Amber Caldwell

CS 405

02/27/2022

Portfolio Reflection

One modern way to help achieve a more secure environment is using Zero Trust. Zero trust is a concept that is about proving your identity. This is an important change considering so many users are working from an area outside of their work office now. As a user, it should appear as though all the normal applications, even local ones, are on the cloud. This can be done by a reverse proxy. The reverse proxy communicates with single sign-on with the help of SAML. Additionally, there is more access control via the tunnel from the proxy. No connections can be accepted that did not originate from the proxy.

When considering how best to move forward with security, it’s best to adopt a secure coding standard and adhere to coding principles. Specifically, it’s important to adhere to industry coding standards such as CERT and OWASP. A defense in depth strategy is needed, as well as a secure coding standard, which is agreed upon. At minimum, the use of a Triple-A framework with data encryption should be in practice. The threat matrix made in the security policy highlights certain security standards which are most likely and least likely to occur as well as low and high priority areas. The most likely standards to occur are Exceptions, SQL Injection, Data Value, and String Correctness. Within the most likely to occur are two priority standards which are String Correctness and SQL Injection. The least likely to occur standards are Assertions, Data Type, File Management, and Function Declaration. However, the lowest priority standards are Assertions, Data Type, File Management, Function Declaration, and Integer Conversion. All these vulnerabilities arguably have their own risk. Emphasis should be placed on SQL injection. It is likely to occur, the severity is high, it has a high priority level, and the mitigation cost is not cheap. Another standard worth highlighting is string correctness. This vulnerability has a high severity, it is likely to occur, it has a high priority level, and it also more expensive to mitigate.

This brings back up the question of why shouldn’t we leave security to the end? It seems harmless if you just build the application and assume it’s working as it should. Right? This might be an acceptable approach for a student who’s learning to program and generally trying to see how the pieces of code work together. However, in the real world there can be consequences. One recent example took place in Washington state at the Department of Licensing. The private information including social security numbers, driver’s license numbers, and birth dates of around 650,000 people was stolen from hackers. This breach was detected January 24, 2022. The database is maintained by Salesforce, but it’s unclear exactly how the breach occurred, and the investigation is still on-going.

No matter what type of prevention is done to prevent breaches, it’s important to a defense in depth strategy. Therefore, if one layer of security fails the system is still protected. Additionally, in the case of our unit testing it helps to have additional effective quality assurance techniques in place. The unit tests are helpful, but penetration testing, audit reviews, or even outside testing could provide insight that was previously missed. In general, it’s a good idea to try and adhere as closely as possible to the security principles. They exist for protection and because they’re effective.