# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Good afternoon, my name is Walter Augustine and today I will be presenting the new security policy that I have developed for Green Pace over the past 7 weeks. |
| **2** | This security policy was created with the intent of providing a comprehensive and consistent set of standards to be applied across the whole company, both within the services we offer and our own internal tooling. A security policy is the first step of implementing a defense in depth strategy, which is the concept of having multiple layers of security, so that if one layer fails or is breached, then another layer can continue to protect the system. |
| **3** | The security policy includes ten coding standards, which I’ll cover in the next slides, organized in this threat matrix by likelihood and priority. The threat matrix makes it easier to visualize which standards should receive more attention and focus to either prevent or remediate. |
| **4** | In addition to ten coding standards, there are also ten security principles, some of which are explicitly coding related. The ten coding standards have each been mapped to a principle, with some principles applying to multiple standards.  Number one is “validate input data” which means never run or evaluate user input unless it has been determined to be safe.  Number two: “heed compiler warnings”. Some compiler warnings indicate vulnerabilities and bad practices that won’t prevent compilation, but may cause serious issues if left unchecked.  Number three: “architect and design for security policies.” Applications and systems should be built securely from the ground up, and not bolted on later as an afterthought.  Number four: “keep it simple.” A simpler system is easier to understand, build, maintain, and secure, which is all beneficial to an organization.  Number five: “Default deny.” When it comes to access management, users should not be able to do anything by default unless they are given explicit permission by an admin user.  Number six: “Adhere to the principle of least privilege” is like number five with one key difference, which is that users are granted as few permissions as needed to accomplish their work.  Number seven is “sanitize data sent to other systems.” Sanitizing data between other systems keeps malformed or incorrect data from being sent more than once between multiple systems.  Number eight “practice defense in depth”, as I mentioned, which is the concept of having multiple layers of security, as mentioned previously.  Number nine “use effective quality assurance techniques.” Use the right tools for the job to deliver and maintain high quality code, such as static analysis tools and unit testing.  Number ten is “adopt a secure coding standard.” This principle should be applied across the entire organization, because it is difficult for a single person to maintain the security of the entire organization by themself. Applying this principle means that everyone is responsible for maintaining security. |
| **5** | With respect to the C++ code owned by Green Pace, there are ten standards which every developer will be expected to follow. Other standards exist for different languages, but this policy only covers C++ at the moment. There are more standards and security rules for C++ that can be found online, but for the sake of simplicity, this policy includes ten that are most relevant to the software development processes at Green Pace. Detailed information about each of the standards, including code snippets, can be found in the security policy.  STD-001-CPP: Never qualify a reference type with const or volatile.    STD-002-CPP: Copy operations must not mutate the source object.    STD-003-CPP: Guarantee that storage for strings has sufficient space for character data and the null terminator.    STD-004-CPP: Sanitize data passed to complex subsystems.    STD-005-CPP: Do not access freed memory.    STD-006-CPP: Assert liberally.    STD-007-CPP: Handle all exceptions.    STD-008-CPP: Do not read uninitialized memory.    STD-009-CPP: Do not modify the standard namespaces.    STD-010-CPP: Prefer special member functions and overloaded operators to C Standard Library functions. |
| **6** | There are three categories in which encrypted data can be described. In flight, at rest, and in use. Encryption in flight means that data gets encrypted before it is transmitted, then decrypted once it reaches the destination. This is one of the most common use cases for encryption because data sent across the internet is inherently risky. Almost everyone who uses the internet has visited a website that uses HTTPS to create a secure connection between client devices and the servers, so that sensitive data such as credit card numbers or passwords can be sent safely. Secure connections should always be used whenever possible, and sensitive data should never be sent over insecure connections if a secure connection is not possible.  Encryption at rest refers to data that’s being stored, for example on a hard disk, solid state drive, or thumb drive. All data at rest should be encrypted, which can be achieved using tools provided by the operating system, or through downloaded software.  Encryption in use means data is encrypted when it is loaded into memory. There aren’t many practical use cases for encryption in use, since encryption in flight can usually achieve the same outcome for a fraction of the cost. Specialized hardware can provide secure execution environments, such as SGX which is made by Intel. Solutions for encryption in use should be evaluated on an as-needed basis. |
| **7** | The Triple-A framework refers to three components of a user management system that when used properly, provide the greatest level of security. The components include authentication, authorization, and accounting. Authentication refers to unique username and password combinations, which provide the users with initial access to a system. The rules for passwords should include a minimum strength threshold, automatic expiration every 60 days, and no reuse of old passwords.  Authorization refers to the permissions that users may be granted to perform certain actions. One of the principles to be applied here is “default deny” which is that newly created users have zero permissions, which must be granted explicitly on an individual basis. This ensures that only users with a business requirement to perform an action as part of their job will be allowed to do so, which makes which makes it easier to audit the history of a system.  The last point is accounting, which refers to maintaining a log of every system interaction. A log may include the name and IP address of the user, the action, and a timestamp of when the action was performed. A log makes all users accountable for their actions, since only logged-in users can interact with the system. |
| **8** | Unit testing is a part of code quality and avoiding vulnerabilities. Unit tests can include assertions which check for common code smells such as accessing freed memory. One unit test framework for C++ is called Google Test, and can be installed in several different IDEs. One future standard that can be implemented for unit tests is a minimum level of test coverage, which is a percentage value indicating how much of the source code logic is covered by a test. Once a coverage standard is implemented and reached, then all new code must be accompanied by a test in order to keep the coverage from declining below the minimum level. |
| **9** | This is a summary of the software development process, which you’ll notice is cyclical, because production software systems are dynamic entities that must be continuously maintained, at the risk of introducing a defect or vulnerability. DevSecOps sits in the middle of this diagram because it is integral to both the planning and production stages of software development. |
| **10** | While DevSecOps requires an active team of professionals who can predict and respond to threats, much of the heavy lifting is done by automated static analysis tools. CPP Check and SonarQube are some of the most widely used tools in the industry for detecting vulnerable code, and many IDEs can provide instant code fixes for many defects. |
| **11** | There are risks of not having a security standard. The biggest risk is that if different standards are followed across the organization, then the defense in depth strategy can’t be adhered to effectively because multiple standards implemented in isolation are likely to be single-layered, and thus have nothing to fall back on. This puts the entire organization at risk because an attack on one area could be duplicated to another and could spiral out of control.  The benefits of a security policy are primarily that a single standard is easier to understand and maintain by the developers, and that it keeps the company safer by combining the efforts of employees in different departments and raising the level of company awareness of security best practices. |
| **12** | There are some areas which this policy does not cover and should be investigated in future versions of this policy.  The first is physical security, which refers to the physical presence of staff and guests in the office where company hardware is located. One attack vector not discussed is not software, but people, using social engineering and other psychological hacks people use to trick people into giving up passwords or letting them into a building to which they are not authorized. Best practices on physical security should be investigated and added to the policy within a reasonable timeframe.  The second area is other programming languages. This policy currently only covers C++, but should eventually extend to all languages used by the company, which can include C, Java, Python, JavaScript, or others. |
| **13** | To conclude this presentation, I would like to restate the top priorities moving forward for implementation of the security policy.  Defense in depth is the primary focus of a successful implementation, as multiple layers of security will always be more effective than any single layer. Each department should investigate their own requirements for defense in depth.  Automation pipelines should be set up within the current workflow to include static analysis on both source code and the networks.  Unit testing should be a requirement for all new source code, and a minimum level of code coverage should be established and maintained.  Finally, this security policy should be reviewed and updated every year to keep up with new technology and industry best practices, and to make sure every developer understands the standards and the purpose behind them. |
| **14** | Thank you for your time and enjoy the rest of your day. |