

The Secure Programming Council's
Essential Skills for Secure Programmers
Using Java/JavaEE

November 2007

Introduction

Criminal hackers are increasingly targeting web applications, as organizations ranging from TJ Maxx to the New Zealand government are acutely aware. Vulnerabilities in web applications give the attackers direct access to valuable personal information or other sensitive information that can be sold for money or used by national intelligence organizations.

To blunt these attacks, most large organizations are establishing application security initiatives led by managers who have broad responsibility to use tools and training to ensure new and existing applications do not have security flaws, whether built in-house, outsourced, or at commercial software companies.

More than 40 of these organizations are cooperating in establishing standards and metrics that they can use for measuring their application security programs – in essence a “minimum standard of due care” for secure programming. The group, called the Secure Programming Council, has just completed its first consensus document, called “Essential Skills for Secure Programmers Using Java/JavaEE,” and is making the document available for public comment. Once a 60 day comment period has been completed and the suggestions appropriately woven in, the Council will publish the Java Essential Skills document for all to use.

The “Essential Skills” series is designed to enable organizations to ensure that the developers who write their applications can demonstrate that they have mastered secure programming. The council has additional minimum skills standards initiatives under way for C, C++, .NET languages, and PHP and PERL. When combined with an effective secure development life cycle, such skills can be enormously valuable in building more secure applications.

Knowledge and skills are essential only in the context of tasks that programmers must complete, so this document is organized by the security-related tasks that programmers do regularly.

The Secure Programming Council has created a set of standardize tests that measure these Essential Skills. These exams can be used in-house to find gaps in programmer skills, and for assessing job candidates, consultants, and outsourcing organizations. A complete schedule of examinations can be found at www.sans.org/gssp. Parallel examinations are also available for on-line administration inside large organizations.

A Community Effort: Your Input Is Welcome

The Secure Programming Council Java and JavaEE steering committee is composed of six of the most skilled Java security people in the world. Ed Tracy of Booz Allen & Hamilton, Ryan Berg and Bruce Mayhew of Ounce Labs, Reza Kopaei of Deloitte and

Touche, Frank Kim of Kaiser Permanente, and Sherif Koussa of Firsthand Technologies. Assisting the steering committee are a team of equally impressive people: Jeff Williams and Dave Wichers of OWASP, Brook Connor of Morgan Stanley, Sanjay Bahl of Tata Consulting, Justin Schuh of Neohapsis, Dave Grant of Watchfire, Erik Cabetas of Fortify, Jesper Johansson of Amazon, Vincent Liu of Stach & Liu, and Rohit Sethi and Nishchal Bhalla of Security Compass. They all welcome suggestions from interested testers and programmers on the front lines.

Please send comments to spa@sans.org. Comments will shape future versions.

GSSP (*GIAC Secure Software Programmer*)

Java/Java EE Implementation Issues

www.sans.org

November 2007

Task 1: Input Handling - Java programmers must be able to write programs that read input from their interfaces and properly validate and process these inputs including command line arguments, environment variables, and input streams. As these sources may ultimately derive from user input or other untrusted sources, Input Handling has security repercussions.

01.1.1 Input Validation Principles - Java programmers must understand that input cannot be trusted, regardless of the interface, i.e., HTTP Requests, Applet sockets, serialized streams, configuration files, backend datastores, etc. Java programmers must understand the white-list approaches and black-list approaches and the tradeoffs between them.

01.1.2 Input Validation Sources - Java programmers must recognize common sources of input to Java applications. This enables them to know when to question the trust level of certain data and weigh it to decide if input validation is warranted.

01.1.3 Input Validation Techniques - Java programmers must understand how to validate common data types such as String data as well as uncommon input structures. Familiarity with Regular Expressions, doValidate() and other tools of Java and J2EE to perform input validation are required.

Task 2: Authentication & Session Management - Java application programmers must have a basic understanding of Java and J2EE authentication APIs as well as a mastery of authentication principles for local and remote applications. For the purposes of this examination, Session Management is considered the process of maintaining an end-user's authenticated identity for an extended period. It is required that Java programmers understand the threats to common authentication and session management operations in order to properly protect these operations.

01.2.1 When to Authenticate - Java programmers must understand that authentication is needed not only for end-users, but also 3rd party services, backend systems, etc.

01.2.2 Authentication Protection - Java programmers are required to know how to use encryption and certificates to protect various authentication processes. This includes an understanding of strength-of-function, credential expiration, credential recovery/reset, and re-authentication.

01.2.3 Session Protection. For the protection of session tokens, Java programmers are required to understand the implications of several topics, including encryption, strength-of-function, lifespan of tokens, and re-issuance.

01.2.4 Rule 4: Authentication Techniques - Java programmers must be familiar with the more common authentication techniques and APIs available within Java and J2EE technologies. This includes the Java Authentication and Authorization Services (JAAS), backend credential storage, and various front-end authentication alternatives such as certificate, forms, and basic authentication. This familiarity assumes the programmer will understand the threats and tradeoffs for each technique.

01.2.5 Authentication Responsibilities - Java programmers must have a complete understanding of what services and protections are provided by using common APIs and what is not provided. For example, maximum session length, re-authentication, and encryption are protections that are not enabled automatically.

Task 3: Access Control (Authorization) - Java application programmers must be able develop applications that guarantee the confidentiality of user data. These applications must also prevent users from performing certain functions. Developers must understand that access control must actively be enforced, not ignored or left to backend systems.

01.3.1 Restricting Access to Resources - Java developers must understand the need for a clear and complete access control policy for system resources: for example, user data objects that should only be accessed by the owner of the data.

01.3.2 Restricting Access to Functions - Java developers must understand the need to restrict access to functions such as privileged functions and privileged URIs, etc.

01.3.3 Declarative Access Control - An understanding of the common APIs (and their tradeoffs) that support access control according to configuration files.

01.3.4 Programmatic Access Control - Java developers must understand how and when to manually perform access control checks in their custom code.

01.3.5 JAAS - Java developers must understand how the Java Authentication and Authorization Service can be used to implement access control.

Task 4: Java Types & JVM Management - Java programmers must understand the security implications of built-in data types and Java-specific memory management.

01.4.1 java.lang.String - Java programmers must have a complete mastery of the String class's immutability and how to compare String objects.

01.4.2 Integer and Double Overflows - Java programmers must understand the limitations of Java's numerical data types and the resulting security implications.

01.4.3 Garbage Collector - Java programmers must have an understanding of how the Java Garbage Collector works and the resulting security implications.

01.4.4 ArrayList vs Vector - Java programmers must understand the differences and the resulting security considerations between the ArrayList and the Vector.

01.4.5 Class Security - Java programmers should be familiar with accessibility modifiers, the final modifier, class comparisons, serialization, clone-ability, and inner classes.

01.4.6 Code Privileges - Java Programmers must understand how to manage the privileges of code as well as the different protection domains. This includes an understanding of the Security Manager and its policy file.

Task 5: Application Faults & Logging - All Java application programmers need to be able to properly handle application faults.

01.5.1 Exception Handling - Java application developers must understand Java's try/catch/finally construct to appropriately handle application and system exceptions. Developers must determine how much information should be logged when an exception is encountered depending on the nature of the exception.

01.5.2 Logging - Developers must understand the principles behind logging security-relevant events such as login, logoff, credential changes, etc. Developers should also be familiar with Java's logging package, `java.util.logging`.

01.5.3 Configuration of Error Handling - J2EE developers should be familiar with the configuration to return a default error page for HTTP 404 and 500 errors.

Task 6: Encryption Services - Java programmers must understand when and how to use encryption to protect sensitive data.

01.6.1 Communications Encryption - Java application developers must be familiar with the Java Secure Sockets Extension (JSSE) packages as well as how to configure SSL communication for J2EE applications. Developers are also responsible for knowing which of their application's external links should be protected with encryption.

01.6.2 Encryption of Data at Rest - Java developers must understand how to store sensitive data in encrypted format.

Task 7: Concurrency and Threading - Java programmers must understand how to properly structure multi-threaded programs.

01.7.1 Race Conditions - All Java application developers must understand race conditions and how they affect system security. This includes avoiding caching security relevant information that can be accessed by multiple threads.

01.7.2 Singletons & Shared Resources - Java developers must understand how to implement the Singleton pattern in Java and how to protect other resources that are accessed by multiple threads.

Task 8: Connection Patterns - Java programs must be able to securely interface with other applications. Developers must be familiar with parameterized queries, output encoding, and fail-safe connection patterns.

01.8.1 Parameterized Queries / PreparedStatements - Java programmers must understand the security risks introduced by using dynamic queries and how to safely use the PreparedStatement to safely and securely interact with databases based on user-supplied input.

01.8.2 Output Encoding - Java programmers must understand when and how to use output encoding to display data to user interfaces, as this is a primary mitigation technique to UI injection attacks, e.g. Cross-site Scripting.

01.8.3 Fail-safe Connection Patterns - Java programmers must properly form connection patterns using Java's try/catch/finally to prevent resource leaks. Resource leaks can occur as a result of failures while operating with connections to external systems.

Task 9: Miscellaneous

01.9.1 Class/Package/Method Access Modifiers - All Java programmers must understand how the Java access modifiers (public, private, protected) can be used to protect class members and methods.

01.9.2 Class File Protection - Java programmers must understand how JAR sealing is used.

01.9.3 J2EE Filters - J2EE programmers must be familiar with J2EE Filters and how they can be used to implement many of the tasks listed above.