Southern New Hampshire University

CS 405 Project Two Script Template

Iesha Sahin
12/22/2024
Project Two: Security Policy Presentation
https://youtu.be/nPHdse-13cE

Slide Number	Narrative
1	Hi, I'm Iesha. Welcome to my Security Policy presentation for Green Pace. Today I will be discussing security standards and policies and demonstrating how coding and architectural issues are organized using a set of 10 guiding security principles. I will then demonstrate how to apply external testing methods to identify potential vulnerabilities showing real-life examples and explain how external testing methods will catch the vulnerabilities. Let's get started.
2	Here we have a defense in depth strategy illustration. Defense in depth is a strategy that uses multiple security measures to protect an organization's assets from cyber threats. I will explain the security risks potentially faced by the organization, policies and standards that can mitigate development issues, and security practices to ensure the safety of both the company and customers.
3	This is a Threats Matrix that will be implemented at Green Pace. There are four categories of the Threats Matrix: likely, priority, low priority, and unlikely. The purpose of this is to prioritize threat levels organized by severity and threat level.
4	Here we have the 10 core security principles. Security standards will usually fall under one or more of these principles. Validate Input Data ensures only valid data is entered into a system to prevent attackers from exploiting vulnerabilities and buffer overflow. This applies to standard two. Heed Compiler Warnings says to pay attention to compiler warnings reporting unusual conditions in code that may indicate a problem. Even though the compilation can continue, it is important to pay attention to each warning as going unattended could lead to as security risk. This applies to standards three, five, six, and ten.

Slide Number	Narrative
	Architect and Design for Security Policies looks into how information security controls are implemented in systems to protect data that is used, processed, and stored in those systems. This applies to standards one and nine. Keep It Simple simply means to avoid complexity wherever possible to lessen the chance of errors and greater the chance of user acceptance. This applies to standards one, two, eight, and nine. Default Deny says if a request is not explicitly allowed, it is denied. This prevents unauthorized access that could have huge security risks. This applies to standard five. Adhere to the Principle of Least Privilege is to limit a user's access rights to only what is required for them to complete their job. This applies to standard five. Sanitize Data Sent to Other Systems is the process of removing harmful elements to ensure a safe and uncompromised system. All unused sensitive data will be cleared as soon as a storage device is no longer in use. This applies to standard four. Practice Defense in Depth: the strategy that uses multiple layers of protection to minimize the risk of a security breach. Use Effective Quality Assurance Techniques like fuzz testing, penetration testing, and source code audits, all incorporated as part of an effective quality assurance program to effectively identify and eliminate vulnerabilities. This applies to standards five and seven. Adopt a Secure Coding Standard. Governing the coding practices, techniques, and decisions that developers make while building software, aims to ensure that developers write code that minimizes security vulnerabilities. This applies to standards one, two, four, six, seven, nine, and ten.
5	Here we have the coding standards in this policy organized by priority and threat level based off of the threats matrix. In order we have: Do Not Attempt to Create a std::string From a Null Pointer Prevent SQL Injection Properly Deallocate Dynamically Allocated Resources Do Not Read Uninitialized Memory User Valid Iterator Ranges

Slide Number	Narrative
	Do Not Access an Object Outside of its Lifetime Obey the One-Definition Rule Write Constructor Member Initializers in Canonical Order Do Not Leak Resources When Handling Exceptions Use a static Assertion to Test the Value of a Constant Expression
6	Encryption at rest is referring to data being in an encrypted state while it is in storage on a disk. The data will be protected regardless of if a user is granted access to the data because they don't have the key to decrypt it. This is essential because this keeps the data safe even if there was a breach of security. If unauthorized access was granted the sensitive information would remain encrypted. Encryption in flight is about keeping data encrypted during transit. While the data is through the network, for example, an email, it will stay encrypted. This is vital now that we are using the cloud more than ever. Encryption in use is all about encrypting data while it is being processed by a system. Even as the data is created, updated, or read it is staying protected. Encrypting in-use data addresses this vulnerability by allowing computations to run directly on encrypted files without the need for decryption.
7	Authentication confirmed a user's identity, making sure they are who they're claiming to be. This includes users creating accounts, logging in, and changing their passwords. Authentication is usually done by sending out one-time passwords, verifying a username and password combination, or answering security questions that were established at the account creation time. As authentication will verify a user's identity, authorization will confirm the access level of a user. The level of access the user will have is dependent on their authorization given to them by the system. This will limit vulnerabilities and interactions users can have with sensitive data by keeping them locked out of certain access points. Accounting refers to the process of keeping track of all activity made by a user. Interactions include logging in, making changes to the database, and accessing files. This policy if vital for keeping a trail of what is always happening in the system. If something ever goes wrong, accounting allows for a backlog of all activity to take the guesswork out of what was happening in the system. Tracking down the issue is made much easier.

Southern New Hampshire University

Slide Number	Narrative
8	Next we have a few screenshots of previous results of unit tests. Each test was done and passed using Google's framework. First up we have a test function validating if five values can be added to the collection.
9	Our next test validates whether the max size is greater than or equal to a specified number of values.
10	This test asserts whether or not the resizing increases the collection works.
11	The final unit test is validating that when a collection is cleared, it is empty.
12	Here we have an automation summary diagram. As you can see the diagram makes an infinity symbol, left side being pre-production, and right-side being production. We start with assessing and planning. Here we are able to visualize potential threats, and how we would respond to them. Then we move on to designing where we ensure we have a security test driven design while following best practices. Moving on to build. This is where we write code for testing. Verifying and testing is when we test for vulnerabilities. On the right side of the infinity symbol, production, we start at transition and health checks. This ensures the deployed code is working as expected. Monitor and detect focuses on keeping track of activity and access rights. If there is any unauthorized access, the next step is Respond. During this step we're able to block attacks and turn off services if needed. Finally, maintain and stabilize. After responding to a potential attack, we're able to return to a stable state. Then the cycle begins all over again.

Slide Number	Narrative
13	DevSecOps integrates security at every step of the software development lifecycle, from initial design to integration, testing, deployment, and delivery This was explained in the previous slide. Some tools used in the pipeline include: Static Application Security Testing (SAST) analyzes code for vulnerabilities during development Dynamic Application Security Testing (DAST) simulates attacks on applications to identify weaknesses Security information and Event Management (SIEM) collects and analyzes security events for incident detection and response Parasoft C/C++ testing is used to verify and test for vulnerabilities
14	When it comes to the risks vs the benefits of implementing security now or later, the benefits will win every time. Acting now will not only potentially save Green Pace a lot of valuable time, but also money. Not making security a priority and adding it in at a later time will increase the chances of attacks tenfold. Waiting until the end can create delays in projects that didn't need to be delayed if there was security planning from the beginning. This frustraters both the employees and the customers and could ultimately give customers the impression of an untrustworthy and unreliable company.
15	Moving forward, I recommended that all employees receive proper DevSecOps training. Everyone should be able to refer to a complete and comprehensive document to ensure best practices in the code. The security policy should be regularly updated to keep all information up-to-date for everyone when needed.
16	In conclusion software security attracts can be avoided. Incorporating coding principals and coding standards will greatly reduce your chance of attack. Because of this, security must be implemented from the beginning. Proper training on the security policy is also a must to ensure it is understood and able to be followed. Testing regularly actually saves time and money, reducing the chance of missed vulnerabilities Finally, the security policy should be updates and reviewed regularly