Global Information Assurance Certification Paper

Netscape Enterprise Server Denial of Service Exploit

GCIH Practical Assignment

Version 2.1 (revised February 15, 2002)
Option 1 – Exploit in Action

SANS Bootcamp Monterey, February 9 – 14, 2002

By Tony G. Smith

May 16, 2002

Table of Contents **Page**

The Exploit

The attack that is presented below was performed using a **fictitious** scenario but on a live test network. Actual IP addresses are masked to protect the innocent and all company and individual names are completely hypothetical.

This paper will show how a specific DoS / buffer overflow exploit is used and how to stop this type of vulnerability on your own network. It will also display a few methods and tools that can be used when troubleshooting an attack.

Specific Exploit Information

Name: iPlanet Web Server Enterprise Edition and Netscape Enterprise Server malformed Web Publisher command causes denial-of-service **-** CERT/CC Vulnerability Note VU#191763

Operating System: Windows NT/2000 platforms

Protocols / Services / Applications: HTTP (tcp port 80) is the protocol used for this exploit and the Netscape Enterprise Server v4.0 SP2,SP6 to 4.1 SP8 running on Windows NT/2000 are the platforms on which the service is vulnerable.

Brief Description: When a malformed ***?wp-html-rend*** command is issued to the web server from a Web Publisher client, it is possible to crash the web server process[1]. This vulnerability requires that Web Publishing is enabled on the Netscape / iPlanet Web Server. This process will be discussed in further detail below. This particular vulnerability does not actually allow the attacker to take over a system or delete files. However the attacker can be very annoying and potentially cost your company thousands of dollars by rendering your web server useless. **Note:** Netscape Enterprise Server was the predecessor to what is now called iPlanet Web Server which is why they are both referenced in the vulnerability notes.

Variants: A variant of this vulnerability is CERT/CC Vulnerability Note VU#985347 – iPlanet Web Server Enterprise Edition and Netscape Enterprise Server Web Publisher command exposes a server to a brute force attack. This attack is also dependent on Web Publishing being enabled on the web server. This vulnerability allows an attacker to make repeated authentication attempts if a server is configured to use HTTP basic authentication.[2]

---

[1] http://www.kb.cert.org/vuls/id/191763, by Art Manion

Another variant can be found at the Internet Security Systems X-Force web page under the listing for netscape-webpublisher-acl-permissons (6058). This attack also relies on Web Publishing being enabled. An attacker can attempt to view or download private files on the web server with this vulnerability.[3] Although this vulnerability runs on Netscape Enterprise Server v3.51, it emphasizes the fact that Web Publishing should not be enabled on a public web server.

Yet another variant is CERT/CC Vulnerability Note VU#32794 - iPlanet Web Server and Netscape Enterprise Server Web Publisher commands allow directory enumeration. This vulnerability uses Web Publishing commands such as *?wp-ver-info* and *?wp-cs-dump* to display directory listings. The vulnerability is due to the incorrect configuration of the "Directory Indexing" component on a Netscape Enterprise Server or an iPlanet Web Server[4]. Directory Indexing determines what type of directory index to generate if the server cannot find one of the index file names specified in the Index Filenames field[5].

References:

http://www.sun.com/service/support/software/iplanet/alerts/iwsalert-5-11-01.html
http://www.securiteam.com/securitynews/5VP0C0A60A.html
http://www.kb.cert.org/vuls/id/191763
http://www.kb.cert.org/vuls/id/32794
http://www.kb.cert.org/vuls/id/985347
http://www.theregister.co.uk/content/55/23609.html
http://www.procheckup.com/security_info/vuln_pr0104.html
http://www.iss.net/security_center/static/7842.php
http://www.iss.net/security_center/static/6058.php
http://online.securityfocus.com/bid/3826

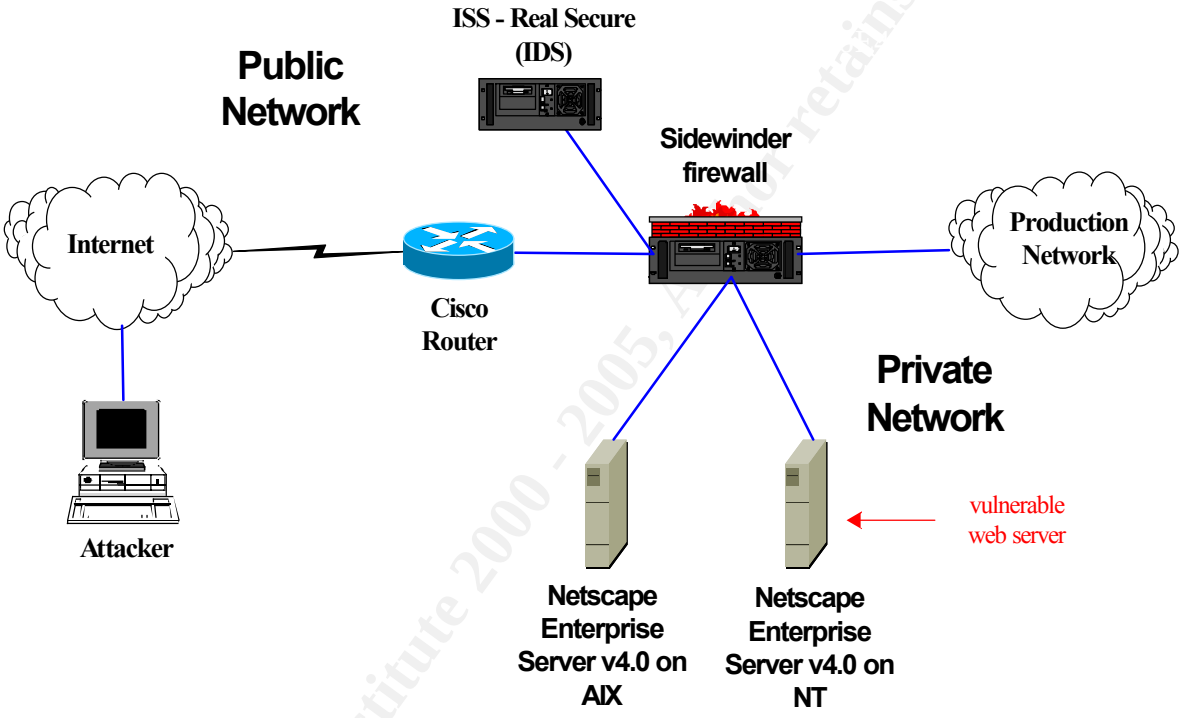
The Attack

Description and Diagram of the Network

The network that was used in this scenario is connected to the Internet via a Cisco router running IOS v12.1(7). This router was configured to only allow tcp ports below 1024 into this network. The Sidewinder v5.2.0.01 firewall was running on a hardened BSD UNIX kernel that allowed all users through tcp port

[2] http://www.kb.cert.org/vuls/id/985347, by Art Manion
[3] http://www.iss.net/security_center/static/6058.php
[4] http://www.kb.cert.org/vuls/id/32794, by Jeff S. Havrilla and Art Manion
[5] Appendix E – Netscape Enterprise Server User Interface On-line Help File

80 to the vulnerable web server. The firewall did not allow any other ports through to either of the web servers from the Internet. The web server was Netscape Enterprise Server v4.0 SP2 running on Windows NT v4.0 SP6a.
It is listening on tcp ports 80 (http – web services), 135 (location service for NT), 139 (NETBIOS session service for NT) and 8888 (web server administration service). Port 8888 was only accessible from the internal LAN segment as well as from selected internal users which was defined in the firewall's access



control list (ACL). Ports 135 and 139 were only accessible from the local LAN segment. The available tcp ports were determined by running ScanPort, which is a tool that scans tcp ports for a given IP range.

As you can see in the network drawing above, I also included a second Netscape web server that was running on an AIX platform. This web server was running a version of the Netscape Enterprise Server on an operating system that is not susceptible to this vulnerability. The reason for including this web server was to simply verify that this particular vulnerability did not exist on the AIX platform running Netscape Enterprise Server v4.0 SP2.

Protocol Description

The protocol that has been captured in this attack is the HyperText Transfer Protocol v1.1. After checking with RFC 2068, I determined that this protocol has been in use by the World Wide Web global information initiative since 1990. It is "a generic, stateless, object-oriented protocol which can be used for tasks such as name servers and distributed object management systems, through extensions of its request methods".[6]

The HTTP protocol uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. This scheme is indicated by an HTTP-Version field in the first line of a message → *HTTP-Version = 'HTTP" "/" 1*DIGIT "." 1*DIGIT.* This versioning policy allows the sender to set up the "rules" or format of the HTTP communications to follow. "The <minor> number is incremented when the changes made to the protocol add features which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender." The <major> number is incremented when the format of a message within the protocol is changed. One note about the version numbers is that both the major and minor numbers must be treated as separate integers and it is possible that each number may be higher than a single number. For example, HTTP/1.6 is a lower version than HTTP/1.12, which is lower than HTTP/10.3. Also, leading zeros must be ignored by recipients and must not be sent. The version of HTTP that an application is using is the highest HTTP version for which the application is at least conditionally compliant.

When accessing a web page, most users type in some sort of a name similar to www.sans.org. This name is technically known as a Uniform Resource Identifier or URI. Other names that have been used to describe URIs are: WWW addresses, Universal Resource Identifiers, and the combination of Uniform Resource Locators (URL) and Names (URN). In the world of HTTP, Uniform Resource Identifiers are formatted strings that identify a resource or in most cases a web page. In HTTP, URIs can be "represented in absolute form or relative to some known base URI" which depends on what context they are used in. The absolute URI form always begins with a scheme name followed by a colon (i.e. http:). One note that should be made about URI length is that servers should be cautious about lengths above 255 bytes as some older clients may not support these lengths.

[6] http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2068.html

The “http” scheme is used to find network resources via the HTTP protocol. The following is the definition for the scheme-specific syntax and semantics for http URLs:

```
http_URL  = "http:" "//" host [ ":" port ] [ abs_path ]

host      = <A legal Internet host domain name
             or IP address (in dotted-decimal form),
             as defined by Section 2.1 of RFC 1123>

port      = *DIGIT
```

If no port is defined, port 80 is used. The destination host is located at the server listening for TCP connections on the port defined and the Request-URI for the resource is abs_path. IP addresses in URL’s should not be used whenever possible. If the abs_path is not specified in the URL, it must be given as “/” when used as a Request-URI for a resource[7].

HTTP is the de facto standard for transferring web pages from a server to a client. HTTP is most commonly used by connecting via tcp port 80 from a client to a server. After a successful tcp connection has been established, a series of request and reply messages are exchanged. The most basic of these messages is a “GET *url*” to which the server sends back the contents of the *url* document. Another excellent source for information on the HTTP protocol can be found at the World Wide Web Consortium’s web site.

The HTTP protocol is one of the most widely used protocols for attacking because of the sheer number of web servers running on the Internet. Furthermore most security administrators allow it through their perimeters and into their web servers. In some instances, this is all that an attacker needs to reek havoc on an innocent web administrator. The attack described in this paper does not exploit a vulnerability in the HTTP protocol itself, rather the attack uses the HTTP protocol to transport a command that exploits a vulnerability that exists on the web server itself.

Denial of Service Exploit Examined

Denial-of-Service (DoS) / buffer overflow attacks date back to the earliest form of

[7] http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2068.html

the modern day computer and yet they continue to be one of the most popular exploits used by attackers. DoS vulnerabilities exist in many forms in our world today. Whether the DoS attack is intentional vs. accidental or malicious vs. annoying, a DoS attack essentially disables a network resource or a computer from its normal operating status. DoS attacks can be split into three basic categories[8]:

- Complete use of a limited resource
- Reconfiguration of a network component
- Physically damaging or crippling of a network component

The first category of a DoS entails the flooding of a service or network connection with traffic until the desired service becomes either unresponsive or unavailable. This can be accomplished by sending a large volume of traffic to a network device or by simply sending a relatively few crafted packets to a particular host. The host cannot process this traffic properly and therefore shuts down a particular process on the server. An attacker might also attempt to fill up disk space by sending a mail server a large number of email messages which could in turn disable your email server[9]. The attack that this paper is focused on falls under this category.

The second DoS category involves changing working configurations in a way that disables their function or causes the device to work improperly. For example, a router administrator can innocently make a change to a router access list which causes all ftp traffic to be denied to your public ftp server. Even though the "attack" was unintentional, this still causes a denial of service to your ftp server.

Finally, physical security is a necessity when trying to secure any network device. All access to sensitive and critical equipment should be limited to only those who absolutely require access. The proper use of physical security can prevent a majority of unauthorized physical access or damage to network devices.

The beauty of many DoS attacks is that they are for the most part simple and don't require an attacker to be an expert in C++ or the like to crash or disable a system. Many web sites exist for the sole purpose of informing attacker "wanna be's" of how to attack vulnerable systems on the web. They even provide a vast array of "hacker" tools that most nine year olds can use. As disheartening as this is, we do live in a free country and can't stop these folks from publishing this material. We can only attempt to keep up on current vulnerabilities and try to

[8] http://www.cert.org/tech_tips/denial_of_service.html
[9] http://www.cert.org/tech_tips/email_bombing_spamming.html

stay one step ahead of the “bad guys”.

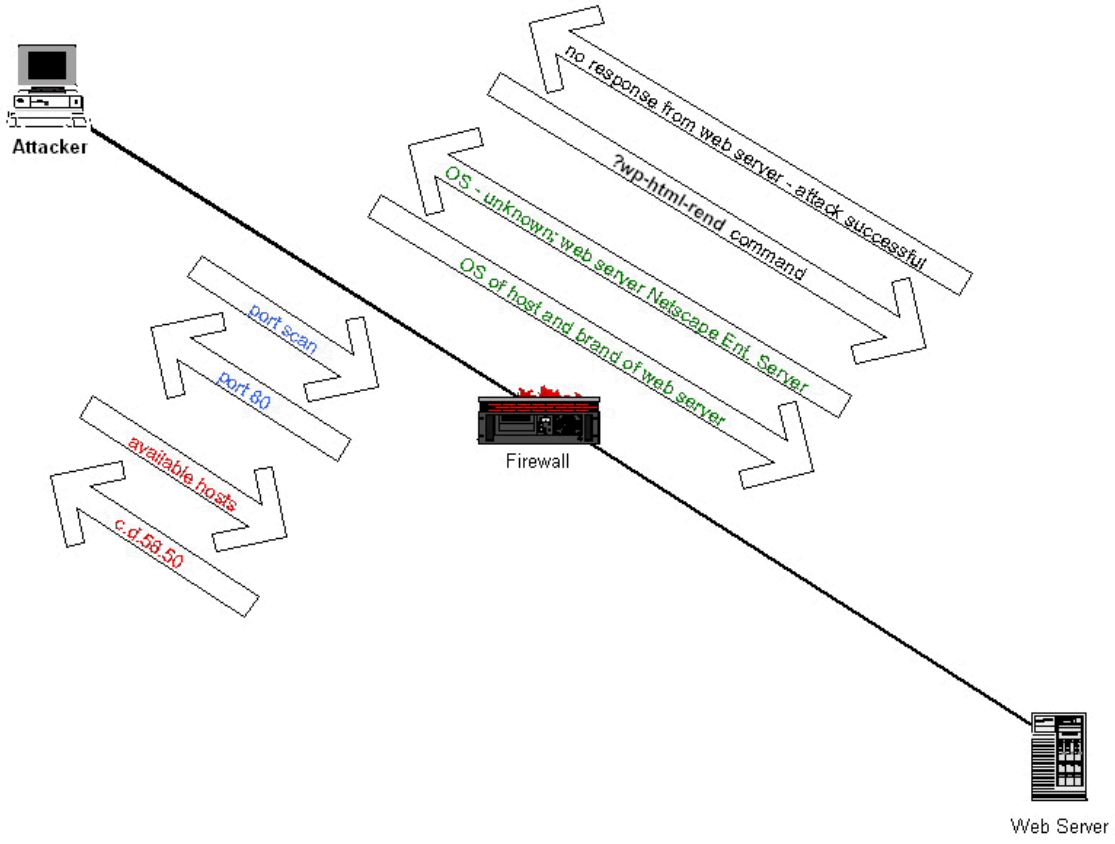
As the Internet grows in popularity and speed, the number of vulnerable hosts is increasing as are the sheer number of attacks. These facts point to a growing number of attacks that can be seen anywhere from a home PC connected to the Internet via a DSL router all the way to the most protected hosts in the Department of Defense. With the advent of automated tools that scan entire Class B addresses with little effort reinforces the fact that no network is safe regardless of how “hidden” you may think you are.

Most security professionals make a valid attempt to balance security with business practicality. As most people say, “The only way to completely secure your network is to pull the plug.” Since this is not a feasible alternative for a security architecture, most networks remain plugged into the Internet and administrators do the best that they can to protect their devices from the big, bad attackers. Security must be a balance between secure business practices along with the flexibility and power of the Internet.

Description and Diagram of the Attack

All attacks require a target of some sort and someone to do the attacking. Some targets are known ahead of time while others are discovered during the reconnaissance phase of an attack. An attacker may be a disgruntled employee, a bored teenager or someone who is out to make an extra buck or two. At any rate, once a target or vulnerability has been determined the attacker can begin to focus in on their final goal of wreaking havoc on their prey.

The attack described below is a denial of service attack on a Netscape Enterprise Server v4.0 SP2 running on Windows NT v4.0 SP6a. With this attack a web server is hit with a malformed **GET /?wp-html-rend HTTP/1.1** command which renders the server useless until an error message has been cleared on the console or until the server can be patched.



Simplified Flow of the Attack (see drawing above)

1. The attacker scans a specific range of IP addresses for live hosts.
2. The attacker then determines what ports are listening on the live hosts.
3. Once port 80 has been discovered to be available, the attacker attempts to determine the OS of the server and the brand of web server running on the target host.
4. When Netscape Enterprise Server has been identified as the web server, the attacker can select from a much smaller list of vulnerabilities even though the OS running on this host is not known.
5. Once the malformed ***?wp-html-rend*** command vulnerability has been selected, the attacker simply submits this command to the web server. If the vulnerability exists on the target web server, it is taken out of service which can be tested by simply trying to bring up the target web page. If the page

does not come up, the attacker's DoS was successful.

Expanded Description of the Attack

The first phase of this attack involves some reconnaissance work to determine who specifically to attack. Once the target has been decided upon, tools such as whois **(whois –h whois.arin.net x.x.x.x)**, nslookup and traceroute can be used to determine IP address ranges and specific hosts to attack. These tools can be used to determine hosts such as routers and dns servers.

Once it has been determined what network to attack, tools such as Nmap or Cheops can be used to determine what hosts are available for attacking. For example the **nmap –sP –PI 10.10.10.1-254** will perform a ping sweep on all addresses in the 10.10.10.0 / 24 subnet. Nmap will return a list of hosts that respond to the specific request that was sent to it. In this case, nmap will return a list of hosts that reply to a ping request.

After a list of available hosts has been gathered, a tool such as ScanPort can be used to determine what ports a specific host is listening on. Scanport will not only report the ports that a host is listening for but also will try to define what service is normally associated with that port (i.e. tcp port 80 – http).

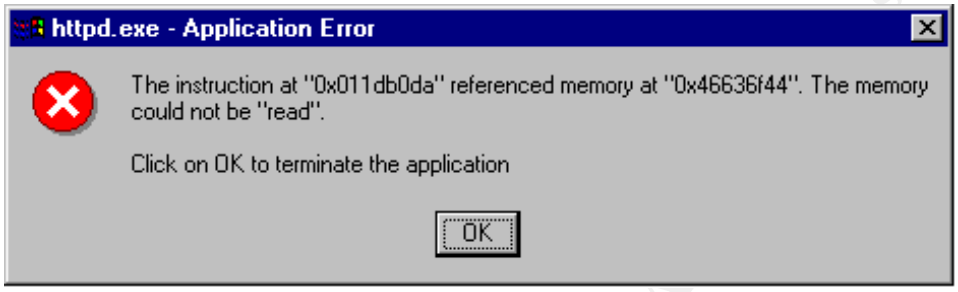
If tcp port 80 has been discovered as a port that is open, a web site such as Netcraft could be used to quickly determine what web server is running on a host. Netcraft attempts to determine the operating system of the targeted host by analyzing the HTTP reply returned by the web site[10]. The web server or operating system information may not always be returned correctly with this type of query, because the web site might be protected by a firewall. Another possibility is that the site may be using a web proxy device or load balancing products. But regardless of the results that are returned by the query, it is a quick and easy way for an attacker to get some basic information about a particular web site. Any information about a web site that an attacker can obtain significantly narrows down the exploits they have to try. Here is an example of a query to Netcraft's web site.

OS, Web Server and Hosting History for c.d .58.50

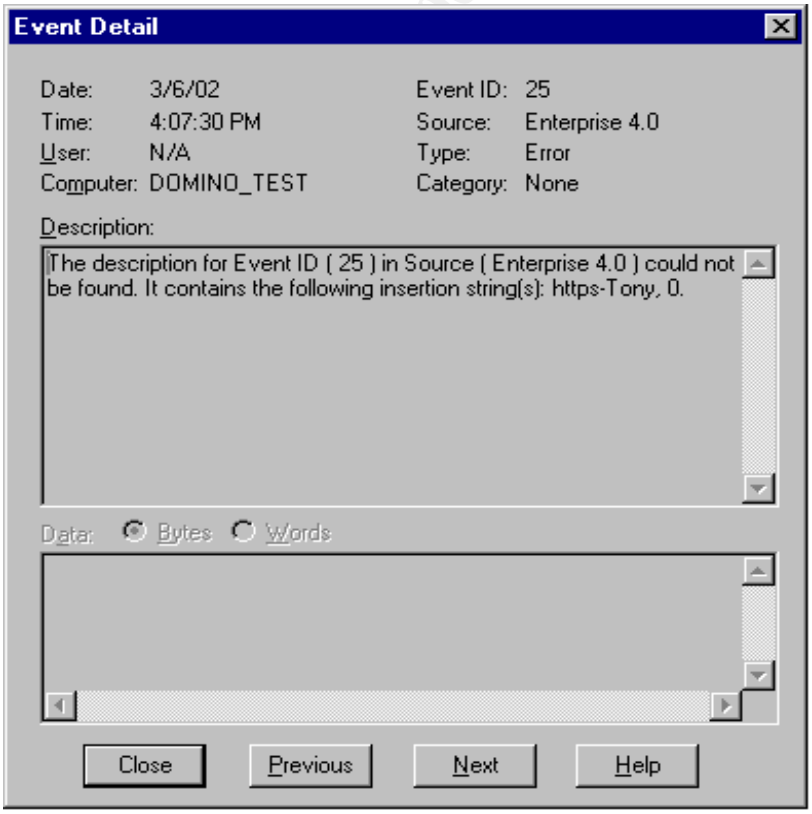
OS	Server	Last changed	IP address	Netblock Owner
unknown	Netscape-Enterprise/4.0	15-Mar-2002	c.d.58.50	Company X

Even though the query does not report an operating system, it does report that the web server was running Netscape-Enterprise/4.0. Once the attacker knows the web server platform but is not quite sure of the operating system, a simple http request containing **GET /?wp-html-rend HTTP/1.1** for example, can provide an immediate confirmation of a specific vulnerability.

Once the malformed command has been issued to a vulnerable web server, the following Application Error message box pops up on the screen.



The message can be cleared by simply clicking on the OK button. This also allows the web server to continue its normal functionality. The only other indication on the web server itself that something has happened is a fairly ambiguous message that appears in the Event Viewer – Application Log.



Even though this specific error message is not terribly informative, it allows the incident handling team to correlate an interesting system event with an event that is logged on the vulnerable web server.

In this scenario, an attacker could perform the above tasks using tools readily available on the web and in a short amount of time. Not all attacks require the attacker to be an all knowing guru in TCP/IP or a specific OS. Some attacks simply require a small amount of time, a little knowledge of the web and a reason to aggravate an unsuspecting host.

Signature of the Attack

One of the areas that did not work as expected was our intrusion detection solution. We were under the assumption that if we were under attack our IDS system would surely identify the attacker and the method of the attack.

Internet Security Systems' (ISS) RealSecure Network Sensor product was selected as our network intrusion detection solution. The reason that RealSecure did not detect this specific vulnerability was that it does not have a signature that can be downloaded from ISS's web site nor can you create a user defined signature for this vulnerability.

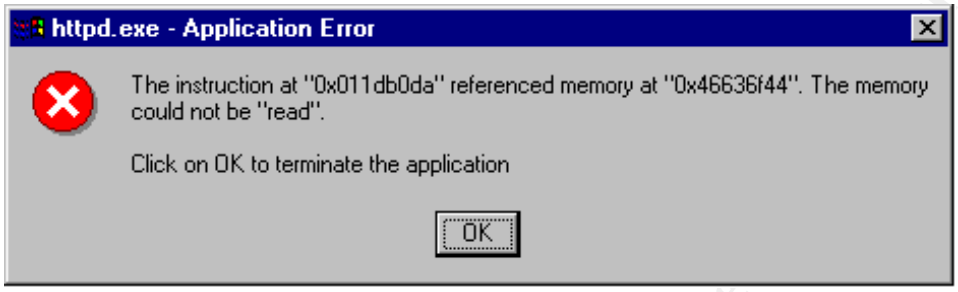
I attempted to create what ISS calls a "User Defined Event" from the RealSecure Management Console, but was unable to do so because of the "?" in the *?wp-html-rend* command. The reason was that in URL data the "?" is a special character that signifies a query. ISS does not have a Context search for Queries and therefore was unable to differentiate any text after the "?"[11]. An escape character such as "\" was also tried when creating a rule for this event but ISS was still unable to detect anything following the "?". I was able to create a User Defined Event that triggered on any "?" that was found within the URL data of a packet. It was decided not to use this rule, because a large number of false positives would be generated due to the frequent use of the "?" in web development. Upon determining that ISS was unable to detect and notify us that this type of an event was occurring, it was decided to look elsewhere in our network for notification of this type of event.

It is possible that other intrusion detection products such as Snort could pick up on this event. However, due to the limited amount of time that I had to work on this project, I was unable to get another product up and running to determine its

[11] This information was provided to me by ISS's technical support staff. They also informed me that this is not an issue that will be addressed in the future and therefore no patches will be available to correct this.

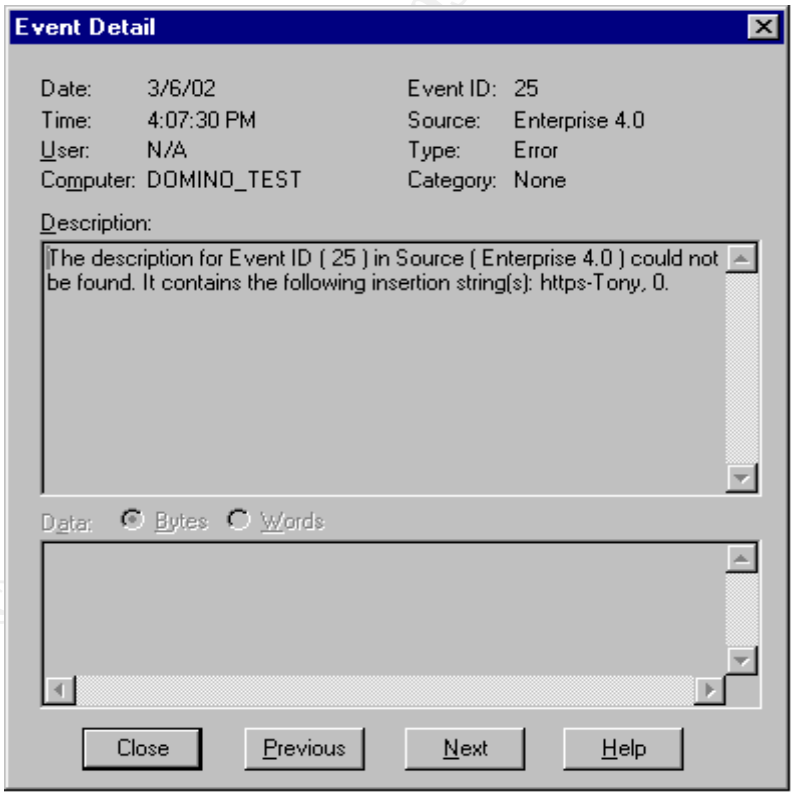
effectiveness on this signature.

By looking at the web server console screen after a successful DoS attack of this nature, the following error message will be displayed.



This message must be cleared by clicking on the OK button in order for the web server to resume normal functionality.

Another tell tale sign that this event has occurred is to look in the NT Event log for the following error message:



How to Protect Against the Attack

A result of this incident was that we emphasized to all administrators that they need to stay current on patch releases. Another suggestion was to have vendors send us updated information via e-mail as soon as new patches become available. In the future it will be imperative that all servers be built by a set of preapproved security guidelines.

One method to protect our network from this attack would be to patch the vulnerable servers. The fix for the vulnerability on a Netscape Enterprise Server version 4.1 is to down load SP9 from the iPlanet web site and apply it to the vulnerable server. Another option is to disable the *?wp-html-rend* command, the details of which were found by referring to the iPlanet Knowledge Base Article ID: 7761. This article indicated that the *?wp-html-rend* command can be disabled by loading *disrend.dll* on a system and by adding the following lines to the *obj.conf* file:

```
Init fn="load-modules" funcs="disRend" shlib="/disrend.dll"
PathCheck fn="disRend"
```

Other possible server based solutions are also described in iPlanet Knowledge Base Article ID: 4302. One of the solutions explained in this article involves disabling "Directory Indexing"[12]. The CERT Advisory – Vulnerability Note VU#191763 indicated that you not only disable directory indexing but that you disable Web Publisher[13] as well. The ProCheckUp Security Bulletin PR01-04, the ISS X-Force Security Database Entry 7842 and SecurityFocus Bugtraq ID 3826 all have basically the same solutions as noted above for this vulnerability.

Another possible solution to protect against this attack would be to install an intrusion detection system that would be able to identify this signature and terminate any connection associated with it. Immediately upon receipt of the *?wp-html-rend* command, the IDS would issue a reset packet to the attacker which would drop the connection to the web server. This would prevent the web server from ever seeing the malicious command.

One other possibility would be to enable content filtering on a firewall and drop any connection at the border that has a content string of *?wp-html-rend.* As with the IDS solution, the firewall would terminate the connection between the attacker and the web server as soon as it detects the malicious command. The web server would never actually see this command and in turn would not crash.

---

[12] http://knowledgebase.iplanet.com/ikb/kb/articles/4302.html
[13] http://www.kb.cert.org/vuls/id/191763

Some security administrators are reluctant to have an automated feature responsible for terminating connections or services. The reason for the reluctance is that if something were misconfigured or went wrong on the IDS or firewall, a number of potentially valid connections may be inadvertently disconnected. The other side of the fence is that a fair amount of attacks happen when there is no one around to watch the alarms and hence no one to react to an attack. An automated process would allow an attack of this nature to be stopped before it could do any damage. An administrator must choose what method is right for their individual circumstance but should proceed with caution if an automated method of protection is chosen.

As can be seen in most buffer overflow attacks, the traffic coming across the wire appears at the surface to be normal traffic and is sometimes undetectable. This is reason enough to maintain a current patch level on all systems. Allowing only services into your network that are absolutely necessary is another great way to cut down on the number of potential vulnerabilities.

The Incident Handling Process

Our story begins with a company called Company X and their somewhat simplistic web site which includes a router, firewall, intrusion detection system and two web servers. Company X is a medium sized corporation that employs about 1,200 people worldwide. Their primary focus is on manufacturing hand held tools and in the past year have just begun marketing and selling their products on the Internet. As with most companies, budgets are tight at Company X and IT budgets are even tighter as upper management has a difficult time justifying them. However, a new project has recently been approved which entails bringing up a new web server on their already established "public" web site that physically resides in Company X's corporate headquarters.

As most of us know, the IT world does not always allow the time for an administrator to configure a web server "properly" and "securely". Sometimes we are told to set up a web server yesterday, so that a newly developed e-whatever application can be publicly available. This new project is no exception. "Has this project been approved by our security department", I ask. "Of course it has", the developers say. When I questioned this with my boss, I got the Nike response, "Just Do It."

Being one of two network administrators responsible for firewalls and web servers, I trudged forward with what I intended to be the most secure web server on the Internet. Then it happened. My boss called and told me that I had to

leave on the 7:30 pm flight to England where their network was "down". He also indicated that before I left that night the web server needed to be ready to go and to make a change to one of the firewalls – all before I made the 45 minute drive to the airport in rush hour traffic! It was 4:30 pm and I had just begun configuring the web server that needed to be up in 5 minutes. My strategy changed to simply getting the web server up and running and plug up the holes when I returned. There was also an insignificant request by the web development team to enable Web Publishing on the Netscape Enterprise Web Server, so that the team could develop their html code from their desks. Web Publishing is a feature of the Netscape Enterprise Server that allows developers to organize and publish documents from their desktops with a web publishing interface[14]. I decided that allowing developers to publish web pages from their desk seemed to make sense so I left Web Publishing at its default setting of enabled and rushed out the door to catch my plane. The problem that I was not yet aware of was that once this server was up and running it would take an act of congress to bring it down for maintenance!

No one intentionally configures their web server to appear like swiss cheese to attackers, it just seems to "happen" that way. Security policies are things that people are supposed to simply talk about not actually live by!

Before I boarded the plane to leave the country, attackers were already at work mapping our network……A little background information on Company X's web network and security information before our story continues.

Preparation

As with a fairly large number of organizations, upper management of our company felt that if a firewall were in place we were "very well protected". For the most part, the firewalls and firewall administrator constitute the majority of the security budget as well as the security plan for our company. Even though a formal security policy was in place it's rarely referenced when security issues are brought up.

Another frustrating part of securing most networks and network applications is that they are usually designed with little or no consideration for security and if security is discussed it is usually an after thought. Most applications on our network are no exception. Security was normally thought of while an application was being implemented and any objections from our security group would cause a delay of the rollout. This could cost our company thousands of dollars in down time. The argument that "We have a firewall so it should be safe

[14] Netscape Enterprise Server On-line Help – Chapter 1

enough" was constantly used. Also, due to poor economic conditions and the lack of a real security budget, our security staff was quite lean. My company's security group consists of a Chief Security Officer (CSO), an Information Management Manager and two firewall administrators.

Our group was eventually able to convince management that we needed a network intrusion detection solution which they reluctantly approved. Internet Security Systems' RealSecure was the IDS that was selected due to the fairly small amount of training involved as well as its ease of use. We realized that a product such as Snort would provide much more decode information than RealSecure, but due to the lack of resources and knowledge, ISS's product won the battle.

With the IDS in place we felt that three layers of network based protection should provide our web site with a fair amount of protection. Physical security was provided for the web site equipment with limited badge access to the computer room for only authorized employees.

At the very outside of our network a Cisco router was used to drop all incoming packets that are destined for any port above 1024. An anti-spoofing access list entry was also on this router, so that only addresses that are sourced from our network are allowed out onto the Internet. The IOS on the router was routinely updated in an effort to stay current on patch and security releases. Secure access to the router was provided with a TACACS+ server which requires a user ID and password in conjunction with a SecureID token.

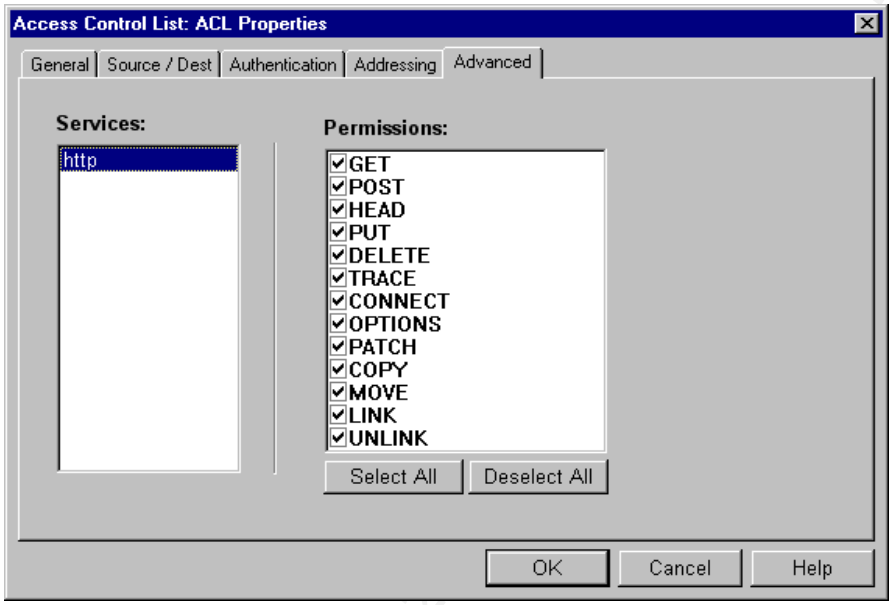
The second layer of protection was a Sidewinder firewall from Secure Computing. The Sidewinder folks created what they call SecureOS™ by modifying a standard BSD/OS UNIX kernel in order to implement their Type Enforcement technology[15]. Basically Type Enforcement separates each firewall application or service into individual "cells" that are not accessible by any other application or service. This prevents an attacker from using a particular service to launch an attack on some other service regardless of what level of privilege they have gained on the firewall. Type Enforcement technology also prevents an attacker from turning off auditing, so they will not go undetected as they might do with other firewall implementations.

The Sidewinder firewall is an application-layer firewall which means that it can analyze a packet at the application layer of the protocol stack[16]. An application-layer firewall utilizes proxies that have the capability to provide a higher level of data protection by analyzing a packet at a higher level. These application-layer

[15] http://www.securecomputing.com/index.cfm?sKey=738
[16] http://www.securecomputing.com/index.cfm?sKey=737

proxies do not allow for a direct connection from the internal side of the firewall to the external side. They actually rewrite the application layer data which "eliminates the risk of network layer attacks based on packet fragmentation and mismatches between protocol stacks." [17]

The Sidewinder that we had implemented only allowed tcp port 80 to the two available web servers. It was also possible to configure the http proxy to only allow the following permissions:



The firewall would filter packets based on the rules for the http protocol (or any other defined proxy) and thus will drop a spoofed http packet. An alarm would then be triggered which could be sent via a pager, e-mail or snmp alarm informing the administrator immediately of a potential attack. Another form of security on our firewalls was that only internal users were allowed to access to them for management purposes. A user would connect via a secured channel for the gui interface and ssh for a terminal session.

The third layer of protection was a RealSecure Network Sensor from Internet Security Systems which is a network layer intrusion detection system. The RealSecure network engine that was used had two network interface cards (NICs) installed. One NIC was in promiscuous mode and was monitoring the external connection of our firewall. The other NIC was connected to our internal network and was used to report events to a centralized RealSecure management console which managed multiple network engines. RealSecure has the capability to terminate connections that are deemed to be inappropriate

[17] http://www.securecomputing.com/index.cfm?sKey=737

activity. However we were using it to simply notify the administrator of interesting activity via email as well as log these events to a database for historic and reporting purposes[18]. ISS provides Xpress Updates to the signatures on a regular basis and these were applied to each of the network engines as they became available. Access to the management console and the network engines require that you physically be in front of the machine in order to operate the host. Both the management console and network sensors run on top of Windows NT. We verified that all intrusion detection systems had the latest Microsoft patches installed and the NT operating systems had been hardened based on the guidelines set forth on Microsoft's Best Practices for Enterprise Security web page.

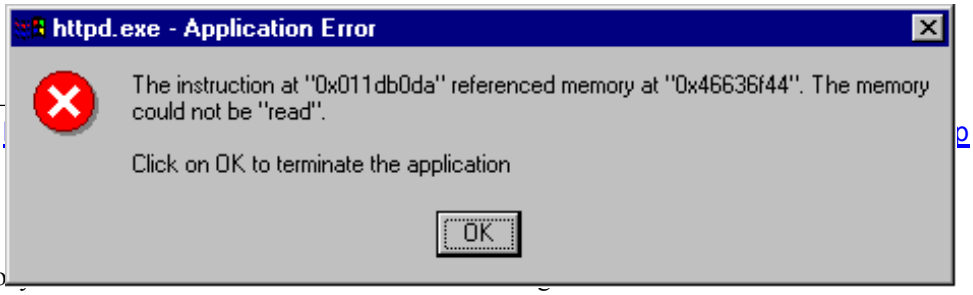
Prior to this incident my company had not deemed it necessary to form a Computer Incident Response Team (CIRT) as this was not in our current or foreseeable budgets. Documentation on any incident consisted simply of e-mails that were exchanged between participants in the troubleshooting process. Therefore we did not have a structured set of steps to follow when anything interesting was discovered on our network.

Our current security team was not aware of our local FBI contact nor did we know of any good forensic contacts or tools in the event of a serious security breech. My guess was that even if we were able to detect a compromise, which I am not convinced that we could have done, we did not have a process in place that would ensure the evidence necessary to prosecute the attacker.

Identification

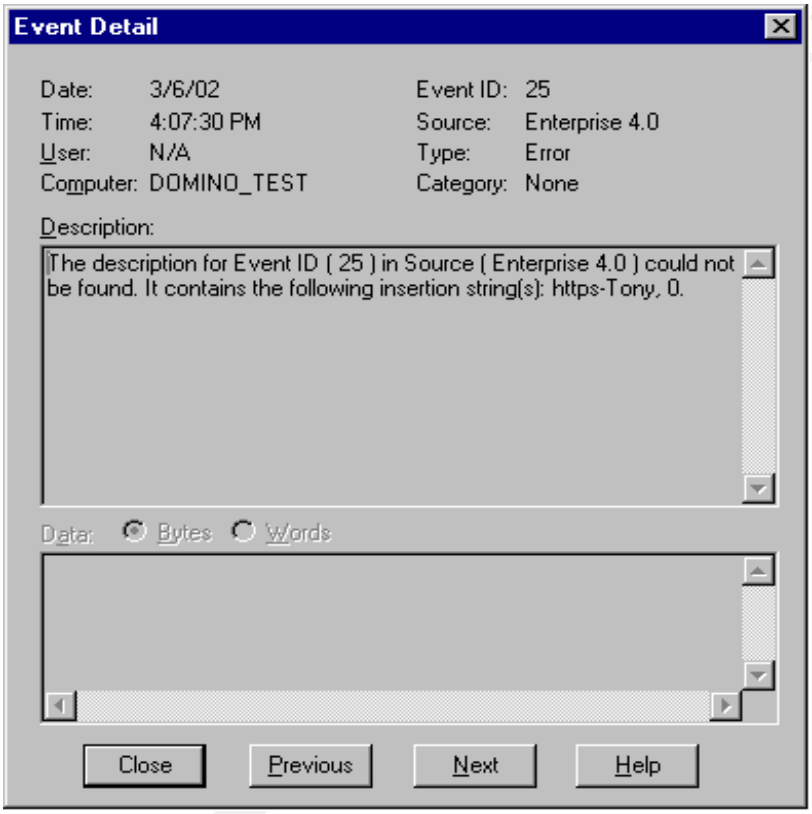
Now back to the story…..The identification process could have been fairly simple in this case. However the necessary process was not in place to provide troubleshooting when a portion of our security team was not in the office. It took about four days for someone to make a firewall administrator aware of the issue as I was out of town and the other admin was on vacation. Everyone assumed that the error was a Microsoft memory issue and that simply clicking on OK was the temporary fix.

When I got back in town, my boss explained that the new web server I set up had been crashing about 10 times each day with the following error message on the screen.



18

No one had been able to determine the reason for this error message. Once the error message was cleared by clicking on the OK button, users were able to access the web server again. I immediately looked at the server and verified that it had the latest Microsoft patches installed. The NT operating system had been hardened based on the guidelines set forth on Microsoft's Best Practices for Enterprise Security web page. I then looked at the server's log files in the event viewer and there appeared to be no errors related to this issue. The only type of error that appeared in the Event Viewer – Application Log is in the diagram below.



The errors, like the one above, vaguely corresponded to the web server crashes, but were of no real help in determining the cause of the web server crashes as they were fairly ambiguous. Even though these error messages did not appear to help us determine why the web server was crashing, all errors surrounding this incident were printed and saved for future reference.

Another member of our security team noted that the other Netscape Enterprise Server which runs on an AIX platform, had not experienced any unscheduled down time for the past two months. This pointed us in the direction of a Windows NT based exploit.

I then looked at the firewall and intrusion detection logs to see if anyone had been snooping around. While looking through the firewall network probe reports, I noticed that someone performed a few port scans on the outside of the firewall at about the same time I was driving to the airport but nothing else of significant interest. I simply blew those scans off because **only** tcp port 80 was open to all Internet users. I also looked at the intrusion detection host and did not see anything interesting in its logs either. Since this was the extent of my forensic information and the attack was continuously happening, I decided to try and catch the attacker in the act. The firewall and a sniffer would come in handy for this exercise. The tcpdump utility would be used to capture packets destined for the web server on the Sidewinder firewall that sits on the outside of the web server. The sniffer would also be used to capture this traffic, but it might be able to give me a more detailed look at the packets.

The command **tcpdump –i exp1 –nXv host c.d.58.50** was run on the Sidewinder firewall between the attacker and the web server in an attempt to capture the event that was crashing the web server's httpd.exe process. Network Associates' Sniffer Pro v4.0.12 was also used to monitor the external interface on the firewall. An input capture filter was created on the sniffer so that it would only capture traffic that was destined for the web server (c.d.58.50).

As soon as the web server crashed, the tcpdump command and the sniffer trace were halted. The traces were scanned in an attempt to see if there was any interesting traffic destined for the web server. Output from both the tcpdump trace and the sniffer trace can be found in Appendix A.

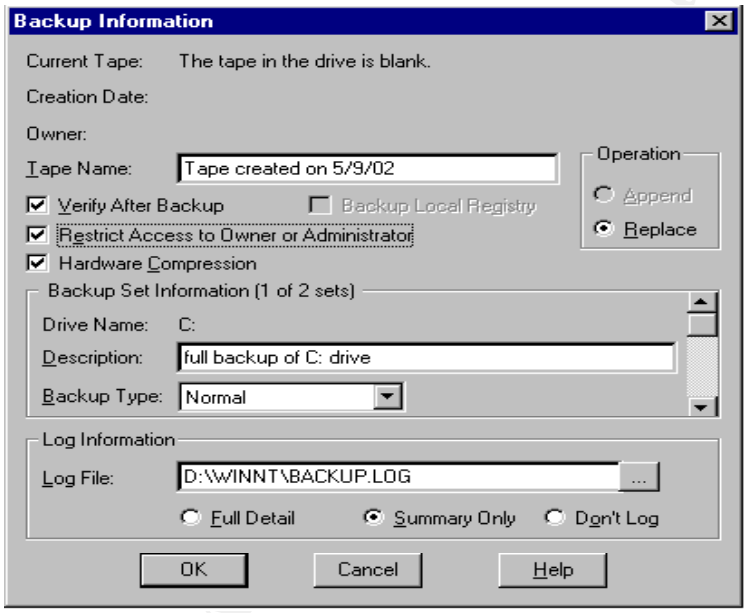
While analyzing the trace files, I immediately noticed after the tcp hand shake was completed the first command that was pushed to the web server contained a string **GET /?wp-html-rend HTTP/1.1** from a host with an IP address of a.b.58.99. This packet indicated that the attacker was using the HTTP/1.1 version of the Hypertext Transfer Protocol. Since the HTTP (tcp port 80) protocol was being allowed through the firewall, further investigation was warranted on specific GET commands for this web site.

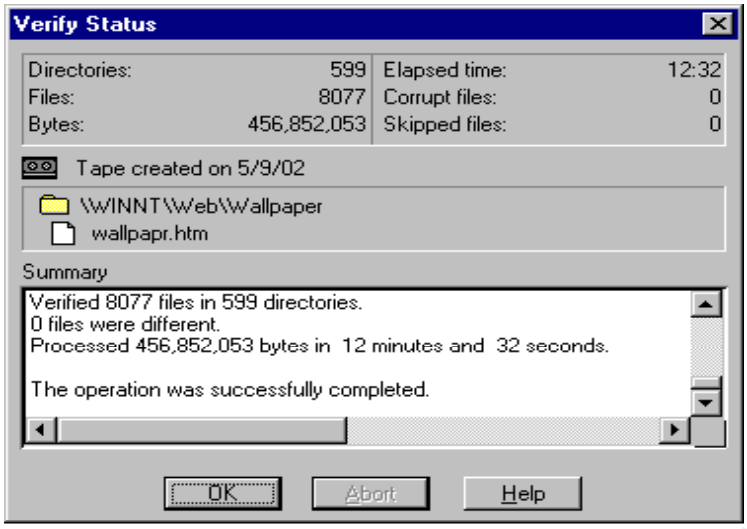
In an attempt to determine "normal" web traffic for this site, data was captured from browsing the home page for this site with the same tcpdump command and sniffer trace rules. These traces were compared with the traffic that were collected when the web server crashed. I immediately noticed that the **GET /?wp-html-rend HTTP/1.1** command could not be found in a "normal" trace. Once the GET command has been received by the web server, it replies with an ack and at that point the host sends a reset packet and the connection terminates. At this point I began to search for instances of this string on the web

using Google. I was greeted with a number of web sites such as CERT, SecuriTeam and ISS X-Force that had information about vulnerabilities that contained the **GET /?wp-html-rend HTTP/1.1** command.

Containment

Once it was determined that the string *?wp-html-rend* was used to exploit a known vulnerability and the attacker's address was identified, we performed a normal / full backup of the web server using Microsoft's Windows Backup program. We were fairly confident that the web server did not get compromised during this process. However in the event that the web server did get compromised during this attack, a full backup might be able to provide us with some of the necessary evidence for prosecuting the attacker. The backup was also performed as a precaution in the event that something happened to get changed on the web server during troubleshooting and repair processes.





Since this attack appeared at the surface to be a DoS attack, the containment process was somewhat different than that of a compromised system. We began the containment process by sending a **ping c.d.58.50** and a **traceroute c.d.58.50** to this host in an attempt to verify that this address was not being spoofed. Both commands returned successfully indicating that the attacking host was currently on-line. I then performed a **nslookup** on **a.b.58.99** and received a **can't find a.b.58.99: Non-existent domain** message which indicated that this host was not registered in any public dns tables.

After making a reasonable assumption that this host was not being spoofed, I immediately placed the following access list entry on our border Cisco router **access-list 101 deny ip a.b.58.99 any.** Access-list 101 was assigned to the Serial 0 interface on this router which was the link to the Internet for this network. This access list entry will drop any packets with a source address of the attacker (a.b.58.99). The purpose for placing the access list entry on the border router was to keep the web server up and running while I began the process of determining how to correct this vulnerability on the web server.

At this point, I began looking through the web pages that I found in the search above and determined what steps were necessary to fix this issue. The first step was to disable the *?wp-html-rend* command, the details of which were found by referring to the iPlanet Knowledge Base Article ID: 7761. This article indicated that the *?wp-html-rend* command can be disabled by loading *disrend.dll* on a system and by adding the following lines to the *obj.conf* file:

Init fn="load-modules" funcs="disRend" shlib="/disrend.dll"
PathCheck fn="disRend"

Other possible solutions are also described in iPlanet Knowledge Base Article ID: 4302. One of the solutions explained in this article involves disabling "Directory Indexing"[19]. The CERT Advisory – Vulnerability Note VU#191763 indicated that you not only disable directory indexing but that you disable Web Publisher[20] as well. The ProCheckUp Security Bulletin PR01-04, the ISS X-Force Security Database Entry 7842 and SecurityFocus Bugtraq ID 3826 all have basically the same solutions as noted above for this vulnerability.

We do not formally have a jump kit but the some of the tools that were used for this incident include: Microsoft's NT Backup Utility, clean backup tapes, Ethernet hub, Windows NT Resource Kit, laptop computer and cell phone. We also have a CD with nslookup, whois, ping, tracert among other NT related programs that are used to troubleshoot an incident.

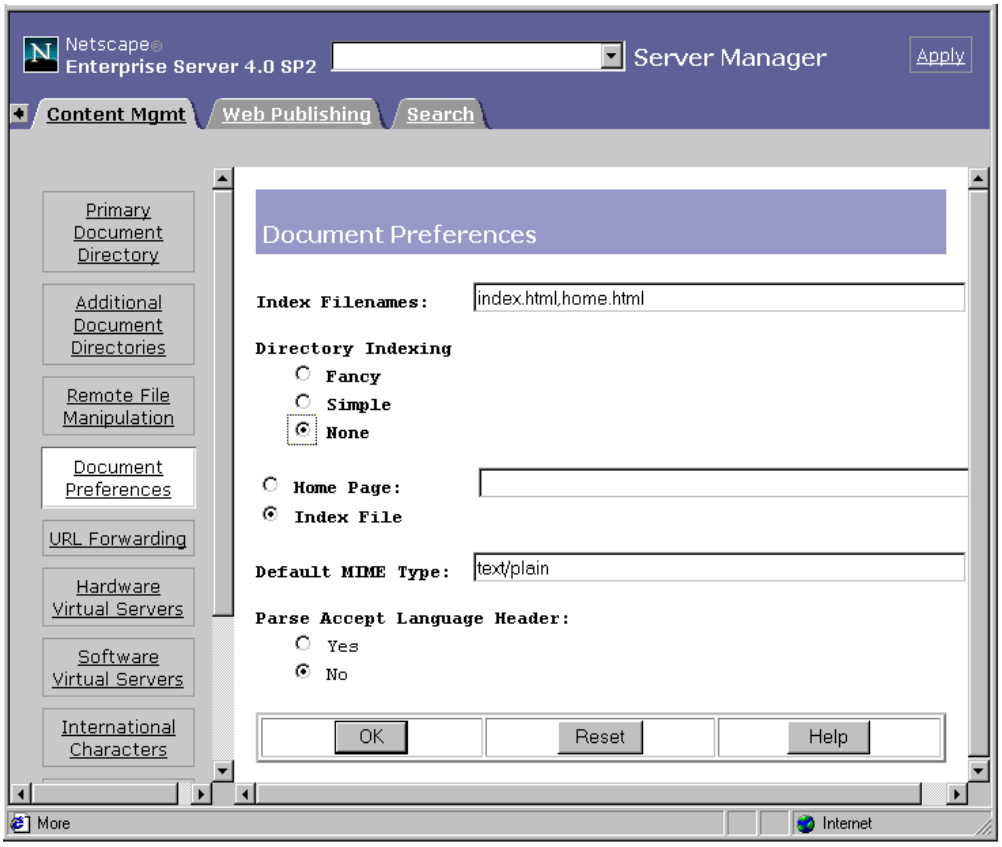
Eradication

Once the attacker was blocked by the access-list entry on our border router, the vulnerable web server was taken off-line. Patching the vulnerable web server began by disabling the *?wp-html-rend* command using the instructions in iPlanet Knowledge Base Article ID: 7761. This article indicates that the *?wp-html-rend* command can be disabled by loading *disrend.dll* on a system and by adding the following lines to the *obj.conf* file:

```
Init fn="load-modules" funcs="disRend" shlib="/disrend.dll"
PathCheck fn="disRend"
```

It was then decided to disable "Directory Indexing" (DI) using the information in iPlanet Knowledge Base Article ID: 4302. DI was turned on by default and is a feature that was not required on this web server. To change the "Directory Indexing" via the Administration Interface, we went to Content Management, selected Document Preferences, and selected "None" from the three checkboxes. Clicked on OK and Apply.

[19] http://knowledgebase.iplanet.com/ikb/kb/articles/4302.html
[20] http://www.kb.cert.org/vuls/id/191763



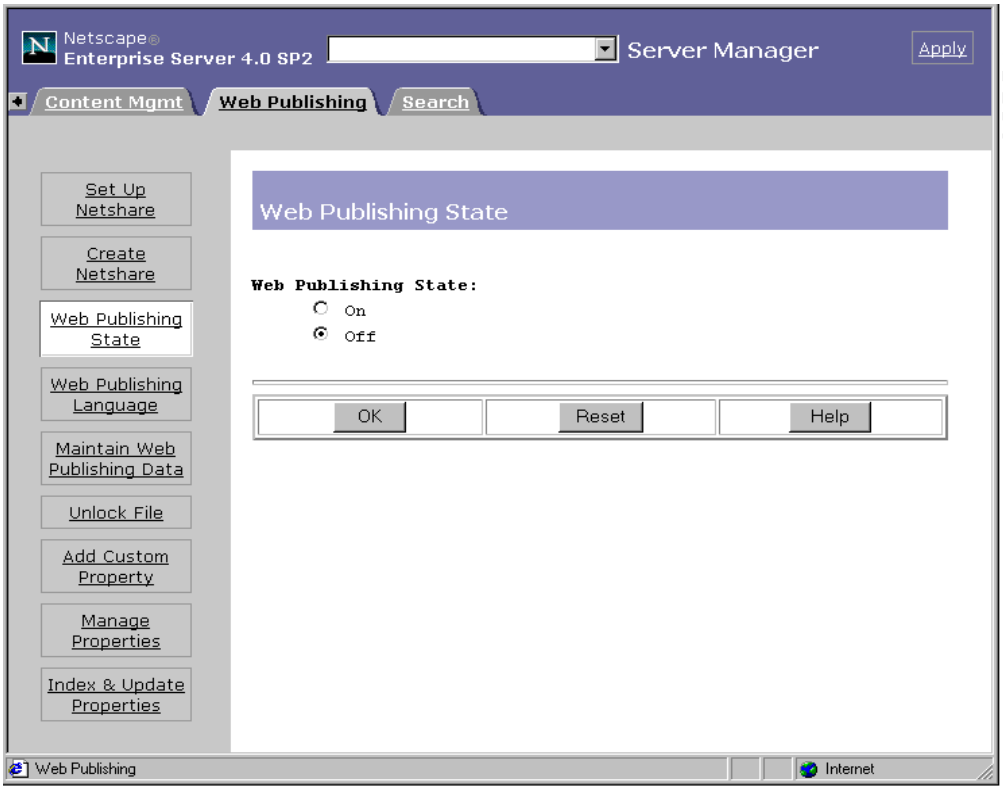
Another option would be to directly modify the obj.conf file. These lines directly control the Directory Indexing behavior:

("Simple" indexing)
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-simple

("Fancy" indexing)
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common

To disable Directory Indexing simply comment out or remove either of the lines shown above.

Web Publishing was also disabled by referencing the CERT Advisory – Vulnerability Note VU#191763. To change the "Web Publishing State" via the



Administration Interface, we went to Web Publishing, selected Web Publishing State, and selected “Off”. Clicked on OK and Apply. Prior to this incident, Web Publishing was enabled on all web servers. After this incident, it was decided that Web Publishing was to be disabled on any publicly accessible web server.

After it was verified that Web Publisher and Directory Indexing were off, we then downloaded Netscape Enterprise Server version 4.1 SP9, which is not vulnerable to this exploit, from the iPlanet web site. The new version was installed onto the vulnerable web server without incident. We verified that submitting the ?wp-html-rend command did not crash the web server with the new service pack.

Recovery

When the service pack was successfully installed, a second normal / full backup was performed on the web server using the NT Backup utility. This was done so that we had a clean copy of what the web server was supposed to look like for future reference. At this point we declared that this particular web server was now immune to the iPlanet Web Server Enterprise Edition and Netscape

Enterprise Server malformed Web Publisher command causes denial-of-service - CERT/CC Vulnerability Note VU#191763 and it could be placed back into production. The once vulnerable web server was continually monitored for about a week after this incident to ensure that we had stopped this particular DoS attack from occurring on our network.

Once we were fairly confident that we had sufficiently protected this web server from this vulnerability, we then decided to do some research on the person responsible for this attack. I began this search by performing a **whois –h whois.arin.net a.b.58.99** and the following was returned:

Bad Company (NET-BADCOMPANY)
666 Fire Lane
Incinerator, AZ 66600
US

Netname: BADCOMPANY
Netblock: a.b.58.0 – a.b.59.0

Coordinator:
Avery, Bill avery@badcompany.com
505-555-0666

Domain System inverse mapping provided by:

NS.BADCPMNY.COM a.b.58.1
NS2.BADCPMNY.COM a.b.59.2

Record last updated on 11-Nov-1998.
Database last updated on 13-Mar-2002 19:58:08 EDT.

The ARIN Registration Services Host contains ONLY Internet Network Information: Networks, ASN's, and related POC's. Please use the whois server at rs.internic.net for DOMAIN related Information and whois.nic.mil for NIPRNET Information.

Instead of performing a command line search, you could also visit the American Registry for Internet Numbers (ARIN) web site and go to the whois section that can be found at ARIN.net.

I then sent an email to their contact, Bill Avery, indicating that a host that he was responsible for (a.b.58.99) had been sending malformed Web Publisher commands that had been crashing our web server, and asked him to

investigate. Included in this message was a portion of the tcpdump output and sniffer trace that I had captured from the attack. It did not appear that this company was an ISP because a quick search on the web determined that this company was involved in the waste management industry. Therefore the host was most likely a user on Bad Company's network or a host that had been compromised by an attacker who was not physically on their network.

After performing some more in depth searches on the ARIN site, I determined that Bad Company's Internet Service Provider was Big Pipe ISP from Chicago, IL. The contact name for Big Pipe who was listed for abuse reporting was Roberta Dunleavy. I sent her an email similar to the message that was sent to Bill Avery, indicating that a host from Bad Company was attacking our network and asked if it would be possible to stop this activity ASAP. Included with the email to Roberta was a portion of the tcpdump output and sniffer trace that I had captured from the attack.

An reply e-mail message was received one day after sending Big Pipe an abuse message. The ISP indicated they had contacted the owners of the host in question. They reported that the network administrators from Bad Company had taken this host off their network and were investigating the background of the attack from this machine. This was verified by checking the logs of the hits on our web server which showed that no one had tried to send the *?wp-html-rend* for the past 20+ hours. Big Pipe also indicated that they would add an abuse strike to Bad Company's account which would indicate that the ISP is proactively trying to stop attacks from being sourced from their customers. This was a good thing!

Lessons Learned

A post-attack meeting was attended by all parties involved to determine what was done right and what could be improved upon for future incidents. The first issue that was glaringly obvious from this event was that a step-by-step incident handling procedure needs to be in place in order to handle future attacks. No one in our group had either the experience or knowledge to properly document or process an attack. It was suggested that a Computer Incident Response Team be formed to handle interesting events in the future. Members of this team would consist of a representative from each of the following groups: networking, operations, security, legal, human resources, upper management and business continuation. In addition to dealing with computer related incidents, the CIRT would also be responsible for ensuring that future network, server, workstation, etc. decisions would adhere to our corporate security policy. The idea of a secured war room was also discussed. This room would be used

for meetings related to CIRT issues as well as a storage location for evidence related to each incident.

Another idea that came from our post-incident meeting was to become more active in the security community. Subscribing to email lists such as CERT or ISS's X-Force would keep us abreast of known vulnerabilities as they are reported. Also, joining a local chapter of a security group such as ISSA or a security forum such as SANS would keep us up to date with the latest industry related issues. Taking a training class once a year would provide updated knowledge on the latest attacker techniques and intrusion detection methods. Since the tools and methods of attackers change on a daily basis, it is imperative that all security personnel stay on top of the latest vulnerabilities and incident handling processes.

Prior to this attack, it was assumed that no attacker would pick on our network since we were such a small entity and we did not have a very large presence on the web. Most of our group used to feel that we had a firewall to protect us from most attacks, so we did not have much to worry about. How wrong we were! It seems that is was only a matter of WHEN we were going to get attacked and not IF!

After this incident, at a minimum all interesting traffic will get a second look. All future servers being built will use a predefined set of hardening installation standards as well. Each network device will be checked on a monthly basis to ensure that the latest available patches have been applied. The idea of a host based intrusion detection product is being considered as is having a third party perform a security assessment on our internal and external networks. Using login warning banners for every server to warn users of inappropriate use is another possibility we are considering to help deter attacks.

A DoS incident was probably the best thing that could have happened on our network to open the eyes of all parties involved in this incident. We were fortunate that downtime was the only result of this attack instead of lost data, a compromised system or even public humiliation. Sometimes it takes an incident such as the one described above to make management realize that all networks, no matter how small, are susceptible to attacks.

Conclusion

When dealing with incident handling, an analyst cannot get stuck in the mold of "this is how we've always done it". As a security professional, you must constantly be looking for new methods of attacks and vulnerabilities. One good

practice is to keep up with attacker information by visiting web sites such as www.wiretrip.net/rfp or www.hackers.com which are both known sources for attacker tools and techniques. Subscribing to mailing lists such as Bugtraq and Carnegie Mellon's Computer Emergency Response Team (CERT) will provide you with email that will keep you up to date on the latest computer attacks and defenses. Having a library of technical reference books is also a great way to "be prepared" in the event of an attack.

The exceptional attackers will always find a way to break into a network. Our job is to be thorough and unrelenting when protecting network systems. Even though this does not guarantee that your network will not be attacked, it will make it extremely difficult for the attacker to discover a vulnerability and exploit it. Just remember that many attackers seem to abide by the saying "Follow the path of least resistance".

Appendix A

Tcpdump Trace

The following is an excerpt from the tcpdump trace using the command **tcpdump –i exp1 –nXv host abc.def.58.50**. The source address / attacker's address is a.b.58.99 and the target address is c.d.58.50. The malformed get command is highlighted in red below.

```
12:37:36.846459 a.b.58.99.1044 > c.d.58.50.80: S 70147:70147(0) win 8192 <mss
1460> (DF) (ttl 128, id 6177)
    0000: 00 03 47 0a 00 1a 00 20   e0 69 65 12 08 00 45 00  __G____  _ie___E_
    0010: 00 2c 18 21 40 00 80 06   35 51 xx xx 3a 63 xx xx  _,_!@___ 5Q_b:c_b
    0020: 3a 32 04 14 00 50 00 01   12 03 00 00 00 00 60 02  :2___P__ ______`_
    0030: 20 00 b4 64 00 00 02 04   05 b4 00 00              __d____ ____
12:37:36.846534 c.d.58.50.80 > a.b.58.99.1044: S 1602976130:1602976130(0) ack
70148 win 17520 <mss 1460> (DF) (ttl 64, id 23574)
    0000: 00 20 e0 69 65 12 00 03   47 0a 00 1a 08 00 45 00  _ _ie___ G_____E_
    0010: 00 2c 5c 16 40 00 40 06   31 5c xx xx 3a 32 xx xx  _,\_@_@_ 1\_b:2_b
    0020: 3a 63 00 50 04 14 5f 8b   79 82 00 01 12 04 60 12  :c_P____ y_____`_
    0030: 44 70 b6 d5 00 00 02 04   05 b4                    Dp______ __
12:37:36.846762 a.b.58.99.1044 > c.d.58.50.80: . ack 1 win 8760 (DF) (ttl 128, id
6433)
    0000: 00 03 47 0a 00 1a 00 20   e0 69 65 12 08 00 45 00  __G____  _ie___E_
    0010: 00 28 19 21 40 00 80 06   34 55 xx xx 3a 63 xx xx  _(_!@___ 4U_b:c_b
    0020: 3a 32 04 14 00 50 00 01   12 04 5f 8b 79 83 50 10  :2___P__ ____y_P_
    0030: 22 38 f0 ca 00 00 00 00   00 00 00 00              "8______ ____
12:37:36.847338 a.b.58.99.1044 > c.d.58.50.80: P 1:334(333) ack 1 win 8760 (DF)
(ttl 128, id 6689)
    0000: 00 03 47 0a 00 1a 00 20   e0 69 65 12 08 00 45 00  __G____  _ie___E_
    0010: 01 75 1a 21 40 00 80 06   32 08 xx xx 3a 63 xx xx  _u_!@___ 2__b:c_b
    0020: 3a 32 04 14 00 50 00 01   12 04 5f 8b 79 83 50 18  :2___P__ ____y_P_
    0030: 22 38 d1 5f 00 00 47 45   54 20 2f 3f 77 70 2d 68  "8____GE T /?wp-h
    0040: 74 6d 6c 2d 72 65 6e 64   20 48 54 54 50 2f 31 2e  tml-rend  HTTP/1.
    0050: 31 0d 0a 41 63 63 65 70   74 3a 20 69 6d 61 67 65  1__Accep t: image
12:37:36.855041 c.d.58.50.80 > a.b.58.99.1044: . ack 334 win 17187 (DF) (ttl 64,
id 23578)
    0000: 00 20 e0 69 65 12 00 03   47 0a 00 1a 08 00 45 00  _ _ie___ G_____E_
    0010: 00 28 5c 1a 40 00 40 06   31 5c xx xx 3a 32 xx xx  _(\_@_@_ 1\_b:2_b
    0020: 3a 63 00 50 04 14 5f 8b   79 83 00 01 13 51 50 10  :c_P____ y____QP_
    0030: 43 23 ce 92 00 00                                  C#____
```

Sniffer Pro v.4.0.12 Trace

The following is an excerpt from the Sniffer Pro trace that was performed during the attack. Again, the source address / attacker's address is a.b.58.99 and the target address is c.d.58.50. For readability purposes, I changed the font to 9 point for the text of the trace. Also, due to the size of the trace file I have only included frame #4 of the trace which contains the GET command that cripples the vulnerable web server.

```
- - - - - - - - - - - - - - - - - - - - Frame 4 - - - - - - - - - - - - - - - -
 Frame Status Source Address Dest. Address Size Rel. Time  Delta Time Abs. Time           Summary
    4       [a.b.58.99] [c.d.58.50] 387 0:00:00.001 0.000.766     03/18/2002 12:52:36 PM HTTP: C
Port=1045 GET /?wp-html-rend HTTP/1.1    ← malformed get command
DLC:  ----- DLC Header -----
     DLC:
     DLC:  Frame 4 arrived at  12:52:36.2307; frame size is 387 (0183 hex) bytes.
     DLC:  Destination = Station 0003470A001A
     DLC:  Source      = Station 0020E0696512
     DLC:  Ethertype   = 0800 (IP)
     DLC:
IP: ----- IP Header -----
     IP:
     IP: Version = 4, header length = 20 bytes
     IP: Type of service = 00
     IP:       000. ....   = routine
     IP:       ...0 .... = normal delay
     IP:       .... 0... = normal throughput
     IP:       .... .0.. = normal reliability
     IP:       .... ..0. = ECT bit - transport protocol will ignore the CE bit
     IP:       .... ...0 = CE bit - no congestion
     IP: Total length    = 373 bytes
     IP: Identification  = 33057
     IP: Flags           = 4X
     IP:       .1.. .... = don't fragment
     IP:       ..0. .... = last fragment
     IP: Fragment offset = 0 bytes
     IP: Time to live    = 128 seconds/hops
     IP: Protocol        = 6 (TCP)
     IP: Header checksum = CB07 (correct)
     IP: Source address      = [a.b.58.99]
     IP: Destination address = [c.d.58.50]
     IP: No options
     IP:
TCP: ----- TCP header -----
     TCP:
     TCP: Source port            = 1045
     TCP: Destination port       = 80 (WWW-HTTP)
     TCP: Sequence number        = 70160
     TCP: Next expected Seq number= 70493
     TCP: Acknowledgment number   = 2238446863
     TCP: Data offset            = 20 bytes
```

```
    TCP: Flags                = 18
    TCP:              ..0. .... = (No urgent pointer)
    TCP:              ...1 .... = Acknowledgment
    TCP:              .... 1... = Push
    TCP:              .... .0.. = (No reset)
    TCP:              .... ..0. = (No SYN)
    TCP:              .... ...0 = (No FIN)
    TCP: Window               = 8760
    TCP: Checksum             = 27E6 (correct)
    TCP: No TCP options
    TCP: [333 Bytes of data]
    TCP:
HTTP: ----- Hypertext Transfer Protocol -----
    HTTP:
    HTTP: Line  1:  GET /?wp-html-rend HTTP/1.1   ← malformed get command
    HTTP: Line  2:  Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
    HTTP:           application/vnd.ms-excel, application/msword, application/v
    HTTP:          nd.ms-powerpoint, */*
    HTTP: Line  3:  Accept-Language: en-us
    HTTP: Line  4:  Accept-Encoding: gzip, deflate
    HTTP: Line  5:  User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.
    HTTP:           0)
    HTTP: Line  6:  Host: testweb
    HTTP: Line  7:  Connection: Keep-Alive
    HTTP: Line  8:
    HTTP:
```

```
ADDR  HEX                                        ASCII
0000: 00 03 47 0a 00 1a 00 20 e0 69 65 12 08 00 45 00 | ..G.... àie...E.
0010: 01 75 81 21 40 00 80 06 cb 07 xx xx 3a 63 xx xx | .u !@.€.Ë.œb:cœb
0020: 3a 32 04 15 00 50 00 01 12 10 85 6b fd 0f 50 18 | :2...P.....…ký.P.
0030: 22 38 27 e6 00 00 47 45 54 20 2f 3f 77 70 2d 68 | "8'æ..GET /?wp-h   ← malformed
0040: 74 6d 6c 2d 72 65 6e 64 20 48 54 54 50 2f 31 2e | tml-rend HTTP/1.     get command
0050: 31 0d 0a 41 63 63 65 70 74 3a 20 69 6d 61 67 65 | 1..Accept: image
0060: 2f 67 69 66 2c 20 69 6d 61 67 65 2f 78 2d 78 62 | /gif, image/x-xb
0070: 69 74 6d 61 70 2c 20 69 6d 61 67 65 2f 6a 70 65 | itmap, image/jpe
0080: 67 2c 20 69 6d 61 67 65 2f 70 6a 70 65 67 2c 20 | g, image/pjpeg,
0090: 61 70 70 6c 69 63 61 74 69 6f 6e 2f 76 6e 64 2e | application/vnd.
00a0: 6d 73 2d 65 78 63 65 6c 2c 20 61 70 70 6c 69 63 | ms-excel, applic
00b0: 61 74 69 6f 6e 2f 6d 73 77 6f 72 64 2c 20 61 70 | ation/msword, ap
00c0: 70 6c 69 63 61 74 69 6f 6e 2f 76 6e 64 2e 6d 73 | plication/vnd.ms
00d0: 2d 70 6f 77 65 72 70 6f 69 6e 74 2c 20 2a 2f 2a | -powerpoint, */*
00e0: 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 | ..Accept-Languag
00f0: 65 3a 20 65 6e 2d 75 73 0d 0a 41 63 63 65 70 74 | e: en-us..Accept
0100: 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c | -Encoding: gzip,
0110: 20 64 65 66 6c 61 74 65 0d 0a 55 73 65 72 2d 41 |  deflate..User-A
0120: 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 34 2e | gent: Mozilla/4.
0130: 30 20 28 63 6f 6d 70 61 74 69 62 6c 65 3b 20 4d | 0 (compatible; M
0140: 53 49 45 20 35 2e 35 3b 20 57 69 6e 64 6f 77 73 | SIE 5.5; Windows
0150: 20 4e 54 20 34 2e 30 29 0d 0a 48 6f 73 74 3a 20 |  NT 4.0)..Host:
0160: 74 65 73 74 77 65 62 0d 0a 43 6f 6e 6e 65 63 74 | testweb..Connect
0170: 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65 0d | ion: Keep-Alive.
0180: 0a 0d 0a                                        | ...
```

References

Specific Articles and Papers

iPlanet Product Alert, dated May 11, 2001

iPlanet Knowledge Base Article ID 7761

iPlanet Knowledge Base Article ID 4302

CERT Advisory – Vulnerability Note VU#191763, by Art Manion, dated January 8, 2002

CERT Advisory – Vulnerability Note VU#985347, by Art Manion, dated January 8, 2002

CERT Advisory – Vulnerability Note VU#32794, by Jeff S. Havrilla and Art Manion, dated January 11, 2002

ProCheckUp Security Bulletin PR01-04, by Richard Brain, dated January 8, 2002

SecuriTeam Security Alert, by Chris Wysopal, dated January 11, 2002

iPlanet Security Flaws Unmasked By John Leyden, dated January 9, 2002

ISS X-Force Security Database Entry 7842, dated January 8, 2002

ISS X-Force Security Database Entry 6058, dated February 2, 2001

SecurityFocus Bugtraq ID 3826, dated January 9, 2002

Netscape Enterprise Server On-line Help

References

General Resources and Tools Used

SANS

ISS X-Force

The iPlanet Web Site

CERT.ORG

World Wide Web Consortium

Microsoft's TechNet

ISSA

BugTraq

Snort.org

Netcraft

Google

Marko.net

Secure Computing

Arin.net

SecuriTeam.com

Wiretrip.net

HDC

Ohio State University

DataSet

Insecure.org

Tcpdump.org

Network Associates - Sniffer