GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials Bootcamp Style (Security 401)"
at http://www.giac.org/registration/gsec

SANS GSEC Practical

"Defeating Perimeter Security With HTTP"

Marcus Bailey
GSEC Practical Version 1.4
08/12/2002

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

TABLE OF CONTENTS

Abstract……..3
Introduction to port 80……..……..3
The HTTP Protocol……..……...……..4
Hardware and Software that utilize port 80/HTTP……9
Security Concerns with Port 80 and HTTP….……11
Penetration Attacks from Outside the Perimeter……12
Hiding Malicious Activity in Outbound HTTP…….……..15
Detecting HTTP Based Attacks……16
Defending Against HTTP Based Attacks..……17
Summary……18
Additional Information……..……..……..18
Bibliography……..……19

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

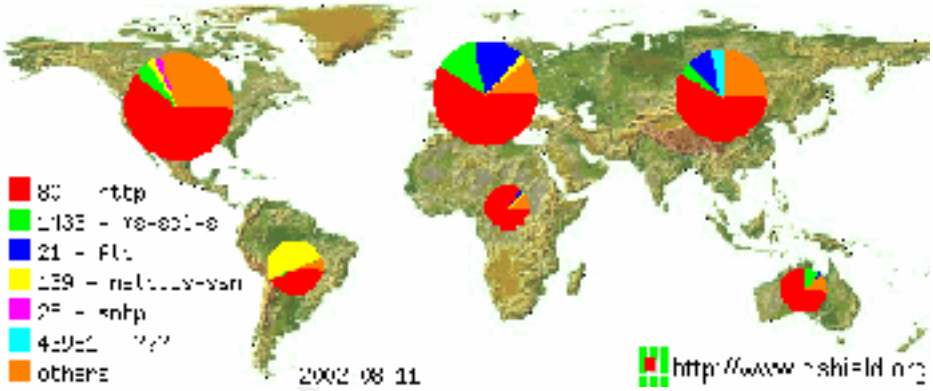
Abstract

This document will discuss TCP port 80 and how it is being utilized to bypass firewalls and intrusion detection systems. The concepts of perimeter security will be discussed, as well as methods malicious users are employing to defeat perimeter security via port 80 (HTTP). In addition, methods for defending against these attacks and/or mitigating the risk will be discussed.

Introduction to Port 80

Port 80, like other commonly known ports, can be used for any number of services/daemons on network connected computers. However, port 80 is most commonly associated with the HTTP (hypertext transfer protocol) protocol. More information on HTTP is discussed in the next session.

Port 80 is commonly attacked by malicious persons in order to take advantage of web server vulnerabilities. Once they've exploited a web server vulnerability, the attacker will often work his/her way into the private network. Port 80 is commonly on the top of the list of the "Top Ten Targeted Ports" as reported by DShield at HTTP://www.dshield.org/topports.html. Figure 1 is from the Dshield.org website, and displays the top ten targeted ports, for date 08/11/2002, organized by geography. A year ago at this time, port 80 was on the top of the list as well.



(Figure 1)

Figure 2 is from the DShield Top Ten Targeted ports site (see URL listed in the first paragraph). Figure 2 lists, in chart format, the top ten targeted ports with an imbedded graph indicating the daily frequency of scans and attacks on the specified port over the month. DShield updates their chart daily, and Figure 2 represents the chart of 8/11/2002.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Service Name	Port Number	Activity Past Month	Explanation
http	80		HTTP Web server
ms-sql-s	1433		Microsoft SQL Server
ftp	21		FTP servers typically run on this port
netbios-ssn	139		Windows File Sharing Probe
smtp	25		Mail servers listen on this port.
[illegible]	[illegible]		NetWare IP
asp	27374		Scan for Windows SubSeven Trojan
[illegible]	8112		
[illegible]	3246		Probably a peer-to-peer file sharing tool
sunrpc	111		RPC: Vulnerable on many Unix systems. Can get root

(Figure 2)

The HTTP Protocol

The accepted and common standard for World Wide Web file transfer is the Hypertext Transfer Protocol (HTTP). Specifics on the HTTP are detailed in RFC 2068 and its update RFC 2616.

In general, HTTP is the protocol used for exchanging various types of files over the Internet. FTP (File Transfer Protocol) and gopher are also used to transfer files over the internet, however this document will focus only on HTTP. HTTP has been in use on the World Wide Web since 1990. Most Internet browsers and web servers utilize, and even require, the Hypertext Transfer Protocol for all non-ftp and non-gopher communications.

HTTP was originally defined as a protocol to handle “raw data transfers” over the Internet. Version 0.9 did not allow for anything more than the data itself to be sent. HTTP/1.0 was introduced to allow for metadata, data that describes data, to be passed along with the request and/or response. Thus if a request involved a specific type of application request, that information could be included within the HTTP request. HTTP/1.1 was introduced to require strict compliance to the HTTP standards. This was necessary as many applications that “advertised” themselves as HTTP/1.0 compliant, truly were not, thus causing some incompatibility.

HTTP resides at the application layer of the OSI model. It is a stateless protocol in that each command is executed without any input from, or “knowledge” of, previous commands. On the other hand, HTTP/1.1 supports persistent connections, allowing a series of file transfers to take place between a client and server over the same connection.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The client-server model was kept in mind when HTTP was built. Clients send a request message to an HTTP server, and the HTTP server sends back a response message. The message can include any number of HTTP resources, including query results, server side output, html files, and so forth. The HTTP request and response are very similar in structure, with some minor differences.

An HTTP message will be composed of 3 basic "components". These components are:

1. Initial line identifying the message as a request or response.
2. Information about the request/response, and/or the object of the request/response.
3. Message body

Initial Request/Response Line - Request

The initial line of an HTTP request message is composed of 3 pieces of data:

1. The method of the request
2. The path to the resource being requested
3. The version of HTTP being used

Here is an example of an initial request line:

GET /wwwroot/security/default.asp HTTP/1.1

The method of the request can be a GET, POST, or HEAD. The GET method asks the server to supply the requested resource to the client. The POST method asks the server to deliver the message body to the specific resource to be processed in some manner. For example: the request could be to have html form data delivered to a CGI script for processing.

The HEAD method is nearly identical to the GET method, except that it only asks the server to return the header information for the resource requested, rather than the actual resource. In addition, the HTTP version will be a HTTP/0.9, HTTP/1.0, or HTTP/1.1.

Initial Request/Response Line – Response

The initial response line is also known as the "status line". This line, like an initial request line, is composed of 3 pieces of data:

1. The version of HTTP being used
2. The response status code
3. The reason phrase

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Below is an example of an initial request line:

HTTP/1.0 404 Not Found

As with the initial request line, the version of HTTP will be HTTP/0.9, HTTP/1.0, or HTTP/1.1. The response status code will be a three-digit number. The first of the three digits represents the type of response that is being sent. The second and third digits will vary. The types of responses include:

- 1XX = Informational Message
- 2XX = Success Message
- 3XX = URL Redirect
- 4XX = Client Error (such as requesting a page that does not exist.)
- 5XX = Server Error (script failure, page not found that should be there, etc)

[Marshall, James. "HTTP Made Really Easy." HTTP Made Really Easy, A Practical Guide to Writing Clients and Servers. 15 Aug. 1997. URL: HTTP://www.jmarshall.com/easy/HTTP/#whatis (13 Dec. 2001)]

The reason codes will be a human-readable equivalent of the status codes. For example, a 404 status code will usually have a "File Not Found" reason phrase.

Message Information

The message information section of the HTTP message is also called a header line. There may be more than one header line, depending on the amount of information being sent. The format of header lines follows the RFC 822 standards. Thus each header line will look like:

"HEADER_NAME: VALUE" <CRLF>

The <CRLF> equates to a Carriage Return Line Feed.

Headers are custom definable. For example, a header may include information about the type, and version, of web server that is responding to an HTTP request. The header may include information about the page that was requested, such as date last modified. For requests, the header may include information about the requestor. For responses, there may be a header that provides information about the message body or resource being returned.

Below are some sample headers.

Program information header:

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Server: Apache/1.3.9 Ben-SSL/1.37

Content information header:

Content-Type: text/html

User information header:

From: mcward@ilstu.edu

Message Body

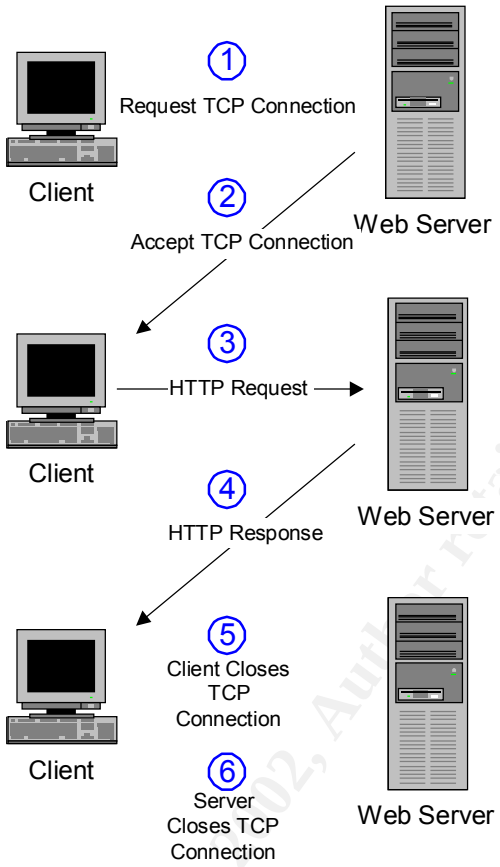
The content of the message body will vary depending on the type of HTTP message being sent. For HTTP requests, the message body may contain user form data to be processed by a server side script, or it may contain files that are to be uploaded to the server, such as on personal web page sites.

For an HTTP response, the message body may contain more detailed information about the response, an html page, or a file that is to be downloaded.

Example HTTP Client-Server Exchange:
Figure 3 below shows the general interaction between client and server in a simple HTTP exchange.

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



(Figure 3)

Below is an example of an actual client-server file exchange. In this exchange, the client requests a particular resource (htm page), and the server will respond with file not found. The client then modifies the request and resends it. The server then responds with the requested resource.

1. The client initiates a TCP connection to the server.

2. The server accepts the connection.

3. The client requests an htm page (HTTP://www.sans.org/infosecFAQ/win/win_lisp.htm) from the server. The request is:

   GET /infosecFAQ/win/win_lisp.htm HTTP/1.1
   From: mvbaile@ilstu.edu
   User-Agent: Mozilla/3.0Gold
   <blank line>

4. Since the file win_lisp.htm (should be win_list.htm) does not exist on the server, the server sends an HTTP response back to the client. The response is:

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
HTTP/1.1 404 Not Found
Date: Sun, 15 Dec 2001 13:42:01 GMT
Server: Apache/1.3.9
Content-type: text/html

<HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD>
<BODY>
<H1>Not Found</H1>
The requested URL /infosecFAQ/win/win_lisp.htm was not found on this
server.

Additionally, a 404 Not Found error was encountered while trying to use
an Error Document to handle the request.
</BODY>
```

5. The user edits the URL in the browser search bar, and resends the request:

```
GET /infosecFAQ/win/win_list.htm HTTP/1.1
From: mvbaile@ilstu.edu
User-Agent: Mozilla/3.0Gold
<blank line>
```

6. The server recognizes the path as a valid page and responds with the requested resource:

```
HTTP/1.0 200 OK
Date: Sun, 15 Dec 2001 13:44:33 GMT
Content-Type: text/html
Content-Length: 5372

<HEAD>
<TITLE>SANS Reading Room: Windows Issues</TITLE>
</HEAD>
<BODY>
(PAGE SOURCE HERE)
</BODY>
```

7. Client and server close the TCP connection.

Hardware and Software that Utilize Port 80/HTTP

Although there is no single service that is always associated with port 80, most web server applications listen on port 80 for HTTP requests. Included

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

below is a list of software and hardware that either listen on port 80 by default, or they are commonly configured to do so:

- 4D WebSTAR
- Alibaba
- AOLserver
- Apache
- Avenida
- BadBlue
- Domino Go Webserver
- DSL Routers
- Enterprise for NetWare
- First Class
- GoAhead
- Hewlitt Packard Print Servers
- iPlanet Web Server Enterprise Edition
- iServer
- Internet Explorer
- Jigsaw
- Load Balancing Switches/Routers
- Lotus Domino
- Microsoft IIS v4.0 and v5.0
- Microsoft PWS
- Microsoft Site Server
- Mozilla based browsers
- NCSA HTTPd
- Netscape
- Netscape Enterprise
- Netscape FastTrack
- OmniHTTPd Pro
- Opera
- RapidSite
- Roxen Challenger
- Roxen Web Server
- Sambar Server
- Savant
- Servertec Internet Server
- SimpleServer:WWW
- Stronghold
- Tcl Web Server
- URL Live!
- vqServer
- Web Commander
- WebSite Pro

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

- WebSite Standard
- Xitami
- ZBServer
- Zeus Web Server

The most common web servers utilized include Microsoft's IIS4 and IIS5, Apache web server, Netscape, and iPlanet. The most common web browsers are Netscape and Internet Explorer. It is also important to note that most web servers can be assigned a "listen on" port other than 80 if desired.

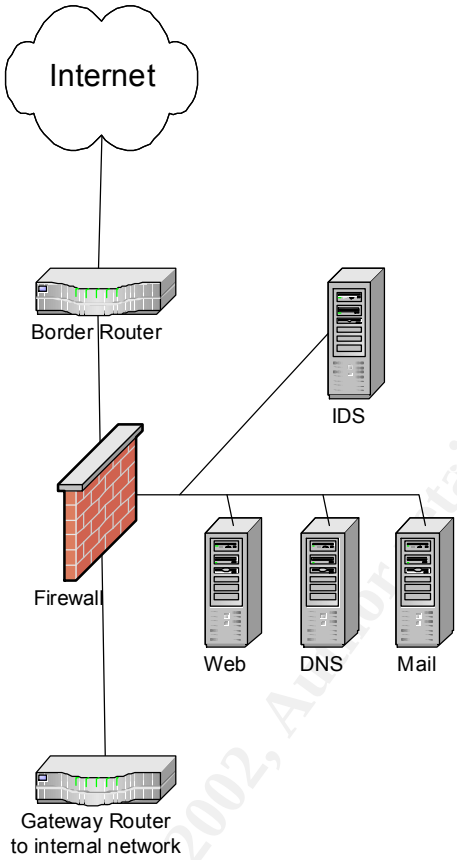
Security Concerns with Port 80 and HTTP

Port 80 remains number one on the DShield list of top ten targeted ports because it is commonly permitted by perimeter security devices and applications, both inbound and outbound. Inbound http traffic over port 80 is commonly permitted to a company's public web servers in their demilitarized zone (DMZ). Outbound http traffic is commonly allowed to provide employees a means of performing activities, such as internet research on sites like Cisco and CNN Financial, to monitor competitor sites, and to conduct business with vendors.

The diagram below (Figure 4) depicts a common DMZ, and some of the common perimeter security implemented to protect the DMZ servers from attacks.

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



(Figure 4)

Border routers are commonly configured as rudimentary firewalls. The border routers may have acl's (access control lists) applied to do some initial packet filtering before the internet traffic reaches the firewall. This helps to reduce a significant amount of "garbage" traffic to the firewall. The acl's will typically allow http traffic inbound (if the company hosts one or more web servers) and outbound for internal users to surf the internet.

For inbound traffic, firewalls will then perform more advanced filtering of traffic to drop specific types of malicious requests, and to drop requests to services or hosts which are not intended for public access. The firewalls may be configured to prevent certain outbound access as well. For example, a firewall may be configured to prevent outbound access for packets with a source address that is not from the internal network, in order to prevent ip spoofing. Proxy firewalls may be used to prevent internal users from accessing specific sites as well.

In spite of all of this, firewalls are typically configured to allow most inbound and outbound http traffic. Depending on the complexity of the firewall, (i.e., packet filtering, stateful inspection, etc), the payload of the http traffic's packets may not be inspected by the firewall at all. This is

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

one of the core issues in attackers utilizing http traffic to defeat perimeter security.

Once past the border router and firewall(s), the traffic will reach the actual destination server, most commonly a web server of some type. Depending on the operating system of the server, and the utilities and applications used in securing the server, many servers will have host level port filtering enabled. Web servers will commonly deny most ports except the absolutely necessary, such as port 80 (http) and 443 (https).

In our example DMZ, the Intrusion Detection System (IDS) server is attached to the service network segment. Intrusion Detection Systems are often deployed in highly secured environments such as DMZs to sniff network traffic, and examine the traffic for specific types of attacks. The weakness in Intrusion Detection Systems is not unlike virus scanners. They can only prevent against known attacks for which a "signature" has been developed. Some IDS systems, like firewalls, are packet filtering and have no concept of state. Attacks composed of properly fragmented packets are often successful at defeating IDS systems. We will examine in more detail how attackers are defeating IDS systems with http traffic as well.

Penetration Attacks from Outside the Perimeter

As mentioned in the previous section, many perimeter security defense methods ignore http traffic. Although this is sometimes out of ignorance, other times it is due to lack of budget. Stateful inspection firewalls and stateful inspection intrusion detection systems are more costly than standard packet filtering firewalls. Performance can also be an issue. For extremely high performance environments, stateful inspection can be too costly in terms of network bandwidth and cpu utilization.

Because HTTP is commonly permitted, and rarely inspected, it is becoming a very common protocol by which attackers hide their activity. This includes both active attacks by cracker tools and also malicious software such as viruses and worms. In attempting to breach perimeter security from outside the network, attackers may use worms (self-propagating malicious code). Prior to there being any IDS or virus detection signatures available, both Code Red and NIMDA performed their attacks over port 80 using the http protocol. The router acl'ing, firewall rules, and server local port filtering saw no harm in letting Code Red and NIMDA through to the web server software, because it was over port 80 and was comprised of http traffic. IDS systems and virus scanners ignored Code Red and NIMDA, because there were no signatures available for the attacks.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Some attackers will craft a URL (sent via an http request) to attempt to exploit vulnerabilities in a web server's handling, or mishandling, of http requests. They may use the malformed URL to attempt to access specific directories of the web server, such as the root folder, or a folder containing scripts that will allow the attacker to execute commands with root, administrator, or SYSTEM privileges. An example of this would be an attacker attempting to access the Microsoft IIS SCRIPTS directory that is installed by default. This directory contains vbscript scripts that may allow the user to perform undesired actions on the server. The actions cannot only affect the web server, but the underlying operating system as well.

There are a number of common malformed URLs that attackers may attempt to deliver to a web server. Some examples, and descriptions, are included below:

The following HTTP request has a URL that attempts to launch cmd.exe, and execute a directory listing of the C: drive. Cmd.exe is the command shell used by Windows systems. Within the URL, the attacker attempts to traverse up two directories, note the ../../, then to the WINNT/system32 directory where cmd.exe is stored. If the attacker is able to launch cmd.exe from the URL, he will be able to execute any number of system commands, including deletion of files and folders, creating, deleting, or modifying users and groups, and setting permissions. If the Windows NT or 2000 Resource Kit is installed on the system, a great deal more damage can be done.

HTTP://host/scripts/something.asp=../../WINNT/system32/**cmd.exe**?dir+c:\

The following HTTP request has a URL by which the attacker attempts to execute a "ps –aux" command on a Unix (or Unix variant) web server. The attacker attempts to traverse the directory structure, note the ../../../../, to reach the /bin folder where the ps binary resides. The ps command will list the processes running on the system, thus granting the attacker knowledge of what other vulnerable processes are running on the system, and possibly what trojan horse software may be running on the system.

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

[Zenomorph. "Fingerprinting Port 80 Attacks." Fingerprinting Port 80 Attacks – A look into web server and web application attack signatures. Nov. 2001. URL: HTTP://packetstorm.decepticons.org/papers/IDS/fingerprint-port80.txt (13 Dec. 2001)]

Although the attacks listed above are known, and have freely available patches and or configuration fixes, they are indicative of the failure of

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

many perimeter security methods to prevent certain attacks executed over http.

Web application attacks are intended to take advantage of flaws in the design of certain web applications, both commercial and “home grown”. Because web application attacks are commonly targeted at home grown web applications, a stateful inspection firewall or IDS will likely not recognize the payload of the http request as being malicious. Some methods of web application attacks are briefly described next.

Database manipulation is conducted by attempting to take advantage of flaws in the underlying dbms of the web application’s backend database. The attacker will attempt to read data from, delete data from, or write data to the database. All actions to which he/she may be unauthorized. The manner in which this is performed includes feeding the web app garbage characters in a user form, feeding the web app various id’s and passwords in a login form in an attempt to gain knowledge of a true id or password (based on the web application’s error return messages), or submitting SQL commands via a user form.

Automation of web application vulnerability exploration has been provided to the security and cracker communities by RainForest Puppy in the Whisker tool. Whisker is an automated cgi script vulnerability scanner that examines a web application’s cgi scripts for hundreds of potential vulnerabilities.

The tool operates over http via port 80 (http) and 443 (https), thereby bypassing perimeter security when firewalls and intrusion detection systems are not designed or updated to recognize the attacks. In fact, Whisker has a number of methods for evading IDS detection. This includes url encoding, extra-long urls, and fragmented url requests. In addition, Whisker was written in Perl and therefore can be run on unix variants and Windows operating systems. Since many script kiddies are running their attacks from Windows 98 desktops, this tool presents them with another weapon in their arsenal.

Hiding Malicious Activity in Outbound HTTP

One of the most common activities an attacker will perform once he/she has compromised a system is to place a “backdoor” or trojan horse program on the compromised system. This backdoor will provide the attacker with a means of easily gaining access to the system in the future, or to easily send commands to the system remotely. Backdoors are also installed on systems via trojan horse software that a user is duped into executing on their system.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

For example, a cracker may create a fun video game which is small in size, and therefore easily transferred via email to friends and family. When a user launches the game, the game plays normally, but in the background it performs nefarious actions on the system that include installing a backdoor to grant the cracker access later. In the cracker's mind, there is hope that this video game will make it's way into a corporate infrastructure where the "fruits" reaped will be great.

One recent trojan that utilizes http as part of it's overall stealth functionality is the Setiri trojan. Setiri was created by the SensePost consulting firm of South Africa, and is a new and improved version of the GatSlag trojan which they introduced last year. Setiri is Windows based and takes advantage of Internet Explorer functionality to control the victim host. Setiri is like other trojans in that it must be installed to be functional. So, either an attacker must somehow compromise the system to the level that he/she can install the script, or the user must be deceived or tricked into installing the trojan.

Once installed, Setiri launches an invisible, or background, instance of Internet Explorer (IE) running under the context of the logged on user. Setiri will direct the instance of IE to Anonymizer.com, (the destination site is configurable), then to another host (Master Controller; also configurable) specified in the Setiri code. Setiri will query the Master Controller for available commands. The commands are commands to be executed on the victim host. The commands can be anything from a directory listing command to a command to install other trojan software. If no commands are available, Setiri will sleep and retry after a period of time. If commands are available, IE brings the commands back to the victim host, and Setiri picks them up to be executed.

Setiri defeats perimeter security in a number of ways. Since Setiri launches IE under the context of the user, all traffic associated with the Setiri traffic will look like standard http requests and responses. As mentioned before, even firewalls that monitor state will typically allow all outbound http requests, and associated responses. Likewise, Intrusion Detection will see the Setiri traffic as standard http traffic from a trusted internal user. Personal firewalls, which most commonly filter applications, rather than filtering packets, will see Internet Explorer as a valid internet accessing application.

Setiri runs under the user's context, however authentication proxy firewalls may be of assistance, given they don't use NTLM authentication. If an authentication proxy is running on a Unix server, the user will be prompted for credentials when Setiri attempts to exit the network. The creators of Setiri are counting on the user to simply provide the credentials, since most users aren't savvy enough to recognize this as an

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

issue. If they are prompted by the computer for something, they typically provide it.

In addition, Setiri has the capability to use SSL, thereby defeating stateful inspection firewalls and IDS that would recognize the payload and/or signature once it's created.

Setiri is one example of how an attacker can hide their activity via outbound http traffic, and http sessions. It is but one of many possibilities for subverting perimeter security measures by using permitted protocols or by mimicking normal traffic.

Detecting HTTP Based Attacks

There are various means for detecting attacks that come over http:

- Firewall Logs
- Web Server Logs
- IDS Alerts
- Web Application IDS Logs and Alerts
- Server Local Auditing Logs and Alerts

You can view the firewall logs for your specific firewall to determine which packets were dropped, and what rule was invoked for the drop. The firewall logs will likely only be of value if a stateful inspection firewall is used, and traffic is dropped based on payload content. Logging of banned ip addresses and subnets may also be of informational value in keeping track of problem hosts.

Web Server logs will provide information about what types of requests were made to the server, which ip the request came from, and the response the server provided. Extended logging is available with most web server software. For example, IIS's extended logging can provide the following helpful information on http requests and the server's response:

- Date and Time of Request.
- Client IP Address.
- Username of client – if available.
- The method (POST or GET) that the client used in their http request.
- The resource accessed – html, asp, etc.
- How many bytes were sent by the web server.
- How many bytes were received by the web server.
- The length of time the action took to complete.
- The type of browser the client used.
- Any site that redirected the client to the web server.

Network and application based intrusion detection products, such as ISS' Real Secure, snort, and others; can be configured to generate alerts based on

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

recognized attack "signatures". Local server auditing can also be configured to alert on specific events related to http attacks. For example, launching of cmd.exe by the IUSR_ account would typically be forbidden. If this were to occur, a host auditing or host based IDS product could recognize the forbidden action and generate an alert.

Defending Against HTTP Based Attacks

There are ways to defend against http based attacks, however the common permission of inbound and outbound http traffic makes mitigation of the threat, rather than elimination of the threat, the only real option. In order to thoroughly defend against http based attacks, both software and hardware options are available:

- Stateful Inspection Firewalls
- Stateful Inspection Network IDS
- Web Application IDS
- Router ACL'ing
- Host Based Defense

Both stateful inspection IDS and firewalls have the capability of watching sessions of traffic, rather than just individual packets. Therefore, they may be able to recognize fragmented attacks over http, as well as known malicious http traffic payloads and patterns. In addition, the firewall rules and router acl'ing can be used to drop traffic to known anonymous proxy sites.

Host based defense can include the use of URLScan for IIS based web servers, and host based firewalls can be used on Linux and Solaris servers to drop traffic with known malicious string patterns in the http requests. In addition, some newer web application IDS products, such as Stratum8, are designed to monitor http traffic for questionable http requests.

And finally, defeating http based attacks by trojans such as Setiri requires user education. A good Security Awareness program can make the user base aware of the potentially negative consequences of installing unauthorized software and opening email attachments from unknown/untrusted sources. A "Defense In Depth" posture which uses a combination of the defense methods mentioned in the section will provide the greatest protection against http based attacks.

Summary

Attacks against organizations and networks are commonly executed over http. This is because most organizations allow inbound and outbound http traffic, often unmonitored by any form of advanced perimeter defense. Even with basic perimeter defense in place, http traffic will often travel in and out of the network unfettered. Advanced and rigorous defense methods are required to defend against and detect http based attacks. Defense in depth is the key. Perimeter

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

protection, host based protection, and user awareness can all combine to provide a meaningful defense against http based attacks.

Additional Information

Vulnerability Information:
HTTP://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2708
HTTP://www.cert.org/advisories/CA-2001-12.html

Exploit Information:
HTTP://www.securiteam.com/exploits/5YP0F204AO.html
HTTP://packetstorm.decepticons.org/0105-exploits/execiis.c
HTTP://packetstorm.decepticons.org/0010-exploits/iisex.c

Product Information:
Stratum8: http://www.stratum8.com/index2.shtml
RealSecure:
http://www.iss.net/products_services/enterprise_protection/rsnetwork/index.php
Whisker: http://sourceforge.net/projects/whisker/

Bibliography

Web Sites

Zetter, Kim. “DEF CON: Trojan horse technology exploits IE hole.” PCWorld.com. 06 Aug 2002.
URL: HTTP://www.nwfusion.com/news/2002/0806trojan.html

McDougall, Bonnie. “Personal Firewalls – Protecting the Home Internet User.” Sans Reading Room. 17 Aug 2001.
URL: http://rr.sans.org/firewall/home_user.php

Hawkins, Iain. "The Clickable Mummy." Akhet Egyptology. 10 May 1997.
URL: HTTP://wkweb4.cableiniet.co.uk/iwhawkins/egypt/index.htm (5 Mar. 1998).

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Brain, Marshall. "How Web Servers and the Internet Work" How Stuff Works. URL: HTTP://www.howstuffworks.com/web-server.htm (13 Dec. 2001).

Server Watch. "Web Servers" Server Watch, Web Servers. 12 Dec 2001. URL: HTTP://serverwatch.internet.com/webservers.html (14 Dec. 2001)

Berners-Lee, T. et al. "RFC 2068" RFC 2068. Jan 1997. URL: HTTP://www.w3.org/Protocols/rfc2068/rfc2068 (14 Dec. 2001)

Berners-Lee, T. et al. "RFC 2616" RFC 2616. Jun 1999. URL: HTTP://www.w3.org/Protocols/rfc2616/rfc2616 (13 Dec. 2001)

TechTarget. "Hypertext Transfer Protocol" Hypertext Transfer Protocol. 5 Oct 2000. URL: HTTP://searchsystemsmanagement.techtarget.com/sDefinition/0,,sid20_gci214004,00.html (13 Dec. 2001)

Marshall, James. "HTTP Made Really Easy." HTTP Made Really Easy, A Practical Guide to Writing Clients and Servers. 15 Aug. 1997. URL: HTTP://www.jmarshall.com/easy/HTTP/#whatis (13 Dec. 2001)

ANU.edu. "Browser-Server Interaction Through HTML." Browser-Server Interaction Through HTML Information Structure of Web Sites. 23 Feb. 1999. URL: HTTP://cs.anu.edu.au/Student/infs3056/LectureNotes/ (14 Dec. 2001)

Zenomorph. "Fingerprinting Port 80 Attacks." Fingerprinting Port 80 Attacks – A look into web server and web application attack signatures. Nov. 2001. URL: HTTP://packetstorm.decepticons.org/papers/IDS/fingerprint-port80.txt (13 Dec. 2001)

Books

Fossen, Jason. Securing Internet Information Server 5.0 (Track 5). Reading: SANS Institute, 2001. 43-45.

Internet Security Systems. Windows 2000 Security Technical Reference. Reading: Microsoft Press, 2000. 478 – 480.

Black Hat Briefings. USA 2002 BRIEFINGS
Black Hat. 77-91.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Last Updated: April 24th, 2017

Upcoming Training

Click Here to {Get CERTIFIED!}

SANS Riyadh 2017	Riyadh, Saudi Arabia	May 06, 2017 - May 11, 2017	Live Event
SANS Security West 2017	San Diego, CA	May 09, 2017 - May 18, 2017	Live Event
SANS Northern Virginia - Reston 2017	Reston, VA	May 21, 2017 - May 26, 2017	Live Event
SANS London May 2017	London, United Kingdom	May 22, 2017 - May 27, 2017	Live Event
SANS Melbourne 2017	Melbourne, Australia	May 22, 2017 - May 27, 2017	Live Event
SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Philadelphia SEC401	Philadelphia, PA	Jun 19, 2017 - Jun 24, 2017	Community SANS
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor