Project Two: Security Policy Presentation

YouTube Video Link: <https://youtu.be/iG1g_FxC6UU>

Kenneth J Gollaher

CS-405 Secure Coding, Southern New Hampshire

Module 7-2 Project Script

Professor Ivan Gappy

Oct 16th, 2022

**CS 405 Project Two Script**

| **Slide Number** | **Narrative** |
| --- | --- |
| **1**  **Introduction** | Hello, my name is Kenny Gollaher, and this is my project two security policy presentation for my CS-405 Secure coding course. During the next few slides, I will touch on a variety of key aspects of secure coding, the principles of coding standards, the risks, and benefits of DevSecOps, recommendations, etc. With that being said,  Let’s get started. |
| **2**  **Overview** | Unauthorized access, or data breaches are becoming a major concern to many organizations. On this slide the diagram showcases the Defense in Depth Model, which is a Multi-layer defense system for secure coding. Defense in Depth (DiD) is an approach that developers use to implement mechanisms in a layered fashion (Yendamury & Mohankumar, 2021).  There are a variety of layers that developers can implement to safeguard the data and prevent unauthorized access. Some examples may be firewalls, antispyware, passwords, encryptions, etc. Each layer of defense is a wall that hackers would have to pass over. One of the major concerns is the threat actor having enough time and resources to overcome any of the set defense layers. A tool that we will mention later in the presentation is the use of automation to detect vulnerabilities within the code. Using automation to detect these vulnerabilities will prevent or reduce the risk an environment may face. |
| **3**  **Threat Levels** | With the layers of defense on our mind, this next slide shows how the level of threat is categorized. The table shown represents the four threat levels, which show the severity of the threat and where it sits in importance to the security team. The four levels are unlikely, likely, low priority, and priority, which is the highest level. The threat levels can be found on confluence website. The link is listed in the references page at the end of the presentation. |
| **4**  **Coding Principles** | Here we listed the 10 principles of secure coding. These principles are used as a guide to help the developing team apply best practices and reduce the risk of a breach. You can locate and read these principles on the Confluence website. The link will be located in the reference page at the end of the presentation. |
| **5**  **Coding Standards** | On this slide, I have listed 10 examples of coding standards. You will see on the data table, it shows the rule, which is the id of the warning, how severe it is, what the likelihood is, and its priority. You will also see the category in which these warnings fall located on the left hand side of the table. This gives a developer an idea of where to look within their code, rectify the concern, and test it again. |
| **6**  **Encryption** | Now, let’s get into Encryptions. There are three that we will go over on this slide: Encryption in rest, in flight, and in use.  **Encryption in rest:** This type of encryption is essential for protecting data from a breach. Encryption at rest is the process of converting any sensitive data into another form using an algorithm. The key is needed to decrypt the sensitive data, and only authorized personnel will have the access to decode the data.  **Encryption in flight:** This type of encryption is essential for protecting data transmitted over aa network, that being internal or over the internet. Data transmitted over the internet is potentially at a high risk. When you Encrypt data in flight, that refers to encrypting the data being transmitted over the network. Some examples may be using VPN’s, avoid using self-signed certificates, and using transport layer security (TLS) for your https connections. These are a few examples that can reduce risk.  **Encryption in use:** Data in use is one of the higher forms of vulnerability. Since this data is stored in clear text during its use, it is the most susceptible to exploitation. Encryption in use will always protect the data and is encrypted throughout the entire data life cycle. Any anomalies that are detected are analyzed and blocks any suspicious requests. |
| **7**  **Triple A** | Here you see three processes of verification, which allow clients/users the privilege to gain access to the computer network, which includes resources, policies, auditing, etc. The three processes, known as the Triple A, which includes Authentication, Authorization, and Accounting. These combined processes are considered vital for effective network management and security.  **Authentication:** This process verifies the identity and credentials of the client/user wishing to gain access and connect to the network. The network in turn verifies the credentials to ensure the identity belongs to that client/user. Authentication is a layer of defense and determines if the request is approved.  **Authorization:** This process determines which privilege is granted to the client/user of a particular credential and given a specified level of access. Clients/users must obtain an authorization before tasks requested are granted. Authorization is needed to grant access to the system and should be limited to the necessary level only for a limited time to prevent unauthorized access.  **Accounting:** This process is a form of logging or collecting information on resource utilization on a specific section or on all network access. Accounting is vital to security on any environment. Having a record of what has transpired is needed to ensure the appropriate work is always conducted on the system. |
| **8**  **Unit Testing** | Unit testing is a method used to focus on testing the functionality of an individual unit or component. In the next few slides, I show a few code examples of how unit testing works. On this slide you see the results of those tests. In the next few slides I will show you an example of the code written to represent the results shown on this slide. |
| **9**  **Unit Testing** | For this first slide, a test was created to verify that max size is greater than or equal to sizes. |
| **10**  **Unit Testing** | The second unit test is verifying that capacity is greater than or equal to the entries. |
| **11**  **Unit Testing** | This third unit test verifies whether resizing increases the collection. |
| **12**  **Unit Testing** | This last slide show verifies the code to determine whether resizing decreases the collection.  Again, referring to slide 8 will show the results of these unit tests. |
| **13**  **Automation** | The illustration shown is a diagram that represents the flow of the DevSecOps. DevSecOps is an IT culture where the responsibility for delivering secure software is shared between the development and operations teams. The diagram shows an automation that will be used to enforce and be compliant to the standards defined in the policy. In the next few slides, we will go over more details on DevSecOps. |
| **14**  **Tools** | First and foremost, DevSecOps, also known as “Development, Security, and Operations,” is an integration of security throughout the entire software development lifecycle. This includes the initial design to the testing and deployment of the software, and the integration and software delivery. The major advantages of DevSecOps are its security and speed. That being said, the team is more susceptible of producing better and more secure code with this integration and can be cost effective for the organization.  Is there any resources or tools that can help developers reduce vulnerabilities prior to deployment. Of course, and below you will see a few examples.  There are a variety of tools that the development team can use to help assist them test for any vulnerabilities. Choosing the correct one may be crucial to the environment so research is key. One that I used recently that I found useful is Cppcheck. Cppcheck is a static analysis tool that provides a unique code analysis that is designed to detect bugs, warnings, errors, etc. One of the best ways to deploy a secure environment is to make a plan and determine what the end goal is and continuously test. |
| **15**  **Risks & Benefits** | There are multiple reasons why an organization should be concerned of a data breach or at least have it on their radar. If you work with software, there is always risks. First, a developer should always consider the concepts of motives an attacker may have and be aware of common practices that should be avoided. The more software an organization uses, the higher risk of vulnerabilities may be imminent. One risk an organization may face during development is human error due to ill-training. Teams may struggle on implementation and following best practices due to lack of training. Just a simple search for data breaches will show you human error and lack of training, which could have been avoided.  An example of this is the Equifax data breach in march of 2017. Equifax was made aware of a vulnerability and the failed to patch the vulnerability which lead to a series of events that affected over 147 million people. The link provided will show a time line of the events.  A key element that should be incorporated during development is testing. My moto that I now have and learned my lesson early is to test early and test often. Don’t wait until the end to search for vulnerabilities. Although there are many benefits of testing early, there are risks as well. Testing throughout the development may extend the projects deadline, but on the same token, it may benefit the team by having a more secure environment.  If the development team waited, there may be compatibility issues for security protocols and could be more costly for the organization. One of the benefits is deploying the environment much quicker, which can potentially save money. That saying, “Time is money.” |
| **16**  **Recommendations** | There are a few notable recommendations that I would like to expand on.  Always use secure coding standards and implement them daily. Pay attention to any compiler warnings and errors. As mentioned above, test early and test often. Its vital to practice best coding standards and ensure the environment is secure. One way to help secure the network is through teamwork. Development teams and operation teams can work together to strengthen the software. Ensure to stay on top of updates for vulnerabilities and update the software to reflect the changes. Don’t wait to rectify the warnings. Simulate threats throughout the planning and building the projects to ensure vulnerabilities are reduced and or eliminated. |
| **17**  **Conclusion** | As technology advances and more software is developed, the industry must be pushed to incorporate high security measures and follow security policies and procedure. It is crucial that organizations and businesses incorporate the appropriate security training and best practices. There are too many threat actors wanting to exploit a database and there are many reasons and or motives they attempt to access them. No software is safe, but as a developer, you can prevent unauthorized access by using tools and security strategies to find vulnerabilities and rectify them before the database is breached. As mentioned in the previous slide, work together, rectify any vulnerabilities, and continuously simulate threats through penetration testing. It is not only the responsibility of the development and operations teams to design and develop secure code, but it is also the responsibility of the organization. |
| **18**  **References** | Mentioned throughout the presentation are some links to reference the material. This slide gives you further information so you to conduct research on your own. This slide shows the references used. |