**Project Two: Security Policy Presentation**

Jehu A. Domenech Ortiz

Department of Computer Sciences, Southern New Hampshire University

CS-405-X6166 Secure Coding 23EW6

Prof. Olga Mill

August 12, 2023

YouTube Link: https://youtu.be/7b\_mAi5h6vY

Green Pace

Security Policy Presentation

Developer: Jehu Domenech

OVERVIEW: DEFENSE IN DEPTH

With the security policy implemented we are defining core security principles which will ensure quality and standard. By using the Encryption and AAA framework we are looking to keep the standards for security.

THREATS MATRIX

Rules with likelihood of likely and probable where put in the likely section for unlikely it was put accordingly. When setting the priority I checked both the likelihood and severity in account and I set them accordingly. Medium severity with unlikely goes to low priority and medium severity and likely goes into priority.

10 PRINCIPLES

* Adopt a Secure Coding Standard
* Use Effective Quality Assurance Techniques
* Architect and Design for Security Policies
* Keep It Simple
* Default Deny
* Adhere to the Principle of Least Privilege
* Practice Defense in Depth
* Validate Input Data
* Sanitize Data Sent to Other Systems
* Heed Compiler Warnings

I went with the following order according to importance. With the order I am looking to build a solid foundation so that the following principles can be implemented easily.

CODING STANDARDS

* STR50-CPP
* MEM50-CPP
* EXP53-CPP
* ERR56-CPP
* CTR54-CPP
* STR53-CPP
* DCL58-CPP
* INT50-CPP
* INT50-CPP
* DCL52-CPP

For the order in the coding standards I decided to use the official priority levels used in the coding standards website. They use the likelihood and severity for setting the priority.

ENCRYPTION POLICIES

Encryption at rest looks to encrypt data when it is stored in the system and not moving from system to system.

Encryption at flight encrypts data when it is being sent over from one end to another end.

Encryption in use encrypts data when it is being used on the system.

TRIPLE-A POLICIES

Authentication looks to confirm that the person who is using the system is the person that is supposed to access it. This uses logins and passwords and other security measures that ensure that it is the person.

Authorization lets the user only access what they need to access and nothing else this prevents security vulnerabilities.

Accounting ties the account with each action they do so that they hold accountability for what they do in the system.

UNIT TESTING

This unit test shows various tests the OutOfRangeTest shows that the exception of std::out\_of\_range is thrown when the vector is at 20 by using myvector.at(20).

AUTOMATION SUMMARY

When automating by looking at the diagram below we would implement the testing in the verify and test portion of the pre-production section.

TOOLS

The DevSecOps pipeline is useful for programming while having security in mind. If the system that is being worked on relies on security a DevSecOps pipeline would be ideal (What is a DevSecOps pipeline?, 2022).

A useful external tool is Check Point CloudGuard it allows you to prevent threats and automize security. This tool would fall in the monitor and detect section of the DevSecOps.

RISKS AND BENEFITS

If security is implemented from the very beginning issues from security will be minimal although the initial set up will take a long time it is much more beneficial for the project.

RECOMMENDATIONS

I believe that my current security policy has a few gaps that would be better filled with other security policies. Although the policies I chose are useful they don’t have enough severity and likelihood for having them in the security policy.

CONCLUSIONS

Looking over the security policy various improvements could be made by having policies with a higher severity level and likeliness. If these are considered, then by automating the process security will be much better for the system.