Vulnerable Beneath the Source Code: Increasing Vulnerability by Decreasing Perceived (and Increasing Actual) Complexity - Shwan Ware


Sensitive Data Availability in High-Level Language Applications

Java (version of the "Programming Language of 2015" award) is used by an estimated 9.4 million developers worldwide and is ranked as a “must have, must know” skill. It’s scalability, maintainability, and efficiency have made it widely used in enterprise applications throughout the world. In addition, the JDBC standard allows developers to interact with databases using SQL statements. Developers can connect to the database and perform queries or insert, update, or delete operations. The queries can be written in SQL or Java, with the latter being more efficient and scalable. However, Java’s automated and abstracted memory management poses large cybersecurity risks. An application must maintain an efficient and practical level of high-level languages, including object manipulation and high-level language efficiency, to reduce the likelihood of sensitive data persisting within memory boundaries.

The purpose of layered solutions is the ability of an institution to use multiple technologies or methods to protect sensitive data. A layered solution can be described as a system where one technology or method is used to prevent unauthorized access to data, another to detect unauthorized access, and a third to respond to unauthorized access. A layered solution can be used to prevent unauthorized access to data, detect unauthorized access, and respond to unauthorized access.

The timeline above is a rough example of what we hope to create by the end of our research. We plan to examine 10 enterprise solutions and determine if they can be used as a layered solution. The timeline will include a timeline of when the vulnerabilities were discovered, the patches were released, and the vulnerabilities were exploited.

REFERENCES & RESOURCES