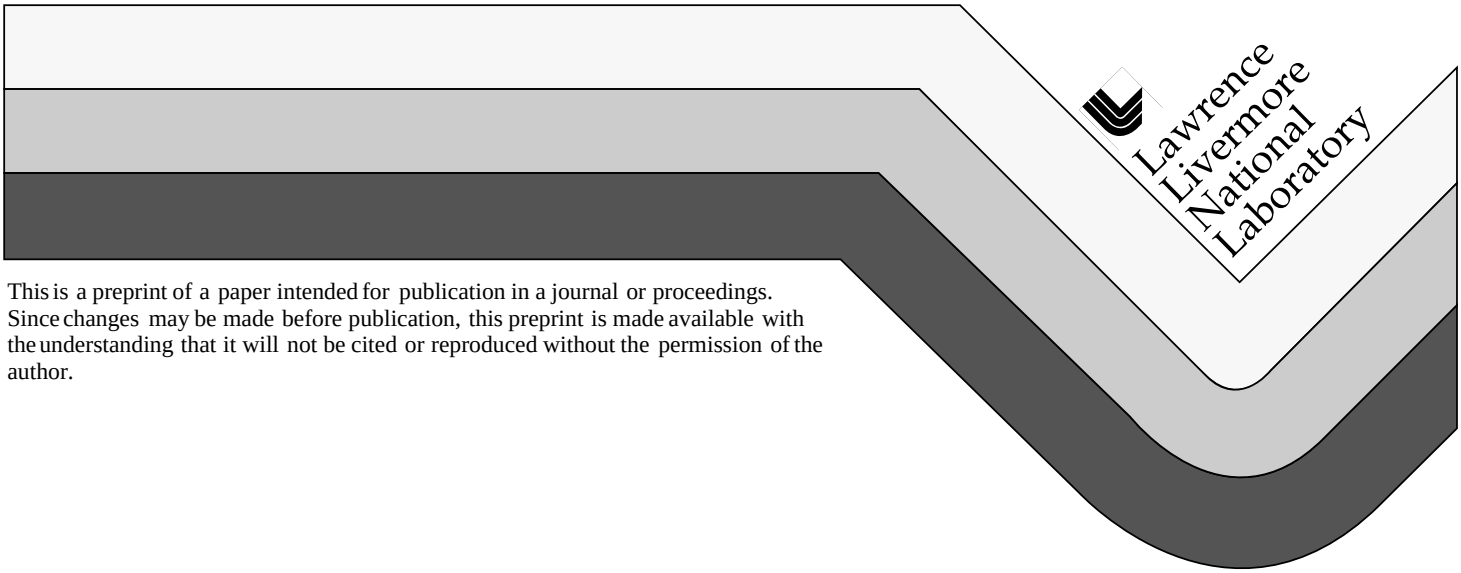
UCRL-JC-128433
PREPRINT

Go Ahead, Visit Those Web Sites, You Can't Get Hurt, Can You?

J.S. Rothfuss
J.W. Parrett

This paper was prepared for submittal to the
20th National Information Systems Security Conference
Baltimore, MD
October 6-10, 1997

February 1997



Go Ahead, Visit Those Web sites, You Can't Get Hurt, Can You?

February 6, 1997

James S. Rothfuss
(510)422-9452
rothfuss1@llnl.gov

Jeffrey W. Parrett
510-423-7577
starman@llnl.gov

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 before 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 by creating the myth that a user visiting a web site 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 web sites. But, either directly or indirectly, others can also learn quite a bit about you. Of even more concern is a web sites' 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 the 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 web site). In most normal web configurations, web sites opens 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 web site 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 central web sites of high-profile agencies has resulted in damage and great embarrassment. These web site break-ins tend to be well covered by both the conventional and 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 web site. 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. In fact, the safety of the client is not insured and your lack of education may be subjecting your computing environment to great risk. The remainder of this paper will focus on educating you about the threats, concerns, and protections on the client side of a web interaction.

Big Deal, We Have Seen It All Before

Computer security was around long before the web. The Internet has not been considered completely "safe" for at least ten years. Many of the "new" threats created by the web are just the same old threats wrapped in a different package. While all of these statements are true, the danger is, many more unsophisticated and uneducated people are accessing the web and being subjected to security risks. The web offers unethical, malicious or subversive individuals a simple mechanism for reaching you. Therefore, we feel it is important to review some of the common computer security threats and how they map into the web.

Programs That Lie

A common desire in current network computing is to find a mechanism to get access to your computer. One mechanism for doing this is to learn your username and password. A common ploy of the past was to create a program that looks like it is presenting a normal login screen. You enter your username and password, thinking you were logging in. In reality, the program would store these two critical items for later use. You might then be presented with a normal login session or given an error message making you think there is a glitch and then proceeding with a real login. The new model of presenting the victim with a HTML form asking for critical information and hoping they are naive enough to respond is not much different then the old password grabber.

Gathering Information & Social Engineering

The gathering of information has always been a desire by the network hackers. For example, a simple network sniffer can gather reams of information from a simple telnet connection. As we will discuss a web browsers can also give out information, sometimes with your knowledge and sometimes without.

It has long been known that one of the most successful weapons used by hackers is social engineering. This is the process of making a non-threatening personal contact with key personnel with the intent of obtaining bits of information that can be used to break into a system. This contact can be by phone, e-mail, in person, or a number of other ways. The web offers an environment where people seem to feel safer because they choose where they were going. It is amazing what types of questions people will answer, even quicker than they would if they were contacted by a stranger.

Programs That Do Bad Things

You hear about new computer virus on the news on a regular basis, some even get national attention. The activity of these viruses could be something as innocuous as displaying a funny message or as detrimental as erasing your hard disk. Because the web is downloading information to your computer, the potential for receiving “a mite”, the web equivalent of a virus, does exist.

Software Has Bugs

As with any computer software, the web browser is built by humans and probably contains bugs.

> It's a maxim in system security circles that buggy software opens up security holes. It's a maxim in software development circles that large, complex programs contain bugs. Unfortunately, web servers (and browsers) are large, complex programs that can (and in some cases have been proven to) contain security holes.[21]

How They Get To You?

The reality is they don't, you go to them, but what they have to offer is changing on what seems like a daily basis. As the desire to increase the capability of what can be delivered to the client from the web server gains momentum, the interactions between the two become more complex. As the complexity grows so does the risk. We will now discuss some of the more popular technologies which are being incorporating into the web and the risks associated with them.

Java, Its More Than Just 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 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 site can download and run a program on the client with no prior indication. For obvious reasons, Java is not always looked upon as a positive advancement by those concerned with security.

What Security Precautions Have Been Built 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 out of the programmer's hands and into run-time and the implementation of the "virtual machine".

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, to do unauthorized actions 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.

Security Depends On The Browser

From the previous discussion, it is easy to see that the responsibility for strict enforcement of language restrictions lies in the browser. Sun initially created a proof-of-concept browser called HotJava. Sun released full source 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 1 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) [31]. 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 1).

Sun and the other browser manufacturers have been very responsive and have taken steps to correct all of the known Java problems (some problems, such as denial of service attacks are not fixable[17]). As of this writing, no outstanding problems remain. However, 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 other security flaws might be found in web browsers from other manufacturers' if their source code were to received the same scrutiny.

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	[1] [2]
	HotJava 1.0a3	Covert channel	DNS	Navigator 2.0 + patch	[1] [2]
	HotJava 1.0a3	Read files in public_html and /TEMP(Windows)	Bad default ACL	In JDK 1.0 and Navigator 2.0	[1] [2]
	HotJava 1.0a3	Read access to username,machine name, and all environment variables	Via System.getenv()	In JDK 1.0 and Navigator 2.0	[1] [2]
	all	Measure real time on client	Part of Java features		[1]
	all	Denial of Service	Consume CPU		[1]
	HotJava 1.0a3	Denial of Service	Acquire locks on browser		[1]
	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	[1] [2]
Feb. 96	JDK 1.0 Navigator 2.0	Arbitrary host connections from client	DNS spoofing attack	JDK 1.0 1 and Navigator 2.01	[3] [4] [5] [6] [18]
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	[3]
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	[7] [9] [18]
Mar 96	JDK 1.0 Navigator 2.0	Expose internal memory mapping of Java applet on client	hashCode() method		[9]
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)		[9]
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	[9] [10] [11] [18]
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	[12] [18]
Apr.	all	Denial of Service attacks			[17] [18]
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	[13] [18]
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	[13] [18]
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	[16]
Aug. 96	Explorer 3.0	Allows arbitrary execution of DOS commands	Bypasses warning notice	Explorer 3.1	[14]

Aug. 96	Navigator 3.0b5	Circumvent Java's security mechanisms	Define class as a special name	Navigator 3.0b6	[15]
Dec. 96	JDK 1.1b1	Allow untrusted applets to run	Signature checking failed	JDK 1.1b2	[18]

Security Related Java Flaws
Table 1

ActiveX Is Very Active

Microsoft has introduced ActiveX into the market as their approach to web based computing [25]. 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 [30].

With the exception of Java applets, the security structure of ActiveX is totally dependent on the trustworthiness of the downloaded object [26,27,29]. 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 operating system. The trust of an object may be determined by cryptographic signatures that are optionally attached to downloaded objects [28]. 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.

JavaScript And JScript, Its Not Coming From Steven Spielberg

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 [24]. 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.

What is not clear is what security restrictions are in place on these scripts once they are running on your machine. For example, JScript can execute ActiveX capability which raises all of the concerns described above. 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 exploitation’s 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 [23].

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. It would at least be nice to be able to view the source code for a script so it could be subjected to scrutiny.

CGI: Can Get Information

The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as web servers [19]. The programs can be written in any programming language or scripting languages. 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. A CGI program obtains information from two sources, the web server and the web browser. The information available from the web server is based on the connection between the server and the client. It may also depend on the type of server software being run. Refer to Table 2 for an example of information available from the server. The CGI program can obtain information from the web browser via the HTML forms capability. The exact content of this information is depends on the HTML form. As with any program, the information gathered by a CGI program can be processed immediately, or stored indefinitely at the web site for some future use. The dynamic nature of CGI allows the web site to initiate action immediately based on information gathered and while you are still connected.

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 =

Information Available From The Server
Table 2

Are CGI Scripts Really Dangerous?

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 [32,33]. From the browser perspective a URL has been requested and a page was returned. While a program can not mount a direct attack on the browser through the CGI, dynamic execution of programs can enhance the server's ability to obtain information and 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 the web comes the ability to allow an automatic process to lure victims into divulging valuable information with seductive web pages and non-threatening questions. CGI allows the web site to dynamically customize the sequence of web pages and form questions with each phase building on past web site interactions. Through the CGI, the web site can also maintain persistent memory across invocations. With the stored information, a web site could build visitor dossiers and eventually, through days, weeks, or months of seemingly innocuous question/answer sessions, piece together damaging information.

Another aspect of CGI programs is that once enough information has been gathered an automated attack can be triggered. Remember that the CGI does not inform the web site visitor that it has started a program unless the program specifically sends HTML commands clueing the visitor that something dynamic is happening. The visitor may assume they are looking at a passive HTML page when in reality, a program has been triggered that is actively attempting to break through one of the other network facilities (i.e. sendmail, ftp).

The Magic Cookie Jar

Did you know that any web site can read and write to your disk through your browser and that you cannot stop them! Of course, 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.

What Is A Magic Cookie?

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 Netscapes Persistent Client State HTTP Cookies documentation[34].

Why Do I Get Them?

The following is an excerpt from reference [34]:

> 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 with in the CGI section and you now have to be concerned with information gathering 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 , you. (Whoops! That's not a security problem, its a marketing feature!)

Taking The Bite Out of Magic Cookies

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 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. The URL above describes the format of cookies if you want to understand what has been placed on your system.

Once you have examined the cookies on your system its simply a matter of deleting the cookie file to remove them (remember you must quit your browser or it will recreate the cookies file from what is currently in memory). This does not effect how the browser will operate as the cookie file will be recreated. One trick you can use if you want to limit what a site can do with cookies is to create any empty text file, name it properly for your browser, place it in the proper location, and lock it or make it read-only. This will allow the browser to use cookies in memory for a given run session but it will be unable to save them for later sessions.

HTML: Hazardous To My Life

Under most circumstances one would not consider the Hyper Text Markup Language to present any security risks, but this is not entirely true. One risk which exists in HTML is a feature of Netscape 2.0 and higher is the ability to upload files from the client system. This is done through the use of HTML forms and a CGI script. It requires the user to select an the input file from their local system. Fortunately this type of field can not be pre-loaded with a file name so only a file the user selects is at risk. However, if you do not understand what is happening you could be tricked into consenting to an undesired file upload.

Beware Of Servers Bearing Gifts

Have you ever clicked on a web site URL and sat back as a file was downloaded and opened by some application which existed on your system. Through the use of MIME types, servers can tag files in a fashion which enables the browser will recognize them and open the proper application locally. Sometimes these full-feature applications don’t exist locally so in there place a “plug-in” or “helper” application handles the display of the material. Of course, these applications resided on client before the web site was even visited, so they are all assumed to be trusted programs.

Envision The Following Scenario

You arrive at a web site that has a URL for a very seductive document, *very_interesting.bum.* When you click on the URL the document is downloaded and you are presented with the instructional pop-up "Cannot open BUM document, please pick the application to use." Of course, you do not have the BUM application, but those courteous folks who prepared the web site, realizing that most visitors do not have the BUM application, have supplied a URL that will download the Big Uncompress Machine (BUM) onto your client. Wanting to get a look at *very_interesting.bum* as soon as possible, you click on the BUM URL and then click on the *very_interesting.bum* URL. The file is downloaded, you tell it to use the BUM application just downloaded and BUM runs...locally...

It is obvious that caution must be taken in downloading seemingly valuable extras because 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.

Being Helped 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. Oh yes, by the way ... 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 ShockWave, they have no control over what developers do in their documents with Lingo. The danger is further compounded by the fact that Lingo can be extended by the developer to the point of making local system calls based on the platform on which it is running. This is of extreme concern as Macromedia has not published any information on what security issues and restrictions are addressed in Lingo. 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 document virus [35].

Jane! Let Me Off This Crazy Thing!

As a kid, one of my favorite shows was 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" before dying. 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 with Barney and Willie, 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 an 8" X 10" browser screen. The Secure Internet Programming team at Princeton has proposed a scenario [20] where a trojan URL is slipped into a legitimate web site. Instead of sending the browser to the expected web server, it sends it to a spoof web server. You think you are in an airplane, but you’re really in a warehouse. Once caught in the spoof web site, 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 line on the top and the status line at the bottom, 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.
- Changing tax information through your company's human resources web server.
- Using your credit card to make a purchase through a web server.
- Changing your password on a web server that requires a password.
- Checking stock information.
- Reading outcomes of recent congressional votes.

In some cases, the damage may only be the embarrassment of being made to look stupid as you accept misinformation as fact. In other cases, actual losses of time and money can occur.

So, What Is New?

Just Like Falling Off A Log

Web sites have one great distinction over older network connections. It is the reason that web site URLs appear on television, magazines, and product labels while ftp addresses never got beyond the "computer geek media". Web surfing is easy. You do not have to know about directory structures, filename syntax, ftp commands, and on and on. The bad news is that the 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 should not always be trust them. Not only is it easy to wander the web in reckless abandon, the current browsers' lack of adequate logging and security options make it difficult to take a cautious route.

Here I Am ... Kick Me!

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[36] or ISS [37] to find unprepared victims. But 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 web site 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 creating a series of web pages on a particular subject, the interest of the victim is immediately determined. By making web page content 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 web site 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 web site host).

There are two important difference 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 web site. Once the web site 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.

Where Is It All Going?

The Commercial World Will Only Make It More Obtrusive

Have you noticed how "free" everything is on the internet? Netscape and Microsoft give away their browsers. Magazines, newspapers, books, and many other copy-written sources are freely available through web sites. Sophisticated network programs such as search engines and locator maps are available for your unrestricted use. Is all this provided merely for your pleasure? Sometimes, but more than likely not. Most of the time the web sites want something from you. It might only be your name. Or it might be your address, computer hardware, habits, and any other information that can be extracted. People will pay for information about you.

As a result of this free-market approach, the commercial web products market is being driven more by the server-side desire to collect market information than the client side desire to stay safe and be anonymous. It seems that the motive for client side security is not to protect the client, but to insure that the client feels safe enough to continue using the service. Feeling safe and being safe are not always the same.

Adding More Technology

S-http and Secure Sockets Layer (SSL)

Both of these technologies address a security problem mentioned briefly in this paper, "sniffing" information as it travels around the network. S-http and Secure Sockets create encrypted channels in a seamless manner, once the client/server connection is made, all communications between the two are kept private. Further, no one can cut in with their own disguised packets. They also provide for strong authentication between the client and server. This does mitigate many of the problems

mentioned in this paper. The creation of this type of connection is at the discretion of the server, your client has no control over creating secure communications channels.

Trusted Executables

Microsoft has based their entire web security structure for ActiveX applications on the concept of digitally signing executables. Netscape is going to provide the capability soon. The concept is based on public/private key cryptography. Each public provider of an ActiveX application is expected to apply to Microsoft for a unique public/private crypto key pair. The provider then signs all of their applications with the private key. As a user, you add the provider's public key to the list in your browser. Downloaded executables will either run, run after a warning is presented, or be denied access based on verifying a digital signature. The browser bases this decision on the user selected security settings, the inclusion or absence of public keys in the browser's list, and the ability to verify the digital signature on the downloaded object. Future releases of the Java Development Kit and the "virtual machine" will include a similar capability to create trusted applets.

Firewalls

If your personal computer is located behind a firewall, most of the browsers allow for a proxy connection through the firewall configured to allow access to web servers. The proxy connection is designed to allow web browsing through the firewall, not to enhance the security of the client. Unless the firewall is doing session level filtering, the proxy connection offers limited security benefit. Most firewalls work at the network transport level, either allowing or disallowing connections, depending on the type of request. Once a legitimate connection is made, the flow of traffic through that connection is not usually monitored. All of the vulnerabilities described in this paper take place after the full initialization of the web server connection. Requests for information, transfer of Magic Cookies, interaction with CGI scripts, and the download of Java, JavaScript, JScript, and ActiveX controls, all happen within the context of HTML. Firewall technology is beginning to progress to the point where they are capable of screening requests to the web server. The firewall can then be configured to disallow the download of files like Java applets. However, this does nothing to protect you from content embedded in the HTML stream. To provide this level of protection, a firewall would need to filter the HTML stream, rejecting HTML interaction that is outside the boundaries of the site policy. We are unaware of any firewall technology that currently provides this capability.

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. 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 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 both the Netscape and Microsoft browser promise to offer the ability to lock certain configuration settings so an uneducated user can not change them.

There's No Such Thing As A Free Lunch

Beware of anything a web site wants to give you for free. Some freebies may be legitimate, some may not. Be aware, 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) 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 web sites.

Many web sites 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. 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.

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

Bibliography

[1]	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
[2]	Marianne Mueller. Sun' Response to the HotJava Security Flaws	http://www.cs.princeton.edu/sip/news/sun-11-09-95.html
[3]	Steve Gibbon. postings to firewalls mailing list	http://www.aztech.net/~steve/java/
[4]	February 1996: DNS-based Attack on Java	http://www.cs.princeton.edu/sip/news/dns-spoof.html
[5]	Java Security: DNS Attack Scenario	http://www.cs.princeton.edu/sip/news/dns-scenario.html
[6]	Marianne Mueller.Sun's Reponse to the DNS Spoofing Attack	http://www.cs.princeton.edu/sip/news/sun-02-22-96.html#DNS
[7]	David Hopwood. Java security bug (applets can load native methods)	http://catless.ncl.ac.uk/Risks/17.83.html#subj13
[8]	March 1996 Security Flaw: Extended Description	http://www.cs.princeton.edu/sip/news/non-expert.html
[9]	Drew Dean, Ed Felten, Dan Wallach. Java Security: From HotJava to Netscape and Beyond	http://www.cs.princeton.edu/sip/pub/secure96.html
[10]	Sun Press Release. Verifier implementation bug	http://www.javasoft.com/sfaq/960327.html
[11]	Netscape Press Release. POTENTIAL VULNERABILITY IN JAVA VERIFIER REPORTED	http://home.netscape.com/newsref/std/verifier_resp.html
[12]	Eric R Williams. Java Applet Security: Sockets	http://www.sky.net/~williams/java/javasec.html
[13]	Safe Internet Programming: News	http://www.cs.princeton.edu/sip/News.html
[14]	August 1996 Internet Explorer Security Flaw: Brief Description	http://www.cs.princeton.edu/sip/news/Aug96-2.html
[15]	August 1996 Security Flaw: Brief Description	http://www.cs.princeton.edu/sip/news/Aug96-netscape.html
[16]	August 1996 Security Flaw: Brief Description	http://www.cs.princeton.edu/sip/news/Aug96-microsoft.html
[17]	Sun Press Release. Denial of service May 10, 1996	http://java.sun.com/sfaq/denialOfService.html
[18]	Sun Press Release. Frequently Asked Questions - Applet Security	http://java.sun.com/sfaq/
[19]	National Center for Supercomputing Applications, The Common Gateway Interface	http://hoohoo.ncsa.uiuc.edu/cgi/
[20]	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
[21]	Lincoln D. Stein. The World Wide Web Security FAQ Version 1.3.1,	http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html
[22]	Netscape Communications Corporation. Netscape JavaScript	http://www.netscape.com/comprod/products/navigator/version_2.0/script/index.html
[23]	John Robert LoVerso. JavaScript Problems I've Discovered	http://www.osf.org/~loverso/javascript/
[24]	Microsoft Corporation, Microsoft JScript and the ECMA standards	http://www.microsoft.com/jscript/us/techinfo/standards.htm
[25]	Microsoft Corporation, ActiveX Resources Area	http://www.microsoft.com/activex/
[26]	David Chappell, Component Software Meets the Web: Java Applets vs. ActiveX Controls, May 1996	http://www.chappellassoc.com/JavaActX.htm
[27]	Dan Meriwether, A simple six step process for certifying your Microsoft® ActiveX® control.	http://www.delux.com/Thoughts/ActiveX.html
[28]	Paul Johns, Signing and Marking ActiveX Controls, October 1996	http://www.microsoft.com/intdev/controls/signmark.htm
[29]	Don McGregor, ActiveX: Threat or Menace?	http://www.stl.nps.navy.mil/~mcgredo/ActiveX.html
[30]	Information Week. Netscape Reverses Position; Embraces ActiveX, October 16, 1996	http://techweb.cmp.com/iw/newsflash/nf601/1016_st1.htm
[31]	CERT(sm) Advisory CA-96.07	ftp://info.cert.org/pub/cert_advisories/
[32]	Paul Phillips, CGI Security	http://www.go2net.com/people/paulp/cgi-security
[33]	NCSA. Writing secure CGI scripts	http://hoohoo.ncsa.uiuc.edu/cgi/security.html
[34]	Netscape Communications Corporation. Persistent Client State HTTP Cookies	http://www.netscape.com/newsref/std/cookie_spec.html
[35]	Microsoft Word Virus	http://ciac.llnl.gov/ciac/virdb/VIRS0844.TXT

[36]	Security Administrator's Tool for Analyzing Networks	http://www.fish.com/satan
[37]	Internet Security Systems home page	http://www.iss.net

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551