GIAC
CERTIFICATIONS

# Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials (Security 401)"
at http://www.giac.org/registration/gsec

# Java and ActiveX Security Concerns

Tod Williamson

## Abstract

There is an increasing number of Internet websites that use Java and ActiveX. Websites that use Java or ActiveX can provide additional functionality that cannot be accomplished by standard html (hypertext markup language).

Anyone who uses the Internet will eventually access websites that contain mobile code. Any code that is transmitted across private or public networks and executed remotely is considered mobile code. Java, ActiveX, Macromedia Flash and Shockwave can be classified as the popular mobile code types.

As mobile code continues to evolve, Internet users have to decide whether to allow, block or scan it. This paper will describe the security concerns of mobile code and discuss ways to minimize the risks.

© SANS Institute 2004, Author retains full rights.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

## Brief History of the Internet

During the early 1990's the Internet was exclusively text based. Colleges and Universities used the Internet for transferring files and sending email. The Internet has evolved from its text-based roots, into an avenue for sending images, sound files, and mobile code.

Websites have also evolved from static web pages, to dynamic content. Web pages can be built with mobile code technologies to enhance the user's experience. This evolution from static content to multimedia content was enabled by the use of mobile code. "Java by far is the most popular implementation of Web-based mobile code used today." [1]

## History of Java

Sun Microsystems with the help of Patrick Naughton, Mike Sheridan and James Gosling started development of Java, which was originally called "Oak" during the early 1990's. The primary objective was to develop software technologies that would work with a wide range of devices, specifically consumer devices and computers.

One of the earliest devices to use Java was the Star 7 device. The Star 7 device was a small hand held device built for home entertainment with an animated touch screen. One of the primary goals of the Java development was to run on processor-independent devices. For example, you could develop software applications using Java and deploy to just about any platform: Macintosh, Unix or Windows based systems.
http://java.sun.com/people/jag/green/ [2]

## How Java Works

Java can be broken down into three major components.

1. **Byte Code**
2. **Java Virtual Machine**
3. **Execution Environment**

The **Byte Code** is a programming language used to compile code that will run on any platform. The **Java Virtual Machine (JVM)** executes the byte code, which then uses the **Execution Environment** to run the code, which contains base java class files. [1]


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Most of the current browsers, Netscape, Internet Explorer and Opera have Java Virtual Machines or JVM plug-ins installed from their default installations. Microsoft Internet Explorer 6.0 will download Microsoft's version of the JVM automatically from a file server at Microsoft, if the web page that is being viewed has an applet embedded in the page.

The Java Virtual Machine is a software-based application that can be downloaded for free from www.java.sun.com. Most operating systems come with Java Virtual Machine, or can be installed from most operating system installation disks.

Applications that are written in Java can be executed on any machine that has JVM installed. Granted, some Java applications are written specifically for certain versions of the JVM. If an application has byte code written specifically for version 1.4.0, the application may not work properly if executed on an older version of the JVM.

For interactive websites, developers will code web pages with Java "Applets" embedded in them. Anyone with a Java enabled browser can now download and run these applets within the browser. Some of these Java based applications allow users to access databases remotely over the Internet, or even view geographic maps.

Internet is being used to distribute more public record information. It's becoming easier, faster and more efficient to access public records over the Internet. Java enabled browsers and web pages with Java applets make this possible. In this example, we are going to access a public website that contains Java. The website will allow you to view different geographic maps in Wisconsin. Try accessing the website:

© SANS Institute 2004, Author retains full rights.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

http://www.bayfieldcounty.org/LandRecords/mapviewer_start.htm
[3]

For example, consider you are thinking about purchasing land in Wisconsin for the dream summer vacation home. You see an ad in the Wall Street Journal™ that someone is selling a 3-acre plot of land in Northern Wisconsin. Ten years ago, you would have a tedious task to find the courthouse telephone number, obtain a parcel number of the land, etc. With a Java enabled web browser and access to the Internet, you obtain this information in a few seconds. By accessing this public website, you can check if the vacation property is on a flood plain.

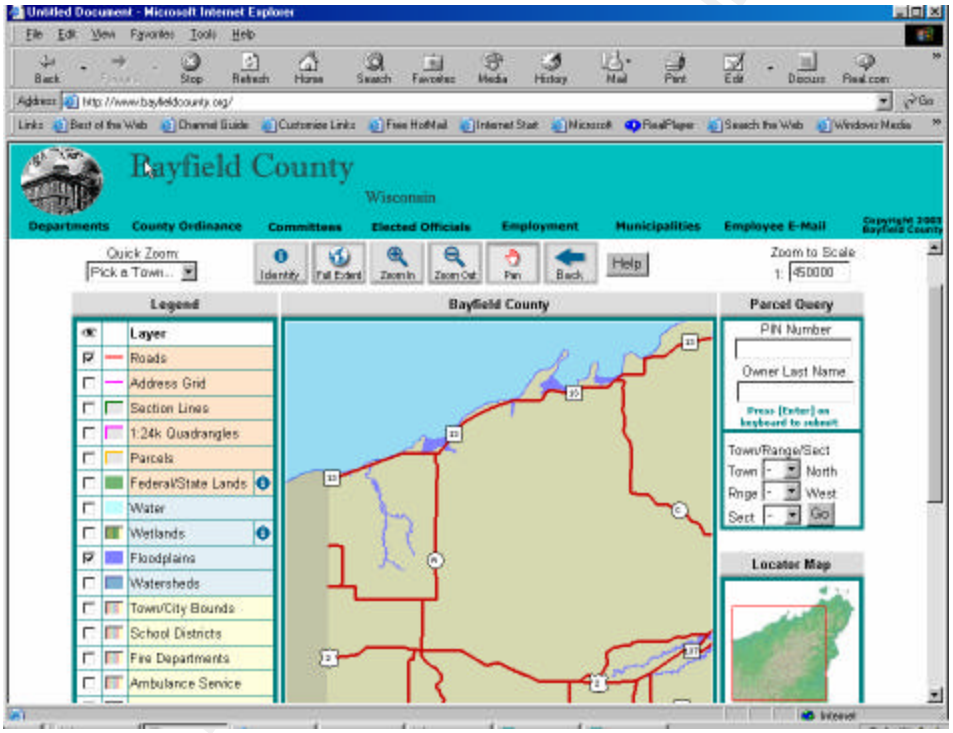


*Figure 1.*

By right clicking on the far left column of the page and selecting view source, you can verify this page is using Java. You can either scroll through the source code, or use the search function to find .class. Notice the html tags representing the applet codebase as mapplet and the mapplet.class files in Figure 2.

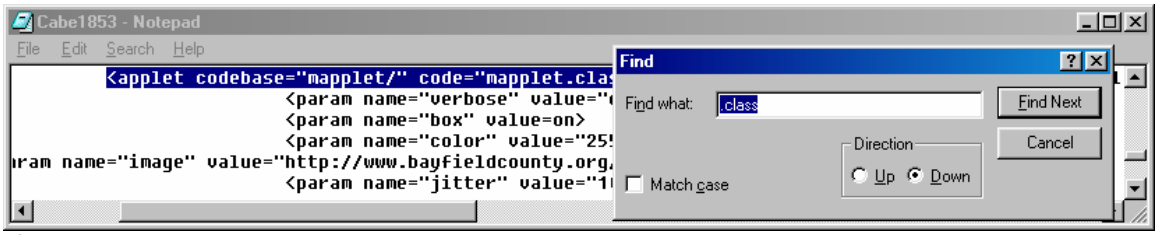


*Figure 2.*



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

## Java Security model

There are two approaches in securing Java. The two techniques are called Sandboxing and Code Signing. [1]

The Sandbox security model is a method of restricting access to a limited amount of resources or files on an end user's computer. The mapplet.class displayed in Figure 1, would be considered as running inside the Java "Sandbox". The mapplet.class code is used for displaying maps from a vendor's webserver. It does not try to create or delete files, nor does it try to copy or email files to unknown locations.

The code signing security model is based upon a third party company who will "sign" the code. This process could be compared to a notary public, who stamps and verifies a person's signature. One of the most popular code signing companies is Verisign.

There are several types of code signing options. Typically, class 2 ID's are designed for individuals. Class 3 ID's are Commercial ID's for companies, especially companies who use the Internets as a means for business. Verisign will run a background check on the company or individual who is requesting their code to be signed. The goal is to provide a level of trust and guarantee the identity of the remote computer.
http://www.verisign.com/products/signing/code/ [4]

One of the most popular uses of code signing is secure SSL (secure socket layer) within e-business web based applications. Within an SSL session, the network connection from a client's web browser to the webserver is encrypted. The certificate will ensure the code has not been tampered with. You can view the details of a certificate by clicking the padlock icon within the browser during an SSL session, as shown in figure 3. The SSL certificate will display the details of the code signing as displayed in figure 4.




*Figure 3.*



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

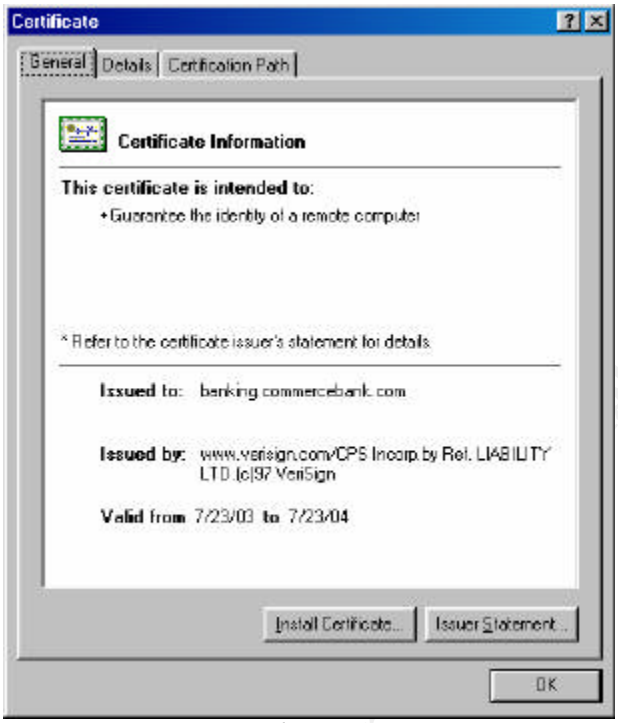


*Figure 4.*

## Hostile Java Applets

Some developers find the Java sandbox too restrictive. They may decide to code Java applications, which run outside of the Java sandbox. For example, a Java Applet could scan the computer's file system, modify files, access memory, or open other applications.

Hostile Java Applets can be classified into four categories as show in figure 5:

| Category | Details of Attack | Severity |
|---|---|---|
| **System Modification** | Modifies or deletes data, or operating system files | **High** |
| **Invasion of Privacy** | Copies or sends information, possible stored in cookies, files, workstation settings and sends them to remote computers | **High** |
| **Denial of Service** | Cause the computer or the browser to freeze. Could consume CPU resources and waste productivity | **Minimal** |
| **Annoyance** | Most hostile applets fit into the annoyance category. May cause a reboot, or have popup ads that keep annoying you | **Minimal** |

*Figure 5.* (Felton, E., and McGraw, G) [1]

Hostile Java applets, which modify files, are a high-risk security concern. There are specific websites that demonstrate all four techniques. You can execute



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

examples of hostile mobile code by accessing http://www.finjan.com/mcrc/sec_test.cfm [5]

**ActiveX**

According to Microsoft, "ActiveX and Java are complementary, not competing, technologies." Microsoft has it's own flavor of the Java Virtual Machine. After a long court battle with Sun Microsystems, Microsoft will no longer support Microsoft JVM after January 2004. [6] ActiveX when first released didn't work on Macintosh or Unix based systems, nor was it supported in Netscape browsers. Netscape users would have to download a special browser plug for ActiveX controls.
http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnaractivex/html/msdn_faq.asp
[7]

Generally, ActiveX can be classified into three categories

**ActiveX Controls**
**ActiveX Documents**
**ActiveX Scripting**

**ActiveX controls** are used in web pages for display animation, audio, and video. There are literally thousands of ActiveX controls. ActiveX controls are also persistent. If a user downloads an ActiveX control it can be stored in the browser cache, or on the hard drive. Once downloaded, it can be accessed by other applications.

One advantage of ActiveX is the ability to reuse existing ActiveX controls to build web-based applications. These ActiveX controls can be purchased from software developers or downloaded from websites. This website has ActiveX controls which can be downloaded for free. http://www.sevillaonline.com/ActiveX/

**ActiveX Documents** are a set of active viewers that can display Microsoft Office documents within a web browser. If the end user has the Microsoft Office Suite, a web page with an ActiveX viewer can automatically launch an application like PowerPoint and display it within the web browser. If the user doesn't have PowerPoint, the ActiveX control could download the correct plug in from Microsoft and install it in the browser. This could be accomplished behind-the-scenes without any user intervention.

**ActiveX Scripting** allows ActiveX supported browsers like Internet Explorer to run Java Applets.
**http://www.geocities.co.jp/HeartLand-Gaien/3046/activex/features.html)**
[8]


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

## ActiveX Security Model

Unlike Java, there is not a security "Sandbox" for ActiveX controls. ActiveX relies on end users ability to make security decisions whether to execute the ActiveX control. Microsoft also works with Verisign to use a code signing security model. This method is referred to as Authenticode. Authenticode is based upon digital signatures.

Authenticode supports multiple file formats. Including, PE format files, Java applets, ActiveX controls, plug-ins, executables, and cabinet files. Signed ActiveX controls will verify the code has not been tampered with. If the code has not been digitally signed, browsers like Internet Explorer can be configured to block execution of the control.

It's important to point out that Authenticode does not guarantee that signed software components, in the form of Java or ActiveX controls is without software flaws. Depending on the complexity of the Java Applets, or ActiveX controls, there is a chance that it may cause problems, even if the code is signed by a third party.
http://www.tutorialbox.com/tutors/J++/ch23.htm#IntroductiontoCodeSigning
[9]

## Java and ActiveX security solutions

### Block Java and ActiveX

From a security standpoint, there are strategies to address Java and ActiveX. When Java and ActiveX were in infancy stages, blocking mobile code from the Internet was common practice. Like most new technologies, Java and ActiveX had security issues when first released.

However, in today's world of interactive, multimedia, business and information websites this technique of blocking all Java or ActiveX websites is becoming less of an option. The option to block is rather easy to implement. Using a Gauntlet 6.0 firewall, as shown in figure 6, blocking Java and ActiveX is a simple process. Checking the deny buttons in the http proxy configuration will block the mobile code at the perimeter. Most popular firewalls have the option to block Java and ActiveX.

Firewalls search for the embed tags and will remove the code between the opening and closing embed tags. The end user will experience a blank or incomplete page which had the <embed> tag data removed. Macromedia Flash or Shockwave websites will be blocked too, because these technologies also use <embed> tags.


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

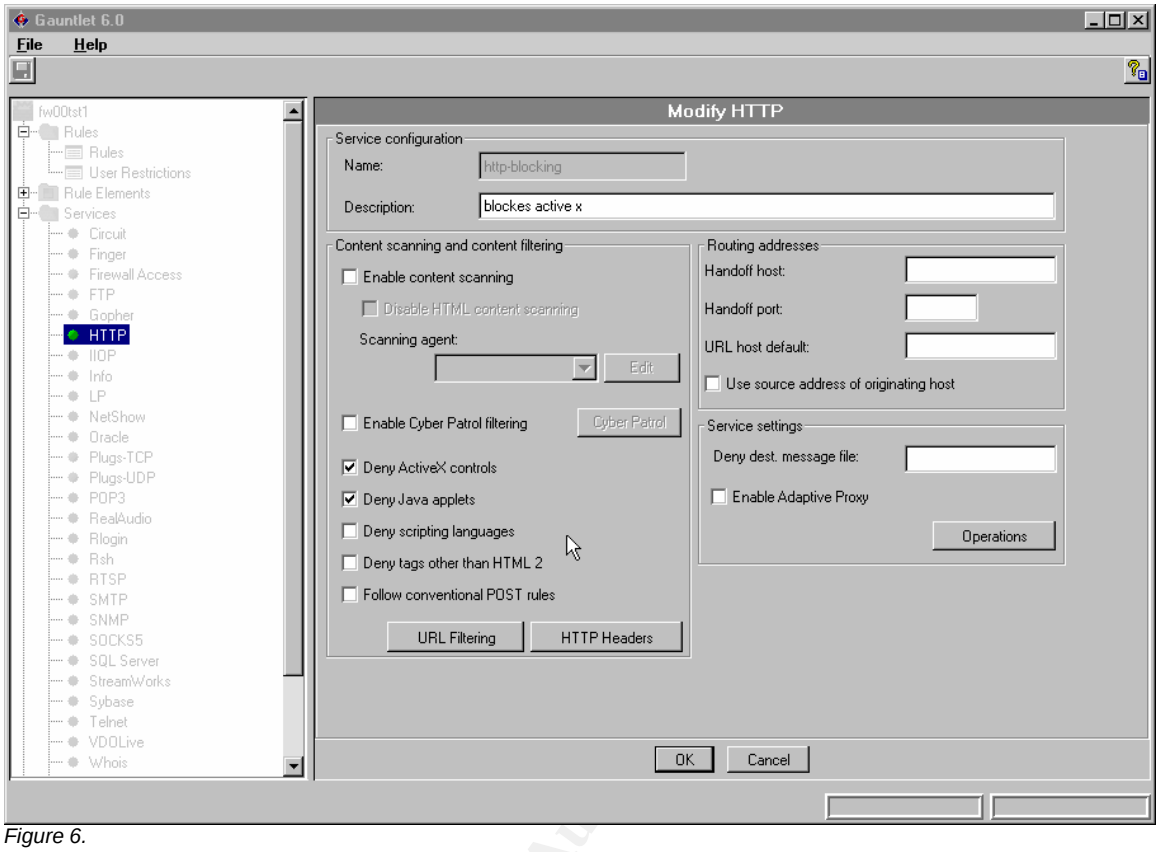

*Figure 6.*

**Important Note:** Firewalls cannot block Java applets or ActiveX controls within an SSL session. SSL (secure socket layer) encrypts the data between the client and web server. Firewalls cannot decrypt the data and inspect for Java and ActiveX. This is an easy way to circumvent firewalls that block Java and ActiveX. Create a website that uses SSL and then embed Java and ActiveX within your web page. The only defense would be a properly configured browser with correct security settings.

The following chart (figure 7) can outline security options for mobile code. The easiest solution is to block all mobile code from the Internet. However, you will also block Java websites that have java class files that would run safely in the sandbox. Your enterprise may not be able to conduct business with external websites that have Java or ActiveX websites. The trade off is security versus functionality.


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

| Options | Risk | Cost | Complexity |
|---|---|---|---|
| **Block** | None | None | Minimal |
| **Allow** | High | Possible* | Moderate* |
| **Scan** | Minimal | Yes | Minimal |

*Figure 7.*

**Allowing Java and ActiveX**

The decision to allow all Java and ActiveX comes with a high amount of risk. The risk factor is increased as you are relying on end users making security decisions. End users who can change the browser's security settings dramatically increases the risk. Changing from high or medium security setting to a low setting is a way for unsigned controls to run. Unsigned ActiveX controls would have the ability to run without user intervention. Information could be stolen in the form of an ActiveX control, which is not restricted by a sandbox.

*The cost associated with allowing Java and ActiveX could be high. Reflected by additional helpdesk support calls and cleaning workstations from hostile or nuisance applets. There are Java based games, which can be downloaded and played. The possibility of downloading unauthorized software that may cause other system related problems. Licensing of software is also an issue, if the end user is accepting the license agreement on behalf of their company.

Some Java applets can run outside the Java sandbox. Even with code signing, these applets can make network connections back to a webserver. Within the Java model, the connection to the external web servers should be only to the originating server. Some organizations do allow outbound Internet connectivity for everyone. This is a dangerous practice as hostile applets or ActiveX controls would have a way to steal data and send it to an unknown host. Firewall rules should be configured to block for all outbound connections, other than required services.

Additional TCP services should have some form of authentication. Split DNS configurations are also helpful. Split DNS is where you have an Internet DNS system that resolves only internal addresses and an external DNS, which resolve Internet addresses. If an ActiveX control used a hostname to connect to a remote host, the internal DNS will not resolve the address. Mobile code may use hostnames; others may use a static IP address within the mobile code. Additional security layers are required if you allow Java and ActiveX. http://www.isaserver.org/tutorials/You_Need_to_Create_a_Split_DNS.html [10]

*Tips to use if you allow all Java and ActiveX

- Consider using a split DNS configuration
- Use firewalls to block or restrict all outbound connections
- Configure internal network to non routable addresses, and proxy traffic
- Secure the browser and lock it down, so it cannot be modified


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

At a minimum configure Internet Explorer to the medium security settings as shown in Figure 8. Browsers should be kept up-to-date with security patches and updates. Properly configured browser may be your best defense against hostile mobile code. Internet Explorer and Microsoft operating system updates can be installed from; http://v4.windowsupdate.microsoft.com/en/default.asp and Netscape browsers can be updated from, http://channels.netscape.com/ns/browsers/download.jsp

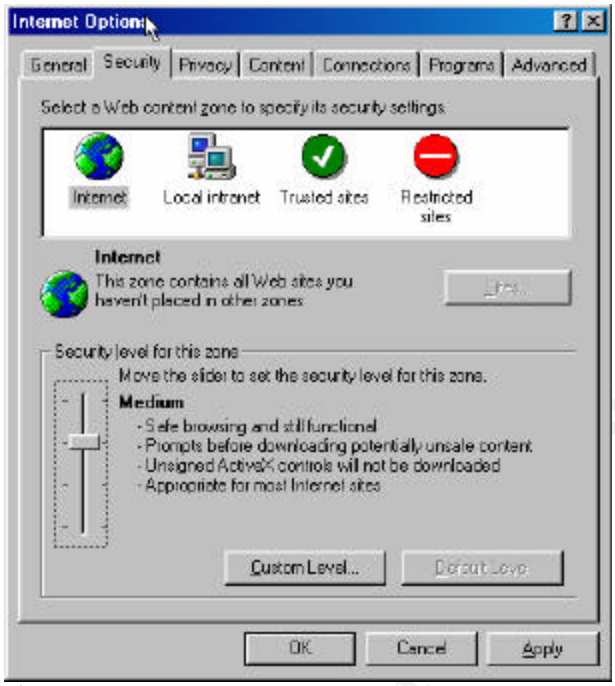


*Figure 8.*

## Scanning Java and ActiveX

There are two vendors who have products that scan for hostile Java and ActiveX code. Finjan Corporation and Trend Micro content servers will scan for hostile applets or ActiveX controls at the perimeter.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Gateway scanning for mobile code is fairly easy to implement. Policies can be made on the gateway device to scan, block or allow mobile code. Surfingate has been on the market for a number of years. The latest version is Surfingate 7.0 SP1 that was released in June 2003. Anti-Virus and URL categorization features which can be purchased with a separate license. Here is a screen shot of Surfingate 7.0 as referenced in figure 9. Surfingate can scan for Java, ActiveX, Executables, Documents, Java Script, VB Script and Plug-ins. Both products look for specific extensions in web pages that contain mobile code. The policies can be very granular as to what is permitted and what is denied. The policies can be configured on a URL or Active content basis.

For example, using Surfingate you could create rules to allow ActiveX content from www.cnn.com or block all Java from www.yahoo.com.

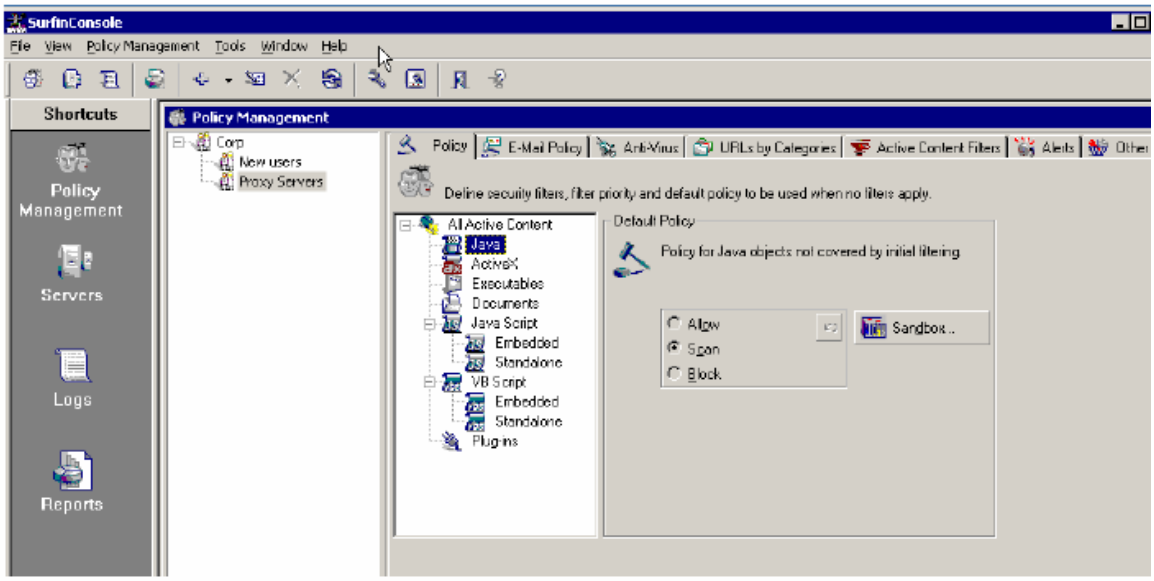


*Figure 9.*



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Surfingate supports two modes of operation, Proxy Mode or ICAP mode.

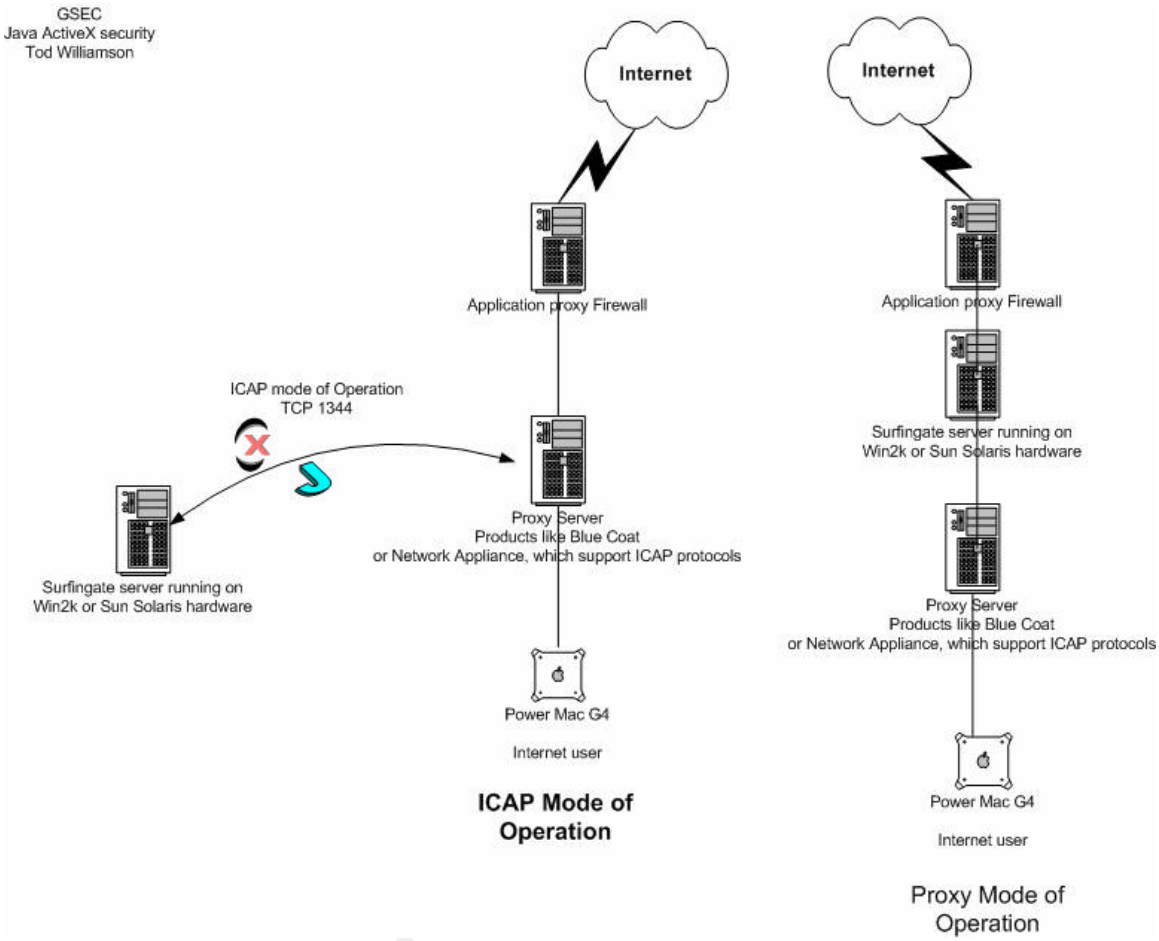


*Figure 10.*

Some organizations could implement Surfingate, as it's first proxy hop to the Internet. Surfingate 7.0 support active directory and users can be grouped into different access rights or accounts.

ICAP (Internet Content Application Protocol) was developed for off-loading specific Internet based content to dedicated servers that handle specific tasks. For example, as shown in figure 10, ICAP protocols could be used to off load http virus scanning duties to dedicated servers, leaving the proxy server to handle additional http connections. In the above diagram, ICAP protocol will forward all traffic that contains the certain extensions related to mobile code, such as Java class files.
http://www.i-cap.org/docs/icap_whitepaper_v1-01.pdf
[11]


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

## Conclusion

Mobile code technologies will continue to gain popularity. The use of Java and ActiveX for a transport mechanism for potentially hostile activity could increase. Are there misconceptions about hostile Java applets? There seems to be a certain amount of "FUD" regarding Java. FUD is Fear Uncertainty and Doubt. http://java.sun.com/features/fudwatch/fudwatch1.2.html [13]

As for minimizing risk, you can block all Java and ActiveX. However, you have effectively denied service to such websites that could benefit your organization. Allowing all Java and ActiveX could have adverse impacts. There are plenty of nuisance applets and ActiveX controls. Try installing gator, from www.gator.com and you will experience nuisance mobile code first hand.

It can be argued that ActiveX model is riskier that Java. As ActiveX controls can be more destructive, and rely on end users to make security decisions. Most end users become frustrated with browser alerts and often click OK to any message to get rid of the security dialog boxes. The case and point, the "I Love You" virus that struck a couple of years ago infected thousands of computers. If a co-worker, or someone you don't know sends an email with "I Love You" in the subject field, they should have suspected "riff raff" activity, yet end users all over the world executed the virus, which infected thousands of computers.

Code signing ActiveX controls doesn't seem to be a widely accepted practice. Code signing costs roughly $400 to have it "signed". If a company signs the code, it appears there could be legal action taken against them. Developers may see this as a risk to accept the liability if their ActiveX control causes damage. Instead of accepting the liability, they choose to not to sign the ActiveX control.

According to a 1997 CNET article, "If a control does something bad to a user's computer, the publisher can then be tracked down and prosecuted. In other words, the Authenticode system does not protect against malicious code; it simply makes it easier to find out who wrote it". http://news.com.com/2100-1023-271469.html?legacy=cnet [14]

There are solutions for scanning Java and ActiveX. There are also ways to circumvent these products. Most notably, websites that use SSL will bypass the scanning engines as the data is encrypted. Scanning engines can be misconfigured with rules to trust all content from business partner's website. If the website was hacked, hostile mobile code would exploit the trust. The result could be one of the following conditions: Denial of service, invasion of privacy, annoyance, or have files modified.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

A properly configured browser may be the best single defense against mobile code. Understanding the basics of Java and ActiveX and when to trust content in the form of signed controls is another line of defense. There are no "silver bullets" to solve all Java and ActiveX security issues. The suggestions outlined in the paper may help minimize these risks.

**References:**

[1] Felton, E., and McGraw, G. "Securing Java," (John Wiley & Sons, 1997)

[2] "Brief History of the Green Project". 1997. URL:
http://java.sun.com/people/jag/green/

[3] "Welcome to Bayfield County Mapviewer!" 2003. URL:
http://www.bayfieldcounty.org/LandRecords/mapviewer_start.htm

[4] "Code Signing Digital Ids" 2003. URL:
http://www.verisign.com/products/signing/code/

[5] "Security Testing" 2003. URL:
http://www.finjan.com/mcrc/sec_test.cfm

[6] "Microsoft to add Java support to Windows XP" 2003. URL:
http://www.itworld.com/AppDev/736/020619winxpjava/

[7] "ActiveX FAQ" 1996. URL:
http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnaractivex/html/msdn_faq.asp

[8] "Features of ActiveX" 2003. URL:
http://www.geocities.co.jp/HeartLand-Gaien/3046/activex/features.html

[9] "Introduction into Code Signing" 2002. URL:
http://www.tutorialbox.com/tutors/J++/ch23.htm#IntroductiontoCodeSigning

[10] "You Need to Create a Split DNS" 2002. URL:
http://www.isaserver.org/tutorials/You_Need_to_Create_a_Split_DNS.html

[11] "Internet Content Adaptation Protocol (ICAP)" 2001. URL:
http://www.i-cap.org/docs/icap_whitepaper_v1-01.pdf

© SANS Institute 2004, Author retains full rights.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

[12] “FUDwatch” 1997. URL:
http://java.sun.com/features/fudwatch/fudwatch1.2.html
http://news.com.com/2100-1023-271469.html?legacy=cnet

[13] “Intuit warns against ActiveX” 1997. URL:
http://news.com.com/2100-1023-271469.html?legacy=cnet

© SANS Institute 2004, Author retains full rights.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.