# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone, and welcome! My name is Cortland Burns, and I am excited to present to you the Green Pace Security Policy. A guide build to make our development stronger, safer, and more unified as our team grows! |
| **2** | At Green Pace, we have already been using some great secure practices. But as our team grows, it is important to make sure everyone moves forward together. This security policy takes what we have been doing and improves it with a defense-in-depth approach. Defense in depth in other terms just means building multiple layers of security so that if one layer is breached, others are still there, standing strong. This makes sure that we stay protected no matter what challenges come our way. |
| **3** | Here, we have the Threat Matric. Each Vulnerability was carefully looked over and ranked by how dangerous it could be, how likely it is to occur, and how much it would cost to fix. As you can see, SQL injection ranks as critical with both being at a very high likelihood and impact. Data overflow, out-of-bounds errors, and unsafe string operation are also high-priority areas. By finding these problems early, we can automate detection, focus on the fixes, and stay ahead of possible threats. |
| **4** | Each of the ten principles you see here directly connects to our coding standards. From validating input data to keeping designs simple, every principle improves our applications against potential vulnerabilities. This map shows that every coding standard we enforce has a clear, strong foundation making sure there is a seamless and secure development culture. |
| **5** | I ranked the standards based on three things. The severity of risk, the likelihood of being exploited, and how much it would cost to be fixed. SQL injection comes first because of how dangerous it can be and the risk of critical breaches. Then comes protecting string, values, and memory, which help prevent an avalanche of vulnerabilities later in production. Everything we build follows this secure focused path. |
| **6** | Encryption is our invisible shield at every turn we make. Whether data is moving, sitting still, or being processed we need to protect it. We use TLS 1.3 for transmission, AES-256 for storage, and advanced memory encryption for data in use. This way, even if attacked somehow get access to our data, it will stay locked and protected from their sight. |
| **7** | Authentication, Authorization, and Accounting are the essential three A’s. We verify identities, control what each role can access, and logs every major action. Together, they form a living, breathing record of security in action, making sure there is trust and traceability throughout the system. |
| **8** | Testing saves us every time. For example, when testing string operation, I created both safe and unsafe scenarios. Positive tests made sure there were safe copies and null termination. Negative tests caught unsafe buffer copying and overflow attempts. Unit testing helps us stop vulnerabilities before they can slip into production. |
| **9** | Automation is the butter to the bread. Tools like Cppcheck, Clang-Tidy, SonarQube, and Coverity continuously scan the code for vulnerabilities. They are integrated directly into the build process, catching problems before they can ever leave development. |
| **10** | Our security tools aren’t just add-ons; they live inside of the DevOps pipeline. From secure design, to building, to testing, and through production deployment and monitoring, security automation supports every single phase. This keeps our systems fast, strong, and secure by design. |
| **11** | Acting now improves our defenses and builds customer trust. Waiting would increase risks and could make future fixes much more expensive. Our current strategy is strong but enhancing runtime protections and fine-tuning real-time threat detection are two areas we will need to keep sharpening. |
| **12** | Looking ahead, our next opportunities include expanding memory exception during runtime, covering even more scenarios in automated testing, and increasing the range of auditing in order to track API more efficiently. By filling these gaps, we can prepare Green Pace for an even safer future. |
| **13** | Security is not just one small trip: it’s an entire adventure. By using continuous monitoring, exploring machine learning threat detection, and refreshing our secure coding training every year, we will stay one step ahead of the evolving threats around us. |
| **14** | These references helped guide and support the building of this power point and with some of the security practices we are putting in place. Following industry standards like SEI CERT and OWASP ensures that out policies are modern, proven, and aligned with the best practices worldwide. |