Advance Java Study and its Approach to Java Security

Minakshi Dhillon & Happy Sharma
Department of Information Technology, Dronacharya college of engineering, Khentawas, Farrukhnagar, Gurgaon-123506,India

Email: minakshidhillon@yahoo.in ; happy22sharma@gmail.com

ABSTRACT

ADVANCE JAVA STUDY AND ITS APPROACH TO JAVA SECURITY, this idea basically deals with the java and securing it. Java is an object-oriented programming language with a built-in application programming interface (API) that is able to handle graphics and user interface further which can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and Windows, and its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics and calculation analysis. Much of the syntax of Java is same as C and C++. One major difference is that Java does not have pointer's. However, the biggest difference is that we must write object oriented code in Java. In Java we can distinguish between applications, which are programs that perform the same functions as those written in other programming languages, and applets, which are programs that can be embedded in a Web page and accessed over the Internet. When a program is compiled, a byte code is produced that can be read and executed by any platform that can run Java. In this paper, we explore importance of java security, basic java security model and guidelines for java security.

Keywords:-
Object-Oriented; Security; Antagonism; Spoofing; multithread; java

1. INTRODUCTION

Java and Object-Oriented technology are a major paradigm shift. The main reason for the popularity of java and its enormous practical applications are due to its open source, free features and the combined set of functions which provide user a vast variety of applications. There are large number of good computer languages. There are relatively few languages with the momentum to make a real difference in software development. Java is one of those languages. The developers of Java had a chance to look at existing computer language and address their deficiencies. Java allows inexperienced users to write high quality code. It incorporates user interface directly in the language. Object-oriented programming focuses on constructs called "objects." An object consists of data and functions known as methods which use or change the data. (Methods are similar to procedures or functions in other languages.) Objects of the same kind are said to have the same type or be in the same class. A class defines what data can be in an object, and what operations are performed by the methods. One or more objects can be created

or "instantiated" from a class. The structure of a Java program consists of various objects exchanging messages. The keyword class defines a blueprint for objects with similar properties.

Java holds great promise as a platform for component-based software, embedded systems, and smart cards. This means Java is poised to play an important enabling role in e-commerce as these systems move from ether-ware to reality. Java components are appearing at a rapid pace and en-capsulate critical functions for transaction-based systems. Java smart cards for e-commerce will debut soon. The Java programming environment from Sun Microsystem is designed for developing programs that run on many different kinds of networked computers. Because of its multiplatform capabilities, Java shows great promise for relieving many of the headaches that developers encounter when they are forced to migrated code between different types of operating systems.

2. FEATURES OF JAVA

The features of Java this torrent computer- speak jargon has often been labelled the "Buzzword description" and was doubtless intended with tongue in cheek, it nevertheless accurately identifies many of the features of Java that they make it so well-suited for programming internet applications. The following Java Buzz-words are as follows:

1) Platform Independent
2) Secure
3) Portable
4) Object-Oriented
5) Robust
6) Architecture-neutral
7) Multithread
8) Interpreted
9) High performance
10) Distributed
11) Dynamic

3. PROPERTIES OF JAVA

3.1. Variable Declaration:

The types of all variables must be declared. The primitive types are byte, short, int, long (8, 16, 32, and 64 bit integer variables, respectively), float and double (32 and 64-bit floating point variables), boolean (true or false), and char. Boolean is a distinct type rather than just another way of using integers. Strings are not a primitive type, but are instances of the String class. Because they are so common, string literals may appear in quotes just as in other languages. Summary of primitive data types.

3.2. Naming Conventions

Java distinguishes between upper and lower case variables. The convention is to capitalize the first letter of a class name. If the class name consists of several words, they are run together with successive words capitalized within the name (instead of using underscores to separate the names). The name of the constructor is the same as the name of the class. All keywords (words that are part of the language and cannot be redefined) are written in lower case. Instance variables and methods can be accessed from any method within the class. constructor refers to the local value of the parameter which is set when Particle is called. We use the this keyword to refer to those variables defined for the entire class in contrast to those defined locally within a method and those that are arguments to a method Classes are effectively new programmer-defined types; each class

defines data (fields) and methods to manipulate the data A new set of instance variables is created each time that an object is instantiated from the class. The members of a class (variables and methods) are accessed by referrring to an object created from thr clas using the dot operator.

3.3. Comments

There are three comment styles in Java. A single line comment starts with // and can be included anywhere in the program. Multiple line comments begin with /* and end with */; these are also useful for commenting out a portion of the text on a line. Finally, text enclosed within /** ... */ serves to generate documentation using the javadoc command.

3.4. Assignments

Java uses the C/C++ shortcuts summarized in Table 1.

Table 1. Assignment shortcuts.

operator	example	meaning
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x*y
/=	x /= y	x = x/y
%=	x %= y	x = x % y

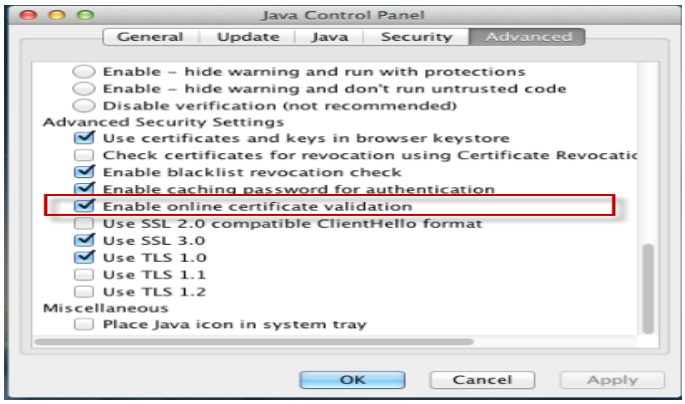
The modulus operator % returns the remainder of a division.

3.5. Type casting

It Changes the type of a value from its normal type to some other type.

4. SECURING JAVA

Java is intended to work in networked and distributed environment by providing security as. All the references to memory are symbolic references, meaning that the user is not aware where is memory program is present; it totally depends on the JVM and the machine on which the program is running. Each applet is loaded on its own memory space, which avoids the information interchange between applets. Introducing viruses, deleting and modifying file in the host computer. The Java enabled web browser checks the byte code of applets to ensure that it should not do anything wrong before it will run the applet. The major security issue in today's software world is BUGS. Unintended bugs are responsible for more data loss than data loss because of viruses. In Java it is easier to write bug-free code than in other languages. Security risks fall into four basic categories:



A. System Modification : It is the most severe class of attacks. Applets that implement such attacks are attackapplets.
B. Invasion of Privacy : If you value your privacy, this attack class may be particularly odious. They are implemented by malicious applets. Include mail forging.
C. Denial of Service : It is also serious but not severely so, these attacks can bring a machine to a standstill. Also implemented by malicious applets and may require reboot.
D. Antagonism : Merely annoying, this attack class is the most commonly encountered. Implemented by malicious applets. May require restart of browser.

5. PARTS OF JAVA SECURITY MODEL

A. The Verifier

The Verifier plays an essential role in Java's language-based approach to security, which is built on the foundation of type safety. As we know that, when a Java program is compiled, it compiles down to platform-independent Java byte code. In fig2, it shows that Java byte code is verified by verifier before it can run. This verification scheme is meant to ensure that the byte code, which may or may not have been created by a Java compiler, plays by the rules. In this sense, the Verifier makes mystery code a bit less mysterious.

B. The Class Loader Architecture

Class loaders determine when and how classes can be added to a running Java environment. Their job is to make sure that important parts of the Java runtime environment are not replaced by impostor code. Spoofing occurs when someone or something pretends to be something it is not. The fake Security Manager shown in the Fig.3 must be disallowed from loading into the Java environment and replacing the real Security Manager. This is known as class spoofing.

C. The Security Manager

This part of the security model restricts the ways an applet uses visible interfaces (Java API calls). The Security Manager implements a good portion of the entire security model and is the part of the security model most often encountered (in terms of a Security Exception) by Java applet developers. The job of the Security Manager is to keep track of who is allowed to do which dangerous operations. A standard Security Manager will disallow most operations
when they are requested by untrusted code, and will allow trusted code to do whatever it wants. The Security Manager is a single Java object that performs runtime checks on dangerous methods.

5. GUIDING THE JAVA SECURITY

A. For Java Developers

 Don't Depend on Initialization
 Limit Access to Your Classes, Methods, and Variables
 Make Everything Final, Unless There's a Good Reason Not To
 Don't Depend on Package Scope
 Don't Use Inner Classes

B. For Java Users

 Know what Web sites you are visiting.
 Learn as much as you canabout Java security.
 Know your Java environment.
 Use up-to-date browsers with the latest security updates.
 Keep a lookout for security alerts.
 Apply drastic measures if your information is truly critical.
 Assess your risks

REFERENCES

1. Stephen J. Chapman, Java for Engineers and Scientists, Prentice-Hall (2000).
2. Bruce Eckel, Thinking in Java, Prentice-Hall (1998). Covers all the syntax, plus some finer points of OOP.
3. Matthias Felleisen and Daniel Friedman, A Little Java, A Few Patterns, MIT Press (1998). Cute little tutorial that does not teach much syntax, but is designed to improve your OOP style.
4. David Flanagan, Java in a Nutshell, second edition, O'Reilly (1997).
5. David Geary, Graphic Java, Addison-Wesley.
6. Joseph O'Neil and Herb Schildt, Teach Yourself Java, Osborne (1999). Good language reference for getting up to speed quickly.
7. Walter Savitch, Java: An introduction to Computer Science and Programming, Prentice-Hall (1999).
8. Sun's Java Documentation or better JavaDoc2.
9. Patrick Winston and Sundar Narasimhan, On to Java, 2nd ed., Addison-Wesley (1999).
10. Also see javaintro and Java development tutorial.
11. Peter van der Linden, Just Java and Beyond, Sun Microsystems.
12. Patrick Chan and Rosanna Lee, The Java Developers Almanac 2000.