# CS 405 Project Two Script Template

**Jacob Griffin**

**15 December 2024**

**Project Two: Security Policy Presentation**

[**https://youtu.be/USYvsH3t9EU**](https://youtu.be/USYvsH3t9EU)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone! My name is Jacob Griffin and I will be doing the Security Policy Presentation for Green Pace today. |
| **2** | Defense in Depth is widely used security strategy that utilizes multiple layers of protection to provide a defense against cyber-attacks. The reasoning behind the multiple layers is so that if one layer is breached, there is another layer of protection to protect the data. Examples you as users, use daily, is 2FA, and some behind the scenes examples are endpoint detection and response. |
| **3** | Moving onto the Threats Matrix I have provided real examples from the policy letter based on vulnerabilities that are likely to be exposed and exploited, Priorities that need to be addressed, Low priority risk, and threats that are unlikely to be exploited. As you can see, threats that are likely to occur also are of high severity, making them a priority to be addressed. Threats that are low priority are not of a high severity and are unlikely to be exploited by an attacker. |
| **4** | When discussing security principles, there are 10 that we will be discussing today. The first principle we will be discussing is validate input data. We must always ensure that data entering our system is validated and meets the specifications and rules of the application. When input is validated we can prevent SQL injections, malicious malware etc. Next is Heed compiler warnings, we as developers want to compile our code using the highest warning level possible, and this will ensure that no vulnerabilities are hidden or not being addressed. Architect and design for security policies will be the next principle. The architect and design needs to ensure our software is implementing the security policies for all user levels, and we will ensure that users do not have a higher level of access than intended. Keep it simple is another principle that is very important, which means we want to keep our software as least complex as possible as this will help create easy maintainable and readable code. Next is default deny, default deny is where a user is initially denied access if they have not been validated. Default deny will ensure un authorized users do not gain access. Adhere to the principle of least privilege is where when processes are being executed they should function at the lowest level of priorities possible. Sanitize data sent to other systems is a principle where we want to ensure that all data being sent is clean and sanitized before being processed to avoid potential vulnerabilities. Practice defense in depth is where we want to practice implementing a layers of security so that we do not have single points of failures. Use effective Quality assurance techniques is principle number nine and this is where the code/software needs to be audited periodically to expose vulnerabilities and unauthorized use of the system. Principle 10 is adopt a secure coding standard, the team needs to have a set standard to follow so that security is practiced through the entirety of the development lifecycle. |
| **5** | Coding standards is where I will discuss the prioritization I have established for the standards and why. To begin with I placed Prevent SQL Injection as my number one because it is the easiest and most simple way to exploit a vulnerability, but can pose a great risk if exploited. Number 2 is Exclude user input from the format strings, this is where an attacker can manipulate strings to gain access to the data and the rest of the system, and if user input is not excluded, it as an easy target for hackers. Number 3 is Do Not Access an Object Outside of Its Lifetime, this is an easy error for us as developers to make but can cause a lot of complications. A hacker can exploit these and leak data over into other memory allocations and corrupt the data. Number 4 is never qualify a reference type with a const or volatile, as this can lead to unintended behavior and cause the crash of the system. Number 5 is use valid references, pointers, and iterators to reference elements of a basic string, this is extremely important because using an invalid reference leaves a vulnerability and an attacker can manipulate those references. Memory protection is number 6, Do not access freed memory, this memory is allocated and should not be accessed as this is what causes data leaks and the misuse of memory. Number 7 is assertions, Use a static assertion to test the value of a constant expression, this is what will help validate that are conditions are true and can help detect bugs early on in the development stage. Number 8 is do not return from a computational exception signal handler because this can interrupt the flow of the normal program, but should be easily detected in the production phase.  Number 9 is ensure that unsigned integer operations do not wrap, We do not want to exceed the maximum value allowed for the unsigned integers. This works together with the assignment from week 2 and buffer overflow. This can cause a security vulnerability and be exploited by hackers. Lastly, number 10 is Do not subtract or compare two pointers that do not refer to the same array, If the two pointers are not referring to the same array it can cause logical errors. The pointers must be part of the same array to prevent these errors. |
| **6** | [Encryption in Flight is Used when data is being transmitted over a network. Extremely useful when transferring data over the open internet, like public clouds. This data is very important when it comes to being protected as a lot of times it is ran through third party services.  Encryption at Rest is Used to protect data that is stored on drivers such as HDD, SDD the cloud, or any virtual storage. This data is encrypted so that if it was accessed it would be unreadable to the attacker.  Encryption in Use is Used when data is being actively used or input into the system. This is when data or networks are at risk for attacks such as SQL Injections, buffer overflows, etc. |
| **7** | The Triple A framework is a framework a set of policies that ensures we are following a set of guidelines that will prevent simple attacks. The first policy is Authentication and this Ensures that the user, system, or outside source trying to access the network is allowed. This is applied with usernames/password, 2FA, biometrics, etc. It is applied because it is a layer of security that ensure unauthorized sources cannot access the network. The policy applies to our specific scenario because it will prevent unauthorized access to accounts and maintain the integrity of the user’s data, and the addition of new users. The second policy is Authorization, This is what determines the type of access a user/account has on the network or system. The way this portion of the framework works is verifying the user’s permission once they have logged in. Each user is stored into a database with specific privileges, and these are accessed once a user logs in. For example, an admin has different permissions than a normal user. This applies to our company and applications because the rules of least privilege should be applied so that a user cannot have access to sensitive data that they should not have access to. The last policy of the framework is Account and this is where Accounting is used for keeping a record of user’s activities so that they can be used in audits and for reporting purposes. The way that is works is it logs activities that are performed by users such as login, changes to the database, files accessed by users, settings changes, etc. |
| **8** | Unit testing is a good way to isolate functions and test them and ensure they are behaving and functioning the way they are supposed to. For our situation I utilized google test within Visual studio and ensured the console gave a pass for each of the test ran. Our first test was seeing the collection could increase in size and this is a Positive test. This is a positive test because we are ensuring the function is properly working as intended. As you can see I have provided the code as well as the console output to ensure it worked correctly. |
| **9** | The next test is Can the collection decrease to a negative integer and this is a negative test. This is a negative test because we want to ensure that there is an exception thrown and that it passes by throwing an exception and an error. As we can see, it threw the exception and passed. |
| **10** | The third test is ensuring a value can be added an empty collection and this is a positive test because this is something we want the user to be able to do. |
| **11** | The 4th test is a negative test and it is ensuring that the value added can be found. This is a negative test because we will throw an exception if this function cannot be performed. |
| **12** | Automation is an amazing way to ensure that security remains at the top of the priority list when discussing the DevSecOps pipeline and the development lifecycle. As we look at the diagram there is not a single stage where automation should not exist. |
| **13** | The DevSecOps pipeline is where the workflow is maintained by automation. The reason the pipeline is extremely useful is because it ensures that security is integrated into every phase of the development lifecycle. Automation will be used in all stages of the DevSecOps pipeline to ensure security maintains a top priority in the development lifecycle. Some automation tools we can use are OWASP, which will check for vulnerabilities within dependencies. CodeSonar can be sued in the build phase and will scan the code for vulnerabilities and exploits that a hacker may use. Intruder is used in the health stage and is usually conducted with a white hat Penetration testing team. Splunk will be used in the maintain and stabilize stage to monitor for bugs and vulnerabilities. |
| **14** | When it comes to security and sensitive data, there is no time to sit back and wait to implement these security strategies. Security protocol and measures must be implemented from the first stage to the end phase. If we wait until the end to implement these measures, it can be too late and attacks can be underway, compromising users as well as sensitive data |
| **15** | Automation needs to be implemented so that security protocol and measures are being taken through every stage of the DevSecOps pipeline, this will allow security to be at the forefront of the production, instead of waiting until the very end.  There needs to be security standards that all teams involved in the production of the application are adhering to and following  Third party systems need to be held to the standards of ensuring that their product and software is safe and not putting our data at risk as it passes through  Lastly, I would suggest periodically implementing audits so that improper use of the systems, unauthorized access for users, is identified and fixed |
| **16** | To come to conclusion we can need to implement Automation to ensure that security is remaining a priority throughout the pipeline. Security cannot be set aside nor can it wait until the end to be considered, when security is put off to the end is when users and organizations become at risk of attack. There also needs to be a set of Security Standards in place for members of the team to follow throughout the stages of development to ensure that everyone has a set of rules to abide by. Lastly, Green Pace needs to perform periodic audits to monitor activity and assess unusual activity, but a white hat team either being hired or brought in would also be of great help to identify any immediate risk and threats that need to be addressed. |