8-2 Journal: Portfolio Reflection- Tom Czubat

I feel that I have learned a lot in this course and have realized that there is a lot more to secure coding than I originally thought. When it comes to adoption of a secure coding standard, and not leaving security to the end, I have learned how that there are so many good resources to help you with this. I have learned that it is important to not leave security to the end and implement it in the beginning phases of the development process. This can have your security specialists give their input that can be incorporated into the project. If security principles are enforced at the beginning of development, it is easier to make the necessary changes to ensure security but if it is left to the end then the security suggestions of the experts might not be possible to implement without compete overhang the project which leads to more time and money needed. One of the most helpful resources in this course was the SEI CERT C++ Coding Standard.

Another important principle that I learned in this course was about analyzing risk and cost benefit of mitigation. This caused me to have an in-depth understanding of different vulnerabilities and how these can leave a system susceptible to a breach. Learning about the most common security practices was key. I learn that I needed to heed compiler warnings so that I can fix any upcoming issues with the code. I also learned about input validation and making sure that users used proper long and complicated passwords that can held protect against brute force attacks.

Learning about the Zero Trust policy was another concept that I found very interesting. It made me realize that a company should not automatically think that a device is safe just because it is in your building. This concept shows how building layers of security can help reduce your company’s risk. This policy also considers the modern workforce where it is more common to work from home (especially the last two years). This makes it much more important to have multifactor authentication. This policy also introduces device authentication that can identify the device you use as your work from home. This is another example of the layered approach to security where if one layer fails, there are others to fall back on to protect yor organization.

When recommending security policies, I have learned that it is important understand the most common types of attacks. Making sure that developer understand secure coding practices is crucial to protect your organization. Developers need to know how to implement good input validation and make sure that long passwords are used as well as strongly suggesting 2-factor authentication. Protecting against SQL inject is also important so having a program to constantly train developers and make sure they are up to date on common attacks is important. Exploring the use of external tools can help your developer find warnings and errors that otherwise may have gone unnoticed. Using the static code analyzer Cppcheck showed me how useful these tools can be to find more warnings and help the code be more performant and more secure.

**References**

*A Practical Guide to Zero-Trust Security*. (n.d.). Threatpost.com. <https://threatpost.com/practical-guide-zero-trust-security/151912/>

*SEI CERT C++ Coding Standard - SEI CERT C++ Coding Standard - Confluence*. (n.d.). Wiki.sei.cmu.edu. Retrieved October 17, 2021, from <https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=8804668>

corob-msft. (n.d.). *Security Best Practices for C++*. Docs.microsoft.com. Retrieved October 17, 2021, from https://docs.microsoft.com/en-us/cpp/security/security-best-practices-for-cpp?view=msvc-160&viewFallbackFrom=vs-2019

‌

‌

‌