures gracefully. Lastly, the management of shared resources between threads was listed last, as this mainly applies to more complex, concurrent systems where the likelihood of issues depends on the architecture. |
| **6** | Encryption poilcies and how they apply:  Encryption at rest is the process of encrypting data while it is not being used or while it is stored on a device such as a hard drive. This protects data from unauthorized access in case the physical storage is compromised, lost, or stolen.  Encryption at Flight is the process of encrypting data while it’s being transmitted across a network. This ensures that data is safeguarded from interception or tampering during transfer.  Encryption in Use is the process of encrypting data while it is actively being used in memory, such as an application or program. This is used to protect data while in use, ensuring it is not exposed even during computation. |
| **7** | Authentication – Only legitimate users or systems can access resources by verifying their identity.  Authorization - Authenticated users or systems are only granted the specific permissions or access they need, according to their role, rights, or policies.  Accounting - Track and record user activities and system events to ensure compliance and detect potential security incidents. |
| **8** | The unit test, CollectionSmartPointerIsNotNull, checks that the collection smart pointer is not null by verifying both the smart pointer itself and the raw pointer obtained with the collection.get method. To enhance the test, you can add checks for correct initialization, behavior under various operations and edge cases such as clearing the collection |
| **9** | The IsEmptyOnCreate unit test checks that the collection is empty upon creation by asserting that the collection's empty method returns true and its size method returns 0. To take this test a step further, you could test the collection after adding elements to verify that it no longer reports as empty, or check for other operations like clearing the collection and ensuring the size is reset |
| **10** | The CanAddToEmptyVector unit test verifies that a collection, initially empty, can successfully add an entry. To take this test a step further, you could add more entries and check if the size grows accordingly, or test adding elements of different types. |
| **11** | The CanAddFiveValuesToVector unit test ensures that the collection can successfully add five entries by calling the add entries with a value of 5 method and then verifying that the collection's size is 5. To take this test a step further, you could check the values of the elements added to the collection to ensure they are correctly stored, or test adding a larger number of entries and verifying the collection handles resizing properly. |
| **12** | Here we have a diagram that summarizes the automation pipeline and outlines how the process works. You can see we have different elements to the diagram such as, Asses and plan, maintain and stabilize, monitor and detect and verify and test. All these functions work in unison to make sure automation is seamless. |
| **13** | A DevSecOps pipeline is an integrated approach to software development and deployment that incorporates security practices throughout the entire Continuous Integration/Continuous Deployment (CI/CD) pipeline. It aims to make security a shared responsibility across development, security, and operations teams, ensuring that security is embedded into every stage of the software lifecycle.  External tools in a DevSecOps pipeline automate security practices throughout the software development lifecycle. Key tools include Static Application Security Testing (SAST) for static code analysis, Dynamic Application Security Testing (DAST) for testing running applications, Software Composition Analysis (SCA) for managing third-party dependencies, and Infrastructure as Code (IaC) security tools for configuration checks. These tools are integrated into CI/CD pipelines to continuously scan for vulnerabilities, enforce security best practices, and ensure compliance, providing real-time feedback and improving overall security posture. |
| **14** | Without a clear security policy, organizations risk breaches, data loss, and non-compliance. Implementing a policy now provides immediate protection and better risk management, though it requires some initial effort. Waiting only increases the chances of security gaps and costly incidents. To stay ahead, companies should create clear policies, integrate security into development, and keep employees trained and informed. |
| **15** | Lack of Regular Updates: Security policies may become outdated without regular updates, leaving new threats unaddressed.  Insufficient Employee Training: Employees may not be adequately trained on security best practices, increasing the risk of phishing or other social engineering attacks.  Weak Incident Response Plans: Without a detailed, tested incident response plan, organizations may struggle to respond effectively to security breaches.  Poor Integration with Development Processes: Security may be treated as a separate, after-the-fact process rather than being integrated into the CI/CD pipeline, leaving gaps in vulnerability scanning and code review. |
| **16** | Implementing security policy standards helps a company proactively protect sensitive data, reduce risks, and ensure compliance with industry regulations. By standardizing security policies, a company not only safeguards its assets but also builds trust with customers, partners, and regulatory bodies, ensuring long-term operational stability and reputation. |