# Common-Sense

# An Alternative Approach to Web Security

## Abstract

This paper discusses general threats to end users from the World Wide Web. It explains why threats to end-users may be of interest in a business systems architecture. It introduces two major forms of threat, mobile code and fraud or theft, and the security models and technical solutions proposed for these threats.

It then examines problems with the technical solutions, including the fact that many traditional security paradigms are less than effective against Web threats. The paper asserts that end-user Web security must ultimately rely on knowledgeable users as part of the overall corporate risk management policy.

Russell M. Shumway

Global Integrity Corporation (www.globalintegrity.com/)
8301 Greensboro Drive
Suite 660
McLean, VA 22102
(703) 917-8460
Fax: (703) 734-2234
Russ.Shumway@globalintegrity.com

## Introduction

Dangers on the Web are receiving a lot of press. From web operators stealing user files to fraudulent stock offers to the hazards of ordering by credit card, advice is widely available and help abounds to secure your business or home computer. In the academic press, significant research has been done on various security models. From the Java Sandbox to the ActiveX Trusted Code model, researchers are discovering vulnerabilities in the applications every day.

These new threats are important to both businesses and individuals. The 'Net is no longer the province of a few restricted academic and government organizations. Electronic commerce is growing at a previously unheard-of rate, and there are merchants such as Amazon.com which have virtually no physical presence. The ability to use the Web to research, download software, and order supplies and services have made access a requirement for both private individuals and companies. Traditional security paradigms such as firewalls, however, are often ineffective against the kinds of threats which exist on the Web; content is difficult to screen. For example, Balfanz and Felten discuss ways to pass Java applets through the firewall in a recent paper.[1] Secondly, end-user threats via the Web are fundamentally different than many other vulnerabilities. Attacking a network using a Sendmail bug is like a shooting a target with a precision weapon. Putting a Trojan control which steals credit card numbers on a Web page is like putting out a minefield. Web threats require the victim to seek out the attacker, either wittingly or unwittingly.

I would suggest, however, that Web security is not a technical issue, *per se*. Although technical solutions can certainly assist in providing a reasonably secure environment, the ultimate responsibility for security has to rely on knowledgeable users and a sound risk-management strategy. The bulk of this article may appear to address individual user concerns and vulnerabilities, but they are an integral part of a sound corporate information security policy as well. When users start taking work home, checking their e-mail, or looking up information on the Web for either personal or business reasons, their individual security becomes a corporate concern. If employees are working at home, their home PC is, in effect, behind your corporate firewall. A brief disclaimer: This paper will discuss Web security only from the client perspective. Security of Web servers or hosts is beyond the scope of this document.

## Threats

The major threats to the end user can be divided into two groups. The first consists of mobile or downloadable code. This includes Java applets, ActiveX controls, documents containing macro or scripting languages, or actual executable programs. The second division is the threat of simple fraud or theft. This is logically no different from any fraudulent offer; the web merely makes it easier. E-fraud offers many of the same competitive advantages to criminals that e-commerce does to businesses.

---

[1] Dick Balfanz and Edward W. Felten, *A Java Filter*, *Technical Report 567-97*, Department of Computer Science, Princeton University, October, 1997, http://cs.princeton.edu/sip/pub/javafilter.html.

## Mobile Code

One definition of Mobile Code is "a low-level language that does not depend on the machine architecture, unlike assembly languages for example. Mobile codes can be transmitted across the network and executed on the other end regardless of the computer type. The Java *bytecode* is one example of a *mobile code*."[2] The important point is that (1) generally Mobile Code is cross-platform (although that's not always true, ActiveX will only run under Windows-series operating systems) and (2) it can be easily distributed across the network. As the constraints of HTML became more apparent, developers looked for ways to increase the capabilities of the internet and browser. Java is (probably) the first important example of mobile code (although the macro programming languages in some Microsoft products could arguably share that distinction).

The major classes of mobile code one is likely to encounter include Java (in its browser implementation, as opposed to its use as a general programming language), ActiveX, macro languages (chiefly in Microsoft products, although they are also available in Lotus and WordPerfect), and executable programs. If the situation wasn't already complex enough, the executables can also contain more traditional viruses. As the features available on the web increase, threats also may increase. For example, some programs use what they call an "active setup" where the downloaded program file automatically extracts other compressed files and connects back to the home site to continue downloading files. Furthermore, the capabilities (and hence the threats) of each type of control have converged dramatically in the recent past. While this classification system is somewhat arbitrary depending on whose definition is used, I will still discuss four somewhat distinct forms of mobile code.

- **Java and JavaScript**

The Java security model is often described as a "sandbox." This sandbox defines what specific actions are permitted or prohibited to the applet. Different rules apply to local applications (those which originated on the client machine) and applets downloaded across the network. For example, a downloaded applet is generally not allowed to read, write, or delete files, while a local one is.

Hostile Java applets can either (1) cause a denial of service (DOS) by overwhelming system resources or (2) perform actions or access resources outside the sandbox. While DOS is indeed a threat, it is not a major problem for the average user. Faced with a DOS attack from a Web site, they can simply re-boot and choose not to visit the site again. A DOS attack against a corporate user could conceivably be more troublesome, but the attacks would again be terminated as soon as the browser is closed. It is hard to postulate Java DOS attacks as anything more than a nuisance. An applet which can access resources or execute actions on the client computer, however, is a quite different question. The basic problem is that, in order to be useful, an applet

[2] Mark Austin, *A Short Java Course*, Department of Civil Engineering, University of Maryland, 1996, http://www.isr.umd.edu/~doc/Java/Course/.

needs **some** access to local resources, "after all, who wants to run a program which is not allowed to do anything?"[3]

Defining what resources ought to be allowed or prohibited is not a trivial task. The history of the Java model shows that permitted resources can often be exploited in ways never foreseen by the system designers. For example, it was logically decided in the original security model that a Java applet should only be allowed to allowed to open a network connection *back to the server from which it came*. This was implemented in Netscape 2.0 by using a reverse DNS lookup to determine the IP address of the originating server. By controlling how the DNS lookup was performed, however, it was possible to open a network connection to a third machine. This applet could then be used to attack machines behind a corporate firewall and resulted in a CERT alert. The rule was changed in Netscape 2.01 so that the connection could only be made *back to the exact IP from which it came*.[4]

Since local Java applications have much different permissions than downloaded applets, a vulnerability called the "CLASSPATH attack" allows an applet to masquerade as a trusted local application. This defeats the sandbox restrictions. This vulnerability has reportedly been fixed by a patch to the Java compiler. However, it is postulated that it is possible to directly produce byte code which is not compiled or verified and may still exceed its authorized resource level.[5] An excellent history of the Java security vulnerabilities and their patches is available from Rothfuss and Parrett in the Proceedings of the 20th National Information Systems Security Conference.[6] The Secure Internet Programming group at Princeton also has a good overview of Java Security.[7]

In more traditional Java implementations the basic rule was that an applet loaded from the user's machine had generally free access to system resources; one loaded from the internet was restricted to the sandbox.[8] In more recent implementations of the Java Virtual Machine, however, Netscape has implemented a digital signature technology for Java applets. This allows "trusted" applets to access resources "outside the sandbox."[9] Digital authentication will be discussed further in the ActiveX section, but it is interesting how much the two previously distinct technologies are merging in their capabilities and potential threats.

JavaScript unfortunately shares a name with the programming language and is often confused with it. While Java is a full-fledged programming language which runs in a virtual machine, JavaScript is "a scripting language embedded in HTML . . . that can be interpreted by

---

[3]Gary McGraw and Edward W. Felten, *Java Security: Hostile Applets, Holes and Antidotes*, Wiley Computer Publishing, New York, 1997, p. 29.

[4]*Ibid*., pp. 60-69.

[5]Mark Ladue, *When Java Was One: Threats From Hostile Byte Code*, Proceedings of the 20th National Information Systems Security Conference, October 7-10, 1997, pp. 104-115.

[6]James Rothfuss and Jeffrey Parett, *Go Ahead, Visit Those Web Sites, You Can't Get Hurt . . . Can You*?, Proceedings of the 20th National Information Systems Security Conference, October 7-10, 1997, pp. 84-85.

[7] Princeton University, *Secure Internet Programming*, http://www.cs.princeton.edu/sip/.

[8] Sun Microsystems, *Frequently Asked Questions – Java Security*, http://java.sun.com/sfaq/.

[9] Taher Elgamal, Sean Cotter, *et al*, *Netscape Security: Open-Standard Solutions for the Enterprise*, January 1998, http://developer.netscape.com/docs/manuals/security/scwp/index.htm.

the . . . HTML browser."[10] Java interfaces through the Java virtual machine with the operating system, JavaScript interfaces through the web browser and commands it to perform actions. Typical JavaScript implementations on Web pages include opening a pop-up advertising window or changing the contents of the status line in the browser. JavaScript, however is not without its own set of problems.

While Java has a formal security model, JavaScript security is based on the implementation of the browser. Several problems have been reported with JavaScript, including the ability to upload or retrieve arbitrary files from a user's machine. These have reportedly been fixed in newer browsers, but denial of service attacks are still possible.[11] Felten et al use JavaScript extensively in their Internet Spoofing demo.[12]

- **ActiveX**

ActiveX is defined by Microsoft as "A marketing name for a set of technologies and services, all based on the Component Object Model (COM)."[13] They are developed in native code such as Visual Basic or C, and may consist of executable files, Dynamic Linked Libraries (DLLs), or various executable extensions (OCX). Some researchers also include setup files (INF or CAB).[14] Microsoft also states that Java applets are also ActiveX controls,[15] but I will continue to discuss them separately as the threats and vulnerabilities are generally different. ActiveX controls are not subject to the same sandbox constraints as Java applets. Running under an open operating system such as Windows 95, these controls can execute virtually any command, and can, in fact even modify system files or the operating system kernel. For example, Sun Microsystems demonstrated a control called "OuterLimits" by Fred McLain which formatted the hard drive, searched Quicken or TurboTax to find sensitive financial data, and opened the CD-ROM drive with a message which said "Here's your cup holder."[16] The Chaos Computer Club demonstrated an ActiveX control which would start Quicken on the target machine and initiate a

[10] Roedy Green, Roedy Green's Java Glossary, 1998, http://oberon.ark.com/~roedy/gloss.html.

[11] John Robert LoVerso, *JavaScript Problems I've Reported*, The Open Group Research Institute, 14 Nov 97, http://www.osf.org/~loverso/javascript/. This URL includes an excellent demo of a JavaScript Denial of Service attack.

[12] Edward Felten, Dirk Balfanz, Drew Dean, and Dan Wallach, *Web Spoofing: An Internet Con Game*, Proceedings of the 20th National Information Systems Security Conference, October 7-10, 1997, pp. 95-103.

[13] Charlie Kindel, Microsoft Developer Relations Group, *An Overview of ActiveX*, http://www.microsoft.com/com/slides/kindel1.zip.

[14] David Hopwood, *A Comparison between Java and ActiveX Security*, presented to the 14th World Conference on Computer Security, Audit and Control, 1997, http://www.users.zetnet.co.uk/hopwood/papers/compsec97.html

[15] Charlie Kindel, Microsoft Developer Relations Group, *ActiveX and the Web: Architecture and Technical Overview*, http://www.microsoft.com/com/slides/kindel2.zip.

[16] Alex Lash, *Sun pays for ActiveX attack*, c|net, April 3, 1997, http://www.news.com/News/Item/0,4,9390,00.html.

wire transfer of funds.[17] There are virtually no technical limitations to what an ActiveX control can be asked to do, and, unlike Java applets, their affects are not audited or logged by the system.

The security model for ActiveX is outlined by Microsoft in their Security Management Architecture White Paper, available on their web site.[18] The model consists of a technology called Authenticode which determines if a control should be loaded. Basically it uses cryptographic signatures on the control to see if the code is authentic and the author is recognized, as verified by a trusted third-party provider. If the code contains a valid digital signature, the control is downloaded and given full access.

In theory, this seems an ideal solution. Only trusted code is allowed access, and it is given full access so it can perform useful tasks. There are, however, still dangers in this model. In 1996 an ActiveX control called Internet Exploder, which performed a clean shutdown of Windows95, was released by McLain (The author of OuterLimits, above). McLain obtained a digital signature from VeriSign. Since the code was properly signed, and VeriSign listed McLain as a trusted programmer, Authenticode's verification allowed the control to be used. VeriSign subsequently revoked McLain's signature key, but this incident is often used to claim that the digital signature process, while technically correct, is fatally flawed in implementation. Others claim that the revocation of McLain's key demonstrates that the system works properly.

Hopwood also poses two additional problems with the model. Regardless of whether the signed object is an ActiveX control or a Java applet, signatures authenticate the control only, not the web page itself. This allows a signed control to be replaced by:

> "an unsigned control,
>
> a control signed by a different principal, or
>
> a different control (including previous versions of the expected one) signed by a different principal."[19]

A signed control can also be used in a different context from where it was originally intended. Hopwood presents an illustration where a signed control written for use on an intranet could be posted on an external web page.[20]

Recently other issues with the Authenticode model have been raised. A posting to BugTraq claims that both the encryption algorithms and their implementation by Microsoft are flawed. The article states that an ActiveX control can be used to steal the signature keys of other controls, which can then, obviously, be used to sign new code under someone else's name. Microsoft has not responded to this posting, which is unverified at this time.[21] Even if a control is

---

[17]Simson Garfinkel and Gene Spafford, *Web Security and Commerce*, O'Reilly and Associates, Cambridge, 1997, pp. 79-80.

[18] *Microsoft Security Architecture White Paper*, http://microsoft.com/ie/security/ie4security.htm

[19] Hopwood.

[20] Hopwood.

[21]Peter Gutmann, *How to recover private keys for Microsoft Internet Explorer . . .*, originally posted on BugTraq on January 20, 1998. Also posted on his personal web site at

properly signed and not modified, it may be possible to use it in ways not foreseen by the developers. Any subscriber to BugTraq can relate stories of buffer overruns resulting in unwarranted access to system resources.

- **Plug-ins and Helpers**

Plug-ins are downloadable programs which are installed into the browser. They are then used to view certain content. Since plug-ins can access any resource they can be used to execute arbitrary code on the target machine. This is true whether or not the plug-in is developed by the attacker. For example, the widely-used Shockwave plug-in (used to play multimedia presentations) was shown to be able to read any file on the target machine.[22] If the plug-in is specifically developed with a specific attack in mind, it can obviously be much more efficient.

Helper programs run outside of the browser and are executable applications in their own right, but their effect is similar. For example, a viewer program might be required to properly display multimedia images. The user would be required to download and install the helper application, then download the content and view it in the helper. The dangers in this are easy to see. A Web site known as "sexygirls.com" offered free pornographic images to subscribers if the downloaded a special "viewer." Unknown to the users, this viewer disconnected the user from their ISP, placed a toll call to Moldavia, and reconnected the user. The long distance phone charges, of course, went to the owner of the Web site. A federal investigation resulted in refunds totaling $2.74 million to over 38,000 consumers.[23]

- **Macro Programming Languages**

The threats posed by macro programming languages combine the ability to execute arbitrary commands (like helpers and plug-ins) with the self-replicating capability of viruses. Macro viruses are the fastest growing segment of the virus taxonomy and are especially difficult to eradicate. Users who would never think of downloading shareware from a bulletin board will open a Word document in an e-mail without blinking. Most of the macro viruses currently seen in the wild are relatively mild, with effects simply ranging from destruction of data to annoyances. However, given the sophistication of macro scripting languages, it is easy to imagine much more sinister effects.

For example, Microsoft's Visual Basic for Applications allows macros to declare functions and call them from Dynamic Link Libraries (DLL's). It would be relatively trivial to write a macro which calls a function from the URL.DLL to automatically connect to the Internet and transmit data. This DLL can even dial the modem, log into the ISP, then re-direct the browser to any valid URL. Obviously it can also pass the user's login and password to the connection. By using Object Linking and Embedding technology, any OLE-compatible program can be made to yield its data to the calling macro. A macro can cloak itself by infecting other documents, perhaps even of a different type.

---

http://www.cs.auckland.ac.nz/~pgut001/pubs/breakms.txt. Russ Cooper's responses on NTBugTraq may be found at http://www.ntbugtraq.com/archives/Default.asp.

[22]Garfinkel and Spafford, p. 72.

[23]Press Release, November 4, 1997, Federal Trade Commission, *Victims of Moldovan Modem "Hijacking" Scheme to Get Full Redress Under FTC Settlements*, http://www.ftc.gov/opa9711/audiot-2.htm.

## Fraud/Theft

As users tread through the minefield of the Web, they are also vulnerable to good old-fashioned fraud. While the vulnerabilities above may be used as the means of attack (as in the sexygirls.com example), it is also possible to simply put up a fraudulent offer and wait for the customers to come. Marcus Ranum, in a 1995 white paper, discusses several important issues in electronic commerce. Two of these are of specific interest to the consumer:

"How do customers know they are dealing with a legitimate business?" and

"What aspects of the transaction does the customer expect or the law require to be private?"[24]

The question of whether of not the business is legitimate is not trivial. In a discussion during last year's National Information Systems Security Conference a panel member stated that it was not economically feasible to counterfeit nickels; the cost of making them was more than the gain. The Internet has changed that. Even if only a small percentage of customers take advantage of the offer, the ability to target literally millions of victims at an extremely low cost makes it profitable. Once the scam is run (in a matter of seconds), the site can fold, leaving no trace for investigators. The SEC is especially concerned with the possibility of fraudulent Initial Public Offering proposals and "pyramid schemes" over the Internet.[25] An excellent general resource for internet fraud is provided by the National Fraud Information Center, a division of the National Consumer League at http://www.fraud.org/.

The Secure Sockets Layer (SSL) protocol prevents the interception of data (for example, credit card numbers) by encrypting it during transmission. Both Netscape Navigator and Internet Explorer support SSL. While this is an important technology and can protect users from some attacks, it is important to remember that nothing is implied by the presence of a secure connection. A fraudulent site can put up a secure server just as easily as a legitimate site.

It is also technically possible (although not yet seen outside of the laboratory) to "hijack" a legitimate connection to a commercial site through the use of Web Spoofing. This can either capture data in transit (even if the user thinks they are connected to a secure site) or to divert data to the attacker's server. Edward Felten *et al* developed an excellent demonstration of this technology; more information is available on their Web page at http://www.cs.princeton.edu/sip.[26] None of the typical security policies previously mentioned (the Java sandbox, Authenticode, SSL) can fully protect against this attack.

[24] Marcus Ranum, *Electronic Commerce and Security*, ca. 1995, http://www.clark.net/pub/mjr/pubs/ecom/index.htm.

[25]Office of Investor Education and Assistance, Investment Fraud and Abuse Travel to Cyberspace, June 1996, http://www.sec.gov/consumer/cyberfr.htm.

[26]Felten, et al, *Web Spoofing*, pp. 95-103.

## Countermeasures

So, given all these threats, how does one protect one's self from the virtual equivalent of a mugging? As shown above, both the Java Security Model nor Authenticode have vulnerabilities which can be exploited. What can one do?

### Technical Solutions

First, it is important to realize that only technical problems can ever be fixed by technical solutions. If a user is going to send a credit card number to a fraudulent web site, no amount of enhanced security in the browser or web can prevent that. Having made that comment, technical solutions do still offer some improvements in security. Both Netscape and Microsoft have been generally responsive in releasing patches to fix technical problems. Most of the Java and JavaScript exploits illustrated in the referenced papers have been fixed, **provided one is using the latest version of the browser**. As new exploits are discovered, both major vendors will undoubtedly continue to issue patches and updates. Antivirus software can also help, at least in detecting documents or archives containing viruses.

Cryptography offers some help, chiefly through digital signatures of downloaded objects and the use of SSL to create a secure link between the browser and web site. But cryptography cannot universally prevent certain forms of attack. Users may still choose to download unsigned objects. JavaScript, since it is embedded directly in the HTML document, may bypass the signature check completely. Digital signatures on an object can only protect the object, not the entire context.[27] And, while secure links may protect sensitive data during transmission, nothing is available to prevent a user from simply sending that data (encrypted en route or otherwise) to a fraudulent site.

### Disable Mobile Code

One option, which offers excellent protection, is to disable the downloading of all active content. Simply disable Java, JavaScript, and ActiveX in the browser. Ensure that no plug-ins are loaded. Of course, to be completely safe, all helper applications need to be turned off as well. For example, a hyperlinked Word Document or PowerPoint presentation should be saved to disk and then read using a viewer which does not enable the macros.

The obvious problem with this strategy is that many Web sites, especially those with any interactive features, require some form of active content. Disabling all content is the virtual equivalent of not carrying money--you can't be robbed, but you can't buy much either. Secondly, as businesses implement new technologies, Java or ActiveX may be required to use applications on the corporate intranet.

Microsoft's latest browser, Internet Explorer 4.x, offers the user the ability to selectively enable and disable mobile code through the use of 4 security zones. These divide the Internet into The Intranet (local sites, which are essentially fully trusted), Trusted Web Sites (defined by the user which are mostly trusted), The Internet (the rest of the world, which is generally not trusted), and un-trusted Web Sites (hackers-r-us.com, which is, obviously, not trusted at all). For each zone, the user can define settings which range from completely disabling all active content, to

[27] Hopper.

prompting the user before running any controls, to fully allowing everything. This approach has the advantage of allowing much more customization than was previously possible, but it is complex. Secondly, it requires the user to define all the sites in the Trusted and Untrusted Web Sites settings. These are the two most useful zones since they define exceptions to the standard security, but they are empty in a default setup.

With the release of Netscape Navigator 4.x (as part of the Communicator suite), Netscape also now supports some implementations of ActiveX and digital signing of both ActiveX controls and Java applets. Netscape's security architecture allows trusted applets to run outside the sandbox, unsigned or untrusted ones are still restricted.[28] Although this is a slightly different approach than Microsoft's security zones, it offers similar functionality.

Java and ActiveX could also be blocked at the firewall but allowed to run on internal web pages. There are difficulties with this approach; it is at least theoretically possible to pass Java applets and/or classes through a firewall (see the Java Filter paper by Balfanz and Felten). Some vendors are selling solutions which purport to block "hostile" applets but there are significant doubts as to the efficiency of such blocks. Mark LaDue, in a review of such a product, discusses these difficulties at length.[29]

## Don't Go There

The second choice is to simply not go anywhere where the user can hurt. Stay out of the dark alleys and the muggers won't bother you. While this is good advice, it ignores the recent dramatic growth of EC. Deciding which alleys are dark and which are well-lighted is not that easy a task. It also ignores the threats of Web spoofing or session hijacking which can strike even on the virtual Main Street at noon.

It has been argued that credit card fraud on the Web is no more inherently dangerous than in any other place. After all, carbons can be found in the trash; con men can always contact their victims directly over the phone. This argument fails to address the fact that the Web gives one a certain anonymity (obviously one is not fully anonymous, but one doesn't have to rent an office or phone to set up a business site on the Web, either). It also doesn't address the fact that the all-pervasive nature of the Web allows previously uneconomical scams to suddenly become profitable. Consumers generally have some level of confidence in the reliability of the merchants, and this is difficult to ensure or assess over an electronic communication.

Most credit cards carry only a $50 liability in the case of theft. Perhaps it's worth the chance of losing the money to engage in this new marketplace. While this may be true, most people don't want any of their hard-earned money going to criminals. The limit doesn't apply to the credit card company; it's liable for the full amount; so that cost is going to be passed on to consumers in the end. Similarly, if merchants lose enough, they will either raise prices or simply stop offering on-line ordering. Incidentally, while credit cards have some level of protection

[28] Elgamal, Cotter, *et al*.

[29] Mark D. Ladue, *Drowning in the Surf: A Review of [Name Deleted*], 1997, http://www.rstcorp.com/hostile-applets/drowning.html.

based on the "Truth-in-Lending Act," debit cards (which store a pre-paid balance of funds which is used for payment) may have much higher limits on liability.[30]

In an interesting report, the National Fraud Information Center surveyed the most common methods of payment for internet fraud reported in 1997. Checks and money orders each outnumbered all credit cards, and cash transactions were more popular than all but one credit card.[31]

## Common Sense

The above two approaches, while providing some level of security, are extreme reactions. Sometimes people do have to carry cash, and they do have to venture out into the streets. Carrying the mugger metaphor further, how does one approach the problem in the real world?

One approaches personal security through a combination of strategies. When traveling, many people don't carry a lot of cash, so they can't lose a lot at a time. They don't go into bad areas (or at least, are very careful if they do). They pay attention to their surroundings. When walking out to the car at night, they watch for trouble before leaving. But they don't lock themselves up in a fortress and never leave the house.

In short, people assess risk all the time. Web and Internet security is no different, users and administrators are just less comfortable with the methodology in this new environment. Before this strategy is successful, users need to be educated on the threats. We in the Information Security business are extremely bad at this. We tend to make the discussion of security overly-complex and technical, which alienates the user. Then, complaining about how clueless the user community is, we implement a technical solution.

The problem with a purely technical solution is that they invariably have one or both of the following two failings:

1. They are too efficient, and prevent legitimate users from accomplishing legitimate tasks, or
2. They don't prevent security breaches.

In any case, holes are appearing faster than patches. This is not to say that technical solutions have no value, only that they are not panaceas.

Both the Java "sandbox" and Microsoft's Authenticode are excellent examples. The Java model often prevents applets from accomplishing anything useful, let alone threatening. It does not, however, prevent all security breaches as demonstrated by the articles in the footnotes. Similarly, Authenticode can prevent users from downloading or executing unsigned content, even if it is benign. It can also be fooled by Web spoofing.

[30] Board of Governors of the Federal Reserve System, *Report to Congress on the Application of the Electronic Fund Transfer Act to Electronic Stored Value Products*, March 1997, p. 43.

[31] Susan Grant, *Fraudulent Schemes on the Internet: Remarks to the Senate Permanent Committee on Investigations*, February 10, 1998, http://www.fraud.org/news/1998/feb98/021098.htm.

## Conclusion

How then, do we as security professionals and administrators fully protect our users and our networks from Web threats? Simple. We can't. But then again, we can't fully protect our them from other threats either. Even in a completely closed system, data can still enter and leave the building. Aldrich Ames took classified computer files out of the CIA by copying them to floppies and carrying them out the front door.[32] I have seen viruses on a closed network, and I find it hard to believe that it was a case of spontaneous generation.

What we can do, however, is two things:

1. Provide users with the appropriate security tools or resources. Ensure that the users have access to the latest versions of their browsers. Make the patches or updates available or, if the network configuration permits, update user installations remotely. Provide them with (current) antivirus software.

2. Define a **reasonable** and **achievable** security program which addresses these issues. Users need to know what to do when faced with a question about executable content, whether this is at home or at work. Now that we can work from home (or at least work **at** home), the security of the user's home PC is now a business concern.

The first is easy; the second, much less. Again it seems that the answer comes back to education. But information security has always been about education. Technical controls can only protect the box; they can't protect the data before it enters or after it leaves the box, whether that's on disk, paper, or in a person's head. Why then should the threats of Web security be any different from any other kind of information security? In the end, protecting the information (personal or financial) is what we're all trying to do in both our personal and business lives.

Both major vendors provide an excellent start to managing risk by site and content. The setup, however, is extremely complex for the average user. Power surfers can customize the systems and their permissions to allow some very sophisticated security models, but they will be required to make conscious choices about sites. System administrators can remotely set the security permissions, but, again, the user has to make a choice when the browser asks whether or not an applet or control should be loaded.

How do they make that choice? They need to make it on the basis of information about what the applet or control can or cannot do if downloaded. Users should understand that using active content can both provide significant enhancements and significant danger. Similarly, they need to understand the dangers of macro viruses and Trojan horses without subjecting them to all the virus hoaxes making the rounds. The final piece is that the decision to allow or not allow active content or executable programs depends very much on the source of the content. Downloading a control, software patch, or applet from www.microsoft.com is very different than downloading it from a pornographic site located somewhere in the third world. Ordering a product from amazon.com is different from ordering it from that mythical pornographic site. Users need to weigh the risks and ask themselves whether the company is likely to be there

[32]*The Aldrich H. Ames Case: An Assessment of CIA's Role in Identifying Ames as an Intelligence Penetration of the Agency*, October 21, 1994, para. 57, posted on http://www.loyola.edu/dept/politics/hula/hizrept.html.

tomorrow.  Users make these kind of decisions all the time when they order items over the phone. They're betting that the company would rather stay in business than clean out their bank accounts and skip town.  The issue is again a matter of training and education.

In conclusion, Web security is not significantly different from any other form of information security.  It provides far greater access to far more users, but the major issues are similar to those previously posed by client-server technologies, networked computers, and de-centralized processing.  In the end, technical solutions can only provide partial solutions, users must be responsible for information security since users are responsible for the information.