# CS 405 Project Two Script – Joseph Veneski -- April 21, 2024

<https://www.youtube.com/watch?v=EYzHPcZmK6g> – Link to presentation video

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Joe Veneski and today I will be presenting Green Pace’s Security Policy. |
| **2** | Through this security policy we hope to bring all teams into alignment and make sure everyone is on the same page. This will serve as a reference for developers at Green Pace, both new and old. Some of the topics covered include our use of Encryption and Triple-A framework, unit testing, and automating security. |
| **3** | The threats matrix gives everyone an idea of where our focus should be when it comes to secure coding. Threats that include SQL injection or buffer overflow vectors are considered high priority and likely. Things such as assertions and file operations are still a concern but a little lower priority. |
| **4** | Here we have the ten secure coding principles that we will follow at Green Pace. These principles are directly mapped to the coding standards and will help us create more robust and secure systems. |
| **5** | This is a breakdown of the ten coding standards listed in the policy. These standards include using smart pointers and safe string functions that protect memory and prevent overflows, ensuring functions return specified parameters, and iterators do not go out of range. |
| **6** | Data should be encrypted at all stages: at rest, in transit, and in use. Whether the data is stored in the server, being sent from the server to the client or back, or while it is currently being accessed, we want to protect it from attackers. Examples include the use of multi-factor authentication, Secure Hash Algorithms, and the Advanced Encryption Standard to secure our data. |
| **7** | The Triple-A Framework includes Authentication, Authorization, and Accounting. We want to know who is trying to access the system, what they are allowed to do, and keep notes of everything that happens. We will authenticate users and entities, limit their authorization to only what is essential to complete their task, and record it all for auditing review. This can help identify threats early and shut them down before major harm begins. |
| **8** | Our next few slides cover some examples of unit tests. Unit testing can be automated so that any new update can be vetted before touching production. The more comprehensive our unit testing is, the less headache we will have later when something goes wrong. The first unit test here simply ensures that popping a collection correctly decreases the size as expected. |
| **9** | We want to include negative testing as well to make sure nothing slips through the cracks, and we get as much coverage as possible. In this example, we attempt to set the capacity of a collection to negative which throws an exception just as we would expect. This can be great for testing exception cases per the coding standards. |
| **10** | The unit tests should be standalone and run independent of one another. They should verify everything works as it is planned to. This test ensures we are able to add to an empty collection. It doesn’t get more basic than this. |
| **11** | One more example of a negative test that verifies accessing an index out of range correctly throws an exception and doesn’t result in a memory access issue. |
| **12** | And final example, straightforward and clear in purpose. Verifying a collection correctly clears. |
| **13** | This graphic shows how automated security checks can be integrated into every phase of the SDLC. Setting up automated trackers to notify teams of new vulnerabilities or regulatory changes with frameworks and libraries they use. Automated static and dynamic runtime analysis and unit testing as we described earlier. |
| **14** | What tools can we use to accomplish all this automated testing? There are numerous tools out there such as CodeSonar, RuleChecker, SonarQube and plenty of others. |
| **15** | Why implement this whole security policy and is it more hassle than it is worth? It can be a lot of upfront set up requiring time and expertise, but it will more than make up for it in the long run. It is not a matter of if our system is attacked, but when and how. The MOVEit File Transfer cyber exploit affected 94 million users and cost over $15 billion in damages, not to mention the loss of customer trust. Once you lose trust, it is very difficult to come back. |
| **16** | There is always room for improvement and this standard should be updated regularly. Some additional opportunities include the use of third-party auditing and security review to identify anything that could be made more secure. We discussed some automated testing tools that could be utilized but a comprehensive list of approved tools, and frameworks to be used could also be of benefit. Additionally, any compliance and regulatory requirements that need to be met could be listed as well. |
| **17** | There are numerous threats out there that we need to be wary of. Security needs to be part of everything we do. Using a Defense in Depth strategy for layers of security won’t stop all threats but make it exponentially harder for any attackers. By building more secure systems we not only protect ourselves, but our clients as well. |