6-1 Journal: Don’t Leave Security to the End

Nathaniel Madore

The term “Don’t Leave Security to the End” has a meaning rooted in the fundamentals of secure coding best practices. Security as an afterthought of a project is a costly mistake because the further development moves along without use of secure coding practices means the wider the gaps in the code. Although this data has not been tested, in theory using secure coding practices reduce the number of mistakes that crop up later in development. In some cases, such as waterfall methodologies for software development leaving security to the end could mean that the mistakes cannot be fixed later in the development cycle. Either way it is much more costly to wait to implement security at the end of the project since the fundamentals of secure coding need to be followed early on to reduce the number of security flaws that enter into the production version of the application.

The best way to prevent this from happening is to follow the is to follow the SEI CERT C++ Coding Standard which is community based and sets a standard for implementation of secure coding practices. Another best practice procedure to follow is to follow some of the built-in security check features that work with Visual Studio /guard is an example of Enable Control Flow Guard which helps to analyze the control flow for indirect call targets. Visual Studio will provide some built-in tools that can be used during the development of software to reduce the security risks. However, these tools do not pick up all security vulnerabilities therefore tools like CPPcheck and Clang Tidy are useful to fine tuning security tests to determine vulnerabilities that the built-in tools inside Visual Studio are not likely to detect.

Unit testing is important in an Agile methodology because it allows for streamlined testing that is broken down into parts distributed along the development cycle. This means more testing can be done as the application is developed which allows for more thorough results and a stronger security posture. Unit testing also increases efficiency in testing because theoretically it is easier to thoroughly test a smaller portion of an application than it is to test an already developed and complex application. This means the product should be published with less bugs that develop into security threats that could increase the risk for an organization.