Global Information Assurance Certification Paper

Malicious Mobile Code Security Case Study

GIAC Security Essentials Certification (GSEC)
Practical Assignment, Version 1.4b

Option 2: Case Study in Information Assurance

Stephen Laird

January 6, 2003

Summary

The ICSA reports that "despite increased spending, the rate of malicious code infection continues to rise…[and] in addition to becoming more prevalent, computer viruses were becoming more costly and destructive." [1] Typical problems caused by viruses include information theft, file corruption, data loss, and productivity loss. Downloading active web content via web browsers represents one means of transmitting computer worms, viruses and other malicious code. What makes it particularly pernicious is that active content is often downloaded and executed without the user's explicit consent. Thus, "web browsing provides a means for malicious code to bypass the firewall and infect a host or network."[2] We recognized this risk at my company and sought means to address it.

This paper first discusses the vulnerabilities associated with downloading mobile code from the Internet, then describes some of the risks associated with two of the more common forms of executable content – ActiveX controls and Java applets. Finally it describes the mobile code server we developed as part of our defense in depth strategy to protect our network from web-based malicious mobile code. This server not only includes common functionality such as control lists and a real-time virus scanner but also implements a mobile code policy that categorizes the different types of mobile code and defines requirements for their downloading and execution.

Before

Existing Internet Browsing Security Measures

My company's existing Internet security measures were somewhat limited. Network security was provided by a firewall and an intrusion detection system. Individual user logins and real-time virus scanners provided host-based security. Also, our Internet Explorer security setting was "Medium". This setting prevents some forms of active content from being downloaded and provides a pop-up verification window before downloading others. Lastly, although we employed nominal monitoring of employee Internet usage, employees had unrestricted use of the Internet.

Vulnerabilities

These Internet browsing security measures were beset by numerous vulnerabilities. The main threat is that individual users could download malicious code and execute it without explicitly approving it. Active content such as ActiveX controls and Java applets can be embedded in web pages. When these pages are downloaded, the active content can be automatically executed with no guarantee that it is benign. A second vulnerability was in allowing individual users to accept web-based code by simply selecting "OK" in the browser's pop-up windows. Individual users may be ignorant of the risks associated with downloading mobile code or may simply desire expediency and accept the code. Finally, by relying on browser settings as part of our web-based policy, we were

open to malicious individuals intentionally changing their own settings or naïve individuals inadvertently changing them. Both of these actions serve to bypass the corporate web-browsing policy and open the organization's computer networks up to the threat of downloading web-based malicious mobile code.

Risks

The risks associated with mobile code include inadvertently downloading viruses or other malicious software into the user's system. Possible actions of malicious mobile code include "crashing the browser, damaging the user's system, breaching the user's privacy, or merely creating an annoyance."[3] Other possible actions are presented in the Nimda worm, ActiveX and Java applet examples given below.

Nimda Worm

A recent, well-known case of malicious mobile code was the Nimda worm that exploited vulnerability in Windows systems starting in September 2001. According to the CERT Coordination Center, the impact of the worm includes the following: "Hosts that have been compromised are at high risk for being party to attacks on other Internet sites. The high scanning rate of the Nimda worm may also cause bandwidth denial-of-service conditions on networks with infected machines."[4]

One of the many propagation modes that Nimda used was through browsers: "As part of the infection process, the Nimda worm modifies all web content files it finds (including, but not limited to, files with .htm, .html, and .asp extensions). As a result, any user browsing web content on the system, whether via the file system or via a web server, may download a copy of the worm. Some browsers may automatically execute the downloaded copy, thereby infecting the browsing system."[5]

Forms of Mobile Code

There are several different technologies that can be used as web-based mobile code: ActiveX, Java applets, Visual Basic for Applications, Javascript, Shockwave and others. This report will examine two of the most common forms, ActiveX and Java applets, to highlight some of the risks associated with mobile code.

ActiveX

A good description of ActiveX, including its security model and associated risks, is found in The World Wide Web Security FAQ and is summarized as follows. ActiveX controls were developed by Microsoft as a means of distributing software over the Internet. They can be embedded in a Web page. ActiveX places no restrictions on what a control can do and as a result, they present a huge risk to computer systems if used maliciously. The ActiveX security model is based on digitally signing each control in such a way that the signature cannot be altered or repudiated. Browsers will recognize unsigned ActiveX controls or those that are certified by an unknown authority and present a dialog box

warning the user that this action may not be safe. The user can elect to abort the transfer, or may continue the transfer and risk downloading and executing a malicious control. While this security model ensures that ActiveX controls cannot be distributed anonymously and that third parties cannot tamper with a control after its publication, it does not ensure that a control will be well behaved. A malicious ActiveX control is capable of crashing a machine, reformatting a hard disk or planting a virus. [6]

Java Applets

Unsigned Java applets are safer than ActiveX controls in that they are typically restricted to a sandbox environment. As such, "they cannot execute arbitrary system commands, load system libraries, or open up system device drivers such as disk drives. In addition, applets are generally limited to reading and writing to files in a user-designated directory only. Applets are also limited in the network connections they can make: an applet is only allowed to make a network connection back to the server from which it was downloaded." [7]

Nonetheless, applets can still leave a system vulnerability to denial-of-service or other attacks. They do this by absorbing system resources such as memory and CPU time to the point where the system is incapacitated. Among other risks associated with applets include "annoyance (playing load sounds), interfering with other applets or sending forged emails." [8]

Protection against hostile applets involves installing the latest browser versions and all patches as they become available, filtering known malicious applets or virus scanning at the client or firewall, resource limiting or checking for trusted signatures before downloading and executing applets. [9]

During

General Requirements

In order to balance the value of Internet browsing with the risks associated with malicious mobile code, we desired to develop a means that allows web browsers to download the maximum content without sacrificing too much security. An additional desire was to remove the security decisions (i.e., pop-up windows) from the individual users and allow Internet policy to be controlled by the system administrator from a central location.

Common Internet Security Measures

Common tools used in implementing a web browsing policy include a control list device and a real-time analyzer. "Devices employing the control list method typically integrate with an organization's firewall, proxy server or Internet caching device. When a web user requests access to a site, the firewall/proxy/cache consults the control list server and determines if the policy for the requesting user will permit or deny access to the requested site." [10] The control list is typically a database of URLs maintained by the vendor of the product. One vendor, Websense [Ref. 4], uses web-site mining techniques to

generate and maintain its list of sites containing malicious code. Real-time analyzers typically perform one or both of the following functions: scanning for unsuitable content that contravenes organizational policy and detecting malicious code. "A real-time analyzer generally employs keyword matching or pattern matching to detect unsuitable content." [11] Some companies, such as Symantec, are developing advanced heuristic methods to detect unsuitable content [Ref. 11].

In addition to the typical solutions mentioned above, we also considered the Department of Defense's published Mobile Code Policy [12]. This policy was designed to "protect DoD systems from the threat of malicious or improper use of mobile code". In short, this document discusses the risks involved in downloading various types of mobile code and defines the necessary restrictions placed on downloading and executing them. ActiveX controls, for example, can have unconstrained access to host systems and resources and are thus ranked in the highest category, "category 1". Java applets are restricted to the sandbox and have other restrictions placed on them and are thus ranked as "category 2". JavaScript allows little access to computer resources. It is ranked as "category 3" which is deemed low risk. The policy specifies the requirements for the reception of mobile code from external sources. Clearly, due to the increased risk of downloading category 1 mobile code, more stringent requirements were placed on category 1 mobile code as compared to category 2 and category 3.

Our Internet Security Requirements

Drawing from these sources, we started to define the requirements for our own Internet policy. The solution would be based largely on the tenets of the DoD mobile code policy and also include some of the more common security measures such as a restricted URL list and virus scanning. Another major consideration was to centralize the configuration. That is, provide a means whereby all the browsers on the network can be controlled and configured from a central location. This removes control from the individual users and allows for easier configuration changes.

The following features were required for the initial implementation:

- Intercept all active web content and allow or block downloading in accordance with local mobile code policy
- Provide a list of unacceptable URL's
- Provide a list of "legacy" URL's – URL's from which active content is acceptable even if it does not otherwise meet the local Internet policy
- Provide virus scanning of active content

Implementation

The core of our solution is a server running on a central machine through which all browser requests are funneled for analysis. This server also acts as the central configuration point – that is, the entire mobile code policy can be implemented and changed at this single location. When the server receives a URL request, it separates the mobile code requests from the non-mobile code

requests based on file extension. The non-mobile code requests (e.g., html, gif) are retrieved and forwarded directly to the browser. Mobile code, meanwhile, is retrieved and checked for viruses and approved signatures. It is then measured against the given policy to see if it is deemed safe to forward to the requesting browser.

The policy we implemented is summarized in the following table.

Category	Examples	Allowed to Execute
1	ActiveX, Unix Shell Scripts	Signed AND from a trusted source Designated as Legacy
2	Java applets, Postscript	From a trusted source Signed Designated as Legacy
3	Javascript,VBScript	Always

Table 1: Mobile Code Policy

The sections below define the terms in the mobile code policy and discuss how the were implemented.

Signed Code

According to the mobile code policy, certain types of mobile code must be signed with approved certificates in order to be downloaded and executed. We added the ability to recognize files signed using Microsoft's Authenticode, Netscape SignTool or JavaSoft. In addition to coding in the ability to recognize signed files of each type, we needed to maintain a list of acceptable certificates. We added the ability to enter a certificate to our list directly -- using the .cer file -- or importing it from a signed file. Typically, for a piece of mobile code to be considered signed, the digital certificate used to sign the code and all the certificates in its certificate chain, up to and including the root certificate, must be entered into the server's certificate list. Under certain circumstances, the root certificate may not be available, so we added the option to accept the certificate even if the certificate chain is incomplete. Here again, we are trading security for functionality.

Trusted Sources

According to the DoD mobile code policy, a trusted source is "a source that is adjudged to provide reliable software code or information and whose identity can be verified by authentication" [13] Basically, we needed a way to define which web sites we trust and then be assured of the identity of those sites when downloading mobile code. For our policy, we decided that a digital signature or an SSL connection is sufficient to validate the identity of a trusted source. So, in

addition to the code signing recognition described above, we needed to generate and store a list of trusted sources. Sources were identified by their Internet host name or IP Address. When our server receives a mobile code request, we compare the hostname to our list of trusted sources. If the name is found, this mobile code is considered trusted so long as it is properly signed or comes through an SSL connection.

Legacy Applications

Legacy applications will be executed regardless of whether or not they otherwise meet the requirements of the mobile code policy. Obviously, designating a piece of executable content as legacy opens your network up to the possibility downloading malicious code, so this option is used with caution. This option, although sparingly used, allows us to retrieve some vital yet unsigned applications from the Internet. Legacy applications are identified by their full URL. When our server receives a mobile code request, we compare the URL to our list of legacy applications. If the name is found, retrieved code is virus scanned and returned to the browser without further analysis.

Blocked Applications

Blocked applications will be prevented from downloading regardless of whether or not they otherwise meet the requirements of the mobile code policy. This feature is similar to a restricted URL list that many proxy servers use. It was included for cases where a few URL's were known to be malicious but a commercial control list application is not available. Blocked applications are identified by their full URL. When our server receives a mobile code request, we compare the URL to our list of blocked applications. If the name is found, the code is blocked.

Virus Scanning

When a piece of mobile code is retrieved from the Internet and found to match the mobile code policy, we perform a virus scan on it before forwarding it to the browser. To do this, we use a commercial real-time virus scanner that has been installed on the host. Any commercial virus scanner will do, we have used virus scanners from both Norton and McAfee. At first, we used the command line implementation of the virus scanners but found that this method resulted in too much latency. Some virus scanning took on the order of 5 seconds per file, which was unacceptable to our users. We then found that our virus scanners ran much faster in the real-time mode. To take advantage of this, we configured them to delete infected files. When our server receives a piece of mobile code, we write a copy to the local disk drive. This triggers the virus scanner -- if it finds a virus, the file will be deleted. If the file remains, we know it is safe to forward to the browser.

Extra Functionality

The sections below describe extra functionality not explicitly defined in our mobile code policy.

Treat MSOffice Documents as Mobile Code

Microsoft Office documents may contain macros, which are considered category 2 mobile code by our mobile code policy. A security hole was found which "allows specially coded macros inside Excel and PowerPoint files to run without the user's knowledge, making it possible for a virus writer to code malicious macros to perform operations such as deleting files, posting data on websites, sending e-mails, and the like."[14] To address this vulnerability, we added an option to treat all MS Office documents as category 2 mobile code. To some users, blocking all MS Office documents represents too much of a sacrifice of functionality, but others require the added security.

Block All Unsigned Applets

Java applets are considered category 2 mobile code by the mobile code policy. As such, unsigned applets may be downloaded if they come from a trusted source. Some administrators prefer that all java applets be signed. We added an option that prevents any unsigned applets from being downloaded.

Display Custom Error Messages

When some forms of mobile code are blocked, we provide custom messages to the browser which state why the code was blocked. Otherwise, the browser will display its default messages.

Log Debug Messages

Selecting this option results in verbose log messages for aid in interpreting any unusual behavior. We use this sparingly since the log files generated can become quite large.

Handling Non-Standard File Extensions

At this point our code was ready for more detailed testing. During this testing we discovered that file extension alone was not always an accurate means for determining the file type. For example, ActiveX files often have the file extension .ocx. However, if an ActiveX file is renamed with another file extension, Internet Explorer will still recognize the file as an ActiveX control and execute it. The use of non-standard file extensions could simply be at the preference of a developer or it could be the intentional attempt of a malicious entity to bypass security. To handle this vulnerability, we needed to develop an alternate means of determining file type. We sought and developed other (trade secret) means of verifying whether or not a file contained executable content. If these means indicate that a retrieved file is a higher category of mobile code than the file extension indicates, the file is blocked. The cost of this addition security measure is that we now have to examine each retrieved file. This is particularly expensive for files with non-mobile code extensions – such as gif – that we heretofore had simply forwarded on to the requesting browser.

This completed the initial development phase of our project. Having satisfied the functional and usability requirements, this method was put into operation at our corporate offices.

After

Initial Feedback

The mobile code server described in this paper is now in operation full-time at our company's head offices. Each browser in our office is redirected through the server we developed. We have not received any complaints regarding latency as we feared at first. The most common report is that some desired piece of mobile code has been blocked. When these events occur, we check out the site and then add the URL to the legacy applications list. One unexpected benefit reported by some Internet users is a reduction in the number of pop-up advertisements.

Continuing Vulnerabilities & Defense in Depth

While the server we developed helps protect our network from inadvertent downloads of malicious mobile code, it is best used as part of a suite of protection measures to provide defense in depth. For example, it is possible for users to bypass the server from their own browsers – one common cause of this is laptop users who are often leaving and reconnecting to the company's network. To prevent users from connecting directly to the Internet, the firewall can be configured to prevent outgoing connections to ports 80 and 443. Additional security can be added by configuring network browsers to disable all ActiveX controls or at least all unsigned ActiveX controls. Furthermore, some browsers permit the user to explicitly grant or deny privileges to applets based on the specific certificate used to sign the applet. These privileges can be denied outright or limited to the minimum set required for the applet to carry out its function. Other defense in depth measures that can be taken include using one of the commercial control list devices mentioned above to block sites known to contain malicious code.

Other continuing vulnerabilities include revoked certificates. Our mobile code server does not currently check Certificate Revocation Lists (CRL) to see if certificates have been revoked. This problem can be exacerbated by the fact that some browsers have flawed CRL checks themselves [15]. Another vulnerability lies in the speed with which viruses and worms can be propagated over the Internet. A virus scanner is no longer a guarantee against malicious code as well. "In today's environment, an unknown threat is much more likely to cause a virus disaster than a known threat due to the speed at which viruses propagate. Anti-virus vendors and end-user organizations can no longer take a reactive approach to combating these threats."[16] Our current work includes finding alternate means for detecting malicious content in mobile code.

Future Plans

Our future plans include adding an interface to our IDS product that will monitor web browsing to check for insider malicious activity. We are also considering adding caching capabilities to our mobile code server. This would serve a two-fold purpose: first, it would reduce latency involved by preventing the same mobile code from repeated downloadings and second, it reduces the chance of downloading malicious code by limiting the number of times we go out onto the Internet. Other plans include developing an email server that checks email attachments for malicious mobile code.

Conclusion

This paper discussed our efforts to improve network security with regard to Internet-based mobile code. We started by developing a mobile code policy that defines the requirements necessary to download the different types of mobile code. We then developed a mobile code server to implement this policy. This server also includes control list and virus scanning capabilities. It also affords us a central location to implement our Internet mobile code policy, thus reducing the ability of individual users can bypass the current policy. This server is currently in use at our company as part of our defense in depth network security strategy.

End Notes

1. Bridwell and Tippett.
2. Stein and Stewart [Ref .9].
3. Stein and Stewart [Ref. 9].
4. “CERT Advisory CA-2001-26 Nimda Worm.”
5. Stein and Stewart [Ref. 10].
6. Stein and Stewart [Ref. 10].
7. Stein and Stewart [Ref. 10].
8. Seruga[Ref. 7].
9. Seruga[Ref. 8].
10. Gray.
11. Gray.
12. “Policy Guidance for use of Mobile Code Technologies in Department of Defense (DoD) Information Systems.”
13. “Policy Guidance for use of Mobile Code Technologies in Department of Defense (DoD) Information Systems.”
14. Kassen.
15. Stein and Stewart [Ref. 10].
16. Bridwell and Tippett.

References

1. Bridwell, Lawrence M. and Tippett, Peter, MD, PhD. "ICSA Labs 7th Annual Computer Virus Prevalence Survey 2001."
2. "CERT Advisory CA-2001-26 Nimda Worm." September 25, 2001. http://www.cert.org/advisories/CA-2001-26.html (4 January, 2003).
3. Gray, Stephen. "Web Content Security: the Requirement, Methods Available and Considerations." March 23, 2001. http://rr.sans.org/malicious/constent_sec.php (4 January, 2003).
4. http://www.websense.com/products/about/datasheets/pdfs/PremiumGroupIII.pdf (4 January, 2003).
5. Kassen, Ron. "MS finds new hole in Excel and PowerPoint." Oct 11, 2001. http://www.geek.com/news/geeknews/2001oct/mac20011011008295.htm (4 January, 2003).
6. "Policy Guidance for use of Mobile Code Technologies in Department of Defense (DoD) Information Systems." November 7, 2000.
7. Seruga, Dr. Jan. "An Investigation of Active Content On The Web." Australian Computer Society Newsletter, May 2001.
8. Seruga, Dr. Jan. "An Investigation of Active Content On The Web." Australian Computer Society Newsletter, June 2001.
9. Stein, Lincoln and Stewart, John. "The World Wide Web Security FAQ." Version 3.1.2. February 4, 2002. http://www.w3.org/Security/faq/wwwsf1.html (4 January, 2003).
10. Stein, Lincoln and Stewart, John. "The World Wide Web Security FAQ." Version 3.1.2. February 4, 2002. http://www.w3.org/Security/faq/wwwsf2.html (4 January, 2003).
11. "Web Security: Protecting Networks from Inappropriate Web Content and Malicious Code." http://www.hoffmanmarcom.com/IS/Examples/Symantec_web_security.pdf (4 January, 2003).