Go Ahead, Visit Those Websites, You Can't Get Hurt ... Can You?

June 3, 1997

James S. Rothfuss
rothfuss1@llnl.gov
Lawrence Livermore National Laboratory
P.O. Box 808 L-315
Livermore, CA 94550

Jeffrey W. Parrett
jeff_parrett@peoplesoft.com
PeopleSoft
4305 Hacienda Drive
Pleasanton, CA 94588

This work was performed under the auspices of the U.S. Department of Energy
by Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48
UCRL-JC-126433

ABSTRACT

Browsing (surfing) the World Wide Web (the web) has exploded onto the Internet with an unprecedented popularity. Fueled by massive acceptance, the web client/server technology is leaping forward with a speed that competes with no other software technology. The primary force behind this phenomenon is the simplicity of the web browsing experience. People who have never touched a computer can now perform sophisticated network tasks with a simple point-and-click. Unfortunately, this simplicity gives many, if not most, web wanderers the impression that the web browser is risk free; nothing more than a high powered television. This misconception is dangerous. It creates the myth that a user visiting a website is immune from subversive or malicious intent. While many want you to believe that surfing the web is as simple as using any other household appliance, it is not like surfing television channels, it is bi-directional. You can learn a lot of useful information from websites, but, either directly or indirectly, websites can also learn quite a bit about you. Of even more concern is a website's potential ability to exert control over the local computer. This paper tries to consolidate some of the current concerns that you should consider as you jump into the surf.

WHO IS THE BAD GUY: SERVER VS. CLIENT

In the past, most network computing was done between two consenting parties with some level of trust between the parties (there are a few exceptions, such as anonymous ftp). With the web architecture the model has changed radically. In any web interaction there are two sides: the client (your browser) and the server (the website). In most normal web configurations, websites open themselves to some or all of the network population without any authentication or authorization. Likewise, the web browser often connects to the server with limited knowledge of who owns or operates the web server. Both must exist to make the web work, but both should be concerned about the intentions of the other.

The website owner must protect himself from those members of the internet community intent on doing harmful or embarrassing things to the server. There have been many cases where infiltration into websites of high-profile agencies has resulted in damage and great embarrassment. These website break-ins tend to be well covered by both the conventional and

PAN EX1028
Palo Alto Networks v. BT Americas
IPR2023-00889

internet news media. Perhaps not so well publicized are the threats to you when you are browsing the web.

The browser is the software that you run on your local computer, enabling it to access information from a website. Currently the two premier browsers are the Netscape Navigator and the Microsoft Internet Explorer. The individual using the browser is often under the delusion that their web connections are completely anonymous and free from any intrusion into their local computing environment.

Web connections have another distinction over older network connections. It is the reason that website Universal Resource Locators (URLs) appear on television, magazines, and product labels while ftp addresses never got beyond the "computer geek" stage. Web surfing is easy. You do not have to know about directory structures, filename syntax, ftp commands, and so on. The bad news is that you do not have to know about Java, JavaScript, JScript, downloading, plug-ins, helper applications, compiling, CGI, or the connected host. You do not have to know where you are, where you are going, what is happening, or what is going to happen with the next click. There may or may not be visual clues. Even if visual clues are presented, you cannot always trust them. It is easy to wander the web in reckless abandon. Unfortunately, the safety of the client is not insured and your lack of awareness may be subjecting your local computing environment to great risk. Even if you are attentive, the current browsers' lack of adequate logging and security options often make it difficult to take a cautious route. The remainder of this paper will focus on educating you about the threats, concerns, and protections on the client side of a web interaction.

TOOLS AND ATTACKS

As the desire to increase the web server capabilities gains momentum, the interactions between the client and web server becomes more complex. As the complexity grows, so does the risk. Following is a discussion of some of the more popular technologies, and associated threats, which are being incorporating into the web.

CGI: Can Get Information

The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as web servers [1]. The programs can be written in any programming or scripting language. The CGI program communicates to the web browser by creating HTML text "on the fly" and sending it across the network to the browser. In effect, a CGI program can create a dynamic HTML page. Certain browser and server environment information is made available to the CGI. Refer to Table 1 for an example of this information. The CGI program also can obtain information from the browser user via an HTML forms dialog. As with any program, the information gathered by a CGI program can be processed immediately, or stored indefinitely at the website for some future use.

A great deal has been written about the perils of using the CGI. However, most of the risks associated with the CGI are to the web server [2][3]. From the browser perspective a URL has been requested and a page was returned. While a program cannot mount a direct attack on the browser through the CGI, dynamic execution of programs can enhance the server's ability to obtain sensitive information or to mount tertiary real-time attacks on other vulnerabilities of the client operating system.

An example of a unique attack that CGI makes possible might be called "cyber social engineering." With CGI comes the ability to produce an automatic process which, through the use of seductive web pages and non-threatening questions, lures the victim into divulging valuable information. CGI allows the website to dynamically customize the sequence of web

pages and form questions with each phase building on the victim's past website interactions. With this stored information, a website could build visitor dossiers and eventually, through days, weeks, or months of seemingly innocuous question/answer sessions, piece together damaging information.

While CGI scripts cannot directly manipulate files on a client system; they can, along with help from HTML forms, upload files with the browser user's consent. Fortunately, the CGI script cannot be pre-loaded with a defined file name and therefore requires the browser user to select the input file from their local system. However, once again through the use of cyber social engineering, the victim could be tricked into consenting to an undesired file upload.

Table 1
Information Available From The Server

The following table was generated by a facility at http://hoohoo.ncsa.uiuc.edu/cgi/examples.html. Anyone can go to this server and trigger the cgi script from their browser. The server will return the information the server is able to obtain from the client.

CGI/1.1 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = NCSA/1.5.2
SERVER_NAME = hoohoo.ncsa.uiuc.edu
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = POST
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
HTTP_USER_AGENT = Mozilla/3.01 (Macintosh; U; 68K)
HTTP_REFERER = http://hoohoo.ncsa.uiuc.edu/cgi/examples.html
PATH_INFO = /foo
PATH_TRANSLATED = /u/Web/foo
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST = cso.llnl.gov
REMOTE_ADDR = 128.115.85.123
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 9
ANNOTATION_SERVER =

Another harmful aspect of CGI programming is that an automated real-time attack can be mounted against the client. The CGI does not inform the website visitor that it has started a program unless specifically programmed to do so. The visitor may be lead to believe that they are looking at a passive HTML page when, in reality, a program has been triggered that is actively attacking the client through one or more of the other network facilities (i.e. sendmail, ftp).

Beware Of Servers Bearing Gifts

Have you ever clicked on a website URL and sat back as a file was downloaded and opened by another local application? Through the use of MIME types, servers can tag files in a fashion which enables the browser to recognize them and open the proper application, "helper", or "plug-in". These applications usually reside on the client before the website was visited, so they are all assumed to be trusted programs. If they do not exist you will often be given instructions on how to obtain them.

Envision the following scenario: you arrive at a website called www.sexygirls.com which advertises several seductive documents. When you click on the URL, the document is downloaded and you are presented with the instructional pop-up "Cannot open document, please install the document viewer". Of course, you do not know where to get the viewer, but those courteous folks who prepared the website have supplied a URL that will download the viewer onto your client. Wanting to get a look at the document as soon as possible, you click on the download URL and then click on the document. The document is easily opened by the locally running viewer. What is not expected is that while you are viewing the document your modem sound is turned off, the connection to your internet service provider (isp) is dropped, and your modem dials up another isp using a 900 phone number. From the time this switch takes place until the time you disconnect, the new isp will be charging $3.00 per minute for their connection. This scenario is real. A civil action is currently in the courts against the owners of www.sexygirls.com [4][5][6].

Caution must be taken in downloading seemingly valuable extras. You are essentially downloading native code to run on your system and trusting the author not to do bad things. The most common defense is to limit your helpers and plug-ins to reputable companies.

Even some reputable helpers will help you in ways you never imagined. The ShockWave web browser plug-in from Macromedia provides the ability to play multimedia presentations created with their Director application which are downloaded via the web. The creator of a ShockWave multimedia presentation can include custom code in their presentation using the Lingo language. So even if Macromedia is conscientious in their creation of the ShockWave plug-in, they have no control over what developers do in their documents with Lingo. The danger is further compounded by the fact that early versions of Lingo could be extended by the developer to the point of making local system calls based on the platform on which it is running[7]. While to date there are no recorded incidents of malicious code being downloaded as part of a ShockWave document, there is a concern that a virus or trojan horse could be created as part of a downloaded document, similar to the Microsoft Word macro virus[8].

Once this was brought to the attention of Macromedia, they responded by adding security features ShockWave. Two sentences are offered by Macromedia to describe their enhancements:

> The new security-enhanced Shockwave player is now available. This version fixes the potential security issues recently reported that relate to Shockwave and Netscape.[9]

This meager explanation does not inspire confidence that Macromedia has adequately covered all the security issues. Even if the new version is “safe”, thousands of users of the older versions of ShockWave are still vulnerable. This same attack can be applied to any helper or plug-in that uses a macro language to enhance their capabilities.

Java, It's More Than A Cup Of Coffee

Java is a new programming language created by Sun Microsystems to correct many of the annoyances found in the C++ language and to take advantage of a new "network paradigm" of computing. An application written in the Java language and delivered over the web is called an applet. An applet written in Java is compiled into bytecode. Bytecode is a machine-independent code that is downloaded to the browser where it is either run on a "virtual machine" that is embedded in the client's browser or is converted by a Just In Time (JIT) compiler into native machine code. In either case it runs locally on your computer. With a simple point-and-click operation, a remote website can download and run a program on your client with no prior indication. Java is not always looked upon as a positive advancement by those concerned with security.

Sun did attempt to build some amount of security into Java Having the history of C and C++ to draw upon, the developers of Java made several improvements that keep the less-than-vigilant programmer from creating security holes. Most of these improvements stem from moving memory management from the programmer's hands and into run-time and the implementation of the “virtual machine”.

Table 2
Security Related Java Flaws

Date	Browser effected	Vulnerability	Details	Corrected	Ref.
Nov. 95	HotJava 1.0a3	Arbitrary host connections from client if covert channel is established.	No restrictions on accept() system call.	In JDK 1.0 and Navigator 2.0	
	HotJava 1.0a3	Covert channel	SMTP/e-mail	In JDK 1.0 and Navigator 2.0	[10] [11]
	HotJava 1.0a3	Covert channel	DNS	Navigator 2.0 + patch	[10] [11]
	HotJava 1.0a3	Read files in public_html and /TEMP(Windows)	Bad default ACL	In JDK 1.0 and Navigator 2.0	[10] [11]
	HotJava 1.0a3	Read access to username,machine name, and all environment variables	Via System.getenv()	In JDK 1.0 and Navigator 2.0	[10] [11]
	all	Measure real time on client	Part of Java features		[10]
	all	Denial of Service	Consume CPU		[10]
	HotJava 1.0a3	Denial of Service	Acquire locks on browser		[10]
	HotJava 1.0a3 JDK 1.0 Navigator 2.0	Intercept network traffic if proxy server is used.	Change the browser's HTTP and FTP proxy server	Navigator 2.0	[10] [11]
Feb. 96	JDK 1.0 Navigator 2.0	Arbitrary host connections from client	DNS spoofing attack	JDK 1.0 1 and Navigator 2.01	[12] [13] [14] [15] [16]
Feb. 96	JDK 1.0 Navigator 2.0	Execute arbitrary code on client	Stack overflow using Javap	JDK 1.0 1 and Navigator 2.01	[12]
Mar 96	JDK 1.0 Navigator 2.0	Bypass all Java security	Load package with / (absolute path name)	JDK 1.0 1 and Navigator 2.01	[17] [18] [16]
Mar 96	JDK 1.0 Navigator 2.0	Expose internal memory mapping of Java applet on client	hashCode() method		[18]
Mar 96	JDK 1.0 Navigator 2.0	An applet can detect and effect the execution of another applet	Handle of the ThreadGroup, stop(), and setPriority(0)		[18]
Mar 96	JDK 1.0.1 Navigator 2.01	Allow file deletion and other damage	Java language restricts calling of ClassLoader, bytecode does not.	JDK 1.0.2 Navigator 2.02	[18] [19] [20] [16]
Mar 96	JDK 1.0.1 Navigator 1.0.1	Allows TCP/IP connections to hosts other than the host from which the Applet was loaded		JDK 1.0.2 Navigator 2.02	[21] [16]
Apr 96	all	Denial of Service attacks			[22] [16]
May 96	JDK 1.0.1 Navigator 2.02 Explorer 3.0b1	Allowed an applet to execute arbitrary machine code	Create a classloader	JDK 1.0.2 Navigator 3.0b4 Explorer 3.0b2	[23] [16]

Table 2 (continued)
Security Related Java Flaws

Jun. 96	Navigator 3.0b4	Allowed an applet to execute arbitrary machine code	Illegal type cast attack	JDK 1.1 Navigator 3.0b5 Navigator 3.0b6 on PCs	[23] [16]
Aug. 96	Explorer 3.0b2	Allows full file and network access	Code packages can set certain security-critical variables such as the access control lists	Explorer 3.0	[24]
Aug. 96	Explorer 3.0	Allows arbitrary execution of DOS commands	Bypasses warning notice	Explorer 3.1	[25]
Aug. 96	Navigator 3.0b5	Circumvent Java's security mechanisms	Define class as a special name	Navigator 3.0b6	[26]
Dec. 96	JDK 1.1b1	Allow untrusted applets to run	Signature checking failed	JDK 1.1b2	[16]

Although the language helps restrict creation of security holes, a devious programmer can bypass the high level Java language by manipulating the bytecode, either directly or through a "custom" bytecode compiler. Through bytecode manipulation, unauthorized actions can be attempted on the client machine. To guard against such an attack, the browser is required to do a bytecode verification pass that checks that the bytecode is within the bounds of the Java language.

The browser also places restrictions on file and network access which can be performed by an applet. The degree of restriction depends on the browser. Netscape Navigator 3.0 does not allow any client side access to local files (no read, write, or delete) and only allows the applet to make network connections to the server from which the applet was downloaded. Microsoft Internet Explorer 3.0 file access is dependent on its configuration settings. It can allow unrestricted file access, access after an affirmative reply is given to a pop-up window, or no access at all.

It is easy to see that the responsibility for strict enforcement of language restrictions lies within the browser. Sun initially created a proof-of-concept browser called HotJava. Sun released full source code for HotJava (which was written in Java) and a group of industrious researchers at Princeton immediately set out to discover any security flaws that might exist in this new technology. An examination of Table 2 shows that they were successful.

Sun has licensed the Java technology to other browser manufacturers (most notably Netscape, and later Microsoft) and replaced HotJava with the Appletviewer in the Java Development Kit (JDK) [27]. The Appletviewer's primary purpose is to aid in writing and debugging Java applets, leaving the creation of commercial browsers to the licensees. Since the commercial browsers base their Java implementations on the JDK, most of the flaws that have been found in the JDK can be found in all Java-capable browsers, and the JDK has had its share of problems (see Table 2).

Sun and the other browser manufacturers have been very responsive and have taken steps to correct all of the known Java problems (however, some problems, such as denial of service attacks, are not fixable[22]). As of this writing, no fixable outstanding problems remain. Some have concluded that there is a lack of a formality in the development of Java security which may

result in future problems. If this conclusion proves to be true, we are likely to see a continuous flow of Java patches, much like the UNIX patch situation we now experience.

To Sun's credit, most of the scrutiny Java has received has been encouraged by their own willingness to release the JDK source code. The open question remains as to how many security flaws might be found if the other web browser manufacturers where to expose their source code to the same scrutiny.

JavaScript And JScript

Realizing that many programmers prefer to write in an interpreted scripting language (such as basic, perl, or tcl), Sun and Netscape Communications Corporation decided to collaborate, combining Java and LiveScript technology to create JavaScript. Not wanting to be left out Microsoft Corporation created the JScript technology [28]. Unlike Java, JavaScript and JScript are not compiled into bytecode before being sent to the browser, nor are they verified or executed on the “virtual machine.” Scripts are embedded directly in an HTML document and delivered to the browser as part of the document. Like Java, JavaScript and JScript are being executed locally on your computer.

It is not clear what security restrictions are in place on these scripts once they are running on your machine. For example, JScript can execute ActiveX capabilities which raises many concerns (described in the next section). Flaws discovered in the implementation of JavaScript have not been as serious as many of those found in Java. Whereas some Java bugs have allowed active exploitations such as execution of arbitrary code, writing/deletion of files, and connection to arbitrary hosts; all of the problems in JavaScript have been related to the passive ability to obtain information from the client that is supposedly off limits [29].

While Sun has total control over the Java programming language, the web browser developers appear to be the driving force behind JavaScript and JScript. The method of converting the script to native machine code is done within the browser at the discretion of the browser manufacturer. As with most commercial software the browser manufacturers have kept a tight lid on the source code for the script interpreters. This means the scripts have not been subjected to the intense scrutiny that Java has undergone. A real concern is the complete lack of any published security model for either scripting language.

ActiveX Is Very Active

Microsoft has introduced ActiveX into the market as their approach to web based computing [30]. ActiveX is not a programming language. It is better described as a set of client side capabilities designed to be download and locally run by different applications. Currently ActiveX supports several application types:

ActiveX Controls	Microsoft describes these as "interactive objects." These controls have taken the place of the old OLE controls. They can be written in almost any popular language.
Documents	Desktop documents such as Excel or Word files. ActiveX allows them to be downloaded and viewed through the Web browser.
Java	Accepts and runs Java applets.
JScript	Microsoft's rendition of JavaScript (The same...but a little different).

VBScript	The Visual Basic language embedded directly into HTML documents.

ActiveX is supported in the Microsoft Internet Explorer 3.0 Browser. Netscape also has plans to support a subset of the ActiveX suite of capabilities [31].

With the exception of Java applets, the security structure of ActiveX is totally dependent on the trustworthiness of the downloaded object [32][33][34]. Java applets are subjected to the same bytecode verification described earlier, but any other object, if determined to be trustworthy, is executed by the ActiveX engine with no security restrains other than what is enforced by the local operating system. The trust of an object may be determined by cryptographic signatures that are optionally attached to downloaded objects [35]. If you are running an insecure operating system (such as Windows 95 or Macintosh System 7) and insist on allowing ActiveX objects to run on your browser without signature verification, you are leaving your machine open to total control by the web server. This vulnerability was demonstrated by an ActiveX control developed by the Chaos Computer Club that, when downloaded from the server, will determine if the Quicken financial software exists on the client, open Quicken, and start making money transfers between Quicken accounts. This is done without any visible clues given to the client owner [36]
.

The Magic Cookie Jar

Did you know that any website can read and write to your disk through your browser and that you cannot stop them? They can. But the area that is available to them is very small and the read/write action is severely restricted. The file they can write to is called the magic cookie file.

A magic cookie is information deposited on your system by the web server. The exact number and content of a magic cookie is determined by the server. The server can write a maximum of 300 magic cookies into the file. Each cookie is associated with a domain name and may be given an expiration date. When you return to the server at a later time all the magic cookies that you were given on your previous visit are returned to the server. This is a very simple description of the magic cookie mechanism, for a more detailed description, refer to the Netscape Persistent Client State HTTP Cookies documentation[37].

The following is an excerpt from that documentation:

> This simple mechanism provides a powerful new tool which enables a host of new types of applications to be written for web-based environments. Shopping applications can now store information about the currently selected items, for fee services can send back registration information and free the client from retyping a user-id on next connection, sites can store per-user preferences on the client, and have the client supply those preferences every time that site is connected to.

The disturbing part of this capability is that information can be stored on your computer and passed around to other web servers (with domain restrictions) at the discretion of the server that creates the magic cookie. The action does not need your consent and is routinely done without your knowledge. Combine this with the “cyber social engineering” concerns described within the CGI section and you now have to be concerned with information gathering on an unlimited time scale. From a paranoid security standpoint it might allow covert information channels through your computer and it might allow detailed user profiles to be built about an unsuspecting target. (Whoops! That's not a security problem, it's a marketing feature!)

With Netscape V3.0 and higher, the browser can be set to notify the user when an attempt to set a cookie is being made. The user can then reject the cookie. The bad news is that for each request to the server you will receive the same notification and have to reject it. This can be rather annoying as each document, image, sound, etc. is considered to be a separate request to the server!

As an alternative, you can allow them to set the cookie and take action after you depart the site. If you are concerned about cookies you can examine the contents of the cookie file using your web browser. Cookies are kept in memory for a given run session of the web browser. To force it to save the cookies quit the browser and restart. You can then use the Open File feature of the browser to examine the cookie file. This file is usually located in a preferences directory for the browser. Searching for "cookie" or "magic" should reveal the name and location on your system. Once open you will see all the cookies on your system. Reference [37] describes the format of cookie entries in the magic cookie file.

Removing the magic cookies from your system is a simple matter of deleting the magic cookie file. Deleting the cookie file does not effect how the browser will operate. The cookie file is recreated each time you quit the browser. But remember, you must quit your browser before you delete the file or the browser will store the cookies that are currently in memory in the newly recreated cookie file.

One trick you can use to limit what a site can do with cookies is to replace the magic cookie file with an empty, read-only text file. The browser will be able to use cookies in memory for a given run session but it will be unable to save them in the cookie file for later sessions.

Being Spoofed In Virtual Reality

Have you ever watched Mission Impossible? One of the IM force's favorite tricks was to place the evil spy into a familiar environment and manipulate the environment to the IM force's advantage. The spy would think he was flying at 20,000 feet in a private jet, but in reality, the turbulence he was experiencing was Barney and Willie shaking a wingless fuselage inside a big warehouse. As the plane was tipped up to simulate a climactic death crash, Jim Phelps would pressure the spy to divulge his "top secret code". Of course, the Spy would blurt out the code. Then, with the suddenness of a commercial break, Jim would step out of the fuselage, jump into a waiting car, and speed off down the deserted streets. The spy was left in his bewilderment to sort out what had happened.

The same ploy can be used in the “virtual reality” of the web, but it is a lot easier and cheaper to do. All that has to be simulated is a browser screen. The Secure Internet Programming team at Princeton has proposed a scenario [38] where a trojan URL is slipped into a legitimate website. Instead of sending your browser to the expected web server, it sends you to a spoof web server. You think you are in an airplane, but you are really in a warehouse. Once caught in the spoof website, all other URL references can be spoofed, turning the whole browser expedition into a charade. Of course the whole game can be exposed simply by watching the location and status lines on your browser...except if JavaScript or JScript is enabled. A script program can replace both of those lines to keep the charade alive.

Is this a simple playful trick, or can it be dangerous? Think about these instances:

- Downloading security patches from vendor site.
- Downloading “plug-in” or “helper” applications for your browser.
- Financial transactions over the web.

- Changing your password on a web server that requires a password.

In some cases, the damage may only be the embarrassment of accepting misinformation as fact. In other cases, actual losses of time and money can occur.

Welcome To My Parlor

One might ask "How are these threats any different than the threats that have existed on the Internet for years?" In the past, the typical attacker used semi-random attacks to locate vulnerable hosts. Perhaps the attacker narrowed the boundary of attacks by looking for interesting domain names or by collecting interesting IP addresses mentioned in netnews, ftp sites, or other network information sources. A really sophisticated attacker might automate much of the assault using scanners like SATAN[39] or ISS [40] to find unprepared victims. The intruder had to do a lot of work and never had any assurance that, once broken, the victim's host would hold anything of value. With the advent of the web, the stalker no longer has to stalk. A website with a theme that will attract a particular crowd can be created and advertised. Soon those who seek and store information that is valuable to the intruder will be declaring themselves as potential targets by following their own curiosity.

Let us assume that the intruder wants to narrow the hit list even further. By making web page content on a particular subject more detailed and intense with each descending link, the victim not only declares his interest, but also his affinity for the subject. Adding in other easily obtainable information such as time between page changes and the number of times each page is "hit" could be used to further define the profile. Finally, well written "cyber-social" CGI or Java programs could top off an information assault, targeting those hosts and users that have a keen interest in the subject that the intruder is trying to obtain. Once a suitable target has been identified the website can start a vulnerability scan on the victim host. Also, if so desired, an immediate or delayed automated attack can begin (note that neither the scan nor the attack has to come from the website host).

There are two important differences between this scenario and earlier conventional attacks. The victims are "screened" for their potential value to the attacker; and the victim, not the attacker, initiates the entire attack sequence by clicking into an seemingly innocuous website. Once the website has been properly built, the attacker does nothing except review the results of his trap. "Maybe the analogy of the web is more accurate than you think" said the spider to the fly.

SAFE SURF'N: A PRACTICAL APPROACH

Let us summarize. As browser functionality becomes more complex, risks are increased. As the browser interface becomes simpler, the actual control you retain to compensate for risks is reduced and you must rely more on the integrity of the browser. Browser design is not always done in your best interest. Anytime any type of executable code downloads to your computer, the potential for receiving “a mite”, the web equivalent of a virus, does exist. In short, the use of the new web paradigm presents you, the network surfer, with some scary prospects. However, no matter how scary, the technology is too valuable to dismiss as an "unacceptable risk." So what can be done? Here are some suggestions.

Know Thy Browser

Here are some critical misconceptions to avoid:

1. The browsers will always protect my computer.
2. All browsers are all alike.
3. The browser will come configured with the most secure settings.

4. The browser will not let me do unsafe things.

There is only one way to protect against these misconceptions and that is to learn about the browser. The motto of the industry at this time seems to be "Let the browser beware".

Stay Out Of Bad Neighborhoods

Its not always possible to know a bad site from a good site, but often it is! If they tell you they are hackers, BELIEVE THEM! (i.e., www.Legion_of_Doom.org). If you find yourself in a site that seems to deal in illegal or immoral material, the chances are that their attitude toward you is probably illegal or immoral. GET OUT FAST.

Leave the dangerous exploratory stuff to the professionals. If you must explore, be prepared. Back up your files, remove confidential information, and set your browser settings in the most conservative modes. Know your browser well enough that you can spot unusual activity.

In a corporate atmosphere, policy or guidance may be needed to determine how far from home one can stray or what browser features are safe. Future releases of browsers from Netscape and Microsoft promise to offer the ability to lock certain configuration settings so a novice user can not change them.

There's No Such Thing As A Free Lunch

Beware of anything a website wants to give you for free. Some freebies may be legitimate, some may not. Not everyone out there shares your ethical beliefs, so as a guide, remember:

1. Cookies have a purpose.
2. Understand the meaning of all data you provide in Forms.
3. Examine the information about a link before clicking it. If you're not sure what's being sent to you, get advice.
4. Don't download helper/plug-ins applications indiscriminately.

Remember, You Are Not Necessarily Talking To A Friend

Even friendly sites can be spoofed. Keep an eye on the location and status line on your browser. If you travel with Java/JavaScript/JScript enabled, turn it off occasionally or check the history logs to make sure you are really where you think you are.

Commercial sites, by the definition of commercial, want something from you. They may only want to pique your interest in a product or service, or they may be collecting (and selling) your personal information. You have no way of knowing. The situation is not new. Is the salesman with the big smile really a friendly person who is sincerely glad to meet you, or is he a shyster interested in separating you from your money? Use discretion when visiting such websites.

Many websites may ask you to register by providing a username and password. Be cautious, do not use a username and password which you use on other computer systems you access; otherwise you've just given it to the owner of the web server.

Remember, you do not have to answer all the questions or click on all the buttons just because the web server requests you to do so. All browsers, even with their flaws, still offer the ultimate control of being able to disconnect at the client's whim. You can leave at any time. You can be a guest or victim...the choice is yours.

Be Careful Out There

The main thrust of this paper has been to point out possible security pitfalls that may arise from web browsing. Hence, the paper takes on a what might be perceived as a negative slant toward browsers. This is not our intent. As you may notice from the bibliography, most of the research

for this paper took place through the window of a browser. However, we fear that too many people are randomly jumping across the Internet under the motto "ignorance is bliss." There are dangers, and, depending on the circumstance, precautions should be taken.

BIBLIOGRAPHY

[1]	National Center for Supercomputing Applications, The Common Gateway Interface	http://hoohoo.ncsa.uiuc.edu/cgi/
[2]	Paul Phillips, CGI Security	http://www.go2net.com/people/paulp/cgi-security
[3]	NCSA. Writing secure CGI scripts	http://hoohoo.ncsa.uiuc.edu/cgi/security.html
[4]	FTC Says Internet Scam Re-Routes “Surfers” to International Telephone Lines, February 19, 1997	http://www.ftc.com/opa/9702/audiotex.htm
[5]	Statement of Jodie Bernstein, Director, FTC Bureau of Consumer Protection, February 19, 1997	http://www.ftc.com/opa/9702/audiotex1.htm
[6]	Civil Action Against Audiotex Connection, Inc	http://www.ftc.com/os/9702/audiotex.htm
[7]	David de Vitry, Shockwave Security Alert, March 10 1997	http://www.webcomics.com/shockwave
[8]	Microsoft Word Virus	http://ciac.llnl.gov/ciac/virdb/VIRS0844.TXT
[9]	Macromedia Inc., New Security Enhanced Shockwave	http://www.macromedia.com/shockzone/info/security.html
[10]	Drew Dean, Dan S. Wallach. Security Flaws in the HotJava Web Browser, Princeton CS Tech Report 501-95	ftp://ftp.cs.princeton.edu/reports/1995/501.ps.Z
[11]	Marianne Mueller. Sun' Response to the HotJava Security Flaws	http://www.cs.princeton.edu/sip/news/sun-11-09-95.html
[12]	Steve Gibbon. postings to firewalls mailing list	http://www.aztech.net/~steve/java/
[13]	February 1996: DNS-based Attack on Java	http://www.cs.princeton.edu/sip/news/dns-spoof.html
[14]	Java Security: DNS Attack Scenario	http://www.cs.princeton.edu/sip/news/dns-scenario.html
[15]	Marianne Mueller.Sun's Reponse to the DNS Spoofing Attack	http://www.cs.princeton.edu/sip/news/sun-02-22-96.html#DNS
[16]	Sun Press Release. Frequently Asked Questions - Applet Security	http://java.sun.com/sfaq/
[17]	David Hopwood. Java security bug (applets can load native methods)	http://catless.ncl.ac.uk/Risks/17.83.html#subj13
[18]	Drew Dean, Ed Felten, Dan Wallach. Java Security: From HotJava to Netscape and Beyond	http://www.cs.princeton.edu/sip/pub/secure96.html
[19]	Sun Press Release. Verifier implementation bug	http://www.javasoft.com/sfaq/960327.html
[20]	Netscape Press Release. POTENTIAL VULNERABILITY IN JAVA VERIFIER REPORTED	http://home.netscape.com/newsref/std/verifier_resp.html
[21]	Eric R Williams. Java Applet Security: Sockets	http://www.sky.net/~williams/java/javasec.html
[22]	Sun Press Release. Denial of service May 10, 1996	http://java.sun.com/sfaq/denialOfService.html
[23]	Safe Internet Programming: News	http://www.cs.princeton.edu/sip/News.html

[24]	August 1996 Security Flaw: Brief Description	http://www.cs.princeton.edu/sip/news/Aug96-microsoft.html
[25]	August 1996 Internet Explorer Security Flaw: Brief Description	http://www.cs.princeton.edu/sip/news/Aug96-2.html
[26]	August 1996 Security Flaw: Brief Description	http://www.cs.princeton.edu/sip/news/Aug96-netscape.html
[27]	CERT(sm) Advisory CA-96.07	ftp://info.cert.org/pub/cert_advisories/
[28]	Microsoft Corporation, Microsoft JScript and the ECMA standards	http://www.microsoft.com/jscript/us/techinfo/standards.htm
[29]	John Robert LoVerso. JavaScript Problems I've Discovered	http://www.osf.org/~loverso/javascript/
[30]	Microsoft Corporation, ActiveX Resources Area	http://www.microsoft.com/activex/
[31]	Information Week. Netscape Reverses Position; Embraces ActiveX, October 16, 1996	http://techweb.cmp.com/iw/newsflash/nf601/1016_st1.htm
[32]	David Chappell, Component Software Meets the Web: Java Applets vs. ActiveX Controls, May 1996	http://www.chappellassoc.com/JavaActX.htm
[33]	Dan Meriwether, A simple six step process for certifying your Microsoft® ActiveX® control.	http://www.delux.com/Thoughts/ActiveX.html
[34]	Don McGregor, ActiveX: Threat or Menace?	http://www.stl.nps.navy.mil/~mcgredo/ActiveX.html
[35]	Paul Johns, Signing and Marking ActiveX Controls, October 1996	http://www.microsoft.com/intdev/controls/signmark.htm
[36]	ActiveX used as hacking tool	http://www.news.com/News/Item/0,4,7761,00.html
[37]	Netscape Communications Corporation. Persistent Client State HTTP Cookies	http://www.netscape.com/newsref/std/cookie_spec.html
[38]	Edward W. Felten, Dirk Balfanz, Drew Dean, Dan S. Wallach. Web Spoofing: An Internet Con Game, Princeton CS Tech Report 540-96	http://www.cs.princeton.edu/sip/pub/spoofing.html
[39]	Security Administrator's Tool for Analyzing Networks	http://www.fish.com/satan
[40]	Internet Security Systems home page	http://www.iss.net