CS-405 Secure Coding – Project Two Presentation Script

Eric Slutz

Southern New Hampshire University

snhu

CS 405 Project Two Presentation Script

Slide Number	Narrative
1	Hi, my name is Eric. This is my presentation of the security policy developed for the Green Pace organization as part of the SNHU CS-405 Secure Coding course. This presentation will cover the various components that make up a good security policy.
2	The security policy developed for Green Pace specifies the standards and best practices that should be followed to maintain the safety and security of the organizations applications and data. The parts of this policy make up the layer of security needed for Defense in Depth. This graphic goes into further details of Defense in Depth, showing all the potential areas and layers that could need to be secured.
3	The threat matrix is used to visualize the severity of the security risks being addressed by this security policy. Starting in the bottom left of the matrix, you have the security risks that are low severity and unlikely to happen. Above that you have the security risks that are likely to happen but still of low severity. On the bottom right of the threat matrix you have the security risks that are unlikely to happen but if they do it presents a high severity security risk. Lastly, in the top right corner, you have the security risks that are of high severity and likely to occur. Using an automated static analysis tool, such as CPPcheck, can identify risks like these so you can then mitigate the security risks presented.
4	In the security policy there are ten principles that the security standards can fall under. • The principle “Validate Input Data” applies to coding standard four. • The principle “Heed Compiler Warnings” applies to coding standards one and two. • The principle “Architect and Design for Security Policies” applies to coding standards six, seven, and eight. • The principle “Keep It Simple” applies to coding standards two, four, eight and ten. • The principle “Default Deny” applies to coding standard nine. • The principle “Adhere to the Principle of Least Privilege” applies to coding standard nine. • The next principle is “Sanitize Data Sent to Other Systems”. • The principle “Practice Defense in Depth” applies to coding standard five. • The principle “Use Effective Quality Assurance Techniques” applies to coding standards three, five, and seven. • The principle “Adopt a Secure Coding Standard” applies to coding standards three, eight, and ten.

Slide Number	Narrative
5	The 10 coding standards specified in this security policy are listed in order of priority based on the threat matrix. 1. Sanitize data passed to complex subsystems 2. Properly deallocate dynamically allocated resources 3. Range check element access 4. Do not modify the standard namespaces 5. Value-returning functions must return a value from all exit paths 6. Handle all exceptions 7. Never qualify a reference type with const or volatile 8. Understand the termination behavior of assert() and abort() 9. Close files when they are no longer needed 10. Write constructor member initializers in the canonical order
6	Encryption at rest refers to data being in an encrypted state while it is in storage. This means that even if access to the data is gained, the data itself is unreadable unless you have the key to decrypt the data. This policy is important and should be used so that if there were a breach and the data was taken, the information within the data would still be safe because it could not be read without the key. Encryption at flight refers to data being in an encrypted state while it is in transit from one place to another. While data is in route form one place to another, it must be encrypted. This ensures that if the data is intercepted, the information within the data would still be safe because it could not be read without the key. Encryption in use refers to the encryption of data as it is being used by the system. It is used to keep data secure at all times, even as it is changing. This is important because as more data is secured with encryption at rest and in transit, then that leave as data is being used as the weakest link to be exploited. For that reason, it is important to figure out how to keep the data encrypted as it is being used.

Slide Number	Narrative
7	Authentication is the process of confirming that a user is who they say they are. It is typically used when adding new users or when a user attempts to login. This policy applies whenever a user is attempting to gain access to a system or something within the system. You need to be sure that the person within the system is who you expect it to be. This is typically accomplished by verify user identification information such as a username and password. Authorization is the process of confirming that a user is supposed to have access to what they are requesting. It determines the level of access a user has with the system. This policy applies whenever a user is attempting to access a system or something within the system. Once authenticated, authorization will determine if the user can gain entry to the system. This helps to keep users out of places they shouldn't be. Accounting is the process of tracking and logging all requests and transactions made by a user. Examples of that would be changes made to a database or files accessed by a user. This policy is important so that you can know what is going on with a system. For example, if the system starts behaving in an unexpected manor, you can review the logs to track down the issue and then hopefully be able to figure out a solution.
8	The next several slides cover different examples of both positive and negative result unit tests. These example all make use of the gTest unit testing framework from Google. In this test you see a test function to validate if five values can be added to a collection.
9	This unit test example shows a parameterized test function. This allows a single test to be run for multiple values. In this case the parameter value specifies the number of entries in a collection. The test function is validating that the max size of the collection is larger than the collection size.
10	This is another test function to validate that resizing the collection to a larger size works as expected.
11	This test function is validating that when a collection is cleared that it is indeed empty.
12	This test function is our first example of a negative test. It is validating that an out-of-range exception is thrown when a collection is access with an index out of bounds.
13	This last example is another negative test to validate the result of setting a collection reserve above the maximum size for the collection.
14	The illustration of the DevSecOps pipeline shows the different stages of producing a secure application. There are many points throughout this pipeline that can benefit from automation to result in a more secure program.
15	The DevSecOps pipeline can be broken into two parts, pre-production, and production. There are many tools that can be integrated into the stages of the DevSecOps pipeline. One good example is to integrate automated static code analysis into the "Verify and Test" stage.

Slide Number	Narrative
16	When thinking about risks and benefits and if you should incorporate security now or later, the answer should be now! It is never too early to start thinking about the security of your application. If you have started planning, you should be planning security right there with it. The benefits include it being easier to implement security features when you're just starting the planning and development process. It also results in more robust and reliable code, as well as more consistent code across your applications. The risks are not having enough buy-in to get the resources needed for adding security. Other risks come from doing security later, where it can be harder to add security to existing code. Also, you're adding security code on top of everything else, instead of integrating it in with the other code, which can result in less secure code.
17	Moving forward, my recommendations are that DevSecOps training should be given to all developers. A security policy does no good unless it is understood and followed. Additionally, automated monitoring for dependency vulnerabilities should be implemented as an additional layer of security.
18	In conclusion, security must be considered from the start and training must be provided on secure coding in general and specifically about the contents of the security policy. A security policy does no good if it is not followed. That's why there should be enforcement mechanisms in place to ensure that any application that is deployed meets the requirements of the security policy. Lastly, the security policy should be regularly reviewed and updated as needed.