# CS 405 Project Two Script Template

**Sarah Sandstrom**

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Sarah Sandstrom, this is my Security Policy Presentation. Today I will be discussing my outline for Security Policy. |
| **2** | To keep our servers secure we need to make sure our users are the only one able to access them. In Supplier cloud, we ensure privacy level agreements and service level agreements are in place. Without our own private cloud, establish a baseline of deny-by-default configurations and defense in depth through tried-and-true encryption mechanisms, as well as enabling Firewall and privileged account management (PAM), and Monitoring for unauthorized changes and access through logging. Vulnerability scans, patch management techniques, and updates of vulnerable APIs can be integrated into the CI/CD pipeline and automated. By protecting our host security, we protect the critical assets of the system and data that attackers seek. Lastly, training of the team on the layered defense compliance protocols is critical. |
| **3** | Threat Matrix:  In the red we see the high threat level and the probability being likely or probable to occur  In orange we see the medium threat level and the probability being likely, probable, or unlikely to occur  In yellow we see the low threat level and the probability being probable or unlikely |
| **4** | Validate Input Data (STD-001, STD-002, STD-003, STD-004, STD-010),  Head Compiler Warnings (STD-001, STD-002, STD-005, STD-009),  Architect and Design for Security Policies (STD-004, STD-007),  Keep it Simple (STD-001),  Default Deny,  Adhere to the Principle of Least Privilege,  Sanitize Data Sent to Other Systems (STD-003, STD-004),  Practice Defense in Depth (STD-001, STD-002, STD-003, STD-008),  Use of Effective Quality Assurance Techniques (STD-005, STD-006, STD-007, STD-009, STD-010),  Adopt a Secure Coding Standard (STD-008) |
| **5** | Chart explaining the rule number, severity, likelihood, remediation Cost, Priority, and level |
| **6** | Encryption in Rest: How data is stored and held on servers in repositories   * Examples of how we will implement this would be file-level encryption, database encryption, full disk encryption and Protection with Digital rights management. This applies to the Defense in Depth principles as this is effectively a last line of defense against Data Breaches. If an attacker managed to get through every level of defense in the system, it is still entirely plausible that the data they recover would be encrypted and would be indecipherable to them.   Encryption in Flight: How the data moves from one computer or device to another. There is always a potential that this data could be intercept by another person, especially in the case of public networks. This data could contain sensitive details about the user, a business, security credentials, or a host of other information. As such, all this data needs to be fully encrypted and preferable sent in a way that is either impossible or unlikely to be decipherable.   * For example, if credentials for a login were sent without this encryption on a public network, the information if intercepted could then be directly used by the interceptor.   Encryption in Use: When the data is currently being used. This policy is enforced by Identity and roll-based management. Here a user is verified and authenticated (often with two factor authentication) to ensure the accessor of the data is who they say they are.   * Additionally, user information is checked against known information, the user lives in Cleveland Ohio and uses an IP address that reflects Shanghai China, for the first time, it is likely that this user would be challenged and authenticated through other means, maybe this user would require a VPN to access the data. * Finally, a user will only have access to data the user is allowed to have access to. This user would be able to view data, or if they could it would remain encrypted if the user attempts to access it. This means that even if a user has complete control over the file (such as a downloaded copy) this user will only be able to see what is decrypted for them, even if additional data is contained, that data will not be completely or even partially accessible without the appropriate user credentials. |
| **7** | Authentication is the process of identifying and verifying a user is who they say they are. This will be handled with user credentials and two factor authentication. Additionally, if user data (location, browser configuration, etc.) are not the expected values then a user may be challenged to provide more verification such as answering secret questions. If these credentials are verified a user will be granted access, if not the user will not be granted access.  Authorization is the process of ensuring a user can do a specified task. After authentication a user will have a set of authorizations such as the ability to view certain documents, write to portions of a database, grant, or remove user privileges, and access specific directories. Each time an action is taken (such as opening a file or making a change to user roles) authorization come into play as the system will ensure that the user is allowed to do the task. Without authorization, any user can access anything, but with it, a user will only be able to access files and perform tasks within the scope of their authorization.  Accounting is the practice of logging user statistics and data. After a user is authenticated and authorized to do something, it is important to note and track what the user is doing. It may be usual for a user to download several files, upload one and leave the system, but is it usual for a user to use up 80% of total system resources, or to download thousands of files? Accounting is done not only to keep track of modifications and usage statistic but to monitor and detect potential intrusions or attacks. A user that is operating in an unusual manner may warrant an investigation into activities, from there it may be determined that account credential was stolen or that a user is attempting to do something nefarious. |
| **8** | Verifying the actual size of the vector is less than maximum allowed vector size. Two google test functions that are used in this example are EXPECT\_TRUE and EXPECT\_EQ you can see the use of the red underline |
| **9** | Verifying that an exception is thrown when know error occurs.  You can see in the example where I underline where the exception is thrown in the code. |
| **10** | Explain the chart of automation summary |
| **11** | Automation can be created upon Build, by automating manual processes such as compilation and static code checking into a CI/CD pipeline, encourages team workflow, can use Docker for container instances, GitLab for versioning, Jenkins for CI.    Verify and test in SecOps by automating virtualized container deployment. Implement automated security tests and regression tests in QA.  Monitor and detection. Automate static application security tests into nightly builds on key sections of code. Embed dynamic application security testing into SDLC to look for vulnerabilities in real time.  Utilize tools such as OWASP Dependency-Check to check code dependency vulnerabilities.  Notable Tools: ClangTidy, Cppchecker, Parasoft, Coverity, Jenkins, Gitlab, Docker |
| **12** | Fix your bugs. This is the biggest risk for your company. It costs you 6 times more money to fix a bug during implementation phase than fixing it in the design phase.  It can cost you 15 times more if you ignore the problem furthermore and wait until the testing phase than fixing it in the design phase. |
| **13** | Add in more standards. This is just the groundwork but start building more and standards to become even more secure. Also hire a white hat consulting cyber security firm to find and vulnerabilities. This will cost you money but it is better to spend money on making sure you system is secure instead of being hacked. |
| **14** | Start taking the extra step to protect your code from the start. Make sure you fix any issues with you code early on otherwise you will be spending more time and money fixing them later.  Implement defense in depth and incorporate daily testing routines into everyday practices. |