**6-1 Journal**

Winnie KwongSouthern New Hampshire UniversityCS-410: Software Reverse EngineeringProfessor PriestJune 16th, 2024

**Defining a Security Vulnerability**

Security vulnerability can be defined as a weakness or flaw in a computer system, security procedures, internal controls, or even the design and implementation of a system security policy. It occurs when there are weak spots that enable hackers to exploit them and cause malicious harm for financial, political, or personal gain. Possible security vulnerabilities include software bugs, weak passwords, phishing, and other malicious viruses secretly hidden in downloadable files.

**Identify: Vulnerabilities Identified in C++ Code**

Many types of vulnerabilities would be identifiable in C++ code. According to GuardRails, there are five common security vulnerabilities within C++: buffer overflows, memory management errors, unvalidated input, integer overflow, and injection attacks. (GuardRails, 2023). Buffer overflows occur when there is an overflow of data written, exceedingly more than what the buffer can hold. As a result, it could lead to corrupted data or hackers injecting malicious code. An example of memory management errors is memory leaks or improper memory allocation that can cause the application to run out of available memory. When there is not enough available memory, it can lead to crashes or termination, disrupting the program. Invalid input validation occurs when user input is left unchecked when it is supposed to ensure it meets the program's expectations. Having an input validation enables users to improve security from being exploited by attackers to steal data. Like buffer overflow, an integer overflow occurs when a calculation exceeds the maximum or minimum value, leading to crashes and exploitation possibilities. Lastly, Injection attacks are insecure activities that occur through user input. Harmful injections can implement malicious malware and modify the code, leaving the program vulnerable to other attacks.

**Purpose: Reasons to Look for Vulnerabilities During legacy to C++ Conversion**

Looking for vulnerabilities during legacy to C++ conversion instead of the testing phase is essential for diving deeper into code analysis and code benefits. When searching for vulnerabilities during the conversion process, code analysis provides a better understanding of how the logic and functionality are translated within C++. Suppose the user tried to look for vulnerabilities during the testing phase; the testers could miss hidden vulnerabilities within the logic structures since testing is more focused on the functionality of the system. Early detection is always cheaper and less time-consuming than the code being deployed. Avoiding waiting during the testing phase can reduce the cost and the overall testing burden of locating the vulnerability.

**Solutions: Determining an Appropriate Fix for Security Vulnerabilities**

To ensure security best practices, there are plenty of appropriate fixes for security vulnerabilities such as understanding the vulnerability, secure libraries, and Open Web Application Security Project (OWASP). When analyzing the vulnerability, developers can understand how the vulnerability can be exploited and potentially prevent a data breach or system crash from occurring. Using secured libraries can ensure security is improved and reduce error-prone codes. Most libraries are created and maintained by experts who leverage their knowledge to prevent vulnerabilities, making them more secure than others who lack expertise in the security field. Lastly, OWASP lists common vulnerabilities and mitigation strategies to improve security. Utilizing OWASP can prepare developers for cyber-attacks, reduce error rates, and develop stronger encryptions.

**References**

GuardRails. (2023, April 27). *The top C++ security vulnerabilities and how to mitigate them - Security Boulevard*. Security Boulevard. https://securityboulevard.com/2023/04/the-top-c-security-vulnerabilities-and-how-to-mitigate-them/