

**Computer Security**

# **Web Security Project**

**November-December 1998**

## **Web Security Group**

### **Netscape Communicator Security Research :**

Fontini Tsekmezoglou, Pearse Kennedy

### **Microsoft Explorer Security Research :**

Vorapranee Khu-Smith, Parichart Vongsakul

### **Server Security Research :**

Lai Chack An, Simeon Xenitellis, David Pybus, Nutdhanai Ruangkritya

### **Threat Model Research :** John Iliadis

### **Java Security Research :**

Christos Stavropoulos, Choy Wai Leong,  
Gaspar Modelo Howard, Kenneth De Spiegeleire

### **Additional Research, Report and Presentation :**

Gaspar Modelo Howard, Kenneth De Spiegeleire

# Table of Contents

## **Chapter 1 : Introduction**

- 1.1 Short security history
- 1.2 What is Web Security ?
- 1.3 The future ?
- 1.4 This report

## **Chapter 2 : Browser Security**

- 2.1 Introduction
  - 2.1.1 What is a Web browser ?
  - 2.1.2 Web browser security
- 2.2 Netscape Communicator Flaws
  - 2.2.1 History
  - 2.2.2 Flaw categories
  - 2.2.3 Detailed Analysis
- 2.3 Internet Explorer Flaws
  - 2.3.1 History
  - 2.3.2 Detailed Analysis

## **Chapter 3 : Server Security**

- 3.1 Introduction
- 3.2 Microsoft Internet Information server
  - 3.2.1 Introduction
  - 3.2.2 Insecurity due to inadequacies in default configuration
  - 3.2.3 Inherent Web Server Software Issues
  - 3.2.4 Active Server Pages security Issues
  - 3.2.5 Denial of Services attacks
- 3.3 Apache Server
  - 3.3.1 Introduction
  - 3.3.2 Flaw categories
- 3.4 CGI-Scripts
  - 3.4.1 Introduction
  - 3.4.2 List of flaws
  - 3.4.3 General comments

## **Chapter 4 : Java**

4.1 Mobile code

4.2 What is Java ?

4.3 Java Security

4.3.1 The Sandbox

4.3.2 Applet Class Loader

4.3.3 Class loaders and name-spaces

4.3.4 Class loaders for applets

4.3.5 Class loaders in the sandbox

4.3.6 Bytecode Verifier

4.3.7 Security Manager

4.3.8 Digital Signatures

4.4 Java Flaws

4.4.1 History

4.4.2 Java Class Loader Security Flaw

4.4.3 How the attack works

## **Chapter 5 : Further threats, attacks and security related issues**

5.1 Web Spoofing

5.2 DNS attack

5.3 Cookies

5.4 Firewalls

## **Appendix**

## Chapter 1 : Introduction

### 1.1 Short security history

The Internet - as we call it nowadays - was created in 1969 as the ARPANET, a project funded by the Advanced Research Projects Agency of the US department of Defence. From the beginning it was designed to reroute traffic automatically around problems in connecting systems. Thus, the Internet was designed to be robust against denial-of-service attacks. Since the sharing of information was the primary goal of the researchers, everyone needed unrestricted access on the network. Confidentiality was not an issue.

Over the years the amount of connected computers and subnetworks rose sharply, but initial 'security breaches' were more practical jokes than anything else. That the ARPANET could possibly be used for destructive purposes, was pointed out for the first time in 1989 with the publication of Stoll's *The Cuckoo's Egg*. The author uncovered an international effort to connect to computers in the US and copy information from them. One year earlier, in 1988, a Cornell university student had written *the Morris Worm*, a program that would connect to a computer, exploit a vulnerability to copy itself onto that computer, and then begin to run to copy itself from that location. 10% of the ARPANET's computers crashed at about the same moment.

By 1989 the ARPANET had moved away from being a government research project to an operational network, that connected more than 100.000 computers. Security issues became increasingly important, while both aggressive and defensive strategies and tools became more sophisticated.

### 1.2 What is Web Security ?

Over the past decades the number of reported security breaches has increased dramatically, but so has the number of connected hosts. Many surveys even show a slowing down of the *relative* rate at which incidents are reported. However, intrusions and attacks have become more and more sophisticated and are very common nowadays.

In this section we will list the most common incidents, vulnerabilities and available security mechanisms. (Definitions and concepts are based on research performed by the Software Engineering Institute at Carnegie Mellon University in the US.)

#### *Types of incidents*

- *probes and scans* :  
probes are attempts to gain access to a system or discover information about the system; scans are simply a large number of probes done using an automated tool;

- *account and root compromise* :  
an account compromise is the unauthorised use of a computer account by someone other than the account owner; the lack of root-level access means the damage can usually be contained; root compromise is very similar but the intruder now has superuser privileges, which means he can do just about anything on the victim's system;
- *monitoring* :  
packet sniffers are programs that capture data from information packets as they travel over the network; this can include the complete monitoring of certain communication lines and may allow the intruder to replay the intercepted message at a later moment, thereby enabling him to establish communication patterns with an unsuspecting host; another popular practice is the stealing of passwords, which are very often transmitted in cleartext;
- *denial of service* :  
this attack aims to prevent legitimate users of a service from using machines or data; attackers may flood a network with large volumes of data or deliberately consume a limited resource; they may also manipulate data in transit, including encrypted information;
- *exploitation of trust* :  
if an attacker can forge his identity, he may be able to gain unauthorised access to other computers or execute commands normally reserved to trusted users; forging of network addresses or other types of masquerading may lead to successful authentication of unauthorised users;
- *repudiation* :  
a malicious user may repudiate his possibly unlawful actions at a later stage, because no means exist in the typical WWW services in order to identify a party in a unique way and without any doubt;
- *malicious code* :  
this is a general term for programs that can cause undesired results on a system; they include Trojan horses, viruses, worms, and so on; this sort of programs can lead to serious data loss, downtime, denial of service and other types of security incidents;
- *internet infrastructure attacks* :  
these rare but serious attacks involve key components of the Internet infrastructure such as network name servers, access providers and large archive sites;

### ***Types of vulnerabilities***

- *feeling of historical unnecessity* :

many early network protocols were designed only with performance and reliability in mind, and not security; in many cases, people's unwarranted trust is a legacy of the past; due to the rapid growth, the philosophy of openness and the anonymous and dynamic nature of the Internet, it is extremely hard to detect, trace or defend oneself against the large variety of possible attacks and compromises on this giant network;

- flaws in software or protocol design :  
very often security is left out of the initial design and implementation of software, and is 'added on' to the system later on; protocols with fundamental design flaws will be prone to exploitation, no matter how well they are implemented;
- weaknesses in implementation of protocols and software :  
well designed protocols can still be vulnerable because of the way they are implemented; intruders might be able to gain access to a victim's system from a remote site by exploiting program weaknesses; a badly implemented electronic mail program might be tricked into sending system or password information to an attacker;
- weaknesses in system and network configuration :  
even when the software and protocols are well designed and implemented, intruders at a remote site could gain access to a victim's system by exploiting weaknesses in the set-up or configuration of the system; a common example is badly configured anonymous ftp archives; very often applications continue to operate in their default settings as supplied by the vendor;

### ***Types of security***

- security policy, procedures and practices :  
a robust defence against all of the attacks mentioned above requires
  - a flexible and well thought-out high-level strategy, adaptable to a changing environment and changing security needs;
  - realistic and effective step-to-step procedures for practical situations involving authentication, encryption, remote access, monitoring, and so on;
  - sound security practices based on constant vigilance in order to achieve good system administration;
- operational technology :  
technologies to prevent, detect or reduce the impact of intrusions include
  - one-time passwords
  - firewalls
  - monitoring tools
  - security analysis tools
- cryptology :

cryptography can provide confidentiality, and in certain circumstances even integrity and authenticity;

This list is by no means exhaustive, and as attackers will devise new ways of compromising system security, new protective tools will have to be developed.

### 1.3 The future ?

New protocols and technologies have to be developed since most of them are still based on concepts dating from the early days of the ARPANET. Important issues to be addressed include :

- internetworking protocols (IPv6 ? more IETF standardisation ?)
- intrusion detection
- software engineering and system survivability
- web-related programming and scripting languages
- intelligent autonomous agents

In the future, companies, government departments, financial institutions, health services and individuals will rely increasingly on information accessible through and/or transported over the Web. In many cases, the security of data transmissions will be of primordial importance. Web security tools, mechanisms and policies will have to ensure that businesses can make use of e-commerce in a reliable and secure way, that individuals can do online banking free from care, that communication (videoconferencing, e-mail, Internet phoning) between individuals is protected from prying eyes, and so on. Web security is one of the hot issues of the future.

### 1.4 This report

In this report we first treat Web *browser* security, which for most users is their gateway to the Internet. We will look at past and present flaws in the two most common browsers, Netscape's Communicator and Microsoft's Explorer. Then, we will have a look at *server* security, namely the security of the Microsoft Internet Information Server, and the Apache server, which is very popular and is used by about half of the Internet hosts. In addition, we will elaborate on the issue of CGI scripts. Because of the future importance of mobile code, we devote an entire chapter to *Java* and Java security. In the last chapter we treat a few *additional web security topics*, such as the spoofing and DNS attacks, cookies, and firewalls. Finally the appendix provides a list of useful references and sources, listed by topic and commented.

This way, we have identified a few of the key areas of Web security. Inevitably, some issues have to be left out or have not been treated in detail. Undoubtedly, a lot more could be said on topics like secure e-mail, key and certificate distribution, the use of Virtual Private Networks, the latest firewall technologies, encryption algorithms and so on. However, we have focused on the core issues of Web security, and have tried to provide a general overview and accessible assessment, as well as a more detailed and technical description of important security flaws.

## Chapter 2 : Browser Security

### 2.1 Introduction

#### *2.1.1 What is a web browser ?*

A browser is the client software that people use to access the Web. The browser sends requests to web servers which contain pages, identified by a unique URL (Uniform Resource Locator) and generally written in a document mark-up language called HTML (Hypertext Mark-up Language). Traffic between browser and server follows a protocol called Hypertext Transport Protocol (HTTP), which uses the TCP transport protocol. The browser uses the server name to open a TCP-connection to that server, usually through TCP port 80. Once the connection is open, the browser sends an HTTP *get*-command that includes the URL and a list of data formats that the browser will accept. If the server can fulfil the browser's request, it sends the requested page back to the browser via the same TCP-connection. As soon as the entire page has been sent, the connection is closed by the server. The same process is repeated for each file.

*Transactions* over the Web require more than one TCP-connection, since the client has to send information to the server. First a form is fetched from the browser to collect data from the user. Then, the HTML *post*-command transmits the data collected by the form to the server. The same command also indicates a CGI-script (Common Gateway Interface) on the server, programmed to collect the form's data, process it and return a Web page in response as an acknowledgement of the transaction. CGI-scripts can be tailored to run custom programs capable of implementing typical automated transactions.

While e-mail and ftp have carried most of the Internet traffic for years, Web browsers have been used more than any other Internet service. Therefore, flaws in browsers could possibly affect a very great number of users

#### *2.1.2 Web browser security*

The Internet is a heterogeneous network of computers where several services are provided. One of the most popular services is the World Wide Web, or Web browsing. Although electronic mail still seems to be the most widely used, browsing has opened much more opportunities for development and growth of different fields because of the multimedia capabilities it provides and the ability to share resources to many people located in different places at the same time. The Web gives the Internet a

feeling of real interactive communities taking place with no boundaries, as opposed to its physical counterpart.

To the end-user, the Web may feel like a safe and anonymous place, but this is untrue. First, every time someone is connected to a network there is the risk to be attacked by an intruder. Second, mobile code opens a new window of insecurities to web browsing because of the chance to download and run code on the client machine. Third, there is a race between the two biggest players in the Browser Industry, Netscape and Microsoft, to gain a bigger share of the web surfing population. This makes them rush into the market with products that are not always tested enough or that are developed without security on mind.

		Browser		
		Microsoft Internet Explorer	Netscape Navigator / Communicator	HotJava
Scripting Language supported	Java	X	X	X
	JavaScript	X	X	
	ActiveX	X		

**Figure : Popular web browsers and mobile code supported.**

There is a collection of security flaws for both browsers, most of them fall into one of two categories: Mobile Code or Memory/Buffer Overflow. Both flaws give the attacker the possibility of crashing the client's machine, damaging the system, breaching the user's privacy or simply annoying him. Not all browsers support every scripting language, they have even implemented them with a subjective approach in some cases. In this chapter, we treat some of the flaws discovered in the two most popular browsers : Netscape's Communicator and Microsoft's Explorer. For each browser, we provide a list of detected flaws, categorise them and/or provide more technical details about a few of them.

## 2.2 Netscape Communicator Flaws

### 2.2.1 History

Oct '98	1. <b>Frame Spoofing Bug.</b> Weakness in object models protection. No fix yet.  <a href="http://www.securexpert.com/framespoof/index.html">http://www.securexpert.com/framespoof/index.html</a> (secureXpert corporation web site)
	2. <b>Caching bug.</b> Involves SSL secured pages incorrectly interpreting HTML meta-tags. Netscape avoidance instructions released.  <a href="http://www.netscape.com/products/security/resources/bugs/nocache.html">http://www.netscape.com/products/security/resources/bugs/nocache.html</a>
	3. <b>Caching bug in Navigator.</b> A bug exploited using JavaScript. Fix not yet known.  <a href="http://www.news.com/News/Item/Textonly/0,25,28111,00.html">http://www.news.com/News/Item/Textonly/0,25,28111,00.html</a> (CNet News Magazine)
	4. <b>MIME type buffer overflow vulnerability.</b> A bug affecting only UNIX versions of Navigator. Netscape avoidance instructions released.  <a href="http://www.netscape.com/products/security/resources/bugs/mimebufferoverflow.html">http://www.netscape.com/products/security/resources/bugs/mimebufferoverflow.html</a>
	5. <b>Preferences Bug.</b> Involves JavaScript code writing to prefs.js file. No action taken.  <a href="http://www.geek-girl.com/bugtraq/1998_4/0145.html">http://www.geek-girl.com/bugtraq/1998_4/0145.html</a> (mailing list archive)
	6. <b>Injection Bug.</b> A bug exploited using JavaScript. Upgrade released.  <a href="http://www.netscape.com/products/security/resources/bugs/injection.html">http://www.netscape.com/products/security/resources/bugs/injection.html</a>
Feb '98	7. <b>Preferences Bug.</b> Involves reading prefs.js file. Upgrade released.  <a href="http://www.netscape.com/products/security/resources/bugs/preferences.html">http://www.netscape.com/products/security/resources/bugs/preferences.html</a>
Sept '97	8. <b>French Privacy Bug.</b> Involves reading prefs.js file. Upgrade released.

	<p><a href="http://www.netscape.com/products/security/resources/bugs/french.html">http://www.netscape.com/products/security/resources/bugs/french.html</a></p>
Aug '97	<p>9. <b>Santa Barbara Privacy Bug.</b> A bug exploited using JavaScript. Fix in upgrade.</p> <p><a href="http://www.netscape.com/products/security/resources/bugs/santabarbara.html">http://www.netscape.com/products/security/resources/bugs/santabarbara.html</a></p>
	<p>10. <b>Tracker Bug.</b> A bug exploited using JavaScript. Upgrade released.</p> <p><a href="http://www.netscape.com/products/security/resources/bugs/tracker.html">http://www.netscape.com/products/security/resources/bugs/tracker.html</a></p>
July '97	<p>11. <b>Singapore Privacy Bug.</b> Involves communication between JavaScript and Java applets. Upgrade released.</p> <p><a href="http://www.netscape.com/products/security/resources/bugs/singapore.html">http://www.netscape.com/products/security/resources/bugs/singapore.html</a></p>
	<p>12. <b>Bell Labs Privacy Bug.</b> A bug exploited using JavaScript. Upgrade released.</p> <p><a href="http://www.netscape.com/products/security/resources/bugs/bell_labs.html">http://www.netscape.com/products/security/resources/bugs/bell_labs.html</a></p>
June '97	<p>13. <b>Danish Privacy Bug.</b> Details kept secret, fix since released.</p> <p><a href="http://www.netscape.com/products/security/resources/bugs/danish.html">http://www.netscape.com/products/security/resources/bugs/danish.html</a>  <a href="http://www.cnnfn.com/digitaljam/9706/12/netscape_pkg">http://www.cnnfn.com/digitaljam/9706/12/netscape_pkg</a> (CNN).  <a href="http://www8.zdnet.com/pcmag/news/trends/t970612b.htm">http://www8.zdnet.com/pcmag/news/trends/t970612b.htm</a> (ZDNet Magazine).  <a href="http://www8.zdnet.com/pcmag/news/trends/t970613b.htm">http://www8.zdnet.com/pcmag/news/trends/t970613b.htm</a> (ZDNet Magazine).</p>
March '97	<p>14. <b>Event Logging Bug.</b> Entered data transferred to next site's logs. No fix found.</p> <p><a href="http://www5.zdnet.com/zdnn/content/inwo/0327/inwo0001.html">http://www5.zdnet.com/zdnn/content/inwo/0327/inwo0001.html</a>(ZDNet Magazine).</p>
Sept '95	<p>15. <b>Random Number Generator Bug.</b> Flaw tested on Solaris and HP-UX UNIX systems. Fixed by upgrade.</p> <p><a href="http://hplyot.obspm.fr/~dl/netscapsec/wsj.txt">http://hplyot.obspm.fr/~dl/netscapsec/wsj.txt</a>  <a href="http://hplyot.obspm.fr/~dl/netscapsec/cypherp1.txt">http://hplyot.obspm.fr/~dl/netscapsec/cypherp1.txt</a>  <a href="http://www-ns.rutgers.edu/www-security/archives/0866.html">http://www-ns.rutgers.edu/www-security/archives/0866.html</a>  <a href="http://home4.netscape.com/newsref/pr/newsrelease46.html">http://home4.netscape.com/newsref/pr/newsrelease46.html</a></p>

## ***2.2.2 Flaw categories***

The security flaws of Netscape Communicator can be grouped into the following categories in terms of the nature of flaw.

### ***1. Cache memory flaws***

These are privacy bugs in Communicator that let a malicious web site get the URLs that are contained in the current browser cache. So, the intruder can find out what sites a user has visited in the recent past. The most interesting thing is that this bug could reveal web site passwords, depending on whether sites include those passwords in their URLs. Most of the time, these bugs are exploited using JavaScript code. A temporal solution could be to set the size of cache memory to zero. See bugs 2, 4, 5, 6, 10 & 13 in history list.

### ***2. Mime type buffer overflow flaws***

This type of flaw occurs when a user visits an untrusted web site that requires a plug-in that is not currently installed. The Null Plug-in dialog box appears to assist the user in locating the correct plug-in, but in reality a hacker operating through that site could exploit a weakness in the Null Plug-in mechanism to force Communicator or Navigator to crash. Users can avoid this vulnerability by setting a preference for Navigator to prompt them before it attempts to download a Plug-in with an unknown MIME type. (MIME stands for Multipurpose Internet Mail Extension and is an Internet protocol that allows a user to send binary files (graphics, photos, sound, video files) across the Internet as attachments to e-mail messages.) See bug 12.

### ***3. Meta-tag flaws***

These problems have to do with the way Navigator negotiates HTML meta tags, especially caching meta tags. Meta tags describe the content of a page or provide specific instructions on how to treat it.

In cases where multiple people use the same copy of Communicator on the same PC, they have the opportunity to find out private and personal information about the other users when certain meta tags exist or do not exist in a web page. See bug 14.

### ***4. Preferences flaws***

All these flaws are concerned with the 'prefs.js' file. Malicious web site operators can read the 'prefs.js' from the hard disk of visiting users through an obscure series of steps, or make 'prefs.js' useless through JavaScript code. Information contained in this file can include e-mail addresses, domain names and passwords. As the most sensitive information that can be located in the 'prefs.js' file is the e-mail password, you can protect against an attacker learning your password by making sure it is not automatically stored in Communicator. See bugs 8, 9 & 11.

### ***5. Reading hard disk information flaws***

This type of flaw can allow malicious web site operators to retrieve known files from a visiting user's hard disk through an obscure series of steps. The only limitation is that the attacker must know the exact name and path of the file. Navigator users should download the updated version of Communicator in order to completely remove any risk of this bug. See bug 3.

### ***6. Second window/frame techniques in web site flaws***

The second window privacy bug can allow a hacker to see a user's browser information through a hacker-generated second browser window. To be exposed, a user must visit a malicious hacker's Web site and then visit other Web sites through the second browser window. The bug does not allow a hacker to access a user's hard drive or steal from, erase, or write information to a user's hard drive. See bug 7.

The frame spoofing bug allows a hacker to insert a frame of his own creation in a trusted web site. The hacker can then read any information entered in the frame. See bug 15.

### ***7. Netscape random number generator flaw***

This flaw relates to the predictability of Netscape's random number generator, which is used for encryption. Netscape uses symmetric key cryptography to scramble sensitive data so that it is unreadable by hackers snooping on the network. The intruders examine the so-called "random number generator" aiming to guess the encryption key by working out the seed value of the random number generator. Netscape corrected this type of vulnerability by increasing the amount of random information used to seed the generator. See bug 1.

## ***2.2.3 Detailed Analysis***

This section looks at two of the flaws in detail. We have decided to concentrate on the Frame Spoofing Vulnerability (see bug 15 in history) and Netscape's Random Number Generator flaw (see bug 1 in history).

### ***1. The Frame Spoofing Vulnerability:***

#### Description of flaw:

This flaw has been found by SecureXpert Labs. The vulnerability enables the author of a nefarious Web site or e-mail message to "spoof" information presented by another Web site.

For example, the nefarious Web site could cause false or embarrassing information to be displayed by another Web site; or it could cause the other Web site to display a form which, if filled in, would send information back to the attacker.

This vulnerability can also be exploited through e-mail. For example, a user might receive a HTML e-mail message appearing to be from a trusted source (since standard e-mail is easily forged) containing a message advertising a product or service. That e-mail could then cause a well-known and trusted Web site to open. The Web site could then be manipulated to confirm the attacker's message.

The vulnerability occurs because Netscape (and Microsoft) browsers fail to correctly prevent a Web site or HTML e-mail message from replacing a frame displayed by the other site with content that is under the attacker's control.

### How the flaw works:

This vulnerability exists in the protections built into the object models in both the Netscape and Microsoft browsers.

The vulnerability occurs simply because Navigator and Explorer fail to protect the frames[] array from cross-domain write access, as they do for the layers[] and fields[] arrays, and as they do for read access to all of these objects and others.

All that is required for a site to exploit the vulnerability is either one of the following:

- The attacker has opened the victim site's window -- either by sending a HTML e-mail, or from a scripted Web page (required for the JavaScript-based variant of the attack)
- The attacker knows, or can guess, the name of a frame in the victim site (for the HTML-based variant)

This attack works irrespective of whether the victim has Java and JavaScript enabled or not. If Java and JavaScript are disabled, the user must have two windows open, and must click on a link to activate the exploit.

The 'TARGET=...' syntax in HTML can also be exploited to write a document into another site's frameset, by the simple expedient of TARGETing a frame name which does not exist in your own site. Since HTML e-mail documents can automatically open a browser window displaying a Web site, setting up the circumstances for a rather damaging exploit of this hole is quite easy. Netscape Navigator can compound the problem by misleading you about the origin of what you are viewing. If you click the padlock security icon (at the bottom left of the Navigator window) while viewing a site that has been altered by this code, you will see that Navigator asserts (incorrectly) that the page is authentic -- under the heading "Verification" Navigator states "This page comes from the site: www.so-and-so.com".

### Flaw prevention:

Webmasters can prevent their sites from being used as unwitting accomplices to this attack as follows;

It appears that the attacker needs to guess the name of the target frame, which for static pages is trivial. If this is true, then dynamically generated HTML can defeat the attack at the host side, by generating an address- or session-based frame name that is not guessable by the attacker. To do this, modify the frame name to use the hash of a secret appended to the remote's IP address:

```
frame_name = original_name + SHA1( browser_IP_address + secret);
```

## ***2. Netscape's Random Number Generator:***

### Description of flaw:

This flaw relates to the predictability of Netscape's random number generator, which it uses for encryption. Netscape uses symmetric key cryptography to scramble sensitive data so that it is unreadable by hackers snooping on the network. In September 1995, two graduate students at the University of California at Berkeley posted a message to the Internet's "Cypherpunks" mailing list, claiming that the random number that generates the key was "fairly trivial to guess" and that the key "usually takes less than one minute to find." Rather than try to break ciphertext to find the key (an exhaustive key search), the two graduate students examined the so-called "random number generator" and discovered that the seeding number isn't so random, allowing them to guess the encryption key. It took the two students, Ian Goldberg and David Wagner, two days to identify the vulnerability and write a software program that could guess the encryption key in less than one minute.

### How the flaw works:

This was tested on Solaris and HP-UX Unix systems.

Netscape seeds the random number generator it uses to produce challenge-data and master keys with a combination of the time in seconds and microseconds, the process identification number (pid) and the parent process identification number (ppid). Of these, only the microseconds is hard to determine by someone who

- (a) can watch your packets on the network and
- (b) has access to any account on the system running Netscape.

Even if (b) is not satisfied, the time can often be obtained from the time or daytime network daemons; an approximation to the pid can sometimes be obtained from a mail daemon (the pid is part of most Message-ID's); the ppid will usually be not much smaller than the pid, and has a higher than average chance of being 1. Clever guessing of these values will in all likelihood cut the expected search space down to less than brute-forcing a 40-bit key, and certainly is less than brute-forcing a 128-bit key. Subsequent http connections after the first (even to different hosts) seem not to reseed the RNG. This makes things much easier, once you've broken the first message. Just keep generating 16 bytes of random numbers until you get the challenge-data for the next message. The next key will then be the 16 random bytes after that.

See also <http://hpilyot.obspm.fr/~dl/netscapsec/cypherp1.txt> and <http://www-ns.rutgers.edu/www-security/archives/0866.html> for a C program to find the key.

### Flaw prevention

In a press release (25/9/95) Netscape claimed to have addressed the potential vulnerability by increasing the amount of random information used to seed the random number generator in its security implementation. "The new solution uses many times more random information than the previous version, ensuring much greater degrees of difficulty in identifying the key used to encrypt a particular session."

<http://home4.netscape.com/newsref/pr/newsrelease46.html>

## 2.3 Internet Explorer Flaws

### 2.3.1 History

21 <sup>st</sup> Oct 98	<p>1. <b>IP address flaw</b></p> <p>The flaw lies in the way Internet Explorer parses IP addresses, which could allow malicious Webmasters to run code on users' machines.</p>
20 <sup>th</sup> Jan 98	<p>2. <b>Mac IE 4.0 problems with SSL Triple-DES encryption.</b></p> <p>When making a secure connection to a server, Internet Explorer 4.0 for Macintosh would offer multiple supported levels of encryption. However, later additional server-side processing was also checking to ensure that 128-bit Release Candidate (RC) 4 encryption was being used. Unfortunately, this process would not also allow Triple-DES connections. This resulted in a refused connection even though the browser does in fact support 128-bit RC4 as well as the stronger Triple-DES encryption.</p> <p>The patch is <a href="http://www.microsoft.com/windows/ie/security/mac128.html">http://www.microsoft.com/windows/ie/security/mac128.html</a></p>
14 <sup>th</sup> Jan 98	<p>3. <b>Internet Explorer mk:// overrun vulnerability.</b></p> <p>A newer exploit occurs when the browser tries to access URLs that have an "mk" prefix.</p>
21 <sup>st</sup> Nov 97	<p>4. <b>Page Redirect Issue.</b></p> <p>The flaw could occur when a user connects to a site that requires basic user authentication information such as user name and password. The Web site then redirects the user to another page where the authentication information could potentially be captured by the second Web site. However, this can happen only if the Web site has malicious intent and set the site in a certain way so that it can obtain the information.</p>
12 <sup>th</sup> Nov 97	<p>5. <b>Internet Explorer 4 has res:// overrun vulnerability.</b></p> <p>This flaw stemmed from Explorer's handling of URLs that began with the "res" prefix and ran longer than 256 characters.</p>
17 <sup>th</sup> Oct 97	<p>6. <b>Internet Explorer 4 local file access "Freiburg" vulnerability.</b></p> <p>Internet Explorer Web browser allows malicious Web sites operators to view the content of certain files on a user computer.</p> <p>The patch is available at</p>

8 <sup>th</sup> May 97	<p>7. <b>Internet Explorer vulnerability concerning PowerPoint.</b>          This problem involves the PowerPoint application. There is a feature which allows an application to be run from within PowerPoint with no warning to the user. Internet Explorer supports the viewing of the PowerPoint files so is potentially for misuse of this feature from the malicious website operators.</p> <p>The patch is available at  <a href="http://www.microsoft.com/msdownload/iesecurity/iepptsecurity.html">www.microsoft.com/msdownload/iesecurity/iepptsecurity.html</a></p>
24 <sup>th</sup> Mar 97	<p>8. <b>Cybersnot Security Problem.</b>          This security flaw involved shortcut and URL files.</p>
14 <sup>th</sup> Mar 97	<p>9. <b>Username and Password vulnerability.</b>          The security flaw can reveal your username and encrypted password by simply visiting a website (<a href="http://www.ee.washington.edu/computing/iebug/passout.cgi">http://www.ee.washington.edu/computing/iebug/passout.cgi</a>) which exploits this hole. There is still no fix for this flaw.</p>
7 <sup>th</sup> Aug 96	<p>10. <b>ActiveX exposes vulnerabilities in Internet Explorer.</b>          Exploder is an ActiveX control with demonstrates security problems with Microsoft Internet Explorer. Exploder performs a clean shut down of Win95 and will turn off the power on machines that have a power conservation BIOS (green machines).</p>
Date: unknown	<p>11. <b>Java Applets</b>          This bug involves users who run network programs from the same machine and then execute potentially dangerous Java applets. These applets can continue to run in the user's cache file and could access non-password protected data. This bug was fixed in version 3.02.</p> <p>Patch : <a href="http://www.microsoft.com/ie/security/?/ie/security/java.htm">http://www.microsoft.com/ie/security/?/ie/security/java.htm</a>.</p>
Date: unknown	<p>12. <b>Delete/Damage file</b>          This bug could be exploited by a malicious website operator to delete or damage files on a users computer's. This is similar to the Cybersnot flaw described above. This bug was fixed in version 3.02.</p> <p>Patch : <a href="http://web.mit.edu/crioux/www/ie/index/html">http://web.mit.edu/crioux/www/ie/index/html</a></p>
Date: unknown	<p>13. <b>The Bell Labs Privacy Bug</b>          This bug allows malicious website operators to see information</p>

	<p>sensitive information) is exchanged between the client and server, as well as track websites you visit. This problem was fixed in the final release version 4.0.</p> <p>Patch : <a href="http://www.microsoft.com/ie/security/?/ie/security/bell.htm">http://www.microsoft.com/ie/security/?/ie/security/bell.htm</a></p>
--	--

## ***2.3.2 Detailed Analysis***

### ***1. Buffer Overrun Issue***

This flaw stemmed from Explorer's handling of URLs that began with the "res" prefix and ran longer than 256 characters. When the browser visited such an HTML document with more than 256 characters, the extra characters were dumped into system memory. That either crashed the browser or, if a programmer wrote those extra characters as executable code, ran the code on the system unbeknownst to the system's owner.

A newer exploit occurs when the browser tries to access URLs that have an "mk" prefix, a protocol normally reserved for compressed HTML resource files that Internet Explorer can extract from the system and read. As with the "res" problem, a programmer must deliberately create a URL with more than 256 characters to cause a crash or overrun. However, unlike the "res" problem, a programmer needs to know details about each specific system and adjust the bug to each target system. .

### ***2. IP Address Flaw***

The flaw lies in the way Internet Explorer parses IP addresses, which could allow malicious Webmasters to run code on users' machines. Internet Explorer has two ways of parsing IP address. Firstly, if the browser encounters an address in the commonly used format of four numbers separated by dots such as 1.2.45.77, it treats the address as a Web site. The second method is that if IP addresses are represented as a single value derived by expressing the four number address as a power series of 256 for example  $(1 \times 256 \times 256 \times 256) + (2 \times 256 \times 256) + (3 \times 256) + 4$ . The browser then assumes such single-value IP addresses locate intranet-based sites.

As a result, if the user has applied less stringent security checks to intranet-sourced content, Web site administrator could use this hole to run active content on the user's computer without obtaining the user's permission to download and run it at first. However, as stated before, the user would have to set IE4's default intranet security zone settings to be affected by this flaw.

## Chapter 3 : Server Security

### 3.1 Introduction

First we have a look at two popular servers, namely the Microsoft Internet Information Server and the Apache server. Next, we have a look at CGI scripts.

### 3.2 Microsoft Internet Information server

#### 3.2.1 Introduction

Microsoft's Internet Information Server provides HTTP, ftp and Gopher services for workstation's running Microsoft NT Server. The server runs on Alpha, Intel x86, PowerPC and Mips processors. However, it will only run on the Windows NT operating systems. Currently it is in version 4.0 and it "comes with" NT server 4.0. Generally, it is quite similar to what other web servers offer with a few main differences. Some of the differences are :

- Can be integrated with NT 4 user database
- Users Active Server Pages
- ISAPIs
- Good integration with Microsoft Back-Office Suite

A table with more detailed listing of the features of IIS is included below :

<p>Launching and Logging</p>	<ul style="list-style-type: none"> <li>➤ Can write to multiple logs</li> <li>➤ Log files can be automatically cycled or archived</li> <li>➤ Performance measurement logs</li> <li>➤ CGI scripts can create their own log entries</li> <li>➤ Can serve different directory roots for different IP addresses</li> <li>➤ CERN/NCSA common log format</li> <li>➤ Runs as Windows NT service and/or application</li> <li>➤ Can listen to multiple addresses and ports</li> <li>➤ Can track individual users in log</li> <li>➤ Logging with syslog (Unix) or Event Log (Windows NT)</li> <li>➤ Can generate browser log entries</li> </ul>
<p>Protocol Support and Includes</p>	<ul style="list-style-type: none"> <li>➤ Comes with SNMP agent</li> <li>➤ Access to server state variables from CGI or other scripting</li> </ul>

	<ul style="list-style-type: none"> <li>➤ Non-supported methods can invoke a script</li> <li>➤ Select documents based on Accept header/user-agent headers</li> <li>➤ Has built-in scripting language</li> <li>➤ Supports Microsoft ISAPI</li> </ul>
Security	<ul style="list-style-type: none"> <li>➤ Integrated certificate server</li> <li>➤ Prohibit access by domain name</li> <li>➤ UID CGI Execution</li> <li>➤ Prohibit access by IP address</li> <li>➤ Prohibit access by user and group</li> <li>➤ Supports S-HTTP</li> <li>➤ Can change user access control list without restarting server</li> <li>➤ Hierarchical permissions for directory-based documents</li> <li>➤ Prohibit access by directory and file</li> <li>➤ Configurable user groups(not just a single user list)</li> <li>➤ Can hide part of a document based on security rules</li> <li>➤ Supports SSL v. 2 and 3</li> <li>➤ Can require password(Authorisation: user)</li> <li>➤ Security rules can be based on URLs</li> </ul>
Additional Security Features	<ul style="list-style-type: none"> <li>➤ NT challenge response</li> <li>➤ X.509 certificate manager</li> <li>➤ Ability to map to NT authentication.</li> </ul>

### ***3.2.2 Insecurity due to inadequacies in default configuration***

The majority of software packages come insecure “out of the box”. It is the job of the person who is responsible for the piece of software to “patch” up the insecure configurations. This is true of most (if not all) of the commercial packages.

The “patching” of software is a deceptively simple task, usually just by changing the default configuration. In the majority of the cases, the difficulty lies in the fact that the person who is installing the software has no idea that there are “inherent” insecurities in the piece of software. Even for a person in the know, it is often not easy to ensure that each and every one of these inherent insecurities due to default configuration has been put right.

The Microsoft Internet Information Server (IIS) is a web server running on the NT platform. It is closely integrated with the NT security model and as such is also heavily dependent on the security architecture of NT. It has a number of default configurations that are insecure.

- 1) Default mapping of .bat and .cmd extensions to cmd.exe. This enables the execution of any command by the execution of an arbitrary file (even if it doesn't

- 2) Installing IIS on the domain server will by default add the IUSR\_<nodename> account to the 'Domain Users' group, allowing anonymous users access rights of 'Domain Users' group.
- 3) The existence of certain default 'tools' that come with an installation of IIS are known to be able to cause security concerns. Some of these tools will allow unauthorised creation of files on the server while others allow the viewing of Active Server Pages source codes. These tools could also allow a denial of service attack to be launched against the server. Some of the tools are :
  - Newdsn.exe
  - Webhits.exe
  - Getdrvrs.exe
- 4) Remote Data Service (RDS) is a component of Microsoft Data Access Components (MDAC), which is installed by default when IIS is installed via the Windows NT Option Pack. RDS enables controlled Internet access to remote data resources through the IIS. One of the component of RDS (DataFactory) allows implicit remoting of data access requests and hence can be exploited to allow unauthorised Internet clients to access OLE DB datasources.
- 5) The following directories are marked executable by default :
  - /ROOTDIR/msadc
  - /ROOTDIR/News
  - /ROOTDIR/Mail
  - /ROOTDIR/cgi-bin
  - /ROOTDIR/SCRIPTS
  - /ROOTDIR/IISADMPWD
  - /ROOTDIR/\_vti\_bin
  - /ROOTDIR/\_vti\_bin/\_vti\_adm
  - /ROOTDIR/\_vti\_bin/\_vti\_aut

In a default installation, the physical drive mapping of the above directories will be :

msadc	c:\program files\common\system\msadc
News	c:\inetpub\News
Mail	c:\inetpub\Mail
cgibin	c:\inetpub\wwwroot\cgi-bin
SCRIPTS	c:\inetpub\scripts
IISADMPWD	c:\WINNT\System32\inetrv\iisadmpwd
_vti_bin	(only with FrontPage installation)

The default NTFS permissions are overly permissive and allow change control (RWXD) to the Everyone group by default, with the exception of msadc which is full control (!) to Everyone.

### ***3.2.3 Inherent Web Server Software Issues***

The software industry is very competitive and there is strong pressure on companies to release new software packages with more functions as well as ‘bells and whistles’. Software packages are often released before they can be fully tested.

As a result, software is being released with progressively more serious ‘bugs’. Some of the software ‘features’ released with IIS are:

- 1) The ability to access files outside of the webserver content root directory by using a URL such as <http://www.domain.com/..\>.
- 2) IIS depends on NTFS to implement the access control of a particular file. If a file is not accessible by IUSR\_<nodename> account, IIS will send back a Web Server Basic Authentication request. The issue here is that for an executable script, even if all rights to the directory are revoked, the script will be executed without challenging the user for credentials.
- 3) IIS 4.0 if installed together with MS Proxy 2.0 with multiple IP addresses bound to the NIC will cause an access violation. This is due to a corruption in the process heap due to the enabled Socks service.
- 4) If IIS is used to manage multiple virtual Intranet sites, the accounts IIS installs to log activity uses the default password of “password” for the second and subsequent accounts. Only the first account IIS creates uses a random password.
- 5) IIS ftp service can be configured to allow connections from only certain “trusted” IP address. However, this could be by-passed by executing a “bounce” attack on the IIS ftp server using a second ftp server.
- 6) When an IIS server is configured to run Perl scripts, data could be read from execute only virtual directories. This is because any URL appended with .pl will be passed to perl.exe, including URLs like <http://www.domain.com/scripts/file.ext%20.pl>
- 7) The IIS SMTP server listens on port 25 as well as port 465 (!) Port 465 was initially an IANA registered port for SSMTP which had been revoked.
- 8) If a file with a long filename is protected, it could still be access via its canonical name without restriction. Example, the protected “someslongfile.htm” could still be accessed via the name “someslo~1.htm”. This applies to directories as well.

### ***3.2.4 Active Server Pages security Issues***

- 1) Any asp pages can create, read, write or overwrite any file on the system regardless of security permissions using the Scripting.FileSystemObject.
- 2) IIS allows users to read the contents of .asp files instead of executing it by appending a ‘.’ or a series of ‘.’s to the end of the URL. Example :  
<http://www.domain.com/default.asp>.
- 3) After a hotfix was released for (2) the same effect could be reproduced this time by replacing the ‘.’ with %2e in the .asp example :  
<http://www.domain.com/default%2easp>
- 4) The source code of an asp page could be read via the unnamed data stream.  
Example : [http://www.domain.com/default.asp::\\$DATA](http://www.domain.com/default.asp::$DATA)

### ***3.2.5 Denial of Services attacks***

Denial of Service (DoS) attacks are one of the simplest methods of compromising the security of a web server. There are a number of reasons why the attacks work, often it is due to the fact that programmers did not handle their memory allocations correctly thereby allowing a particular variable to overflow its allocated memory space.

This form of attack is difficult to find and is usually difficult to execute as well. However, a successfully implemented attack will usually cause the infamous “Blue Screen of Death” on the NT machine. Some of the DoS attacks that IIS is vulnerable to are as follows :

- 1) Telnet to port 80 of the IIS server and issue the command GET ../ .. <Enter>
- 2) Sending around 15 commands (just clicking reload a lot of times etc) and the CPU usage shoots up to 100% and the file server runs amok. (IIS v2)
- 3) IIS services will stop when it receives a CGI request from a browser that contains between 4k to 8k chunk of data in the URL. Example :  
<http://www.domain.com/script.exe?variable=XXXXXXXXX.....>
- 4) If a site is running IIS 4.0 FTP server with more than 100 different FTP virtual directories or virtual sites, a DoS can be performed by sending more than 10 simultaneous PUT or DELETE ftp orders against a public ftp directory.
- 5) When multiple passive connections are made to a single FTP server via the PASV FTP command, it is possible to use up all available system threads for IIS and subsequent connections will fail.

## 3.3 Apache Server

### 3.3.1 Introduction

The Apache Web server is a public domain Web server developed by a loosely knit group of programmers, now called the Apache group. The first version was published in 1995, based on the NCSA httpd Web server. Because it was developed from existing NCSA code plus various patches, it was called *a patchy server*, hence the name.

Apache has become the world's most popular Web server because of its sophisticated features, excellent performance and free availability. The original version was written in UNIX, but there are now versions that run under OS/2, Windows and other platforms.

### 3.3.2 Flaw categories

Apache servers present an extensive array of sophisticated features, only to be completely understood by experienced Apache programmers. The flaws described in this paragraph were published by the Apache group as part of a 'Security Advisory'. They should not be present anymore in version 1.2.5. The technical details below are intended for those who have a solid grounding in how Apache works.

#### 1. *Buffer overflow in `cfg_getline()`*

`cfg_getline()` is a function that the Apache core and several Apache modules use to read certain types of files from disk. Some examples of the type of files that read with this are `htaccess`, `htpasswd` and `mod_imap` files. It is possible to create a sequence of data such that a buffer overflow occurs while `cfg_getline` is reading from a file. If someone has access to create any of these types of files on the server, this hole is generally exploitable to gain full access to the user Apache runs as. On most systems, this is of little consequence since users already have such access via methods such as the creation of their own CGI scripts. If, however, the server is secured so that the user has no access to the server other than to create and modify files (eg. a "ftp only" account with no ability to create CGI scripts) this could allow increased access to the server.

#### 2. *Several coding errors in `mod_include`*

There are several coding problems in `mod_include` that can result in a buffer overflow or in the child process going into an infinite loop. The same comments about the nature of the risk apply here as do for the `cfg_getline()` overflow. Generally, a user already needs to have access to the server to exploit this. Note that it is possible to

setup a document that deliberately allows this to be remotely exploited, however such a document would be very rare in practice. If users are not allowed to use `mod_include`, then they can not exploit these holes.

### ***3. Inefficient removal of duplicate '/'s ("beck" exploit)***

The code in the `no2slash()` function used to collapse multiple '/'s in a request for access checking purposes is very inefficient. It is  $O(n^2)$  in the number of '/'s in the input. What this means is that as the input size grows, it very quickly requires vastly increased CPU time to process the request. By sending many requests with a large number of '/'s in to a server, it is possible to cause a large amount of CPU time to be used in processing these requests. Making multiple simultaneous requests of this nature could result in a high load average, high CPU usage, and possibly starving other processes for CPU resulting in a denial of service attack. This does not allow for any compromise of the server. The fixed version of the `no2slash()` function is  $O(n)$  and does not allow for this attack.

### ***4. Possible buffer overflow in "logresolve" program.***

The `logresolve` program is used for non-realtime processing of logfiles to convert numeric IP addresses into host names. In some cases, it may be possible for a remote user who has control of a DNS server to return a hostname specifically designed to exploit a coding hole in `logresolve`. This can only happen on a system where either the `MAXDNAME` define does not exist and the resolver can return names longer than 256 characters or where the `MAXDNAME` define does exist but is less than the maximum length of hostname that the resolver can return. Even on such (arguably broken) systems, this would be very difficult to exploit. The number of systems which are impacted by this is very small. This problem is a potential concern only if you use the `logresolve` program.

### ***5. Insufficient data validation in mod\_proxy***

The `ftp proxy` part of `mod_proxy` accepts directory listings from remote `ftp` servers and converts them to HTML to send to the client. It is possible to deliberately create a listing that will cause Apache to dump core. This hole does not compromise the server; the only risk is that it would be possible to use this to create a denial of service attack which would render the server effectively inoperative. If `mod_proxy` is not used, the risk is avoided. When the use of `mod_proxy` is restricted, then only those users allowed to use it can attempt to exploit this problem.

### ***6. Possible buffer overflow reading from the proxy cache***

When caching is enabled in `mod_proxy`, Apache writes cached files to disk as the user that the server runs as. If an attacker can gain access to this user id (eg. by running a CGI script from a pre-existing account on the machine) then they can modify the filenames on disk resulting in a buffer overflow. Because the data is limited to what can be stored in a filename (not the file, just the filename), and they already need to have access to the user ID the server runs as to exploit this, the risk is minimal. The main instance where this may be a cause for concern is if there is privileged information stored in memory by the web server, such as an unencrypted SSL key. This same caution, however, applies to the other buffer overflows listed. If `mod_proxy` is not used, this problem can not be exploited.

### ***7. Unreadable htaccess files were ignored***

Previously, if a `htaccess` file was unreadable Apache ignored it. This is, from a security standpoint, a poor idea because it goes against the principle of "if in doubt, deny access". This had already been corrected in the 1.3 development tree, but the Apache Development Team had refrained from making the change in 1.2 because it could cause unexpected behavior on existing sites. After some reconsideration, and as of 1.2.5, Apache will now reject requests if there is a `htaccess` file present in the relevant directory tree that is unreadable for any reason. It is also possible, in very rare conditions, for this to be used to bypass `htaccess` files restricting access to a directory or file. The only case where this can happen is if the attacker can form a request that results in the full path to the `htaccess` file being too long (on most systems, meaning over 1024 characters) yet the request for the protected file in the same directory is not too long. The only normal case where such an attack could be possible is if there is a symbolic link such as "`somedir -> .`" created in the document tree.

## 3.4 CGI-Scripts

### 3.4.1 Introduction

CGI stands for Common Gateway Interface and is a specification for transferring information between a Web server and a CGI program. A CGI program is the most common way for web servers to interact dynamically with users. For example, many HTML pages that contain forms, use a CGI program to process the form's data once submitted. While Java applets, Java scripts or ActiveX controls run on the user's machine rather than on the Web server (so-called client-side solutions), the use of CGI is a server-side solution, since all processing occurs on the Web server. For busy Web sites, this can noticeably slow down the server, because a new process is started each time a CGI script is executed.

When a Web daemon retrieves an HTML-file, this document is static : it exists in a constant, non-changing state. A CGI program on the other hand is executed in real time, so it can output dynamic information. Since a CGI program is executable, it is as if one lets the world run a program on your system, which calls for quite some security precautions.

CGI scripts are a major source of security holes. Although the CGI protocol is not inherently insecure, CGI scripts must be written with just as much care as the server itself. Quite a number of widely distributed CGI scripts contain known security holes. First we provide a (non-exhaustive) list of popular CGI scripts that are known to (have) contain(ed) security flaws. Next, we will explain some general security issues.

### 3.4.2 List of flaws

*(Assembled by Lincoln D. Stein, author of the WWW security faq)*

<b>1. Matt Wright's TextCounter</b> Versions 1.0-1.2 (Perl) and 1.0-1.3 (C++)	Earlier versions of the TextCounter program, which is used to place page hit counts on pages, fail to remove shell metacharacters from user-provided input. As a result, remote users can execute shell commands on the server host. This affects both the Perl and C++ versions.  Fixes/Patches : fixed in higher versions (Perl) <a href="http://www.worldwidemart.com/scripts/textcounter.shtml">http://www.worldwidemart.com/scripts/textcounter.shtml</a> (C++) <a href="http://www.worldwidemart.com/scripts/C++/textcounter.shtml">http://www.worldwidemart.com/scripts/C++/textcounter.shtml</a>
<b>2. Guestbook scripts</b> Versions : Selena Sol	This flaw takes advantage of scripts that don't strip HTML tags from user-provided input and which write the guestbook file to a directory that allows SSI (Server-

<p>others</p>	<p>Side Includes = snippets of server directives included in HTML documents).</p> <p>Fixes/Patches : <a href="http://www.eff.org/~erict/Scripts/guestbook.html">http://www.eff.org/~erict/Scripts/guestbook.html</a></p>
<p><b>3. Excite Web Search Engine (EWS)</b>          Versions 1.0-1.1, Both Unix and NT versions of the search engine</p>	<p>The Excite Web Search Engine fails to check user-supplied parameters before passing them to the shell, allowing remote users to execute shell commands on the server host. The commands will be executed with the privileges of the Web server.</p> <p><i>Note:</i> This bug only endangers your web site if you have the search engine installed locally. It does not affect sites that link to Excite.com search pages, or sites that are indexed by the Excite robot.</p> <p>Fixes/Patches : <a href="http://www.excite.com/navigate/patches.html">http://www.excite.com/navigate/patches.html</a></p>
<p><b>4. Info2www</b>          Versions 1.0-1.1</p>	<p>Info2www, which converts GNU “info” files into Web pages, fails to check user-provided filenames before opening them. As a result, it can be tricked into opening system files or executing commands containing shell metacharacters.</p> <p>Fixes/Patches : fixed in version 1.2</p>
<p><b>5. Count.cgi</b>          Versions: 1.0-2.3</p>	<p>Count.cgi, widely used to produce page hit counts, contains a stack overflow bug that allows malicious remote users to execute Unix commands on the server by sending the script a carefully crafted query string.</p> <p>Fixes/Patches : <a href="http://www.fccc.edu/users/muquit/Count.html">http://www.fccc.edu/users/muquit/Count.html</a></p>
<p><b>6. Webdist.cgi</b>          Part of IRIX Mindshare Out Box          Versions 1.0-1.2</p>	<p>This script is part of a system that allows users to install and distribute software across the network. Due to inadequate checking of the CGI parameters, remote users can execute commands on the server system with the permissions of the server daemon.</p> <p>Fixes/patches : not fixed</p>
<p><b>7. Php.cgi</b>          Versions: all up to 2.0b11</p>	<p>This script, which provides an HTML- embedded programming language in HTML pages, database access, and other nice features, allows anyone on the Internet to run shell commands on the Web server host machine. In addition, versions through 2.0b11 contain known security holes. The Apache module version of Phpis said not to contain these holes since it does not run as a cgi script</p>

	Fixes/Patches : <a href="http://php.iquest.net/">http://php.iquest.net/</a>
<p><b>8. files.pl</b> part of the Novell WebServer Examples Toolkit Versions: 2.0</p>	<p>Due to a failure to check user input, the scripts that come with the Novell WebServer installation allow users to view any files or directory on your system, compromising confidential documents, and potentially giving crackers the information they need to break into your system.</p> <p>Fixes/Patches : none</p>
<p><b>9. Servers running Microsoft FrontPage Extensions</b> Versions: 1.0-1.1</p>	<p>Under certain circumstances, unauthorised users can vandalise authorised users' files by appending to them or overwriting them. On a system with SSI enabled, remote users may be able to exploit this bug to execute commands on the server.</p> <p>Fixes/Patches : <a href="http://www.microsoft.com/frontpage/documents/bugQA.htm">http://www.microsoft.com/frontpage/documents/bugQA.htm</a></p>
<p><b>10.Nph-test-cgi</b> Versions: all versions</p>	<p>This script, included in many versions of the NCSA httpd and apache daemons, can be exploited by remote users to obtain a file listing of any directory on the web server.</p> <p>Fixes/Patches : none</p>
<p><b>11.Nph-publish</b> Versions: 1.0-1.1</p>	<p>Under certain circumstances, remote users can attack and modify world-writable files on the server.</p> <p>Fixes/Patches: <a href="http://www.genome.wi.mit.edu/~lstein/server_publish/">http://www.genome.wi.mit.edu/~lstein/server_publish/</a></p>
<p><b>12.Phf.cgi</b> Distributed with NCSA httpd and Apache Versions : all versions <b>AnyForm</b> Versions: 1.0 <b>FormMail</b> Versions: 1.0</p>	<p>Remote users can execute commands on the server.</p> <p>Fixes/Patches: <a href="http://hoohoo.ncsa.uiuc.edu/">http://hoohoo.ncsa.uiuc.edu/</a> Fixes/Patches: <a href="http://www.uky.edu/~johnr/AnyForm2">http://www.uky.edu/~johnr/AnyForm2</a> Fixes/Patches: <a href="http://alpha.pr1.k12.co.us/~mattw/scripts.html">http://alpha.pr1.k12.co.us/~mattw/scripts.html</a></p>

### 3.4.3 General comments

Overall, CGI-scripts can present security holes in two predominant ways :

1. Scripts may intentionally or unintentionally leak information about the host

2. Scripts that process remote user input may be vulnerable to attacks in which the remote user tricks them into executing commands.

Since a CGI program runs as the userID of the HTTP server, it is not a good idea to set the latter as *root*, but rather as *nobody* or *wwwuser* or some other restricted, no-login userID. Even then, a CGI script running as *nobody* still has enough privileges to mail out the system password file, examine the network information maps, or launch a log-in session on a high numbered port. This is one of the reasons why non-administrators should not have the permission to put CGI programs on the server without careful prior checking. It is also recommendable to restrict CGI scripts to one single directory in the document tree (usually called *cgi-bin*), so it is much easier to keep track of what scripts are installed on the system.

A CGI program can be written in any language that allows it to be executed on the system : in a programming language (such as C,C++, Fortran or VisualBasic) or in a scripting language (such as Perl, TCL or a Unix shell). In general, the use of compiled languages such as C is considered to be safer than the use of interpreted languages like Perl. There are several reasons :

- One would rather not give an intruder access to the script's source code. If one writes a CGI script using a compiled language, one can compile it to binary form before placing it in the *cgi-bin* directory, thereby not giving intruders direct access to the source code.
- Furthermore, big software programs like shell and Perl interpreters are more likely to contain bugs and therefore security holes.
- As mentioned above, the invocation of system commands from within scripts is one of the major potential security holes. In most compiled languages this invocation is much more difficult than in scripting languages, in which it is easy to send data to system commands and capture their output.

On the other hand :

- Interpreted scripts tend to be shorter and thus more readable and understandable by users other than the author. This could allow to discover certain security holes.
- Languages like Perl contain a number of built-in features designed to catch some common potential security holes.

CGI scripts are without any doubt interesting tools, but when writing or using them, one should bear in mind the following :

- Do not give too much information about your site and server host.
- When coding in a compiled language, do not make assumptions about the size of the user input. This could cause a buffer overflow and crash the program. It could allow a hacker to execute system commands remotely.
- Never pass unchecked remote user input into a shell command.
- Avoid unnecessary complexity.
- If the script reads or writes files on the host system, or if it interacts with other programs, check that this is done securely.

## Chapter 4 : Java

### 4.1 Mobile code

Mobile Code is another example of what people define as “the same problem with a different twist”. Trying to download code from another machine across a network is not new.

A couple of years ago, with the proliferation in use of modems and Bulletin Board Services (BBS) the discussion focused on the security of downloading files from another machine. Most of the time, people agreed that all depended on trust. Then came the Internet and a new media to distribute more shareware and freeware arrived. People could use services like Archie and ftp to download them. But the same question was asked again and again, is it secure to download files ?

The risk of downloading another person’s code is clear. Most of the time the person who downloads is not an expert programmer and can’t be sure if the file has a virus, logic bomb or Trojan horse attached to it.

In 1992 Tim Berner-Lee invented the World Wide Web, then the GUI browsers arrived and the real explosion of downloading files began. As a user-friendly interface, the browser gave more people the ease to connect to remote sites and download files. The HTML-based web was static at the beginning. Then companies like Netscape, Sun and Microsoft introduced their own scripting languages to produce dynamic pages. The scripting languages would be used to produce code that should be run on the browser. So that meant downloading files across a network from an another host, same problem with a different twist.

The production of secure mobile code involves not only the code itself but also the platform where it is going to be run. Each implementation of mobile code - Java, ActiveX or JavaScript - has a different approach for this. To our belief, Java seems to be the front runner in this race. JavaSoft, Sun’s Java development division, is pushing toward a sound system with the combination of type safety and dynamic loading. During this process, they have encountered several risks and flaws. The implementation of the security model they want to achieve is not easy by any means. This is the reason why we chose to analyse the Java Security Model and the flaws they have found as a representative of mobile code technologies.

## 4.2 What is Java ?

A high-level programming language called Java was developed by Sun Microsystems in 1990. Originally Java was called *OAK*, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.

With most languages, developers translate their programs into the zeros and ones of machine code with a tool called a compiler. This means that the final results can be understood only by a specific operating system.

Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java source code files (files with a *.java* extension) are compiled into a format called *bytecode* (files with a *.class* extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as *Java Virtual Machines (VMs)*, exist for most operating systems, including UNIX, the Macintosh OS, and Windows. Bytecode can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

This is a bonus for the programmers because they only need to write one version of a program to run on any machine. The users on the other hand don't have to worry about whether their new application will run on their computer. Maybe one day it will even run other devices, like a pager, smart card readers, or cellular phones. This promise of portability, what Sun likes to call "write once, run anywhere," is why Java has caused such excitement.

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Java can be used to develop complete applications, programs or small applications, called Java applets. These applets can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer. Each browser has what is called a Java Virtual Machine, where the actual code is run.

There is nothing more magical about applets than any other kind of application you run. But applets were one of the first technologies that could animate Web pages and make them interactive and thus sparked a lot of excitement. Applets are now used to deliver everything from animations and games to charts and interactive programs that let each user get different information from the same Web page.

(The following two figures are taken from Sun's website on Java)

## HOW JAVA™ TECHNOLOGY WORKS ...in a Web browser

Here's how it works on a desktop computer in a Web browser:



References to **Java software** are embedded on a Web page. This page can be stored on the network or local disk.

When the **Web browser** sees these references, it loads the Java software (called an "applet") transparently and securely.

The applet is then processed by the Java virtual machine (below), which is built into the browser.

The **Java virtual machine** first does stringent security checks, and then runs the applet, which appears and interoperates inside the browser

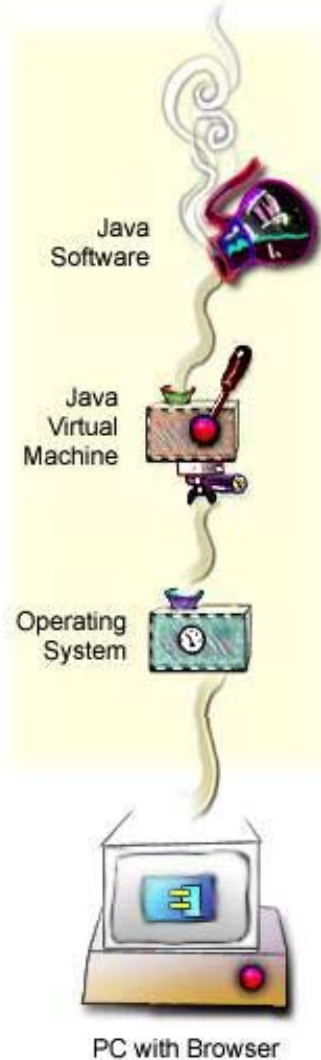
The computer's **operating system** provides machine-specific support for many of the actual operations and interactions.

**Result:** The user gets an interactive Java applet running in a browser. The Java virtual machine serves as consistent platform on different kinds of computers and operating systems, so that the same Java applet runs in browsers on PCs, Macs, UNIX® workstations, Network Computers, and elsewhere.

## HOW JAVA™ TECHNOLOGY WORKS

*...on a server.*

*On servers, the distributed nature of Java technology is ideal for implementing two-tier, three-tier, and n-tier architectures. Java technology integrates well with existing "legacy" computing systems, and without platform lock-in. Here's how Java technology works on a server:*



**Java software** is stored on the network or local disk.

The **Java virtual machine** on the server first does stringent security checks, and then runs the software.

The server's **operating system** provides machine-specific support for many of the actual operations and interactions.

**Result:** A "servlet" or other Java program running on the server and interacting with other systems on the network. The Java virtual machine serves as consistent platform on different kinds of computers and operating systems, so that the same Java server software runs on UNIX® servers, mainframes, mid-range computers, and PC and Mac servers. Deploying Java technology on the server helps you prevent 'lock-in' to a single architecture.

## 4.3 Java Security

Java has two different purposes: as an ordinary programming language and as a tool to build and handle untrusted applets on the web. Each of these purposes requires a completely different security policy, also about where it should be implemented.

When Java is used as a programming language to produce applications, there are no extra security requirements involved. This is assuming that the applications are not imported from outside an organisation's network. It should be expected that the underlying operating system would take all the necessary precautions to protect itself and regulate the application through its own security policy. Security would only be enhanced by maintaining physical control over the systems where the applications are run by using traditional measures like virus checkers and so on.

When it comes to an applet, it would depend on its source location to know what it is allowed to do. If it is originally stored on an Internet or Intranet webserver, the Java Virtual Machine establishes a specific space in the local machine for the applets to run, known as Sandbox, preventing them from accessing data outside of it. Since our work is about Web Security, much of our Java Security explanation will be devoted to the Sandbox model. If the applet resides on the client's local disk and in a directory that is on the client's CLASSPATH, then the file system loader and not the Sandbox loads it. The most important differences are: applets loaded via the file system are allowed to read and write files, load libraries on the client, execute processes, exit the virtual machine and skip the Sandbox (byte code verifier).

### 4.3.1 *The Sandbox*

Java's security model is focused on protecting end-users from hostile programs downloaded from untrusted sources across a network. To accomplish this goal, Java provides a customisable "sandbox" in which Java programs run. A Java program can do anything within the boundaries of its sandbox, but it can't take any action outside those boundaries. The sandbox for untrusted Java applets, for example, prohibits many activities, including:

- reading or writing to the local disk
- making a network connection to any host, except the host from which the applet came
- creating a new process
- loading a new dynamic library and directly calling a native method

By making it impossible for downloaded code to perform certain actions, Java's security model protects users from the threat of hostile code.

The Sandbox is actually composed of the following systems operating together:

- Bytecode Verifier
- Security Manager

### ***4.3.2 Applet Class Loader***

This is the first link in the security chain of an applet downloaded from the Internet or an Intranet. Its job is to fetch the executable code from the network and enforce the name space hierarchy. A name space controls what other portions of the Java Virtual Machine an applet can access. It is through a separate name space for each applet that the Class Loader can prevent an untrusted applet from accessing privileged parts of the system.

In the JVM, class loaders are responsible for importing binary data that defines the running program's classes and interfaces. In reality, there may be more than one class loader inside a JVM. The JVM has a flexible class loader architecture that allows a Java application to load classes in custom ways.

A Java application can use two types of class loaders: a "primordial" class loader and class loader objects. The primordial class loader (there is only one of them) is a part of the JVM implementation. For example, if a JVM is implemented as a C program on top of an existing operating system, then the primordial class loader will be part of that C program. The primordial class loader loads trusted classes, including the classes of the Java API, usually from the local disk.

At run time, a Java application can install class loader objects that load classes in custom ways, such as by downloading class files across a network. The JVM considers any class it loads through the primordial class loader to be trusted, regardless of whether or not the class is part of the Java API. It views with suspicion, however, those classes it loads through class loader objects. By default, it considers them to be untrusted. While the primordial class loader is an intrinsic part of the virtual machine implementation, class loader objects are not. Instead, class loader objects are written in Java, compiled into class files, loaded into the virtual machine, and instantiated just like any other object. They really are just another part of the executable code of a running Java application.

Because of class loader objects, you don't have to know at compile-time all the classes that may ultimately take part in a running Java application. They enable you to *dynamically extend* a Java application at run time. As it runs, your application can determine what extra classes it needs and load them through one or more class loader objects. Because you write the class loader in Java, you can load classes in any manner: You can download them across a network, get them out of some kind of database, or even calculate them on the fly.

### ***4.3.3 Class loaders and name-spaces***

primordial or object - loaded the class. When a loaded class first refers to another class, the virtual machine requests the referenced class from the same class loader that originally loaded the referencing class. For example, if the virtual machine loads class Volcano through a particular class loader, it will attempt to load any classes Volcano refers to through the same class loader. If Volcano refers to a class named Lava, perhaps by invoking a method in class Lava, the virtual machine will request Lava from the class loader that loaded Volcano. The Lava class returned by the class loader is dynamically linked with class Volcano.

Because the JVM takes this approach to loading classes, classes can by default only see other classes that were loaded by the same class loader. In this way, Java's architecture enables you to create multiple *name-spaces* inside a single Java application. A name-space is a set of unique names of the classes loaded by a particular class loader. For each class loader, the JVM maintains a name-space, which is populated by the names of all the classes that have been loaded through that class loader.

Once a JVM has loaded a class named Volcano into a particular name-space, for example, it is impossible to load a different class named Volcano into that same name-space. You can load multiple Volcano classes into a JVM, however, because you can create multiple name-spaces inside a Java application. You can do so simply by creating multiple class loaders. If you create three separate name-spaces (one for each of the three class loaders) in a running Java application, then, by loading one Volcano class into each name-space, your program could load three different Volcano classes into your application.

A Java application can instantiate multiple class loader objects either from the same class or from multiple classes. It can, therefore, create as many (and as many different kinds of) class loader objects as it needs. Classes loaded by different class loaders are in different name-spaces and cannot gain access to each other unless the application explicitly allows it. When you write a Java application, you can segregate classes loaded from different sources into different name spaces. In this way, you can use Java's class loader architecture to control any interaction between code loaded from different sources. You can prevent hostile code from gaining access to and subverting friendly code.

### ***4.3.4 Class loaders for applets***

One example of dynamic extension with class loaders is the Web browser, which uses class loader objects to download the class files for an applet across a network. A Web browser fire off a Java application that installs a class loader object -- usually called an *applet class loader* -- which knows how to request class files from an HTTP server. Applets are an example of dynamic extension, because when the Java application starts, it doesn't know which class files the browser will ask it to download across the network. The class files to download are determined at run time, as the browser encounters pages that contain Java applets.

The Java application started by the Web browser usually creates a different applet class loader object for each location on the network from which it retrieves class files. As a result, class files from different sources are loaded by different class loader objects. This places them into different name-spaces inside the host Java application. Because the class files for applets from different sources are placed in separate name-spaces, the code of a malicious applet is restricted from interfering directly with class files downloaded from any other source.

### ***4.3.5 Class loaders in the sandbox***

In Java's sandbox, the class loader architecture is the first line of defence against malicious code. It is the class loader, after all, that brings code into the JVM -- code that could be hostile.

The class loader architecture contributes to Java's sandbox in two ways:

1. It prevents malicious code from interfering with benevolent code.
2. It guards the borders of the trusted class libraries.

The class loader architecture guards the borders of the trusted class libraries by making sure untrusted classes can't pretend to be trusted. If a malicious class could successfully trick the JVM into believing it was a trusted class from the Java API, that malicious class potentially could break through the sandbox barrier. By preventing untrusted classes from impersonating trusted classes, the class loader architecture blocks one potential approach to compromising the security of the Java runtime.

### ***4.3.6 Bytecode Verifier***

The Bytecode Verifier is called before a newly imported applet is run. This helps prevent the possibility of attack via the use of a hostile compiler. The browser that downloads the class files does not know if a trustworthy Java compiler produced the bytecodes. The Verifier checks that the applet conforms to the Java language specification and that there are no violations of Java rules like stack overflows, name space violations, illegal data type casts, etc. All this also helps to enhance the performance of the interpreter, eliminating the runtime checks it would have to do otherwise. The interpreter meanwhile, assumes that the checking has been performed.

The bytecode verifier and the class loader must work as a team to guarantee that only safe and valid code is executed. The verifier is invoked by the class loader to perform a series of tests on class files that are regarded as potentially unsafe. Only when the tests have been passed is the file made available for use.

It is important to note that the Java specification requires the JVM to behave in a particular way when it encounters certain problems with class files, still the implementation varies from vendor to vendor.

### ***4.3.7 Security Manager***

The third component involved in loading and running Java program is the security manager. Even when untrusted code has been verified, it is still subject to runtime restrictions. It enforces the boundary of the sandbox through a security policy, which is why it is also known as the Security Policy Manager. The applets are restricted with respect to what they are allowed to do by the Security Manager object. The Security Manager does not ordinarily allow applets to execute arbitrary system commands, to load system libraries, or to open up system device drivers such as disk drives. Whenever an applet performs an action that is a potential violation, the security manager decides whether it is approved or not.

The security manager is similar to the class loader in that it is a Java class (`java.lang.SecurityManager`) that any application can extend for its own purpose. In a browser the security manager is provided by the browser manufacturer and is the component of the JVM which prevents applets from reading or writing to the file system, accessing the network in an unsafe way, making inquiries about the runtime environment, printing and so on.

Although the three elements of JVM security – class loader, bytecode verifier and security manager – each have unique functions they have to intercooperate tightly. The security manager relies on the class loader to keep untrusted classes and local classes in separate name spaces and to prevent the local trusted classes from being overwritten. The class loader relies on the security manager to prevent an applet from loading own class loader, which could flag untrusted code as trusted. And everything relies on the bytecode verifier to make sure that class confusion is avoided and that class protection directives are honoured.

### ***4.3.8 Digital Signatures***

Starting with JDK version 1.1, a Java application or applet can create its own digital signature in the PKCS7 format. When the applet is about to do something that the sandbox normally restricts, the browser user or network administrator can guarantee that the applet comes from a trustworthy source.

The signature on the applet or application links the code to the programmer or administrator who created or packaged it. The signer has to provide a public key certificate to allow the user to check the signature.

JDK 1.1 introduced the JAR (Java Archive) format that gives the chance of packaging

improve performance during applet downloads. JAR signing allows generating digital signatures for any of the files in the archive. Files can be signed by more than one signer, giving the opportunity for a web of trust to develop.

## 4.4 Java Flaws

Symantec, the makers of the famous Norton AntiVirus, publicised the existence of a program called “Strange Brew”, which was written in Java and supposedly modify class files on the local file system. The company claimed they have discovered the first Java-targeted virus. Nevertheless JavaSoft replied refusing the existence of such virus due to a flaw in the Java language security, and we agree.

To be “infected”, one would have to run a program in a mode that allows it to access the local file system. The “virus” would have to be installed on the file system, bypassing the Sandbox and giving the program a trusted status. This attack is merely physical and doesn’t have to do with the Java language security or its implementations. If someone would try to download such “virus”, the Java Sandbox and the inherent security features of secure browsers such as Netscape Navigator and Internet Explorer defeat such viral intrusion.

It is good to point out that with the release of JDK 1.2, Java expects to control even these physical attacks. The new security policy will give users and administrators the chance to set more detailed security levels than the two presented (trusted/untrusted) in prior JDKs.

### 4.4.1 History

July 22, 1998	<ol style="list-style-type: none"><li data-bbox="503 1375 1360 1988"><b>1. Princeton Classloader Attack</b> <a href="http://www.cs.princeton.edu/sip/History.html">http://www.cs.princeton.edu/sip/History.html</a>  A malicious applet can disable all security controls in Netscape Navigator 4.0.x. After disabling the security controls, the applet can do whatever it likes on the victim’s machine, including arbitrarily reading, modifying, or deleting files.  This situation doesn’t apply to applications based on the standard JDK but only to applets running on a Netscape browser.  The flaw is not directly exploitable unless the attacker can use some other secondary flaw to gain a foothold. Netscape 4.0x has such a secondary flaw (security manager bug found by Mark LaDue).  The law is fixed in Navigator 4.5</li></ol>
---------------	---

<p>June 23, 1997</p>	<p>2. <b>Kimera Verifier Implementation Bug</b>  <a href="http://kimera.cs.washington.edu/flaws/vacuum/">http://kimera.cs.washington.edu/flaws/vacuum/</a></p> <p>The JDK 1.1.2 verifier allows a type-unsafe applet to execute (type safety flaw). This allows the applet to search and locate strings that are stored in the browser's address space. This could lead to passwords and browser history information, revealing user's details.</p> <p>This flaw applies to JDK 1.1.2, 1.1.1 and the HotJava Browser.</p>
<p>May 16, 1997</p>	<p>3. <b>Kimera Verifier Implementation Bug</b>  <a href="http://java.sun.com/security/UW.html">http://java.sun.com/security/UW.html</a></p> <p>The JDK 1.1.1 bytecode verifier does not check if the number of arguments passed into a method is less than the amount of space allocated to local variables of that method. This could lead to the JVM crashing.</p> <p>There are some discrepancies between the number of flaws found by the Kimera Research Project (24) and the number of bugs fixed by JavaSoft (1). For more information, go to: <a href="http://java.sun.com/security/Uwdetails.html">http://java.sun.com/security/Uwdetails.html</a></p>
<p>April 29, 1997</p>	<p>4. <b>JDK 1.1.1 Signing Flaw</b>  <a href="http://www.cs.princeton.edu/sip/news/april29.html">http://www.cs.princeton.edu/sip/news/april29.html</a></p> <p>The bug applies to the way a JVM manages applet's signatures internally. An unprivileged, but signed, applet scan through the list of trusted signatures on the local machine and impersonate one of them.</p> <p>The flaw affects JDK 1.1.1 and HotJava version 1.0</p>
<p>March 17, 1997</p>	<p>5. <b>Privacy Attack</b>  <a href="http://www.alcrypto.co.uk/java">http://www.alcrypto.co.uk/java</a></p> <p>An applet calls getLocalHost() to find out the IP address of the computer that it's running on. Problem comes when the computer running the applet resides inside a firewall, where the IP address should remain hidden from the visiting applet. The loopback host call is a generic handle to the local computer and doesn't reveal any private information that should remain private.</p> <p>The flaw was actually fixed in May 1996 with the release of JDK 1.0.2</p>

<p>March 5, 1997</p>	<p>6. <b>Virtual Machine Bug</b>  <a href="http://java.sun.com/sfaq/chronology.html">http://java.sun.com/sfaq/chronology.html</a></p> <p>The JavaSoft bugging team found a security flaw on the verifier of the Java Virtual Machine. No real details of the flaw were given by Sun. They claim that an attack based on this theoretical flaw is complex and extremely difficult to exploit. The flaw was fixed with JDK 1.1.1</p>
<p>December, 1996</p>	<p>7. <b>Web Spoofing</b>  <a href="http://www.cs.princeton.edu/sip/pub/spoofing.html">http://www.cs.princeton.edu/sip/pub/spoofing.html</a></p> <p>Web Spoofing is not a Java language or implementation -related flaw. This type of attack takes advantage of Java networking capabilities, helping the attacker to hide the real identity of a malicious and “shadow copy” of a website. When the communication has been established, the attacker can monitor all victim’s activities and send misleading data to him/her as well as impersonating him/her.</p>
<p>August, 1996</p>	<p>8. <b>Array Name Bug</b>  <a href="http://www.cs.princeton.edu/sip/news/Aug-96-netscape.html">http://www.cs.princeton.edu/sip/news/Aug-96-netscape.html</a></p> <p>The flaw was caused by the incorrect handling of type definitions in the Java internals. Under special circumstances, an applet could define a class that had one special name, normally predefined for its array types. Although the verifier threw an exception, the malicious definition was mistakenly left in one of the system’s array tables. The applet could then redefine one of Java’s array types, breaking its type system. This bug affected Netscape Navigator 3.0 beta 5 and earlier.</p>
<p>August, 1996</p>	<p>9. <b>The Package Access Bug</b>  <a href="http://www.cs.princeton.edu/sip/news/Aug-96-microsoft.html">http://www.cs.princeton.edu/sip/news/Aug-96-microsoft.html</a></p> <p>The bug allows an untrusted applet to become a member of a security-critical Java package (module) whose membership was supposed to be limited to Java classes that are built-in to the browser (trusted applets). The Java Security Manager should be expected to detect and prevent this bug from happening. Nevertheless, Microsoft did not implement correctly the Java model. This bug affected Microsoft Internet Explorer version 3.0 beta 2.</p>
<p>June 2, 1996</p>	<p>10. <b>Illegal Type Cast Attack</b>  <a href="http://ferret.lmh.ox.ac.uk/%7Edavid/java/bugs">http://ferret.lmh.ox.ac.uk/%7Edavid/java/bugs</a></p>

	<p>David Hopwood of Oxford University discovered a bug that used with some techniques from the Princeton March attack can be used to execute arbitrary machine code.</p> <p>Under this flaw, an applet can manipulate the way objects are assigned and the way they collaborate, in order to undermine the Java type system.</p> <p>This problem was fixed in Navigator 3.0beta5 on all machines except PCs, 3.0beta6 on PCs.</p>
May 18, 1996	<p><b>11. Cargill's Classloader Attack</b>  <a href="http://java.sun.com/sfaq/chronology.html">http://java.sun.com/sfaq/chronology.html</a></p> <p>Based on bad declarations of private methods in the Java Sandbox, it was possible to get past system restrictions on creating a classloader. Once an applet had its own classloader, it could define and execute classes that would otherwise be barred from execution.</p> <p>This problem was fixed in JDK 1.0.2 and Netscape Navigator 3.0beta4.</p>
April, 1996	<p><b>12. Denial of Service</b>  <a href="http://java.sun.com/sfaq/denialOfService.html">http://java.sun.com/sfaq/denialOfService.html</a></p> <p>A number of hostile web pages, using Java applets, appeared on the Internet. Some applets exploit the uncontrolled CPU or memory resources assigned to them on the local machine.</p> <p>Other applets tried to fool people into thinking a dangerous attack has been launched on their computer.</p> <p>This attack got the attention of JavaSoft, after a student from Georgia Tech University, George Ladue, collected a list of malicious applets and published on the web.</p>
April 3, 1996	<p><b>13. URL Name Resolution Attack</b>  <a href="http://java.sun.com/sfaq/dns-spoof2.html">http://java.sun.com/sfaq/dns-spoof2.html</a></p> <p>For a specific firewall-protected network configuration, an applet downloaded from a client inside the firewall would be able to connect to a single specific host behind the firewall.</p> <p>The Domain-name-spoofing scenario occurs when both networks involved have the same domain name, but one network has not been registered with InterNIC and the other has. The attacker has to be on the InterNIC-registered network.</p> <p>Fixed in JDK 1.0.2 and Netscape Navigator 2.02</p>
March, 1996	<p><b>14. Verifier Implementation Bug</b>  <a href="http://www.cs.princeton.edu/sip/news/non-expert.html">http://www.cs.princeton.edu/sip/news/non-expert.html</a></p>

	<p>The flaw exposes a Netscape browser user to the risk of having his machine compromised (read, delete or corrupt local files). The applet can generate and execute raw machine code. Due to the serious implications involved, neither JavaSoft nor the discovering research team wanted to give much details about it.</p> <p>Fixed in JDK 1.0.2 and Netscape Navigator 2.02</p>
March, 1996	<p><b>15. Class Loader Implementation Bug</b>  <a href="http://ferret.lmh.ox.ac.uk/%7Edavid/java/bugs">http://ferret.lmh.ox.ac.uk/%7Edavid/java/bugs</a></p> <p>David Hopwood at Oxford University found a bug in the class loader that could be exploited to load illegal byte code, which could then be used to load a class referenced by an absolute path name.</p> <p>Fixed in JDK 1.0.1 and Netscape Navigator 2.01</p>
February, 1996	<p><b>16. DNS Attack</b>  <a href="http://www.cs.princeton.edu/sip/news/dns-spoof.html">http://www.cs.princeton.edu/sip/news/dns-spoof.html</a>  <a href="http://www.hks.net/cpunks/cpunks-23/1995.html">http://www.hks.net/cpunks/cpunks-23/1995.html</a></p> <p>An applet could open a connection to an arbitrary host on the Internet, defeating Java's rule of only allowing connection back to the host from which it was loaded. This could then be used as a door to exploit any other TCP/IP-based network service.</p> <p>Fixed in JDK 1.0.1 and Netscape Navigator 2.01</p>

To give a better understanding of the implications of the flaws found in Java, we decided to categorised them according to the basic security concepts affected and the implementation involved (language or browser interface). On the same path, we decided to generate a timeline with the same flaws, which we hope can show in a graphical way how a language related bug has always produce a bug on any of the browsers that support the language. At the same time, it shows how fast JavaSoft produces new releases of its JDK and includes the security fixes into its product.

### ***4.4.2 Java Class Loader Security Flaw***

Just in the nick of time for its Communicator 4.5 beta release, Netscape Communications has moved to fix a serious security hole that affects certain versions of its Web browser. The flaw, discovered and brought to Netscape's attention by the Secure Internet Programming group at Princeton University, lets a malicious Java applet disable the browser's security controls, leaving the user's computer defenceless against attacks over the Internet.

"The potential consequences are as severe as they could be," said SIP director Edward Felten. "Once you penetrate the security of the browser, then there isn't more protection. Someone can write an applet that can seize control of the victim's machine and delete or modify files, spread viruses, or whatever."

The flaw, which affects only versions 4.0x of Netscape's Navigator browser, lies in the implementation of what are called "class loaders" in the Java programming language. These units load and put together classes, or units of Java code, within the Java virtual machine (JVM), the software that lets applications written in Java run on multiple platforms.

The hole allows a maliciously designed class loader to confuse the JVM about the type of object it is processing. Felten and his group designed a class loader that let them gain privileges they should not have had, access memory they should not have been able to, and in turn disable the rest of the security mechanism.

While the flaw discovered in this case is specific to the Navigator 4.0x browsers, Felten and his group lay much of the blame with the Java security architecture. "Despite changes in the ClassLoader implementation in JDK 1.1 and again in JDK 1.2 beta, ClassLoaders are still not safe," reads a note on the bug report posted to the SIP Web site. "A malicious ClassLoader can still override the definition of built-in 'system' types like `java.lang.Class`. Under some circumstances, this can lead to a subversion of Java's type system and thus a security breach."

### ***4.4.3 How the attack works***

Java allows classes in the same package to grant each other special access privileges that aren't granted to classes outside the package. So, a malicious applet could have a class loader loads a class that brazenly declares itself to be part of the Java API (for example, a class named `java.lang.Virus`). If loaded, such a class could gain special access to the trusted classes of `java.lang` and could possibly use that special access for devious purposes. Normally, the security manager of the JVM should have detected such a security violation and prevents the class from being loaded. However, if there is a flaw in the implementation of the security manager, as in versions 4.0x of Netscape's Navigator browser, such violations go undetected and the malicious applet is able to bypass the Java security system and gain privilege access to your computer resources.

## Chapter 5 : Further threats, attacks and security related issues

### 5.1 Web Spoofing

A spoofing attack is an attack in which the attacker creates misleading context in order to trick the victim into making an inappropriate security-relevant decision. In the case of web spoofing, this is done by creating a convincing but false copy of the entire World Wide Web. The false Web will look exactly like the real one – including all the same pages and links – but it is controlled by the attacker, which means all network traffic between the victim’s browser and the Web goes through the attacker’s server. Thus it is another example of a ‘man in the middle attack’ and it is performed in the following steps :

#### *URL rewriting*

The first step (for the attacker) is to rewrite all of the URL’s on some Web page so they point to the attacker’s server rather than to some real server. Thus

`http://home.netscape.com`

could become

`http://www.attacker.org/http://home.netscape.com.`

When the victim requests a Web page, the attacker’s server will look for the page on the real server, from which it then retrieves the requested page. Then the attacker’s server rewrites the page by splicing `http://www.attacker.org` onto the front of all the URL’s. The rewritten version is then provided to the victim, so the victim remains trapped in the attacker’s false Web and can follow links forever without leaving it.

#### Diagram

#### *Forms*

If the victim now fills out a form on the false Web, the result appears to be handled properly by the genuine server. Since form submissions are encoded in Web requests and the replies are ordinary HTML, forms can be easily ‘spoofed’ as well. When the victim submits a form, the attacker’s server will receive it and can change and even modify the submitted data, before passing it onto the real server. Furthermore, the attacker’s server can also modify the data returned in response to the form submission.

#### *‘Secure Connection’*

One very distressing property of the attack is that even if the victim uses a secure Web access using Secure Sockets Layer, everything will appear normal. The victim’s browser *has* established a secure connection, but with the attacker’s server instead of

the intended server. The reason is that it was told to access a URL at <http://www.attacker.org>, so it made a secure connection to *that* server.

### ***Starting the actual attack***

The attacker must somehow lure the victim into his false Web, and can do this in one of the following ways :

- He puts a link to his false Web onto a popular Web page.
- If the victim is using web-enabled e-mail, the attacker can e-mail the victim a pointer to his false Web (or even the contents of a page in his false Web).
- He can even trick a search engine into indexing part of his false Web.

### ***Fine-tuning of the attack***

Since Web browsers are easily customisable, further evidence of an attack in progress can be easily eliminated. :

- In an actual attack, the victim could notice he is (about to be) connected to the attacker's server by looking at the **status line** (= the single line of text at the bottom of the browser window that displays various messages concerning the status of the pending Web transfers). For example, when the mouse pointer is held over a Web link, the status line displays the URL the link points to. Secondly, when a page is being fetched, the status line briefly displays the name of the server being contacted. Thus the victim might notice he is being 'spoofed'. The attacker can cover up both these clues by using JavaScript programs, which can write to the status line. By binding JavaScript actions to relevant events, the attacker can make sure that the status line shows to the victim what would have been there in the real Web.
- The browser's **location line** displays the URL of the page currently being shown, thereby showing if a URL has been rewritten. A JavaScript program can hide the real location line and replace it by a fake location that looks right and is in the expected place. This fake line can show the URL the victim expects to see and can even accept keyboard input, allowing the victim to type in URL's normally, which are then rewritten by the JavaScript program before being accessed.
- The victim could possibly spot rewritten URL's when looking at the HTML **source code** of the displayed page. A JavaScript program can be written by the attacker, replacing the menu bar with a fake one and open a new window showing the original (non-rewritten) HTML-source, if the user would chose to view the document source.
- The same trick applies if the user would choose to see the **document information**, which contains the document's URL. Again, the menu bar can be replaced.

- After an attack, there will almost certainly be evidence of the location of the attacker's server. However, most of the time the perpetrator will break into some innocent user's machine to launch his attack from there. So **tracing** the attacker is not an easy option either.

## 5.2 DNS Attack

This attack has been concisely and accurately described by the Princeton SIP group, so we quote the explanation directly in this paragraph :

### *Scenario*

The victim has two machines, `stooge.victim.org` (IP address 10.10.10.1) and `target.victim.org` (IP address 10.10.10.2). The attacker has a machine `www.attacker.org` (IP address 172.16.16.16).

The victim has a firewall that prevents machines outside the victim's organisation from making unauthorised network connections to any of the victim's machines. This prevents the attacker from launching a direct attack on the victim's machines. The victim's security depends on the firewall.

### *What the attacker does*

The attacker creates a bogus machine name "bogus.attacker.org" and creates a DNS mapping from `bogus.attacker.org` to the pair of IP addresses (10.10.10.2, 172.16.16.16).

The attacker also writes an innocent-looking Java applet and attaches it to a web page installed on `www.attacker.org`.

### *Triggering the attack*

The victim, running his web browser on `stooge.victim.org`, innocently visits a web page on `www.attacker.org`. This causes the attacker's applet to be loaded into the victim's browser, and to start running.

The applet performs some innocent function that is visible to the victim. It also silently attacks the victim's machines.

First, the applet asks to create a network connection to `bogus.attacker.org`. The Java system looks up the address "bogus.attacker.org," getting the IP address pair (10.10.10.2, 172.16.16.16). The Java system compares this address pair to the address of the machine that the applet came from (172.16.16.16). Since the two have the address 172.16.16.16 in common, Java allows the connection. However, the Java system actually connects to the first address on the list, namely 10.10.10.2 (`target.victim.org`).

The attacker's applet now has a network connection to `target.victim.org`. It can proceed to attack the defences of `target.victim.org`, using any one of several common network security weaknesses.

A more sophisticated version of the attack allows the attacker's applet to systematically attack all of the machines in the victim's organisation. The attacking applet can tell the attacker's DNS server which IP addresses to return, by encoding the IP addresses into the DNS name that is looked up. For example, the applet could look

up bogus-10-10-10-2--172-16-16-16.attacker.org if it wanted the DNS server to return the address pair given above.

### ***Why the attack works***

The key to the success of the attack is that the victim's firewall is helpless to prevent it. The firewall is supposed to protect the victim by preventing machines outside the firewall from opening arbitrary network connections to the victim's machines inside the firewall. In this attack, however, the dangerous network connections come from one of the victim's own machines, so the firewall is useless.

In effect, the attacker causes the victim's web browser to attack the victim's own machines.

### ***Using SATAN***

Since the attacking applet can make network connections back to attacker.org, the applet can operate under the direction of a "real attacker" that is running back in attacker.org. For instance, a variant of the notorious security-probing program "Satan" could be used to direct the attack.

### ***Third party attacks***

If the attacker can compromise a machine at third-party.org, it can still carry out the attack on victim.org. The attacker plants his applet on a web server on www.third-party.org. When the victim loads a web page from www.third-party.org, the attacking applet is loaded into the victim's machine. The applet can still use the DNS server at attacker.org to fool Java into allowing arbitrary connections. As above, the applet can connect to any desired machine on the internet, so it can attack the victim's machines, and it can operate under the direction of a program or person somewhere in attacker.org.

### ***A Web virus***

The third-party version of the attack can be used to create a virus. The virus would be attached to an innocent-looking Web applet. When the applet was run by some person, the applet would attack machines in that person's organisation. If it penetrated one of those machines, it would append the attacking code to any web pages it found on the penetrated machines. The virus could spread from web-server to web-server in this manner.

## 5.3 Cookies

### 5.3.1 Why are cookies a security issue ?

For many sites cookies are interesting instruments in order to personalise information, help with on-line sales/services, track popular links or demographics and keep site content fresh and relevant to the user's interests. Cookies can store database information, custom page settings and anything that makes a site individual and customisable. In a way, cookies are amongst the first attempts to shift work and storage load from the server to the client. Storing this information allows a user to pick up where he has left off during the previous 'browsing session'.

Since HTTP is a 'stateless' (non-persistent) protocol, it is impossible for a server to differentiate between visitors, unless it can somehow 'mark' the visitor. A cookie does just that by storing user-specific information in the memory of the browser. As such, cookies are more a threat to your privacy, than to your computer system. Yahoo for instance, uses cookies to keep track of your data on their servers.

Cookies can be created with JavaScript, Perl, LiveWire, ActiveServerPages, VBScript, etc.

### 5.3.2 How do cookies work ?

Cookies are simply sent from server to browser and vice versa by adding an additional line in the HTTP-header. Such a line could be :

*Content-type: text/html*

*Set-Cookie: biscuit=bar; path=/; domain=www.myserver.com; expires Mon, 01-Jan-2000 00:00:01 GMT*

*biscuit=bar*

the *name* of the cookie in this case is 'biscuit' and its *value* is 'bar';

*path=/*

the *path* specifies the URL-path the cookie is valid within; pages outside the path are not allowed to read the cookie; / means it is valid for the entire site;

*domain=www.myserver.com*

this parameter establishes to which server within your *domain* you want to assign cookies; for obvious security reasons, the server issuing the cookie must be situated in the domain that it tries to set in the cookie

*expires...*

this string determines the *lifetime* of the cookie; it defaults to end-of-session

*secure connection ?*

it is possible to set a flag in order to use a *secure connection* through SSL; since most sites do not require secure connections, this defaults to false;

### ***5.3.3 What attitudes towards cookies ?***

After the cookie has been transmitted through an HTTP-header, it is stored in the memory of the Web browser. If you stop browsing or shut down your computer, and the lifetime of the cookie has been set to exceed the amount of time the browser remains open, it is saved on the hard drive. It is not difficult to find the folders in which Netscape's Communicator and Microsoft's Explorer save the cookie files. Since they are simple text files, they can be easily deleted. (This has to be done when all copies of the browser are closed, otherwise the cookie file will be reloaded after deletion.) However, complete deletion means you 'start from scratch' and you may lose certain customised features on specific sites. The most sensible attitude is to open the cookie file and delete any unwanted entries. In addition, most browsers allow some level of cookie verification (to be set in the browser options) and 'alert before accepting cookies'.

Overall, cookies pose more a privacy threat than a real and direct security threat. Cookies cannot be used to transmit viruses, cannot get access to your hard drive, cannot compromise the integrity of your system. It can collect information about your browsing habits, but overall cookies can only gather information if you yourself provide it to a site and that site saves it to a cookie. As in many cases concerning web security, there can be a thin line between caution and paranoia.

## 5.4 Firewalls

Firewalls are systems designed to prevent unauthorised access to or from a private network and can be implemented both in hardware and software, or a combination of both. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria. Firewalls are often used to prevent unauthorised Internet users from accessing private networks connected to the Internet, especially intranets. They are frequently used as the first line of defence for the protection of private information.

Firewalls have many benefits :

- They define a centralised ‘choke point’ that keeps unauthorised users out of the protected network. The security of the private network does not depend on the ‘hardness’ of each host’s security features and it would be only as secure as the weakest point.
- Firewalls offer a convenient point where security can be monitored, alarms can be generated and logs can be checked.
- They can also offer a central point of contact for information delivery service to customers. It is a suitable location to deploy WWW and ftp servers.
- Even if the connection to the Internet fails due to a firewall failure, the internal private network can still remain operational.

On the other hand :

- They cannot protect against attacks that do not go through the firewall, for instance, if users - irritated with the cumbersome authentication process required by certain firewall proxy servers – establish a direct connection to an ISP. This creates the possibility of back-door attacks.
- Along the same line of thought, they cannot offer protection against threats posed by unwitting users, corporate spies, etc. For instance, employees might be tricked into revealing passwords.
- Firewalls cannot protect against virus-infected software.
- Similarly, they cannot protect against data-driven attacks, in which seemingly harmless information is mailed or copied to an internal host and is executed to launch an attack.

There are several types of firewall techniques :

- *Packet filter* : it looks at each packet entering or leaving the network and accepts or rejects it based on user-defined rules; it is transparent to users and fairly effective; on the downside, it is difficult to configure and susceptible to IP spoofing;
- *Application-level gateway* (often referred to as *bastion host*): direct flow of packets between inside and outside system is not allowed; it applies security mechanisms to specific applications, such as ftp and telnet servers; it is very effective but can cause a performance degradation

- *Circuit-level gateway* : it applies security mechanisms in the case of TCP or UDP connections; once the connection has been made, packets can flow freely between the hosts without further checking;
- *Proxy server* : it intercepts all the messages entering and leaving the network; it effectively hides the true network addresses;

Very often these are just building blocks, which can work together to build an effective Internet firewall system. Each type has its shortcomings, but together they can compensate for their respective flaws. In many of the more successful configurations, a DMZ or demilitarised zone is created in-between two firewalls. The intermediate network serves as a buffer between the private network and the Internet.

Different vendors and different technologies deliver different levels of security. When buying and installing a firewall system, one should keep in mind that it should fulfil a number of properties :

- Good performance : speed of operation should not be reduced too much.
- Ease of configuration and operation : it should be easy to use.
- Security : in particular, one should have a good robustness against denial of service attacks. Many surveys and tests have shown the vast majority of firewalls cannot withstand persistent DoS attacks.
- Additional options : such as the possibility of remote and centralised management, low cost, extra optional management tools, etc.

## Appendix : Further References and evaluation of the sources

### A.1 Netscape's Communicator

The first URL refers to Netscape's security home page. Naturally, Netscape tend to play down the seriousness of the flaws, but the information is generally reliable. The references for each bug have been mentioned in the paper itself.

The information for the Random Number Generator Bug was mostly compiled from various mailing lists. As Netscape issued a fix for this bug, we can assume that the information is accurate.

The geek-girl bugtraq site is a full-disclosure UNIX security mailing list started by Scott Chasin. It lists most documented computer security flaws chronologically and by thread. Another useful resource was ZDnet Magazine.

1. <http://www.netscape.com/products/security/resources/notes.html>
2. [http://www.geek-girl.com/bugtraq/1998\\_4/0145.html](http://www.geek-girl.com/bugtraq/1998_4/0145.html) (mailing list archive)
3. <http://www8.zdnet.com/pcmag/news/trends>

### A.2 Microsoft's Explorer

1. <http://www.netcratt.co.uk/security/diary.html>
2. <http://www.microsoft.com/windows/ie/security/cybersnot.htm>
3. <http://www.zknet.com/zdnn/content/pcwk/1410/pcwk0087.html>
4. <http://www.microsoft.com/windows/ie/security/buffer.htm>
5. <http://www.ee.washington.edu/computing/iebug>
6. <http://www.symantec.com/avcenter/security/browser/browserie30.html>
7. <http://www.microsoft.com/windows/ie/security/max128.htm>

### A.3 Microsoft Internet Information Server

Good websites for IIS vulnerabilities include :

1. <http://www.rootshell.com>
2. <http://oliver.efri.hr/~crv/security/bugs/list.html>

### A.4 Apache Server

Good information on Apache servers is hard to find. The best reference on security is the Apache Security Advisory and is published by the Apache Group. Other links provide more general information about this freeware project. Most links tend to refer to the official Apache Web site for reliable information. The flaws described in this

report are copied from the first URL mentioned below, and this without any modifications, since only a small audience of insiders can grasp the technical details.

1. [http://www.apache.org.uk/info/security\\_bulletin\\_1.2.5.html](http://www.apache.org.uk/info/security_bulletin_1.2.5.html)
2. <http://webreview.com/wr/pub/freeware/apache.html>
3. <http://www.apache.org/>

### **A.5 CGI scripts**

By far the best reference on CGI scripts security is the WWW Security FAQ. The WWW Security FAQ is a document from the World Wide Web Consortium (W3C). W3C is hosted by the Laboratory for Computer Science at MIT, Inria and Keio University, with support from DARPA and the European Commission. The other links provide an introduction into CGI scripts.

1. <http://www.w3.org/Security/Faq/>
2. <http://hoohoo.ncsa.uiuc.edu/cgi/intro.html>
3. <http://www.usi.utah.edu/bin/cgi-programming/counter.pl/cgi-programming/intro.html>

### **A.6 Java**

The majority of the references come from three sources – namely the Java security website of the computer security department of Princeton University, JavaSoft at Sun Microsystems and project Kimera of Washington University.

The other references do not provide clear explanations of the nature of the security flaws, and hence do not contribute substantially towards the Java security report.

By far, the best documentation is found at the computer security department of Princeton University web site. The site provides a detailed history of the Java security flaws in chronological sequence. Each flaw is explained in terms of the platform affected, the cause of the flaw, how the attack works and remedy rendered to the affected products. For a good understanding of Java security, this is one site that cannot be missed.

The web site at JavaSoft contains excellent documentation of Java security architecture and the list of uncovered flaws. One can easily cross reference between this and the materials presented at the Princeton site, and this is advisable as you may not get a reasonably objective analysis of the flaws at the JavaSoft site.

1. <http://java.sun.com>
2. <http://cs.princeton.edu>
3. <http://java.sun.com/security/23jun97.html>
4. <http://www.alcrypto.co.uk/java>
5. <http://ferret.lmh.ox.ac.uk/%7Edavid/java/bugs>

## **A.7 Other threats, attacks and security related issues**

### **A.7.1 Web Spoofing**

By far the best paper on web spoofing is the Technical Report 540-96 of the Secure Internet Programming Group of the Department of Computer Science of Princeton University. Don't bother looking elsewhere.

<http://www.cs.princeton.edu/sip/pub/spoofing.html>

### **A.7.2 DNS attack**

The same comment applies as for Web spoofing : once again, Drew Dean, Ed Felten, and Dan Wallach of Princeton University deliver high quality material. The article can be found at the same website as above.

<http://www.cs.princeton.edu/sip/news/dns-spoof.html>

### **A.7.3 Cookies**

There is an unlimited supply of information available on the Web about cookies. Some of the more serious ones include :

1. [http://www.cookiecentral.com/unofficial\\_cookie\\_faq.htm](http://www.cookiecentral.com/unofficial_cookie_faq.htm)
2. <http://www.w3.org/Security/Faq>

### **A.7.4 Firewalls**

A lot of interesting links are available, such as :

1. <http://www.3com.com/ncs/500619.html>
2. [http://www.data.com/lab\\_tests/firewalls97.html](http://www.data.com/lab_tests/firewalls97.html)
3. <http://www.interhack.net/pubs/fwfaq>