# CS 405 Project Two Script

Romain Tomlinson

August 14, 2024

[Recorded video](https://youtu.be/2KgBbschJWQ)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone and thank you for tuning in for this presentation that is about best practices in Secure coding and Security Policies. |
| **2** | In this presentation, we'll discuss the critical role of secure coding standards in building a strong foundation for secure software development. The defense-in-depth strategy, as shown in the diagram, further emphasizes this approach. By adopting this policy, we create a comprehensive framework for secure coding practices that integrates seamlessly into every aspect of our development processes. |
| **3** | Let's now examine our threat matrix, an essential tool for identifying and prioritizing potential security risks. The matrix organizes security threats into four key categories: Likely, Priority, Low Priority, and Unlikely. Each threat is assessed based on its likelihood and priority level, helping us direct our security efforts toward the most critical risks. |
| **4** | In this section, we'll explore the ten principles of secure coding. These 10 principles form the foundation of our security policy. Each principle is associated with specific coding standards, ensuring that our policy is both comprehensive and aligned with best practices. By adhering to these principles, we can effectively reduce the risk of security breaches and maintain a high standard of code quality. |
| **5** | Here, we will examine the ten standards that are the foundation of our secure coding practices. These standards offer essential guidelines and best practices for crafting secure and robust code. |
| **6** | Now, let's review our encryption policies, which are essential for safeguarding sensitive data against unauthorized access and maintaining its confidentiality and integrity. By enforcing these encryption policies, we can enhance our data protection strategies, reduce the risk of unauthorized breaches, and ensure adherence to regulatory standards. |
| **7** | Here, I will go over The Triple-A framework which are—Authentication, Authorization, and Accounting— this ensures that only authorized users have access to resources, and their actions are monitored and recorded for security and compliance purposes. This framework is critical for maintaining control over who accesses sensitive data and how that data is used. |
| **8** | Here we have our four Unit testing results, as you can see the testing was entirely successful.  Test 1: Is the Smart Pointer Not Null?  Test 2: Can the Collection Add a Single Element?  Test 3: Can the Collection Resize Appropriately?  Test 4: Does the Collection Handle Invalid Arguments Gracefully? |
| **9** | In our DevSecOps approach, security is seamlessly integrated into every stage of the development lifecycle, making it a shared responsibility across all teams.  We begin with **Assess and Plan**, where we analyze potential threats and prioritize our security tasks to address the most critical risks.  Next, in the **Design** phase, we implement security-driven design practices, guided by OWASP best practices, to ensure that security is considered right from the start.  During the **Build** phase, we focus on securing our builds by using trusted repositories and incorporating automated security checks. This helps us maintain the integrity of our code.  In the **Verify and Test** stage, we utilize tools like SAST, DAST, and IAST to thoroughly identify and address any vulnerabilities that may be present in the code.  Before moving to production, we conduct a **Transition and Health Check** where we apply necessary security settings and perform penetration testing to ensure that the application is secure.  Once in production, **Monitor and Detect** becomes critical. We continuously monitor the system using SIEM and IDS tools to detect any potential threats in real-time.  If an incident occurs, our **Respond** phase allows us to automate responses, such as rolling back to a previous state, if necessary, to minimize impact.  Finally, in the **Maintain and Stabilize** phase, we regularly assess our systems to ensure they remain secure over time, adjusting our strategies as needed.  To support these processes, we rely on a range of **External Tools**. For code and application testing, we use SAST and DAST tools like SonarQube and OWASP ZAP. For in-depth security validation, we perform penetration testing with Metasploit. Continuous monitoring and threat detection are handled by SIEM tools like Splunk, ensuring that we stay ahead of potential threats.  By integrating these practices and tools into our development lifecycle, we create a robust and resilient security posture that protects our systems and data from potential threats." |
| **10** | On this slide, we will explore the **Risks and Benefits** associated with our current security practices and the proposed solutions. |
| **11** | Here we will continue with our Risks and Benefits with our solutions |
| **12** | Here I will introduce the recommendations such as Gap Analysis and Standards to adopt. |
| **13** | In conclusion, the Green Pace Security policy provides a robust framework for securing our applications. By implementing the recommendations and closing the identified gaps, we can continue to improve our security posture and protect against emerging threats. |
| **14** | Here are the Sources used throughout my presentation. |