**7-2 Project Two: Security Policy Presentation**

Jessica Ayer

December 11th, 2022

CS 405: 22EW2

Instructor: Aaron Demory

YouTube presentation Link: https://youtu.be/9uySd6nbXlQ

# CS 405 Project Two Script Template

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | The presentation will introduce you to Green Pace’s new security policy. The policy is designed to support the transition from a DevOps to a DevSecOps workflow. It covers both what standards and policies are being implemented as well as why. |
| **2** | Defense in Depth protects your data. It is the practice of implementing multiple layers of security measures to protect one’s assets. These layers include but are not limited to antivirus software, firewalls, secure gateways, and best coding practices. Each layer filters out more vulnerabilities to create an overlapping net and preventing security breaches from penetrating though any individual layer’s holes. It can also thwart attacks that have already breached the system. Human negligence is often what lead to security breaches so practicing Defense in Depth is crucial (Fortinet, 2022). |
| **3** | The matrix shows how the severity level is assessed for each coding standard. A Static analysis test can be used to detect these threats and offer suggestions for correction. As you can see standards 3, 4, and 5 address issues that are both likely to lead to a vulnerability and have high relevance. |
| **4** | This slide shows a list of the top 10 secure coding principles.   * Validate Input data * Heed Compiler Warnings * Architect and Design for Security Policies * Keep it Simple * Default Deny * Adhere to the Principles of Least Privilege * Sanitize Data Sent to Other Systems * Practice Defense in Depth * Use Effective Quality Assurance Techniques * Adopt a Secure Coding Standard   For details on each principle see the Security Policy Documentation. |
| **5** | The coding standards listed from the highest priority to the lowest based off severity and likely hood of creating an exploitable vulnerability are:   * Guarantee that container indices and iterators are within the valid range * Guarantee that storage for strings has sufficient space * Sanitize all input data * Do not access dangling pointers * Make sure that performing operations with signed integers does not result in overflow * Copy operations must not mutate the source object * Handle all exceptions * Close files when they are no longer needed * Understand the size of standard data types * Use a static assertion to test the value of a constant expression   Notice the top priorities all refer to memory management vulnerabilities.  Once again, details for each standard can be found in the Security Policy Documentation. |
| **6** | Three key encryption policies will be implemented to protect Green Pace’s data.   * **Encryption at Rest** protects stored data when it is not in use to prevent an attacker from gaining access even if they can access the storage server. Identify the data, how it is stored, and who needs access to decrypt it. Block all other potential users from reading the data. * **Encryption at Flight** protects data as it is transfered from its resting location to the user. We can prevent “Eavesdroppers” by using both symmetric and asymmetric key encryption. * **Encryption in Use** protects data while it is being accessed by a user. Utilize AES-256 encryption in all data fields and analyze data requests in real time to block suspicious requests. |
| **7** | The Triple-A policies address the questions:   * who are you? * What resources are you permitted to use?   (And)   * What resources were accessed, at what time, by whom, and what commands were issued?   This is done through:   * **Authentication:** Identifying a user and verifying that they are who they claim to be with a unique set of credentials * **Authorization:** Determining what access level a user is granted to the system and what functions they can perform   (And)   * **Accounting:** Documenting what resources and the timeframe each user utilizes while accessing the system. |
| **8** | The next few slides are examples of unit tests.  The first checks to see if container indices and iterators are within the valid range. The test is checking to make sure an exception is thrown if an out-of-range index is called.  You need to verify that an index call is not out-of-bounds. If you use an iterator that is out-of-range overflow may occur. |
| **9** | The second example addresses pointers.  Here we are testing to see if a pointer is NULL. Accessing a dangling pointer will result in undefined behavior and exploitable vulnerabilities. |
| **10** | Example three makes sure there is sufficient space available for the data type or in a container before trying to add values.  By making sure the capacity of a container is large enough for the data we desire it to hold we prevent improper memory management that could create vulnerabilities. This slide shows a failed test. The container “collection” has a capacity of zero and therefore cannot hold data without being resized. |
| **11** | The fourth and last unit test example in this presentation checks to make sure the size of a container cannot be negated. An exception is thrown if the user attempts to negate the size. It is important to know what data type is being used and what size range it can be to prevent underflow or overflow. |
| **12** | This visual show the Pipeline for DevSecOps and how automation can be incorporated into the software development lifecycle. As you can see, security has been brought to the forefront of every phase. |
| **13** | DevSecOps stands for Development, Security, and Operations. It integrates security into each phase of the software development lifecycle with automation tools. Security protocols should be implemented when planning, designing, building, testing, and maintaining the system.  The tools that were used to run some sample testing for Green Pace were Google Test and CPPcheck.  Google Test was used for the unit tests. It is simple to use and integrates right into the IDE.  CPPcheck is one of many available static analysis testing applications. The tool is used to discover warnings, and errors in the developed code. It offers a short description and suggestions on how to fix each of the issues found. |
| **14** | There are always going to be risks and benefits to adopting a new workflow. If Green Pace chooses to adopt DevSecOps right now.  Risks would include:   * Lack of training: * More upfront costs   The benefits to switching to DevSecOps right now include:   * Catching vulnerabilities sooner and protecting your data now. * Code being more secure sooner rather than later resulting in less remediation.   Should Grean Pace prefer to wait until later to implement DevSecOps Risks include:   * Insure code * Security gaps due to manual testing practices * More remediation which is costly when development is already complete   Benefits to waiting are:   * More time to learn the new tools and become comfortable with change * Faster code development as the developers won’t be bogged down with new requirements * The current production cycle will not be interrupted and therefore deadlines will not be affected. |
| **15** | Here are some recommendations:   * Slowly introduce the DevSecOps pipeline into the software development lifecycle. Start with Secure coding standards then introduce new automated testing tools one at time. * Research the appropriate unit testing platform and static analysis applications and thoroughly educate the team before implementation. * Define user roles and their access levels and implement a Tripple A framework. * Make sure data is always encrypted. * Always Practice Defense in Depth |
| **16** | Practicing security from start to finish with DevSecOps by following secure coding practices, using a Triple-A framework, constant data encryption, and adopting automated tools will help protect Green Pace from falling victim to an attack. A Defense in Depth strategy will give your clients peace of mind and allow them to trust Green Pace with their sensitive data. |
| **17** | Thank you for taking the time to watch this presentation. |