Programming III

Week 6: Focus on Java: Improving Software Security

Over the past five weeks, we have been introduced to the C++ language and explored secure programming practices for C/C++. In week 6, we explored principles of securing programs as it relates to the Java programming language. Specifically, we learned about data sensitivity in Java and work with the Java Security Manager class.

Learning Outcomes

1. Identify techniques for securing sensitive data in Java.
2. Describe access permissions in a Java application.
3. Discuss techniques to minimize privilege code access.
4. Identify factors that influence noncompliant code.
5. Develop a solution to ensure application compliance in Java.

1. Understanding Data Sensitivity in Java

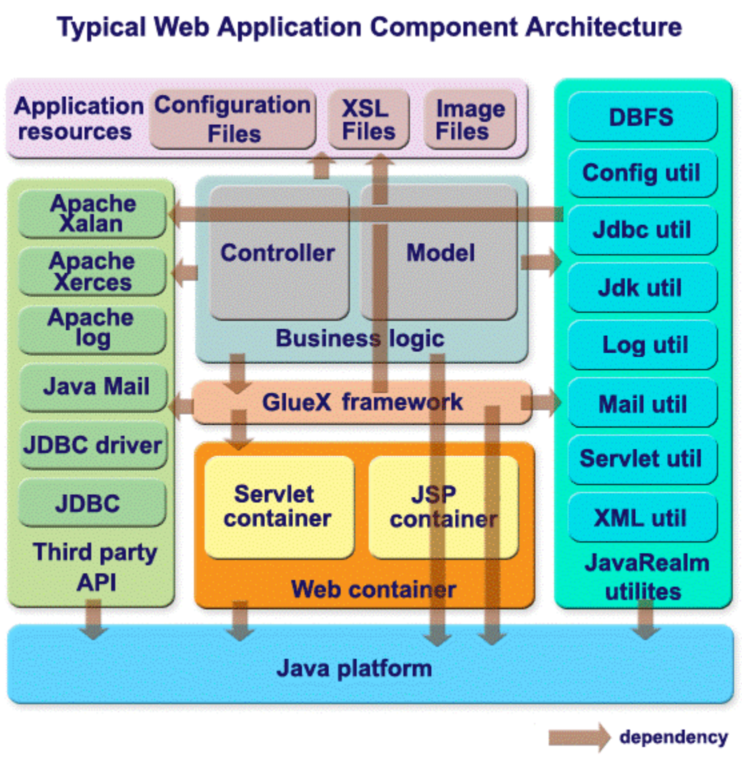
The Importance of Security in Java

One of the most important principles or tenets associated with the Java language is that it was initially designed to keep data management to a minimum for programs. Languages that emerged in the 80s and early 90s were more susceptible to security problems because they allowed for programmers to have more control over memory management. On the other hand, the Java language keeps memory management to a minimum by eliminating the use of pointers in the language.

There are, however, other issues that can cause security problems within a Java application. Security problems or errors can occur in Java through many different mechanisms. Errors can occur when an array goes out of bounds, or a string is no longer properly stored in memory. Throughout this module, we will examine the factors that influence data sensitivity within a Java application.

Data Sensitivity

One of the most critical factors that can influence an application's performance is the data that is utilized. When a Java application executes, it is essential that the data used in the application is properly stored and accessed in the system from appropriate objects. A Java application that does not correctly remove data from the system will allow for performance vulnerabilities. These vulnerabilities occur because an outside user is likely able to access data that has previously executed in a Java application and is stored in memory, the operating system, or on a hard disk. In essence, this means that it is important for data access to be limited to the lifetime of the application's execution. This will ensure that sensitive data is not fully accessible after application execution. The Java Programming language can be used to create a variety of applications. Web applications can be created using the Java language.



Typical Web Application Component Architecture. (2003). Solutions. JavaRealm Software Development Team. Retrieved from http://www.javarealm.com/solutions.html
Figure 1. Web Application Stack using Java

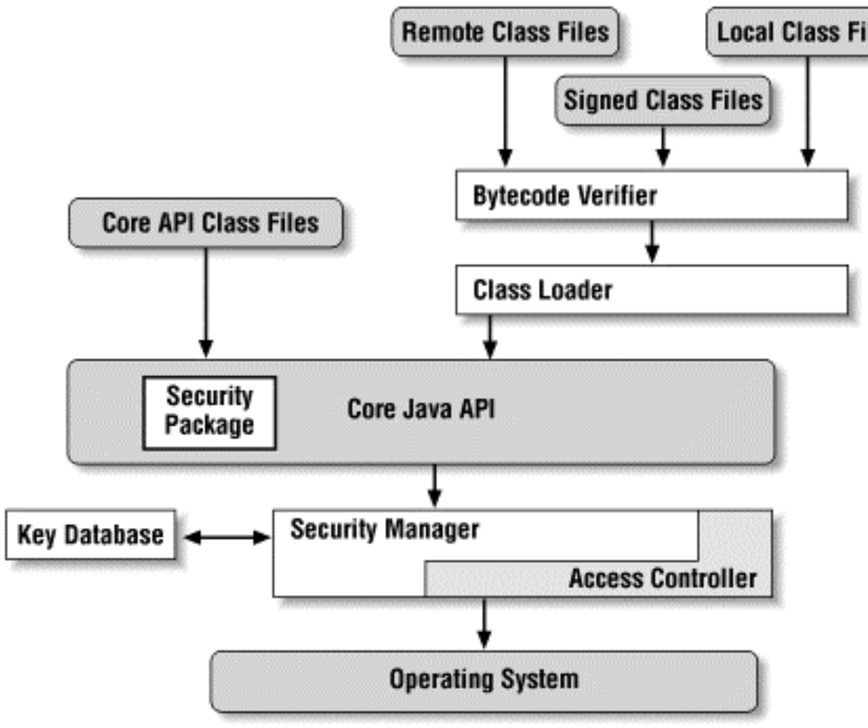
Figure 1 illustrates the various technologies that can be involved in the creation and development of a web application using Java. The versatility of the Java language is beneficial in fostering development. However, it can also lead to application vulnerabilities.

There are a variety of components that are utilized in developing web applications, which can include HTML files, SQL scripts, XML files, and additional executables. The processing of additional files can lead to vulnerabilities as malicious code can be embedded within input that is processed by an HTML file or XML file.

2. Security Through the Java Security Manager

Utilizing Security Managers in Java

To create an application that contains a secure policy for application implementation, a security manager class can be incorporated. The **Security Manager object** is a part of the java.lang.SecurityManager class and can be utilized to define appropriate application security constraints and policies.



Anatomy of a Java Application. (2016). The Java Sandbox. Java Application Security. Retrieved from http://www.onjava.com/pub/a/onjava/excerpt/java_security_ch1/?page=4
Figure 2. Java Security Manager in Java API

Figure 2 provides an overview of how the Java SecurityManager class is represented in the Java API. A **Security Manager** can be utilized in a Java application to provide the application from software vulnerabilities. This provides an extra layer of protection for minimizing application vulnerabilities. By utilizing a Security Manager, a developer can minimize application vulnerabilities by placing the security context of an operation or code segment into perspective.

See Java Security Manager in action in the following video.

Java Security Manager: https://www.youtube.com/watch?v=hRPFvtF1UP4

Summary: This video demonstrates security restrictions on writing a file, connecting to a URL, and rebooting the VM using Java Security Manager.

One of the benefits of utilizing a Security Manager in Java is its application to many different application types. These application types include:

- Desktop applications
- Web applications
- Mobile applications