# CS 405 Project Two Script Template

Complete this template by replacing the bracketed text with the relevant information.

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Today will be explaining the company's Green Pace security policy guide and implementation guidelines and recommendations for maintaining it in the future. |
| **2** | Defense in Depth (DiD) is a cybersecurity method that layers a succession of defensive procedures to protect critical data and information. If one mechanism fails, another immediately steps in to prevent an attack. This multi-layered strategy with intended redundancy improves overall system security and handles a wide range of attack vectors. The model below shows its a multi-layered approach. |
| **3** | By looking at the threats matrix the likely section has a high severity of attacks and the priority section is priorities that have a high number and higher severity levels. The low priority has little exploitable vulnerability but should not be disregarded. For the unlikely section, the attacks are of low severity, but should also not be ignored. Everything should be prioritized from highest to lowest severity. |
| **4** | Validate input. Validate input from all untrusted data sources. Proper input validation can eliminate the vast majority of software vulnerabilities.  Heed compiler warnings. Compile code using the highest warning level available for your compiler and eliminate warnings by modifying the code. Use static and dynamic analysis tools to detect and eliminate additional security flaws.  Architect and design for security policies. Create a software architecture and design your software to implement and enforce security policies.  Keep it simple. Keep the design as simple and small as possible. Complex designs increase the likelihood that errors will be made in their implementation, configuration, and use.  Default deny. Base access decisions on permission rather than exclusion. This means that, by default, access is denied and the protection scheme identifies conditions under which access is permitted .  Adhere to the principle of least privilege. Every process should execute with the the least set of privileges necessary to complete the job. Any elevated permission should only be accessed for the least amount of time required to complete the privileged task. This approach reduces the opportunities an attacker has to execute arbitrary code with elevated privileges].  Sanitize data sent to other systems. Sanitize all data passed to complex subsystems such as command shells, relational databases, and commercial off-the-shelf (COTS) components. Attackers may be able to invoke unused functionality in these components through the use of SQL, command, or other injection attacks.  Practice defense in depth. Manage risk with multiple defensive strategies, so that if one layer of defense turns out to be inadequate, another layer of defense can prevent a security flaw from becoming an exploitable vulnerability and/or limit the consequences of a successful exploit.  Use effective quality assurance techniques. Good quality assurance techniques can be effective in identifying and eliminating vulnerabilities. Fuzz testing, penetration testing, and source code audits should all be incorporated as part of an effective quality assurance program.  Adopt a secure coding standard. Develop and/or apply a secure coding standard for your target development language and platform. |
| **5** | Ensure that operations on signed integers do not result in overflow.  Ensure that unsigned integer operations do not wrap  Do not attempt to modify string literals  Sanitize data passed to complex subsystems  Do not access freed memory  Incorporate diagnostic tests using assertions  Guarantee exception safety  Do not use pointer-to-member operators to access nonexistent members  Functions should validate their parameters  Do not access an object outside of its lifetime |
| **6** | Encryption in rest - By guaranteeing that the data is encrypted when it is on disk, encryption in rest is intended to stop the attacker from obtaining the unencrypted data. You are essentially transforming your customers' sensitive data into another type of data when you encrypt data while it is at rest. This often occurs using an algorithm that a user without access to the encryption key to decode it cannot comprehend. Your data will remain secure because these files will only be accessible by authorized employees.  Encryption at flight - The encryption of data in flight refers to data transmission through a network. This is crucial for people that use the open internet to transport data, which is a component of the majority of public cloud systems. The underlying FASP protocol employs the Cipher Feedback with Checksum technique to guarantee the security and integrity of data in flight each time you send content using an Aspera web app.  Encryption in use - Data is deemed to be in use when it is currently accessible and utilised. Data security must be taken care of before actual data use can start since data must be decrypted in order to be put to use. Secure Encrypted Virtualization is a technique for data encryption in use (SEV). It uses specialized hardware, an AMD EPYC processor, and an AES-128 encryption engine to encrypt RAM memory. Although this field is still relatively young, other hardware suppliers are also providing memory encryption for data in use. |
| **7** | Authentication - Before access to a network device is given, authentication verifies identification. Based on the username and password combination given by the entity attempting to obtain access, it is the process of confirming the identity of the person or device accessing a network device.  Authorization - Access control is provided by authorization. It is the process of putting together a group of characteristics that specify what the user is permitted to do. Authorization often takes place after authentication, when your identity has been verified, AAA security authorization puts together the collection of attributes that characterize what you are permitted to do.  Accounting - Accounting offers a way to gather data, record it locally on a network device, and transfer it to a AAA server for billing, auditing, and reporting. The accounting function keeps track of each management session that is used to grant access and keeps a record of it. |
| **8** | When debugging the code with the edited changes it displayed the information just like the original code in the Google Test Guide. All the tests passed except one which was the AlwaysFail test. Below is the image of the unit test and when I debugged the test using the local window's debugger. Unit testing is primarily concerned with testing and determining if written code is working as expected. As part of the software development and maintenance process, testing and debugging are crucial activities. |
| **9** | This test verifies that resizing decreases the collection to zero. |
| **10** | This test verifies that the out\_of\_range exception is thrown when calling at() with an index out of bounds. |
| **11** | This test verifies that reserve increases the capacity but not the size of the collection. |
| **12** | Automation in software development is a method of reducing errors during the development process, making it faster and less expensive, and improving team collaboration and productivity. Test automation is essential for software engineering teams responsible for complicated systems. When adding new improvements or features, regression (breaking existing functionality) is not unusual. Automated tests make it feasible to detect regression quickly. Aside from regression detecting, automated tests provide wider test coverage and can perform more testing than even an army of human testers. Testing systems with highly complicated concurrent test scenarios and enormous data sets is fast becoming the only option. |
| **13** | A DevSecOps pipeline, which is a CICD pipeline with integrated security practices and tooling, extends the software development lifecycle with practices and services such as scanning, threat intelligence, policy enforcement, static analysis, and compliance validation (SDLC).  Tools For Automation  Astrée - is a static code analyzer that proves the absence of run­time errors and invalid con­current behavior in safety-critical software written or gen­er­ated in C or C++.  Parasoft C/C++test - is an integrated set of tools for testing C and C++ source code that software developers use to analyze, test, find defects, and measure the quality and security of their applications.  Cppcheck - is a static analysis tool for C/C++ code. It provides unique code analysis to detect bugs and focuses on detecting undefined behaviour and dangerous coding constructs. The goal is to have very few false positives. |
| **14** | Risks of Insecure Software - An insecure application lets hackers in. They can take direct control of a device or provide an access path to another device. This can result in:  Denial of service to a single user.  Compromised secrets.  Loss of service.  Damage to the systems of thousands of users.  Loss of life.  Adopting secure coding methods is critical since it eliminates commonly exploited software vulnerabilities and avoids intrusions. Furthermore, optimizing for security from the beginning helps decrease long-term expenses that may develop if an exploit leads in the release of critical user information. |
| **15** | Gaps  Being unprepared to deal with cyber-attacks. Plan ahead on how to deal with attacks.  Address any gaps as soon as possible. Don’t leave any gaps and wait in the last minute to deal with it.  Mitigate gaps and implement a plan.  Different cyber-threats. There will always be new cyberattacks, and we must have that in mind and implement and plan ways to combat them.  Recommendations  Follow the 10 principles of secure coding practices  Implement the Defense in Depth model  Implement the Triple-A security  Test software continuously  CT's purpose is to assess software quality throughout the SDLC, providing vital feedback earlier and enabling higher-quality, faster delivery. |
| **16** | In conclusion, because a malware attack can do severe damage to any piece of software while jeopardizing integrity, authentication, and availability, software security is vital. Damage can be avoided if programmers consider this throughout the development stage rather than subsequently. We should implement the 10 principles of secure coding practices. Implement the Defense in Depth model, encryption policy, and triple-A policy. Regular secure code reviews, together with automated tools that check your code for certain vulnerabilities, can assist prevent such attacks. |