Risks with web programming technologies

Steve Branigan
Lucent Technologies

Risks with web programming technologies

Abstract

Java applets and their kind are bringing new life to the World Wide Web. Through the use of web programming technologies developers can make web pages interactive. Developers can now allow web pages to learn about the user and the user's system in order to operate more efficiently. These web applications are transported to the user's workstation through the magic of the hypertext transport protocol (http). The major web browsers are capable of receiving and running these web applications on the local system.

Of course, this ability to download code over the network and run it on a local system is inherently dangerous. With most of the programming technologies, source code is not delivered to the user's system. A compiled program is delivered to the system, which the user has to either trust or not trust, without being able to inspect the code.

The paper will examine some of the dangers that exist with the web based programming languages in today's Internet. This paper will focus primarily on Java, ActiveX and JavaScript. These three technologies deal with code that is downloaded and executed on a user's behalf from an unknown and/or untrusted party.

Risks with web programming technologies

The basics

Of the web programming technologies, Java and JavaScript are basically platform independent programming languages. (ActiveX uses Windows API calls.) For example, a single Java application (known as a Java applet) will run on a Windows95 architecture just as it will run on any UNIX architecture. Well, this is pretty exciting to the development community, which has been struggling for years with platform dependent issues in creating software.

First, we will examine Java in some detail throughout this paper. Java appears to have been developed with security constructs in mind. For example, Java programs run inside a Virtual Machine that is designed to allow very limited functionality to the underlying operating system. Further, compiled Java code is run through a byte code verifier, insuring that the code is type safe. Java uses type safety in part to ensure security, by attempting reduce the stack overflow problems that we see in security arena today. Java applets are only allowed to communicate with the web server that delivers the Java code. Finally, Java is providing support for digital signatures, which will allow for a more robust trust model in the future. For the web programming technologies that are available, one could argue that Java is the most advanced in the arena of security.

Java applets are platform independent, allowing a single compiled applet to execute on a variety of computing architectures. Java accomplishes platform independence through two major mechanisms; a Java compiler that turns Java source code into Java byte codes, and a Java virtual machine which translates Java byte codes into the native operating system calls specific to the hardware/software where the Java applet is currently running. (Pretty slick, huh?)

In order to develop a Java applet, a software engineer needs to obtain a Java Development Kit (JDK) specific for his/her hardware and software architecture. For example, in the research for this paper, I downloaded a JDK specific for the Windows95 architecture. This allowed me to compile Java applets right on my own system. These applets can run on any architecture where Java can run.

In order for a user to run Java applets directly from the Internet, he/she needs to obtain a web browser that is Java enabled. A web browser that is Java enabled has a Java virtual machine coded into the browser itself. Common examples of Java enabled browsers are the latest releases of Netscape’s Navigator, Microsoft’s Internet Explorer and Sun’s HotJava Browser. (Of course, this is not a complete list of all the web browsers that are Java enabled.) The only other step that a user needs to take is to visit a site on the World Wide Web that contains a Java applet.

If a web user visits a site that is Java enabled, a Java applet contained in the web page will start to execute automatically. The user is not asked for permission to start a specific applet, just notified that a Java applet is running. A web browser can be configured to disable Java and JavaScript execution. However, by default, Java and JavaScript execution is enabled on most browsers. (Digital signatures are starting to be used to verify applets come from trusted sources, which will be helpful in determining which applets are more trust-worthy.)

In staying with our focus on Java for the moment, let's examine exact how a Java applet is invoked.

Sample Applets

TicTacToe

The JDK includes quite a few sample Java applets. The first of these applets that we will examine is the TicTacToe Java applet. This applet displays a tic-tac-toe board inside a web page. The Java applet actually plays a game against the user, responding to the user's moves and sometimes winning.

A Java applet is started for a user through a simple html file. For this applet, the html file is called example1.html.

```
<title>TictacToe
</title>
<applet code=TicTacToe.class width=120 height=120>
</applet>
```

A pretty simple file. Note the **<applet …>** and **</applet>** entries in the file. These are referred to as applet tags in the html file. When a web browser sees these applet tags, it knows to expect a Java binary to be transported over the network. In this case the browser will execute the file TicTacToe.class, which is the Java applet file.

SortDemo

The Java applet SortDemo displays the multi-threaded capabilities of the Java virtual machine. The applet demonstrates the differences in the efficiency of three different sorting algorithms, bubble sort, bi-directional bubble sort, and quicksort. All three sorting algorithms are contained in a single applet, called SortItem.class. These three sorting algorithms can be started independently of the others, and complete independently. Inside the Java Virtual Machine, a thread manager handles the task of assigning CPU time to each thread.

Reasons for concern

Many corporations are configuring their networks to allow http traffic to be transported to the desktop. This is being done to allow users access to the wealth of information that exists on the web today. In addition to the traditional information on the World Wide Web that can now reach the user's desktop, so too can web programming technologies reach a user's desktop.

Imagine that you are responsible for maintaining the network security of a corporation. Now imagine that someone told you that your configuration suddenly allows untrusted computer programs to be downloaded and executed in your corporate network from anywhere on the Internet. These programs have the ability to run on any and every user's desktop that is connected to the corporate network. As frightening as this may sound, it is quickly becoming the current state of affairs with the Java, JavaScript and ActiveX technologies.

These technologies are transported through firewall configurations by the magic of the hypertext transport protocol. Firewalls, of course, are the systems that are assigned the task of screening out bad traffic from the Internet, while allowing the „good“ traffic in to the corporate network. By using a firewall, a corporation can make use of the benefits of worldwide connectivity that the Internet has to offer, while protecting the internal network from the „bad“ traffic on the Internet.

When a corporation adopts this type of configuration, it can bring the services of the Internet straight to the desktop of the corporate workers. This is the case in most corporations today with two very popular Internet services, electronic mail and World Wide Web access. This is done by allowing the smtp protocol and the http protocol access through the firewall.

Another variation that is becoming very common is a configuration where the http, or World Wide Web traffic, is delivered to a proxy server after it passes the corporate firewall. This is done for security and auditing reasons, and performance reasons.

The Firewall

Web programs such as Java applets are difficult to screen at a firewall. Firewalls make their screening decisions based upon connection information, not on content information. For example, many corporate firewalls are configured to reject telnet connections that are coming into the firewall from the Internet. However, these same firewalls are configured to accept smtp (electronic mail) traffic that originates from the Internet. The firewall makes these decisions based upon the source address, destination address, source and destination port, and various flags in the packet header. Since Java applets are transported after a connection is allowed, as part of the message content, it is difficult for a firewall to screen out Java. (Similar for JavaScript and ActiveX).

Firewalls are not aware of the contents of a network connection. The firewall knows how to screen telnet traffic, but it does not know how to process telnet traffic. This is an important point, since Java applets are part of the content of an http connection. The firewall will know how to screen http traffic, but doesn’t know how to parse http traffic.

reaches.

does understand http traffic, and can be designed to monitor the contents of an http

http session.

program from being downloaded and executed over the http link. This stems from the

encrypted. Interesting how one security technology defeats another.)

Currently, the types of attacks that can be mounted through Java and other web

- The denial of service (demonstrated through the noisybear, wasteful and triplethreat Java applets).
- The spoofing attack (demonstrated through the penpal and login screen Java applets).
- The gathering of information (demonstrated through the Netscope attack).

Denial of service attacks

In looking at the denial of service types of attacks, let us first turn our attention to the Java applet called Wasteful. The simple goal of this applet is to consume of the CPU resources on a system, without performing any useful work. (This is the first of a couple of „hostile“ Java applets that we will examine. All of the hostile applets that we will be examining come from Mark D LaDue. He authored these applets in February of 1996, while at Georgia Tech University in the United States.)

To stop the applet, the user generally will need to actually terminate the browser where the applet is running. This applet was tested on a Windows95 system, with Netscape Navigator 4.0 running the Java applet. (This is the standard configuration for all of the tests documented in this paper.) The applet caused severe impact to the Navigator process, and could only be stopped by invoking the task manager and forcefully terminating the process.

Another interesting denial of service applet is the Java applet called Triplethreat. Once invoked, it creates window after window, until memory is exhausted. On the Windows95 system that it was tested on, it created a mess. The task bar on the Windows95 system filled with new buttons, until a scroll bar appeared! To shut down the Triplethreat applet, our old friend the task manager needed to be invoked again.

The final denial of service applet to discuss is the Noisybear applet. This Java applet is designed to play a sound over and over. It will keep play this sound even after the browser has stopped running the applet. The only way to stop the sound from playing is, of course, to stop the browser. (Surprised?)

The Spoofing Attack

A Java applet has been created that lies to collect information. This Java applet, called Ungrateful, looks to connect user ids and password information. The way that this Java applet operates is clever. Once the applet is invoked, it operates on a time delay. After about 5-30 seconds, the applet starts. It displays a large black window that completely covers the entire screen. In the window, it states that Netscape encountered a security exception. In order to regain access to your browser, you must login again.

Once the applet collects this information, it is shipped over tcp port 7000 (the default) to another Java applet that is collecting the ip address, user id and password information. Of course, the applet does not allow the user to regain access to the system.

This applet failed to operate on the Windows95 architecture. However, it did operate quite well on a Linux architecture running OpenWindows. (Platform independence?)

Information gathering

The web site **Fehler! Textmarke nicht definiert.** is a very good place to visit to understand how much information a browser can yield about its system and user. Information is collected on the originator of the http connection, with surprising results. The anonymizer site can not, however, gather network information about systems that are behind http proxies. This is due to the fact that the http proxy appears to the anonymizer (and other web sites) as the originator of an http request.

When testing the anonymizer site from behind an http proxy, the anonymizer displays information about the http proxy, but not about the actual source system. Enter the site at **Fehler! Textmarke nicht definiert.**. (Major Malfunction and Ben Laurie have created this site.) The Java code at this site is able to determine the system names and network addresses of systems that are behind the firewall and http proxies. A Java applet executing on a local system can determine information about local system, such as the system name and ip address. The Java applet can communication with the web server. Since the information is transported through http, it is delivered right through the firewall and the http proxy.

While an http proxy is able to screen out the anonymizer, it does not block this attack. An http proxy could block this type of attack if it were to disable Java from entering the network. However, with the current state of the technology, a proxy that blocks Java blocks Java for everyone behind the proxy.

The options

It should now be fairly clear that Java and the Java like technologies are exciting in the way they are changing the World Wide Web. Along with these new technologies, however, come new challenges in practicing „safe networking“. If you allow http traffic into your network, you have to evaluate how you handle this issue. It is a difficult challenge since it is up to the end user to determine whether or not they will run Java once http has been allowed into the network. In order to protect against some of the problems that these new technologies may allow, consider the following recommendations.

- Disable all Java at the desktop through the browser. Most browsers are capable of disabling Java and JavaScript. Educate the user community to disable these technologies when visiting untrusted web sites, to minimize risk.
- Disable Java based upon source address on a network level. If not available already, hopefully it will be available in http proxies.
- Disable Java based upon source address by the JavaFilter. This new program, available for the Secure Internet Programming group at Princeton University, adds new functionality to the Netscape browser. Hopefully it will soon be available for other browsers.
- Disable http access. This is a drastic recommendation, but it will limit the risk to a network based upon these new technologies.

References:

The following text provided great information on the hostile Java applets and how to protect yourself.

Java Security: Hostile Applets, Holes and Antidotes. Gary McGraw and Edward W. Felten. John Wiley and Sons, New York, 1996

In keeping with the fact that this topic is an Internet topic, most of the material for this topic came from many valuable web pages. Below are listed the web pages that provided valuable information on the web programming technologies.

Fehler! Textmarke nicht definiert. - Provided information about the Java programming language, the Java virtual machine, and the byte code verifier.

Fehler! Textmarke nicht definiert. - Provided information about hostile Java applets. Also, provided code and examples on the JavaFilter product.

Fehler! Textmarke nicht definiert. - Provided information on how Java can be used to obtain information on a web browser that is resident behind a firewall and http proxy.

Fehler! Textmarke nicht definiert. –Provided information on some of the exploits possible with Java and JavaScript technologies.