# CS 405 Project Two Script

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Brief introduction |
| **2** | DiD must be maintained through constant implementation of secure principles, standards, and best practices. Policies that are uniformly defined, implemented, managed, and maintained throughout the lifecycle, require consistent reproach. Defense in Depth is a visual diagram of how security is maintained within a company's infrastructure from networking protocol to security for the cloud. This simple tool is to help guide policies and procedures. It applies to all staff that create, deploy, or support customer software, with respects to employees and where they fall within the illustration. On the bottom left, under the section titled policies, there are rules that govern and manage cybersecurity policies. |
| **3** | There are many ways to ensure code is rid of vulnerabilities. One of the many ways is to Dynamically test the application during production. This is important because new errors or issues may be found after the application has been published. This may include new hackers. |
| **4** | 1. Adopt a secure coding standard 2. Design security policies 3. Practice Defense in depth 4. Keep it simple 5. Use effective quality assurance 6. Validate User Input 7. Heed Compiler warnings 8. Default deny 9. Adhere to principles of the least privileged 10. Sanitize the data sent to another system |
| **5** | 1. Implement abstract datatypes using opaque types 2. Prevent SQL Injection 3. Use assertions 4. Use exceptions 5. Use Loops 6. Do not encode noncharacter data as a string 7. String correctness 8. Memory protection 9. Object Access |
| **6** | 1. Encryption in flight – The protection of data integrity during transit from location such as removeable drive to another form of storage. 2. Encryption at rest – Data encryption cannot be read properly without a decryption key, only the authorized users have access to the data. It can only be interpreted via decryption key. 3. Encryption in use – To ensure the security across all types of encryption throughout the DevSecOps lifecycle. |
| **7** | * Authentication – Verify a person is who they say they are. * Authorization – Give only the appropriate persons access to what is necessary to do the job * Accounting – Documentation of who accessed data, when it was accessed, using what commands, and the tasks needed to be complete will be recorded. |
| **8-10** | * There are several types of testing methods, automation testing, unit testing, and manual testing. Also, we have black, white, and grey box testing.   Developers are familiar with the internal functionality of the code base. The developer has the knowledge of the algorithms and programming languages that were used. A step further with white box testing would be to give developers the task of testing the codebase with tools such as exceptions, and error handling. If a developer knows the code already, he or she may be able to create simple try catch statements and call the functions they know are already there within the main entry point of the program.  The practice of developers not needing to know every internal function of the given program. These types of unit testing is used for dynamically testing application. The applications are put through different scenarios to see how they will perform. Although, it is not guaranteed they will be successful in every scenario. To take this a step further, a developer may test an application by running code on different platforms and devices such as a Linux tablet or the Apple MacBook M1 Pro. |
| **11** | Developers can use the white box and the black box approach at the same time to test code without knowing the internal workings of it all. This approach allows for a better experience when looking for coding defects. It has proved useful in some regards and can drain more time since one is two types of methods into one single method. One step further would be to use grey box testing at the very beginning of each iteration of the development lifecycle until the production phase. |
| **12** | There are many frameworks. How to implement them can be found here –  <https://learn.microsoft.com/en-us/visualstudio/test/writing-unit-tests-for-c-cpp?view=vs-2022>   1. Gtest 2. MSTest 3. Catch 4. CPP Unit 5. Boost.Test 6. DocTest 7. Lest 8. xunit++ 9. Mettle |
| **13** | The pipeline allows the Software Development Lifecycle to be streamlined through automation. During the stages of Build, verify and test, health check, monitor and detect are all using automated security tools. Using the Contrast security tool is widely used to monitor and detect defects within code and architecture. This is a stage of the lifecycle this can be implemented. |
| **14** | There are various tools developers can use within the lifecycle to fulfil each step of the lifecycle. The tools below are only a handful of the many security tools developers may use. All tools mentioned can be used throughout the entire lifecycle to create a more fluid DevSecOps process.   * + OWASP Zap – Web application defense.   + Gitlab – Protect against run time vulnerabilities.   + Codacy – Monitors the quality of a developer's code.   + Contrast security – Provides security checks by analyzing developer code |
| **15** | The benefits of choosing to act now will prevent more vulnerabilities and attacks in the long run. The risk to this is One might not be familiar with the type of cyber threat. This strategy is lacking. How are we supposed to prevent an attack from a source we know nothing about? The risks of acting now without gathering information and planning accordingly is more effective. The best course of action will be to use the DevSecOps lifecycle.  For example, using the first step of the illustration on slide 13 will give us time to prepare for what may come in terms of attack. To combat cyber threats, we must understand threats by looking into them first. Following this, we design around the threat. Next, we build the architecture, and the code needed to prevent the attack. |
| **17** | * Defense in depth is used to visually see how security is applied in different sectors of companies. We can see where policies fall into place to help keep companies secure. * The types of unit tests can be used by experienced developers and inexperienced developers. This is ideal for tasking the new developers on the team with debugging the code base without the need for them to know how to code was written. * Unit testing is a great way to ensure at the end of each phase, the code is functional, free of logical errors and vulnerabilities. Of course, testing should be done multiple times throughout every phase to ensure the code works. (write a little code, then debug). * Automation must be used wherever it fits within the DevSecOps lifecycle. This will drive workflow upward, giving companies the edge to create high quality, error free software on time. |