# CS 405 Project Two Script Template

# Quintin B. Rozelle

8/18/2024

7-2 Project Two

CS-405

<https://youtu.be/hheypXIGL6w>

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi. My name is Quintin Rozelle. I will be presenting a new security policy for Green Pace. |
| **2** | To begin, I would like to provide an overview.  Green Pace is moving from its DevOps model to a DevSecOps model with Defense in Depth as its focus. Auditors will be performing audits to ensure compliance. This security policy has been developed to assist with this audit.  In the same way that DevOps is the intersection of development, IT operations, and application delivery, DevSecOps is the interaction of DevOps and security. All are equally important and all affect and influence the other. As such, using a DevSecOps model requires that security be an equal focus from the beginning of the development process.  A common aspect of the DevSecOps model is the concept of Defense in Depth. This is the practice of using multiple layers of defense to prevent security issues. No one layer can prevent everything, while multiple layers have a better chance of blocking issues. |
| **3** | One element of this new security policy is that of coding standards. I will present a summary of those standards in just a moment, but before doing that, it is important to showcase the broad level of threats that these policies target.  This chart is the Threats Matrix. It helps to rank the threats targeted on a scale from low to high priority. The matrix is arranged so that the likelihood of the threat increases as we move up and the cost of dealing with those threats increases as we move right.  To help point out the severity of these threats, each box is color coded with red being the highest priority down to green as the lowest priority. Each standard is shown in the threat box that it targets. |
| **4** | In addition to these threats, there are 10 core principles taken into account in this security policy. I won’t read each one as they are presented on the screen but do notice that each of the 10 coding standards is also assigned to one or more security principles based on applicability.  You will notice that some principles have multiple applicable standards while some have none. This is not to imply that these principles are less important or don’t relate to coding in general. These principles are ones that are more broadly focused and are better applied to the code and project as a whole instead of just a coding standard. |
| **5** | These are the 10 coding standards. Again, I won’t read out each individually, but do note that they cover a wide range of coding areas and are intended to help standardize the code we produce while also helping to eliminate insecurities.  This table is arranged according to priority level (the last two columns) which is based on the severity, likelihood, and cost of the vulnerabilities they target. Those aspects can be seen in their respective columns. |
| **6** | The actual determination of the priority level of each standard is based on a slightly modified version of the determination that SEI CERT uses.  Severity, likelihood, and cost are all assigned a score from 1 to 3 based on their respective levels as seen in the first table. Do note that cost is assigned differently than severity and likelihood as lower costs to remediate should be targeted over higher costs since those produce the most bang for the buck.  Once those 3 values are determined, they are multiplied together. This resultant value is compared against the second table to determine the priority and level. A higher multiplied value results in a higher priority level. |
| **7** | The next element in the new security policy focuses on encryption. This provides guidance on encrypting data at rest, in flight, and in use.  First, encrypting data at rest is the practice of encrypting stored data. This helps to ensure that a hacker is unable to use any stored data even if they do steal it. For this policy, all stored data should be encrypted using a symmetric encryption scheme as this is a fast and easy way to encrypt data that doesn’t need to be decrypted by another party. Along with this, all keys use to encrypt this data must be stored separately from the data.  Next, encrypting data in flight is the practice of encrypting data while it is being transferred through a network. Since data transfer is an action that comes with a high risk of interception, encrypting it helps to ensure that it can’t be read. To implement this policy, data in flight should be encrypted with an asymmetric encryption scheme.  Lastly, encryption in use is the practice of encrypting data while it is being actively used. This further helps to limit the chances that data is stollen as historically, data had to be decrypted to be used. Advances in cryptography have made it possible to act upon data while still encrypted. To implement this, data should be encrypted with either the TFHE or Microsoft SEAL schemes. |
| **8** | The next element in the security policy focuses on Triple-A, that is authentication, authorization, and accounting.  Authentication is the process of determining that a user is who they claim to be when accessing our systems. This helps to ensure that only the individuals who should have access to our systems do. To accomplish this, logins must be required and should be unique and include minimum password requirements. Two-factor authentication can be considered if needed.  Authorization is the process of granting access rights to users once authenticated. This helps to ensure that users have access to only what they should and aren’t able to modify information they shouldn’t. To implement this, a well-defined access roles and rights scheme is needed. All users should be assigned to this appropriately and any new user should be given no access rights until they can be appropriately assigned.  Accounting is the process to tracking and reviewing user activity within the secured system. This allows for auditing and prevention of inappropriate access to various resources and can help to stop attacks in progress before they be more severe. To accomplish this, the secured system should log user activity which should be reviewed frequently. Automated tools can be used to expedite this process. |
| **9** | The next layer in the security policy revolves around unit testing. Unit tests function as a way to ensure that code functions as it is intended to on a granular level.  As an example of using unit tests, we will look at some demo unit tests to test correct functionality of code targeted by the second coding standard. As a reminder, the second standard is “Ensure that operations on unsigned integers do not wrap”. We will look at four positive tests and two negative tests. These tests will specifically look at various edge cases and non-edge cases. |
| **10** | The first positive test shows that this function (addIntegers) adds two zeros together correctly. As can be seen, 0 + 0 = 0 was expected and what was produced. This test passed. |
| **11** | The next positive test shows that this function adds a zero and the maximum unsigned int together correctly. It tested for UINT\_MAX being both the first and second arguments and expected UINT\_MAX as the result for both options. This test passed. |
| **12** | The third positive test shows that two non-edge case numbers are added together correctly. It looks at adding 100 + 100 and 1000 + 1000. In both cases, the test passed. |
| **13** | The last positive test looks at the case of adding two numbers together that add up to the UINT\_MAX. It looks at two options in which the arguments are swapped in the second case. For both cases, UINT\_MAX is expected to the be result, and both tests pass. |
| **14** | The first negative test looks at adding one to the UINT\_MAX. Since this would result in a number that is greater than the UINT\_MAX, it should be expected that wrapping would be detected, and an error should be thrown. As with similar tests, this is run with two different options swapping the arguments in the second case. In either case, the function throws an overflow\_error and the test passes. |
| **15** | The last negative test looks at adding two UINT\_MAXs together. Again, as expected, the function throws an overflow\_error and the test passes. |
| **16** | The final main element of this new security policy focuses on automation and the tools used to accomplish that. Shown is our current DevOps process updated to include security elements. Briefly, this shows the steps taken to move software through the development stage (incorporating planning, designing, building, and testing) and the production state (covering transition, monitoring, responding, and maintenance). At each stage, the respective elements of security can be seen. |
| **17** | The DevSecOps pipeline covers the process of software development when development, IT operations, application delivery, and security are combined. Security is as integral to this process as the other three, and as such must be factored into all steps from the beginning, not as an afterthought.  There are many tools that can be used throughout this process to help with security automation. Examples include static testing, continuous integration and development, and infrastructure security tools.  Static application security tools review the source code for potential security risks and is commonly used during the build and verify/test stages. Examples of these tools include Cppcheck and SonarQube.  Continuous integration and continuous deployment tools are tools that help to automate the software building, testing, and deployment steps. As such, they are frequently seen in the build, verify/test, and deployment stages. Examples of these tools include Jenkins and GitLab CI/CD.  Lastly, infrastructure security tools help to monitor for and prevent threats on deployed systems. A common security threat prevented by these tools is a DDOS attack. The are frequently used during the monitor/detect and respond stages. Examples of these include Cloudflare, and Wazuh. |
| **18** | Given the presented security policy, we are presented with two options: adopt it now or hold off an adopt it later. Both have benefits and neither are without risks and problems.  If we chose to adopt now, we do run the risk of increased development time (especially in the short term) as staff work to become comfortable with the new development process, requirements, and tools. This might be made more difficult as we work toward gaining staff buy in. Though, good training programs, advocates, and showcasing the benefits can help with this transition. Speaking of which, the benefits seen with adopting this now include a better focus on security early on leading to improved security overall, and less development rework in the long run as security issues are considered and prevented early.  If we chose to wait and adopt at a later point, we run the risk of having security risks in the code which can contribute to increased costs due to rework and being forced to patch those risks after release. The main benefit is maintaining the status quo for the staff which can help to ease the transition into a more security minded focus at some point in the future as they have more time to come to terms with the changes. |
| **19** | Based on these benefits and risks, I recommend adopting the policy now. Our current policy doesn’t adequately address the security threats seen in the field and there is no standard for coding, no use of unit or static testing, and no encryption or Triple-A policies. Adopting the proposed policy will help to close these gaps.  To provide an example of the urgent need for a robust security policy, I would like to briefly discuss Life360. They are the company that owns the Tile tracking devices and software. In June of this year, they experienced a security breach in which large amounts of private data were accessed. It was determined that the hacker gained access to the system by using a former employee’s login credentials. Utilizing the Triple-A standards in this security policy could have helped to prevent an attack like this as it would have removed access from that former employee on separation, would have audited for anomalous activity, and quickly addressed the security breach.  All of this being said, it is impossible for any security policy to be 100% perfect in all cases and this one is no exception. There are a few potential future gaps. First, the data encryption policies should be further developed by incorporating more specifics. Next, there are likely unknown gaps in the current proposal. When these are discovered, the policy must be amended to close these gaps. Lastly, the field of cybersecurity is everchanging. When new threats emerge, we must adapt to prevent those. |
| **20** | In conclusion, the proposed policy helps to address current security concerns by focusing on a defense in depth approach. This includes coding standards, encryption policies, Triple-A policies, and static and unit testing recommendations.  Continued development is still needed to address future changes in security trends, to train staff, audit for compliance, and explore additional security resources and tools. |
| **21** | Thank you for listening to my presentation. |