# CS 405 Project Two Script Template

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

Dominic Clapper

April 16, 2023

CS405

Assignment: CS 405 Project Two: Security Policy Presentation

Youtube Link: [(32) CS 405 Project Two Presentation Template - Finished - Dom Clapper - YouTube](https://www.youtube.com/watch?v=voKAAi1OhQg)

[](https://www.youtube.com/watch?v=voKAAi1OhQg)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone! My name is Dominic Clapper. Today, I will be explaining to you what our security policy is, how it is designed, and why we should be using it as much as possible. |
| **2** | Defense in Depth is an important concept to consider when working with software security. In general, Defense in Depth is the process of utilizing multiple levels of security measures for a system or parts of a system. The idea is that if one of the levels is compromised, then the others still present will defend the system as a backup. With how complex cyber security threats are becoming nowadays, it is crucial to develop with the concept of Defense in Depth in mind. The chart on this slide presents a glimpse of what Defense in Depth covers in the range of security. |
| **3** | Moving on from the discussion of Defense in Depth, the following is a table that represents what threats are discussed in our policy, as well as their likelihood and priority level. Some of the threats mentioned are of high priority, while others are not as urgent. And some of our threats are more likely to occur than others, which is important to consider when developing software. The big purpose behind this table is to show you all that the threats shown in our policy are complex and require a good bit of consideration before working on them. |
| **4** | As I was just mentioning, our security policy has quite a few threats that are present in it, which are related to the principles that also appear on the policy. In total, there are 10 principles that are on the security policy. They are the following: Validate Input Data, Heed Compiler Warnings, Architect and Design for Security Policies, Keep It Simple, Default Deny, Adhere to the Principle of Least Privilege, Sanitize Data Sent to Other Systems, Practice Defense in Depth, Use Effective Quality Assurance Techniques, and Adopt a Secure Coding Standard. These security coding principles act as a guide for developers during the development process on how to secure their code. |
| **5** | Following the 10 security principles are 10 coding standards that are present in the security policy. Each of these coding standards represents a rule that our developers should follow as they write code for projects. These standards are listed to help our developers understand how to code in a secure way, and why it makes our code more secure for users. The following coding standards listed in the security policy are:   1. STD-001-CPP: Obey the one-definition rule which basically means to have an object or function only have one definition for an entire piece of software. 2. STD-002-CCC: Ensure that operations on signed integers do not cause an overflow. 3. STD-003-CCC: Make sure string variables do not overflow due to improper input and buffer overflow. 4. STD-004-JAV: Prevent SQL Injection for any type of input. 5. STD-005-CCC: Free dynamically allocated memory when no longer needed. 6. STD-006-CCC: Use static assertions to test the values of constant expressions. 7. STD-007-CPP: Guarantee exception safety. 8. STD-008-CPP: Use valid references, pointers, and iterators to reference elements of a container. 9. STD-009-CPP: Detect and handle memory allocation errors. 10. STD-010-CPP: Handle all exceptions. |
| **6** | The next topic we have here is encryption. Encryption is utilized in many areas of the IT industry. It is very useful for anyone trying to protect data in its different forms. Encryption is typically the process of using an algorithm and key to encrypt or scramble data so that it is unreadable without using the key. This process makes obtaining useful info from stolen data much harder. As the slide shows, there are three different states or forms of data that matter to our situation involving encryption. These three states are Data in Flight/Transit, Data at Rest, and Data in use. Each of these three states of data are different from one another and utilize different ways of encryption.  The first one, Data in Flight utilizes encryption in an interesting way. Typically, data in flight is encrypted by traveling through secure channels that have encryption employed in them. A good example of a secure channel that is encrypted is TSL or SSL. Both are well tested and very beneficial for encrypting data in transit.  The second one, Data at Rest utilizes encryption in the typical way. Basically, the data is run through an algorithm, where it is scrambled and made impossible to read while it is stored. There are a lot of different encryption algorithm ciphers that are useable for software security, so it is important to do proper research and find a suitable algorithm.  Data in use utilizes encryption in probably the most unique way. The memory that is being used to hold data is actually encrypted to protect it from being accessed by attackers.  Overall, encryption is an important part of any software application that stores, transfers, or uses data. It supports DiD by providing an extra layer of defense for software. |
| **7** | Beyond encryption, there are also other tactics that we can apply to many software projects that will bolster security. These other tactics fall under the concept up the Triple-A policies. These three policies are Authentication, Authorization, and Accounting. Each of these policies represents a type of security level that can be implemented to make software more secure for application owners and users.  The first A stands for Authentication, which is the process of identifying the users of a system and ensuring they are who they say they are. Authentication can come in handy in many software applications because it acts as a first line of defense. You have probably seen authentication before in the form of logging in via username and password or even as a fingerprint scan on a phone. This first line of defense will help in many scenarios where we need to limit who can access the system.  The second A stands for Authorization, which is the action of allocating permissions to users based upon who they are in relation to the system. Authorization typically comes after the Authentication in an application, so they go hand in hand most of the time. In many systems, authorization is dealt with by using a role-based access system, which divides permissions based upon the roles assigned to users. And, usually, users are only given the most basic permissions they need to accomplish the functions they need to.  The last A stands for Accounting, or the process of recording how data and functions are accessed in an application or system. Think of this as a form of auditing the system to determine who, what, when, and why in relation to data inside a system. Typically, the concept of accounting is useful in future situations where audit logs are needed to track bugs or transactions. There are many ways we can keep track of our logs, so we should try to utilize at least more than one, so we have backups. |
| **8** | In the next few slides, I will be showing you some examples of unit testing and how it can be performed in our development cycles. Testing is a major part of the development cycle as it plays a large role in ensuring our code is secure and operates correctly.  The first Unit Test I have here is from a sample of Unit Testing I performed a while back. This test is used to determine if an error is thrown whenever an attempt is made to access a vector item after it has been popped from the vector. The test is successful because it proves that an error is thrown when it needs to be thrown. This test utilizes the EXPECT\_THROW command to determine if an exception is thrown. |
| **9** | The next unit test that we see here is from the same set of sample tests I ran quite a while ago. This test checks to see if a resize command works in changing the size of the vector. The test utilizes the ASSERT\_TRUE command to compare the vector size with the initial vector size. You can apply this testing method to many different pieces of code because what it is doing is trying to test that a comparison is true. As for how this test can be improved, the main way I feel it should be improved is by making the test more complex with more steps of adding and resizing the vector. Now, onto the next test! |
| **10** | This unit test, or test number 3 is similar to the last test, but it utilizes a different ASSERT statement. The test checks to see if a clear command erases all of the contents stored inside of a vector. Instead of using an ASSERT\_TRUE statement here, I used an ASSERT\_EQ statement to check and see if two values were equal. This type of ASSERT statement can be used in many different scenarios involving two values that you want to be equal. This test can also be improved in many ways, but the one I feel is clear is that the vector can be tested more than once by clearing the vector, adding more values to it, and clearing it again. |
| **11** | This is the fourth unit test that I performed in the set of sample tests a while back. This test is a type of negative test that checks to see if a code fails. The test checks to see if an error is thrown when the code tries to access an item in a vector that is out of the bounds of the vector. This test utilizes the EXPECT\_THROW command used in test number 1. This test could definitely be improved by adding more complexity to it, aka, reserve more than once, and then try to access the item in the vector. |
| **12** | This test, aka test number 5, is the last test I will be showing you today. This test revolves around checking to see if the reserve command increases the capacity of a vector specifically. Similar to test number 2, this test also utilizes the ASSERT\_TRUE command to compare two values and see if they are the same. This test is a type of positive test where we are looking for a positive result. The one way I can see this test being improved is by using the reserve command more than once. Use it for the first time and check the capacity. Then use it again and check the capacity once more to see if consecutive reserves work correctly. And with that, we have finished going through all of the Unit tests I have prepared as examples to show you how they can be used to check the functionality and security of code. |
| **13** | Now that we are done with all of the unit testing examples and how they work, let's move on to something similar, but still very distinct. As you can see, all that is on this slide is a diagram depicting the DevOps life cycle. The tools that we have available to us fit into many spots present on this diagram. So, let's investigate that a bit.  The first place that automation should be implemented is within the Verify and test phase of pre-production. Unit testing, integration testing, penetration testing, etc... can all be automated in some way, shape, or form.  Unit testing should also be used and automated in the build phase of the DevOps process because it will facilitate the process of the applications development.  In the Monitor and Detect portion of the Production process, we must use a tool to automate the process of gathering audits on how the users are using the system. Then, in the Maintain and stabilize portion of the process, there needs to be more automation tools used to streamline the process of updating/maintenance the systems/applications.  Automation testing should also be utilized in the Transition and health check portion of the DevOps cycle to ensure our code works correctly and is secure. |
| **14** | This slide is somewhat of a continuation of the previous slide, except this slide goes into a bit more depth about the specific tools that are available for us to use in our DevOps process.  In terms of tools, the DevSecOps pipeline is a very useful tool in our arsenal. The DevSecOps pipeline is a process that makes sure security is being considered throughout the software development process. Utilizing the process of the DevSecOps pipeline makes it much easier and more efficient to manage the security in software projects. Beyond this, the DevSecOps pipeline process is important to implement as a tool because it helps with the optimization of security automation in the DevOps process.  Outside of DevSecOps, there are also many external tools that can be used to assist us in our quest for automation. It is our job to use these tools to their maximum potential in order to make our lives as developers easier. Some of the tools that I have found that could be very useful are CPPCheck, built in testing libraries, PolySpace Bug Finder, LDRA tool Suite, Google Testing Framework, and so many more. All of these tools can be of some help to us in our projects. They will help us finish work faster, and more precisely. |
| **15** | So, we are now getting closer to the closing arguments of this presentation. But first, we need to discuss the risks and benefits of whether we implement this security policy now, or later. There are risks and benefits to both sides. However, I feel that acting now, or ASAP is the best option because of the pros/benefits it offers versus waiting.  To start, let's look at the risks of acting now. If we were to act now and implement the security policy right away, it would be a hectic process. It would cost a lot of time and money to get the new security policy implemented with our employees because of all the training that would be required, all the tools necessary, and all the planning needed to make things work. Despite this, the benefits heavily outweigh the risks here because implementing the security policy now would make insane improvements to our product quality and user security. If we were to act now, the biggest benefit would be that we would save so much time not having to go back at the end of a development cycle to fix something within the software before release. This would also help save some money thanks to not having to waste more time at the end. Another major benefit of implementing the policy now is that we would save a ton of time on maintenance since our documentation and auditing would be vastly improved. Also, if we were to implement the policy now, the entire development cycle would become more structured in terms of tasks for each phase.  Now, moving on to if we were to wait to implement the security policy. If we did end up waiting to implement the policy, there would be quite a few risks. The first one that comes to mind is that the implementation of security will most likely be delayed for software projects. This could lead to some serious issues at the end of the development cycle that cost a lot of time and money to fix. The process of testing code will be put off until later on in the development cycle instead of being a continuous process starting as soon as code is written. This will make it much harder to fix errors as we make them. Both of these problems will increase the likelihood of a security breach occurring with an application. However, the main benefit of waiting is that the development life cycle will be completed faster. |
| **16** | At this point, we have discussed risks, tools to use, automation, unit testing, the triple-A policies, encryption, coding standards, and even coding principles. Now is pretty much the time to conclude things and discuss what we should do. First off, it is important to mention that there are some areas of security that the security policy does not cover completely. These areas can be filled in later, but for now, I will just address them briefly.  The first gap, or area of security not covered well in the security policy, is that not much is mentioned about using the principle of least privilege or the default deny concept. Both topics are discussed a little in the security policy, but they should be fleshed out more in the next version of the doc.  The next gap is that the security policy does not go into very specific detail about how automation is supposed to be used in the DevOps process. This is an oversight on my part because I did not do enough research on the tools that could be used by the system. As for the last gap that I feel is blatantly evident, is the lack of specifications for training models. There is no idea of how to train people to follow the rules of the policy, which makes it hard to believe in the fact that employees will follow the policy. All of these gaps should be filled in a future version of the security policy.  The one major recommendation I would make in the scenario of the security policy is that annual training should be implemented to make learning and understanding the new policy easier for employees. Also, providing an incentive for them to follow the policy is a good idea as well. It will help give some extra motivation to the employees if they know there is a proper incentive. |
| **17** | Beyond what I just mentioned about training and incentives for employees, I believe that there are other standards that should be included in future versions of the policy. The first being that we should help prevent future problems by making security a top priority during development. We also should keep ourselves updated with the ongoing security trends to ensure our systems and policy are not outdated with useless info in it. As for the third recommendation, I would like to recommend that Prioritizing a DevSecOps model is the best option currently because it helps to promote the idea of involved security. Lastly, I would recommend to never being afraid to ask others for help in the team. It is better to work together to accomplish something than do it without asking for help and suffering. |
| **18** | And with that, this presentation is now over! If you have any questions, please ask them now, or send them to my email. BTW, this slide just represents the references slide you would see in a typical APA format presentation. Thank you very much and have a good day! |