# CS 405 Project Two Script Template

**Shingirai Chiremba**

**16 April 2023**

**Project Two: Security Policy Presentation**

[**https://youtu.be/s6aPWcraK4Q**](https://youtu.be/s6aPWcraK4Q)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Title slide |
| **2** | The purpose of this security policy is to take the implicit policies that are applied daily in practice and explain how they have been standardized. This policy explains Green Pace’s security standards and policies, including the surface area of an attack and assumption of vulnerability. Detail was taken to demonstrate how the coding and architectural issues are organized using a set of 10 guiding security principles. There are examples to demonstrate how to apply external testing methods to identify potential vulnerabilities by adding screenshots from coding tests and explaining how external testing methods will catch the vulnerabilities. Unit tests were written to check for the vulnerabilities using the unit testing framework for C++ in Visual Studio. |
| **3** | Principle 1 is about Validating Input Data. The 3 Standards are all about data inputting. Compiler warnings are an indication of something bigger, an issue with the code, and should not be taken lightly. Syntax issues usually lead to compiler warnings which can lead to compiler errors and the program fails to read the machine code. When this happens, the system will fail and with systems down trojans can be planted and harm the system once it is back online. Following OWASP’ secure architecture (SA) practices is recommended as well as the use of subsystems, reliable libraries and plugins. SEI CERT also provides a list of rules and policies that can be used as a reference for security policies.  Security controls should not be overwhelming and easy to navigate and understand. The code should be readable and in most cases reusable.  This policy should be applied across the board to new and old user accounts and access accounts. New settings and changes, features should all be secure until it’s time to release them.  Permissions should be set for all requests, changes, and access attempts and they must be validated. It should be rule of thumb to validate permissions before any new feature is released and permissions should be reviewed often.  String data passed to other systems may contain special characters that can trigger commands or actions, resulting in a software vulnerability.  Multiple layers of security stacked on top of each other in a single system. The idea is to have the security protocols of the systems checking each other for vulnerabilities working in an umbrella-type environment.  Feedback is always key with new developments and projects, so it is always advised to have QA involved from initial planning. Regular product testing and improvements made around proven tests and test results are encouraged for a better user experience with the product.  Security standards that promote secure coding have been put in place over the years. Best and recommended practices have been published and should be followed because they come from a place of deep research and development. Developers need to be aware of these secure coding standards and best practices, so it becomes second nature when they are developing code. |
| **4** | Some of the risks involved with DevSecOps include its complexity when it comes to cloud implementation especially for continued processes and updates. This in turn gives it compatibility issues with cloud platforms and in some cases local machines and networks. It is very challenging to balance speed and security that is if you want security to be done right. Security done right is time consuming because of the tests that need to be done, unit testing. Some organizations have a hard time getting their employees on board with DevSecOps especially if they are accustomed to other traditional methods. |
| **5** | - DCL12-C. Implement abstract data types using opaque types - Abstract data types are not restricted to object-oriented languages such as C++ and Java. They should be created and used in C language programs as well (Sei Cert, 2016).  - INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data – According to the Sei Cert database, Integer conversions, both implicit and explicit (using a cast), must be guaranteed not to result in lost or misinterpreted data (2016).  STR30-C: Do not attempt to modify string literals. Modifying a string literal frequently results in an access violation because string literals are typically stored in read-only memory (Sei Cert, 2016).  IDS00-J. Prevent SQL injection – SQL query can have elements originating from untrusted sources causing SQL Injection weaknesses and attacks.  MEM50-CPP: Do not access freed memory – Memory Protection is a very important standard that must be adhered to protect from known vulnerabilities like arbitrary code execution.  DCL03-C. Use a static assertion to test the value of a constant expression – Assertions are used to detect and eliminate program defects that may cause vulnerabilities in the software.  ERR51-CPP. Handle all exceptions. When an exception is thrown, control is transferred to the nearest handler with a type that matches the type of the exception thrown. If no matching handler is directly found within the handlers for a try block in which the exception is thrown, the search for a matching handler continues to dynamically search for handlers in the surrounding try blocks of the same thread.  MEM31-C. Free dynamically allocated memory when no longer needed - Before the lifetime of the last pointer that stores the return value of a call to a standard memory allocation function has ended, it must be matched by a call to free() with that pointer value.  STR53-CPP. Range check element access – The std::string index operators const\_reference operator[ ](size\_type) const and reference operator[ ](size\_type) return the character stored at the specified position, pos. When pos >= size(), a reference to an object of type hart with value hart() is returned (Sei Cert, 2016).  EXP47-C. Do not call va\_arg with an argument of the incorrect type - The variable arguments passed to a variadic function are accessed by calling the va\_arg() macro (Sei Cert, 2016). |
| **6** | Data At Rest Encryption (DARE) is the encryption of the data that is stored in the databases and is not moving through networks. With DARE, data at rest including offline backups are protected. According to Microsoft, “Encryption at rest is designed to prevent the attacker from accessing the unencrypted data by ensuring the data is encrypted when on disk. If an attacker obtains a hard drive with encrypted data but not the encryption keys, the attacker must defeat the encryption to read the data” (MSM Baldwin, 2016).  Encryption at flight is the encryption of data that moves over a network. Basically, it is the encryption of data in transit. When data moves over a network it is vulnerable to interceptions over the network creating points of weakness.  The general encryption methods only protect data when it is at rest (disk encryption), and when it is in transit of networks. This leaves vulnerabilities when the data is in use by on-premises or cloud applications. Encryption policies to implement protection of data in use are practiced by organizations protecting cloud information. This produces weak endpoints with cloud based systems and storage. According to Microsoft, “In-Use encryption takes a new approach that ensures that sensitive data is never left unsecured, regardless of or lifecycle stage (at rest, in transit, or in use) source, or location (on premise, cloud, or hybrid)” (MSM Baldwin, 2016). |
| **7** | Authentication - After the AAA policy extracts the service requester identity and resource, it authenticates the claimed identity. Authentication is most accomplished through an external service (for example, a RADIUS or LDAP server), but custom processing methods, such as site-specific XML or XPath based solutions, are supported. During policy definition, you select a single authentication method, and, depending on the selected method, provide more required information.  Authorization - Successful server-based authentication generates a set of credentials that attest to the identity of the service requester. You can then map these credentials to a set that is more appropriate to the authorization method. You can accomplish this optional mapping through an XPath expression, an XML mapping file, or a custom method. As with identity credentials, the extracted resource name can be mapped to a more appropriate authorization method. The methods to achieve this optional mapping are the same as the methods for credential mapping. The resulting credentials, along with the resultant resource name, are the basis for client authorization. Client authorization determines whether the identified client has access to the requested resource.  Accounting - the final process in the framework, is all about measuring what's happening within the network. As part of the protocol, it will collect and log data on user sessions, such as length of time, type of session, and resource usage. The value here is that it offers a clear audit trail for compliance and business purposes. |
| **8** | 13 unit-test names were created with the help of Google name generator. The Google Test fixture was executed, including the 11 tests already defined and added the two negative tests. ASSERT and EXPECT macros are used appropriately to handle cases where processing should terminate or continue upon failure. To prove the tests, the Google Test framework is initiated and double checked that each test proved the defined condition of the test. The use of appropriate naming conventions, in-line comments, and a software design pattern approach is practiced throughout the code. |
| **9** | The Unit Testing process begins with creating a test class to house shared data between tests. The test class created was called CollectionTest, essentially creating a smart point to hold our collection. After that, we create a new collection to be used in the test and erase all elements in the collection, if any remain. Lastly, add a helper function to add random values from 0 to 99 count times to the collection. |
| **10** | In this slide, the screenshot is a Test that a collection is empty when created.  Use ASSERT when failure should terminate processing, such as the reason for the test case.  Prior to calling this (and all other TEST\_F defined methods),  CollectionTest::StartUp is called.  Following this method (and all other TEST\_F defined methods),  CollectionTest::TearDown is called |
| **11** | This slide has two tests in it, one to test AlwaysFail and a second test CanAddToEmptyVector to test adding a single value to an empty collection. So, for this test we check if the collection is empty, if empty, the size must be 0. We add an entry (1), then next is the collection still empty if not empty, figure out what must be the size. |
| **12** | This slide shows a screenshot of a test called MaxSizeGreaterThanSize. This is a test to verify that max size is greater than or equal to size for 0, 1, 5, 10 entries. |
| **13** | Automation simplifies implementation and monitoring of operations. According to an online article, “DevSecOps is the practice of integrating security into a continuous integration, continuous delivery, and continuous deployment pipeline. By incorporating DevOps values into software security, security verification becomes an active, integrated part of the development process” (Hristov, 2019). |
| **14** | The plan phase is the least automated phase of DevSecOps, involving collaboration, discussion, review, and strategy of security analysis. The build phase begins once developers commit code to the source repository. DevSecOps build tools focus on automated security analysis against the build output artifact. The test phase is triggered after a build artifact is created and successfully deployed to staging or testing environments. If the previous phases pass successfully, it's time to deploy the build artifact to production. The security areas of concern to address during the deploy phase are those that only happen against the live production system. Once an application is deployed and stabilized in a live production environment, additional security measures are required. Companies need to monitor and observe the live application for any attacks or leaks with automated security checks and security monitoring loops (Hristov, 2019). |
| **15** | According to IBM articles,  Incorporating security into DevOps helps speed up iterations.  DevSecOps helps in developing high quality products without compliance issues.  It helps developers think critically, understand security requirements, and design the software properly from the beginning. It eliminates manual configuration of security consoles, which reduces cycle time. Security functions like identity and access management, firewalls, and vulnerability scans can be automated throughout the DevOps cycle. Vulnerabilities are identified earlier which helps to avoid cyber-attacks. It helps improve communication and collaboration between teams.  (Javed, 2022). |
| **16** |  |
| **17** | Audit Controls and Management - Every software development effort must be able to provide evidence of compliance for each software deployed into any Green Pace managed environment. Evidence will include the following: Code compliance to standards, Well-documented access-control strategies, with sampled evidence of compliance. Well-documented data-control standards defining the expected security posture of data at rest, in flight, and in use and historical evidence of sustained practice (emails, logs, audits, meeting notes).  Enforcement - The office of the chief information security officer (OCISO) will enforce awareness and compliance of this policy, producing reports for the risk management committee (RMC) to review monthly. Every system deployed in any environment operated by Green Pace is expected to be in compliance with this policy at all times. Staff members, consultants, or employees found in violation of this policy will be subject to disciplinary action, up to and including termination.  Exceptions Process - Any exception to the standards in this policy must be requested in writing with the following information: Business or technical rationale, Risk impact analysis, Risk mitigation analysis, Plan to come into compliance and Date for when the plan to come into compliance will be completed. Approval for any exception must be granted by the chief information officer (CIO) and the chief information security officer (CISO) or their appointed delegates of officer level. Exceptions will remain on file with the office of the CISO, which will administer and govern compliance. |
| **18** | These are the citations used in this presentation. |