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

Joel Meza

Professor Trevor Hodde

February 10, 2022

CS 405 – Secure Coding

Southern New Hampshire University

**Never Leave Security at the End**

Security is the main idea to keep intruders away that are unwanted from designated properties. It should always be considered a top priority when it’s applied into a software development and operation lifecycle environment. The emphasis of a security protocol such like defense in depth (DiD), that gives anyone an insight and the peace of mind from any intruders with the intention to gain access to an unauthorized domain field of a system/program. Therefore, many users that have used a program had required to gain access into a system first where they’ve used security measurements like a VPN, changing passwords for a certain period, and have done a two-step verification before they’re allowed a level of access into the program. Threats can be prevented by avoid clicking rigid links in email, advertisement and using unprotected messaging app (Bianco, 2021). However, these previous examples do invite intruders to surveil companies and institutions with large scale consequences. Best practices are to always keep a focus on security measures and maintain threat checking points at a priority by enabling firewall, enabling antivirus software, and maintaining stronger passwords. Keeping these rules of defense in depth (DiD) layered security system is the way to protect a program and to avoid future threats or vulnerabilities.

Unit Testing is one the best methods to enforce security protocols in a software program. While it’s developed, we learn the hack methods that intruders use the most which are the SQL injection or Overflow Buffer which introduce malicious ware such like bugs into a program/system. Unit Testing is a great security method to limit the number of characters or integers that should properly be used within the compiled code. The major advantages of using Unit Testing are the limited results where we may want; for example, a date format of XX-XX-XXXX (month-day-year) will limit the range of integers that’ll be returned if it was implemented correctly. In addition, it will avoid any blank spaces between integers/characters where it can cause a buffer overflow exploit and the effect of an unwanted behavior within the compiled code. It also limits the row counts being processed as it avoids the return of overflow data. Despite of this overflow data result, the benefit of limiting data that is processed can help the compiled code to be free of bugs and maintain a faster compiled performance.

Validating data prior to processing it in Unit Testing method can make a safer, reliable, and secure code. Validation checks data to ensure it is logically correct and does not contain invalid or unwanted values. At first glance, validation may not seem relevant to security, but oftentimes validation errors will lead to the discovery of more interesting problems. These problems might involve bad data, code problems, poorly cleansed data, or a malicious user trying to find security holes. (Pollack, 2019) Which makes a strong emphasis on a vulnerability that can even dig more deeper discovery insight; for example, if we use validation analysis and implement them in reports or warehouse data then we can discover potential threats with misbehavior that yet have not been established as an important bug. Overall, we must not leave security until the end, but join them in union when building code throughout each phase of the DEV-OPS (development/operations), transformed better to be understood as DEV-SEC-OPS (development/security/operations).

**Citations**

1. Bianco, D. (2021). *10 Ways to Prevent Computer Security Threats from Insiders.* Retrieved From: <https://www.techtarget.com/searchsecurity/feature/Ten-ways-to-prevent-insider-security-threats>
2. Pollack, E. (2019). *SQL Injection: Detection and Prevention.* Retrieved From: <https://www.sqlshack.com/sql-injection-detection-and-prevention/>