An Introduction to HTTP fingerprinting

Saumil Shah

saumil@net-square.com
19th May, 2004

Table of Contents

1. Abstract

HTTP Fingerprinting is a relatively new topic of discussion in the context of application security. One of the biggest challenges of maintaining a high level of network security is to have a complete and accurate inventory of networked assets. Web servers and web applications have now become a part of the scope of a network security assessment exercise. In this paper, we present techniques to identify various types of HTTP servers. We shall discuss some of the problems faced in inventorying HTTP servers and how we can overcome them.

We shall also introduce and describe a tool, **httprint**, which is built using the concepts discussed in this paper.

2. Theory of Fingerprinting

A fingerprint is defined as:

1. The impression of a fingertip on any surface; also: an ink impression of the lines upon the fingertip taken for the purpose of identification.
2. something that identifies: as (a) a trait, trace, or characteristic revealing origin or responsibility (b) analytical evidence that characterizes an object or substance.

The process of fingerprinting can be broken up into two sub processes, namely gathering and classification of fingerprints, and comparision of unknown fingerprints with the stored database of known fingerprints.

While gathering fingerprints, it is essential to capture all the key characteristics of the object revealed in the fingerprint. Capturing more details and traits helps in the comparision process. While comparing fingerprints, there may be

chances that a fingerprint can be improperly matched, because of subtle differences that can be easily mistaken.

Fingerprinting is a known technique in network security. Operating system fingerprinting is a common task in any network assessment or inventorying exercise. There are many techniques to perform operating system fingerprinting. What makes operating system fingerprinting successful and accurate is the fact that each operating system implements the TCP/IP stack slightly differently. The way a system responds to malformed packets, either the presence of an error response, or absence thereof, is one example of an implementation difference. A detailed discussion on operating system fingerprinting, or TCP/IP stack fingerprinting, is presented in Fyodor's paper, titled "Remote OS detection via TCP/IP Stack Fingerprinting" [1]

The theory behind HTTP fingerprinting is more or less on the same lines - identifying HTTP servers by their implementation differences in the HTTP protocol. HTTP fingerprinting gets slightly more complicated than TCP/IP stack fingerprinting. The reason being that it is easily possible to customize the responses of an HTTP server by just changing its configuration file, or adding plug-ins or modules, whereas customising the behaviour of the TCP/IP stack requires access to the network code at the kernel layer. Despite this difficulty, it is possible to devise tests to overcome the various customizable features of a web server.

3. Banner grabbing

The simplest and most basic form of identifying HTTP servers is to look at the Server field in the HTTP response header [2]. Using a TCP client like netcat [3], it is possible to send an HTTP request to return the HTTP response header of the server, as shown below:

```
$ nc 202.41.76.251 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 16 Jun 2003 02:53:29 GMT
Server: Apache/1.3.3 (Unix)  (Red Hat/Linux)
Last-Modified: Wed, 07 Oct 1998 11:18:14 GMT
ETag: "1813-49b-361b4df6"
Accept-Ranges: bytes
Content-Length: 1179
Connection: close
Content-Type: text/html

$
```

Three examples of the HTTP response header are shown below:

From an Apache 1.3.23 server:

```
HTTP/1.1 200 OK
Date: Sun, 15 Jun 2003 17:10:49 GMT
Server: Apache/1.3.23
Last-Modified: Thu, 27 Feb 2003 03:48:19 GMT
ETag: "32417-c4-3e5d8a83"
Accept-Ranges: bytes
Content-Length: 196
Connection: close
Content-Type: text/html
```

From a Microsoft IIS 5.0 server:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Expires: Tue, 17 Jun 2003 01:41:33 GMT
Date: Mon, 16 Jun 2003 01:41:33 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 28 May 2003 15:32:21 GMT
ETag: "b0aac0542e25c31:89d"
Content-Length: 7369
```

From a Netscape Enterprise 4.1 server:

```
HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
```

```
Date: Mon, 16 Jun 2003 06:19:04 GMT
Content-type: text/html
Last-modified: Wed, 31 Jul 2002 15:37:56 GMT
Content-length: 57
Accept-ranges: bytes
Connection: close
```

4. Applications of HTTP fingerprinting

From a network management standpoint, HTTP fingerprinting comes in very handy when keeping track of the various web servers on a network. HTTP fingerprinting can also be used to automate information systems and security audits. Automated security testing tools can use HTTP fingerprinting to narrow down the set of tests required, based on the specific platform or the specific web server being audited.

Some of the applications of HTTP fingerprinting are:

- Network management: Web server inventory
- Penetration testing / Auditing: Selecting the right attacks or audit tests for web servers
- Wireless networks: Detecting 802.11x access points from the wired network, since most APs have a web enabled interface
- Web enabled devices: Printers, storage servers, switches, etc. Many times, web enabled devices do not return a server banner string at all, making it difficult to identify and track them.

5. Obfuscating the server banner string

Banner grabbing proves to be a good method of HTTP fingerprinting in many cases. However, many times, server administrators chose to disguise the server banner string, by providing one of their own. Such, "security-by-obscurity" methods help thwart a lot of automated attacks against web servers.

It is easy to configure servers to return different server banner strings. In open source servers such as Apache, one can change the banner string in the source code and re-compile the server. For non-open source servers such as Microsoft IIS or Netscape Enterprise, it is possible to "patch" the binary by opening it up in a hex editor and changing the string embedded in the binary. It is not so easy to do this always, but it has been done successfully. Another way of obscuring the server banner string is to write a custom plug-in for the web server, which can provide customized HTTP responses. A commercial product, called ServerMask [4] from Port 80 Software performs such obfuscation on HTTP responses.

The example below shows the response from an HTTP server with a customized server banner string:

Apache Server recompiled with "Unknown-Webserver/1.0" as the server banner string

```
HTTP/1.1 403 Forbidden
Date: Mon, 16 Jun 2003 02:41:27 GMT
Server: Unknown-Webserver/1.0
Connection: close
Content-Type: text/html;
charset=iso-8859-1
```

The example below shows the response from an HTTP server using ServerMask:

IIS Server using the ServerMask plug-in

```
HTTP/1.1 200 OK
Server: Yes we are using ServerMask
Date: Mon, 16 Jun 2003 02:54:17 GMT
Connection: Keep-Alive
Content-Length: 18273
Content-Type: text/html
Set-Cookie: It works on cookies too=82.3S3.O12.NT2R0RE,4147ON3P,.4OO.;
path=/
Cache-control: private
```

As we can see from the above examples, relying purely upon the contents of the server banner string is not enough for identifying the type of HTTP server.

6. Protocol behaviour

Almost all HTTP servers differ in the way they implement the HTTP protocol. In the case where the HTTP request is well formed and legitimate, the response returned by all HTTP servers is more or less compliant with the specifications laid out in the RFCs for HTTP[5]. However, when confronted with malformed HTTP requests, these servers differ in their responses. Differences in the way the HTTP protocol is handled by various HTTP servers form the basis of the HTTP fingerprinting technique.

Let us illustrate these differences with examples. We shall analyze the response to four HTTP requests, coming from an Apache 1.3.23 server, a Microsoft IIS 5.0 server and a Netscape Enterprise 4.1. The requests are:

HTTP Test	What to expect
`HEAD / HTTP/1.0`	Normal HTTP header response
`DELETE / HTTP/1.0`	Response when operations such as DELETE aren't generally allowed
`GET / HTTP/3.0`	Response to a request with an improper HTTP protocol number
`GET / JUNK/1.0`	Response to a request with an improper protocol specification

In each of these responses, we shall identify key differences between the responses of Apache 1.3.23, IIS 5.0 and Netscape Enterprise 4.1. We shall not take into consideration differences in customizable parameters such as the server banner string.

6.1 HTTP header field ordering

Taking the first request **HEAD / HTTP/1.0**, we shall analyze the HTTP response header and inspect the order of appearance of the various fields returned within it.

Response from Apache 1.3.23

```
$ nc apache.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 15 Jun 2003 17:10:49 GMT
Server: Apache/1.3.23
Last-Modified: Thu, 27 Feb 2003 03:48:19 GMT
ETag: "32417-c4-3e5d8a83"
Accept-Ranges: bytes
Content-Length: 196
Connection: close
Content-Type: text/html
```

Response from IIS 5.0

```
$ nc iis.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://iis.example.com/Default.htm
Date: Fri, 01 Jan 1999 20:13:52 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 01 Jan 1999 20:13:52 GMT
ETag: W/"e0d362a4c335be1:ae1"
Content-Length: 133
```

Response from Netscape Enterprise 4.1

```
$ nc netscape.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
Date: Mon, 16 Jun 2003 06:01:40 GMT
Content-type: text/html
Last-modified: Wed, 31 Jul 2002 15:37:56 GMT
Content-length: 57
```

```
Accept-ranges: bytes
Connection: close
```

If we observe the ordering of the response header fields **Server** and **Date**, we notice that Apache orders the fields differently than IIS and Netscape.

6.2 HTTP DELETE (forbidden operation) response

Next, we shall take the request **DELETE / HTTP/1.0** and observe what the response of each of the servers is, when the requested operation is generally forbidden.

Response from Apache 1.3.23

```
$ nc apache.example.com 80
DELETE / HTTP/1.0

HTTP/1.1 405 Method Not Allowed
Date: Sun, 15 Jun 2003 17:11:37 GMT
Server: Apache/1.3.23
Allow: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH,
 MKCOL, COPY, MOVE, LOCK, UNLOCK, TRACE
Connection: close
Content-Type: text/html;
charset=iso-8859-1
```

Response from IIS 5.0

```
$ nc iis.example.com 80
DELETE / HTTP/1.0

HTTP/1.1 403 Forbidden
Server: Microsoft-IIS/5.0
Date: Fri, 01 Jan 1999 20:13:57 GMT
Content-Type: text/html
Content-Length: 3184
```

Response from Netscape Enterprise 4.1

```
$ nc netscape.example.com 80
DELETE / HTTP/1.0

HTTP/1.1 401 Unauthorized
Server: Netscape-Enterprise/4.1
Date: Mon, 16 Jun 2003 06:03:18 GMT
WWW-authenticate: Basic realm="WebServer Server"
Content-length: 223
Content-type: text/html
Connection: close
```

Apache responds with a **405** "Method not allowed" response, IIS responds with a **403** "Operation on resource forbidden" response, and Netscape responds with a **401** "Authorization credentials required" response. Each of the servers differs in their response to the **DELETE** request.

6.3 Improper HTTP version response

The next test consists of sending an HTTP request with an improper HTTP version number, such as **GET / HTTP/3.0**, to the server. HTTP 3.0 is not even in existence as of this writing, and none of the candidate servers implement it.

Response from Apache 1.3.23

```
$ nc apache.example.com 80
GET / HTTP/3.0

HTTP/1.1 400 Bad Request
Date: Sun, 15 Jun 2003 17:12:37 GMT
Server: Apache/1.3.23
Connection: close
Transfer-Encoding: chunked
```

```
Content-Type: text/html;
charset=iso-8859-1
```

Response from IIS 5.0

```
$ nc iis.example.com 80
GET / HTTP/3.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://iis.example.com/Default.htm
Date: Fri, 01 Jan 1999 20:14:02 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 01 Jan 1999 20:14:02 GMT
ETag: W/"e0d362a4c335be1:ae1"
Content-Length: 133
```

Response from Netscape Enterprise 4.1

```
$ nc netscape.example.com 80

GET / HTTP/3.0
HTTP/1.1 505 HTTP Version Not Supported
Server: Netscape-Enterprise/4.1
Date: Mon, 16 Jun 2003 06:04:04 GMT
Content-length: 140
Content-type: text/html
Connection: close
```

Apache responds with a **400** "Bad HTTP request" response, IIS ignores the improper HTTP protocol number, and responds with a **200** "OK" along with the contents of the HTML data for the root document, and Netscape responds with a **505** "HTTP version not supported" response.

6.4 Improper protocol response

The next test involves observing the response to the request **GET / JUNK/1.0**.

Response from Apache 1.3.23

```
$ nc apache.example.com 80
GET / JUNK/1.0

HTTP/1.1 200 OK
Date: Sun, 15 Jun 2003 17:17:47 GMT
Server: Apache/1.3.23
Last-Modified: Thu, 27 Feb 2003 03:48:19 GMT
ETag: "32417-c4-3e5d8a83"
Accept-Ranges: bytes
Content-Length: 196
Connection: close
Content-Type: text/html
```

Response from IIS 5.0

```
$ nc iis.example.com 80
GET / JUNK/1.0

HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Fri, 01 Jan 1999 20:14:34 GMT
Content-Type: text/html
Content-Length: 87
```

Response from Netscape Enterprise 4.1

```
$ nc netscape.example.com 80
GET / JUNK/1.0
```

```
<HTML><HEAD><TITLE>Bad request</TITLE></HEAD>
<BODY><H1>Bad request</H1>
Your browser sent a query this server could not understand.
</BODY></HTML>
```

In this case, Apache ignores the improper protocol "JUNK", and responds with a **200** "OK" along with the contents of the root document, IIS responds with a **400** "Bad Request" response and Netscape does not even return an HTTP response header, but instead just returns an HTML formatted error message stating that this request is a bad request.

6.5 Summary of test results

The following table summarizes the various tests and the responses from each of the HTTP servers. It is easy to figure out how to distinguish HTTP servers from such tests.

Server	Field Ordering	DELETE Method	Improper HTTP version	Improper protocol
Apache/1.3.23	Date, Server	405	400	200
Microsoft-IIS/5.0	Server, Date	403	200	400
Netscape-Enterprise/4.1	Server, Date	401	505	no header

6.6 Choosing the right tests

In the above example, we discussed four HTTP tests, and observed the differences in the responses from three popular HTTP servers. For an industrial strength HTTP fingerprinting engine, we require more than four HTTP tests. The larger the number of HTTP tests, the better the results of fingerprinting, and the greater the accuracy in matching fingerprints. On the other hand, fewer HTTP tests imply faster execution time.

Fingerprinting tests are of two types

- Decision tree bases tests
- Statistical analysis tests

Decision tree based tests rely on the construction of a tree of tests, which eliminates possibilities progressively, much in the same manner as testing for an unknown chemical compound, based on the results of progressive chemical reaction tests. Decision tree based tests are difficult to scale, and each HTTP server would have specific contributions to the construction of the decision tree. Adding tests for a new HTTP server would involve re-writing the entire decision tree.

Statistical analysis tests usually involve a fixed set of tests, which result in an array of weights being generated for each type of HTTP server. The decision as to what the outcome is, is based on comparing the various weights generated for each server. The accuracy of statistical analysis tests depends on the algorithms used to assign and compare the weights for each HTTP server. Statistical based models yield themselves quite easily to be adapted into neural machines, which can be trained with a set of known values.

7. Statistical and Fuzzy analysis

The rest of the paper focusses on using statistical and fuzzy logic techniques in analysing the responses from the HTTP tests. The technique consists in perfomrming signature analysis with a set of stored signatures, and assigning a confidence rating to each candidate signature. The signatures with the highest confidence rating are then reported as potential matches for the unknown server being tested.

7.1 Assumptions

The fingerprinting engine operates with a known set of server signatures. It can therefore only identify HTTP servers that it knows about. If a server's signature is not present in the set of known signatures, the fingerprinting engine shall report the next closest server, in terms of server behaviour and characteristics.

While performing HTTP fingerprint tests, there shall be no HTTP proxy server present between the system running the fingerprinting engine and the target web server.

7.2 Terms and Definitions

Term	Description
Signature Set	$S = \{s_1, s_2, ..., s_n\}$ n = number of web server signatures known to the fingerprinting engine.

	s_i = i^{th} signature in the signature set.
Reported Signature	s_R The signature that is derived from running HTTP fingerprinting tests against an unknown web server.
Comparision function and Weight	$w_i = fw(s_R, s_i)$ w_i = Weight when reported signature s_R is compared against the i^{th} signature in the signature set S. $fw(s_a, s_b)$ = Comparision function, which compares signature s_a against s_b, and returns a resultant weight.
Weight Vector	$W = \{w_1, w_2, ..., w_n\}$
Confidence Rating	$c_i = fc(w_i, W)$ c_i = likelihood that signature s_i, with weight w_i is the best match amongst the entire signature set S, whose weight vector is represented by W. $fc(w_a, W)$ = Fuzzy logic function to calculate the likelihood, in percentage terms, of w_a being the best weight amongst the weight vector W.
Confidence Vector	$C = \{c_1, c_2, ..., c_n\}$
Max Confidence	c_{max} Maximum confidence rating amongst all the confidence ratings in the confidence vector C.
Best match Vector	$M = \{s_{maxA}, s_{maxB}, ...\}$ s_{maxA}, s_{maxB} = signatures whose confidence ratings equal to c_{max}

7.3 Analysis Logic

The following piece of pseudo code outlines how the best match is chosen out of the signature set.

1. Load the signature set $S = \{s_1, ..., s_n\}$
2. Run the fingerprinting tests and obtain s_R for the unknown web server.
3. for i = 1 .. n
   $w_i = fw(s_R, s_i)$
   next
4. for i = 1 .. n
   $c_i = fc(w_i, W)$
   next
5. $c_{max} = max(C)$
6. M = {}
7. for i = 1 .. n
   if $c_i = c_{max}$ then
   $M = M \cup \{s_i\}$
   end if
   next
8. print M

8. httprint - the advanced HTTP fingerprinting engine

The **httprint** [6] fingerprinting engine uses statistical analysis, combined with fuzzy logic techniques, to determine the type of HTTP server.

httprint can be used to both gather as well as analyse signatures generated from HTTP servers.

Although **httprint** is not open source, it is available at no cost for personal, educational and non-commercial use.

8.1 httprint signatures

HTTP signatures generated by **httprint** are hexadecimal encoded ASCII strings, as the ones shown below:

```
Microsoft-IIS/5.0
CD2698FD6ED3C295E4B1653082C10D64050C5D2594DF1BD04276E4BB811C9DC5
0D7645B5811C9DC52A200B4C9D69031D6014C217811C9DC5811C9DC52655F350
FCCC535BE2CE6923E2CE69232FCD861AE2CE69272576B769E2CE6926CD2698FD
```

```
6ED3C295E2CE692009DB9B3E811C9DC5811C9DC56ED3C2956ED3C295E2CE6923
6ED3C2956ED3C295811C9DC5E2CE69276ED3C295

Apache/2.0.x
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C11DDC7D7811C9DC5811C9DC58A91CF57
FCCC535B6ED3C295FCCC535B811C9DC5E2CE6927050C5D336ED3C2959E431BC8
6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923
```

httprint maintains a set of signatures in a text file [7], and uses these to analyse the results generated from an unknown server. It is easily possible to extend the signatures database, by simply cutting-and-pasting the signature output of **httprint**, when used against a known server whose fingerprint is not in the database. The next time **httprint** is run, the newly added signature will be used in the comparision.

8.2 httprint command line and GUI interfaces

httprint is available in both command-line and GUI versions, running on Windows, Linux, Mac OS X and FreeBSD for this release - v200.

httprint's command line options are as under:

```
# ./httprint
httprint v0.200 (beta) - web server fingerprinting tool
(c) 2003, net square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

Usage:

httprint {-h <host>
| -i <input file>
-s <signatures>
[... options]

-h <host>
           host can be either an IP address, a symbolic name,
                     an IP range or a URL.
-i <input text file>
file containing list of hosts as described above
                     in text format.
-x <nmap xml file>
  Nmap -oX option generated xml file as input file.
                     Ports which can be considered as http ports are taken
                     from the nmapportlist.txt file.
-s <signatures>
     file containing http fingerprint signatures.

Options:

-o <output file>
    Default output file is "httprintoutput.html". Use this
                     option to override the output filename.
-oc <output file>
   output in csv format
-ox <output file>
   output in xml format
-tp <ping timeout>
  Ping timeout in milliseconds.
                     Default is 1000 ms. Maximum 30000 ms.
-t <timeout>
        Connection/read timeout in milliseconds.
                     Default is 10000 ms. Maximum 100000 ms.
-r <retry>
         Number of retries. Default is 3. Maximum 30.
-P0                  Turn ping off.
-?                   Displays this message.

Examples:
```

```
httprint -h www1.example.com -s signatures.txt
httprint -h https://www2.example.com/ -s signatures.txt
httprint -h http://www3.example.com:8080/ -s signatures.txt
httprint -h 10.0.1.1-10.0.1.254 -s signatures.txt -o 10_0_1_x.html
httprint -x nmap.xml -s signatures.txt -oc report.csv
httprint -x nmap.xml -s signatures.txt -ox report.xml
httprint -i input.txt -s signatures.txt -o output.html
```

The options are self-explanatory.

For the Win32 platform, **httprint** is also available in a GUI interface. The screenshot of the GUI version is shown below:

8.3 Running httprint

An example of the output generated by **httprint** is shown below:

```
# ./httprint -s signatures.txt -o apache1.html -h apache.example.com
httprint v0.200 (beta) - web server fingerprinting tool
(c) 2003, net square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

---------------------------------------------------
Finger Printing on http://apache.example.com:80/
Derived Signature:
Apache-AdvancedExtranetServer/2.0.44 (Mandrake Linux/11mdk) mod_perl/1.99_08
 Perl/v5.8.0 mod_ssl/2.0.44 OpenSSL/0.9.7a PHP/4.3.1
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5811C9DC511DDC7D7811C9DC5811C9DC58A91CF57
FCCC535B6ED3C295FCCC535B811C9DC56ED3C295050C5D336ED3C2959E431BC8
6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC56ED3C295E2CE6923

Banner Reported: Apache-AdvancedExtranetServer/2.0.44 (Mandrake Linux/11mdk)
 mod_perl/1.99_08 Perl/v5.8.0 mod_ssl/2.0.44 OpenSSL/0.9.7a PHP/4.3.1
Banner Deduced: Apache/2.0.x
Scores:
Apache/2.0.x: 126 81.29
Apache/1.3.[4-24]: 118 64.73
Apache/1.3.27: 117 62.83
Apache/1.3.26: 116 60.96
Apache/1.2.6: 113 55.59
Apache/1.3.[1-3]: 113 55.59
Stronghold/4.0-Apache/1.3.x: 66  6.89
Netscape-Enterprise/4.1: 59  4.07
Com21 Cable Modem: 56  3.11
Oracle Servlet Engine: 55  2.82
Microsoft-IIS/5.0 ASP.NET: 55  2.82
Microsoft-IIS/5.1: 55  2.82
Microsoft-IIS/6.0: 55  2.82
Lotus-Domino/6.x: 51  1.81
Netscape-Enterprise/3.6 SP2: 50  1.60
EMWHTTPD/1.0: 50  1.60
dwhttpd (Sun Answerbook): 49  1.39
Netscape-Enterprise/6.0: 49  1.39
thttpd: 48  1.20
Netscape-Enterprise/3.5.1G: 46  0.85
Microsoft-IIS/4.0: 45  0.70
Microsoft-IIS/5.0: 45  0.70
Zeus/4.0: 26  0.53
Zeus/4.1: 25  0.52
Xerver_v3: 25  0.52
CompaqHTTPServer-SSL/4.2: 23  0.50
Orion/2.0x: 23  0.50
AOLserver/3.4.2-3.5.1: 23  0.50
```

```
Jana Server/1.45: 23  0.50
Netscape-Enterprise/3.6: 20  0.45
Microsoft-IIS/URLScan: 20  0.45
NetWare-Enterprise-Web-Server/5.1: 20  0.45
HP-ChaiServer/3.0: 32  0.43
Oracle XML DB/Oracle9i: 17  0.38
Hewlett Packard xjet: 16  0.35
Domino-Go-Webserver/4.6.2.8: 35  0.30
Linksys AP2: 14  0.29
CompaqHTTPServer/1.0: 36  0.24
Zeus/4_2: 36  0.24
Netscape-Enterprise/3.5.1: 36  0.24
Stronghold/2.4.2-Apache/1.3.x: 36  0.24
TightVNC: 36  0.24
SunONE WebServer 6.0: 12  0.23
Netscape-Enterprise/4.1: 12  0.23
Lotus-Domino/5.x: 11  0.20
Cisco-HTTP: 11  0.20
MiniServ/0.01 Webmin: 37  0.18
fnord: 10  0.17
WebLogic Server 8.1: 10  0.17
RemotelyAnywhere: 10  0.17
WebLogic Server 8.x: 10  0.17
3Com/v1.0: 10  0.17
CompaqHTTPServer/4.2: 40  0.08
Snap Appliances, Inc./3.x: 1  0.00
Linksys Router: 0  0.00
Linksys AP1: 0  0.00
EHTTP/1.1: 0  0.00

---------------------------------------------------
Dumping results in html file..
```

In the above example, **httprint** first displays the signature it generates from the server "apache.example.com". It then proceeds to compare the signature with those stored in its database, and assigns weights and confidence ratings for every fingerprint. The signature with the highest confidence rating is chosen to be the best match. In this case, it is "Apache/2.0.x" server with a confidence rating of 81.29%. The next best matches are "Apache/1.3.[4-24]" with a confidence rating of 64.73% and "Apache/1.3.27" with a confidence rating of 62.63%.

8.4 The significance of confidence ratings

We may ask ourselves, why do we need confidence ratings? Picking the highest weight alone may seem to suffice in choosing the best match for the web server. The significance of confidence ratings can be best illustrated by an example. Let us assume that there are no signatures for any version of Apache present in the signature set. Therefore, if we run **httprint** against an Apache server, it will never be able to identify the Apache server. Instead, it will try and pick out the closest approximation to Apache, in terms of behaviour and characteristics, from the signature set.

Given below is the output of **httprint** running against "apache.example.org" (as shown in section 8.3), but this time, without any Apache signatures present in its signature set.

```
# ./httprint -s reduced_signatures.txt -o apache2.html -h apache.example.com
httprint v0.200 (beta) - web server fingerprinting tool
(c) 2003, net square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

---------------------------------------------------
Finger Printing on http://apache.example.com:80/
Derived Signature:
Apache-AdvancedExtranetServer/2.0.44 (Mandrake Linux/11mdk) mod_perl/1.99_08
 Perl/v5.8.0 mod_ssl/2.0.44 OpenSSL/0.9.7a PHP/4.3.1
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5811C9DC511DDC7D7811C9DC5811C9DC58A91CF57
FCCC535B6ED3C295FCCC535B811C9DC56ED3C295050C5D336ED3C2959E431BC8
6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC56ED3C295E2CE6923

Banner Reported: Apache-AdvancedExtranetServer/2.0.44 (Mandrake Linux/11mdk)
 mod_perl/1.99_08 Perl/v5.8.0 mod_ssl/2.0.44 OpenSSL/0.9.7a PHP/4.3.1
```

```
Banner Deduced: Netscape-Enterprise/4.1
Scores:
Netscape-Enterprise/4.1: 59 38.06
Com21 Cable Modem: 56 30.85
Microsoft-IIS/6.0: 55 28.65
Microsoft-IIS/5.1: 55 28.65
Microsoft-IIS/5.0 ASP.NET: 55 28.65
Oracle Servlet Engine: 55 28.65
Lotus-Domino/6.x: 51 20.82
Netscape-Enterprise/3.6 SP2: 50 19.10
EMWHTTPD/1.0: 50 19.10
dwhttpd (Sun Answerbook): 49 17.46
Netscape-Enterprise/6.0: 49 17.46
thttpd: 48 15.91
Netscape-Enterprise/3.5.1G: 46 13.06
Microsoft-IIS/4.0: 45 11.76
Microsoft-IIS/5.0: 45 11.76
CompaqHTTPServer/4.2: 40  6.36
MiniServ/0.01 Webmin: 37  3.94
TightVNC: 36  3.25
Netscape-Enterprise/3.5.1: 36  3.25
Zeus/4_2: 36  3.25
CompaqHTTPServer/1.0: 36  3.25
Domino-Go-Webserver/4.6.2.8: 35  2.63
Netscape-Enterprise/3.6: 20  1.34
NetWare-Enterprise-Web-Server/5.1: 20  1.34
Microsoft-IIS/URLScan: 20  1.34
Oracle XML DB/Oracle9i: 17  1.29
Hewlett Packard xjet: 16  1.23
CompaqHTTPServer-SSL/4.2: 23  1.19
AOLserver/3.4.2-3.5.1: 23  1.19
Jana Server/1.45: 23  1.19
Orion/2.0x: 23  1.19
Linksys AP2: 14  1.09
HP-ChaiServer/3.0: 32  1.07
Xerver_v3: 25  0.95
Zeus/4.1: 25  0.95
SunONE WebServer 6.0: 12  0.90
Netscape-Enterprise/4.1: 12  0.90
Cisco-HTTP: 11  0.80
Lotus-Domino/5.x: 11  0.80
Zeus/4.0: 26  0.78
3Com/v1.0: 10  0.70
fnord: 10  0.70
RemotelyAnywhere: 10  0.70
WebLogic Server 8.x: 10  0.70
WebLogic Server 8.1: 10  0.70
Snap Appliances, Inc./3.x: 1  0.01
Linksys Router: 0  0.00
Linksys AP1: 0  0.00
EHTTP/1.1: 0  0.00

-------------------------------------------------
Dumping results in html file..
```

In this example, we notice that **httprint** has reported the best match to be "Netscape-Enterprise/4.1". However, if we look at the confidence ratings for each signature, we notice that "Netscape-Enterprise/4.1" has a confidence rating of only 38.06%. The other close candidates are "Com21 Cable Modem" with a confidence rating of 30.85% and "Microsoft-IIS/6.0" with a confidence rating of 28.65%. Compare this with the confidence ratings generated when the Apache signatures were present in **httprint**'s signature set in section 8.3. These seem to indicate a much better level of confidence in the best match. Also, the top three matches all belong to the Apache group of servers, which, again goes to re-assure us of **httprint**'s inference.

Looking at this, we can infer that **httprint** has not been effective in picking the best choice out of what it knows from its signature set, and hence, the signature set needs to be updated.

Another tool, HMAP [8], uses a similar approach in sending HTTP tests but it does not perform fuzzy fingerprint comparisions and confidence ratings calculations.

8.5 httprint Reports

httprint, by default, generates reports in HTML format, along with some verbose output results embedded as HTML comments, which may be useful for further analysis. A sample report is shown below:

With version 200, **httprint** can also generate reports in CSV and XML formats (available with the enterprise version only).

8.6 Customising httprint

httprint uses ASCII text files for storing server signatures. It is possible to extend **httprint**'s set of signatures, for covering a wider variety of web servers, by simply running **httprint** against the unknown server, and then including the generated signature in the signatures file. For reporting, it is also possible to associate GIF files having server icons with each signature, which will be then used when generating the HTML report.

9. Trying to defeat HTTP Fingerprinting

The technique of system fingerprinting is not yet as foolproof as human fingerprinting. It is possible to disguise and customize HTTP servers quite sufficiently to ensure that they give unexpected results for all HTTP tests.

The following is a list of some of the techniques that can be used to attempt to defeat HTTP fingerprinting:

- Changing the HTTP server banner string
- Stripping or re-arranging the HTTP headers
- Customising HTTP error codes such as 404 or 500
- Using an HTTP server plug-in

Out of the above techniques, the first three techniques are quite obvious. The last one, using a plug-in, is discussed a little more in detail.

One such product on the market is ServerMask [4], which is a plug-in to Microsoft IIS servers. ServerMask not only obfuscates the server banner string, but also re-arranges the HTTP response header field order, to mimic servers like Apache, obscures internal server generated cookies, and even has the ability to pose as a random HTTP server for every HTTP request.

However, ServerMask can yet be defeated by fingerprinting engines like **httprint**, which use fuzzy logic analysis on the test results, as shown in the example below:

```
# ./httprint -s signatures.txt -o unknown.html -h unknown.example.com
httprint v0.200 (beta) - web server fingerprinting tool
(c) 2003, net square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

---------------------------------------------------
Finger Printing on http://unknown.example.com:80/
Derived Signature:
Yes we are using ServerMask
FACD41D36ED3C295811C9DC5811C9DC5811C9DC5505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC59D69031D6014C217811C9DC5811C9DC580FF2CD2
FCCC535BE2CE6923FCCC535B811C9DC5E2CE69272576B769E2CE6926811C9DC5
811C9DC5FCCC535B811C9DC56ED3C2956ED3C2956ED3C2956ED3C2956ED3C295
6ED3C2956ED3C295811C9DC568D17AAD68D17AAD

Banner Reported: Yes we are using ServerMask
Banner Deduced: Microsoft-IIS/5.1, Microsoft-IIS/5.0, Microsoft-IIS/4.0
Scores:
Microsoft-IIS/5.1: 83 53.55
Microsoft-IIS/5.0 ASP.NET: 83 53.55
Microsoft-IIS/4.0: 83 53.55
Microsoft-IIS/5.0: 73 33.22
Apache/1.3.27: 69 26.74
Apache/1.3.[1-3]: 68 25.26
Apache/1.3.[4-24]: 68 25.26
Apache/1.2.6: 68 25.26
Apache/1.3.26: 68 25.26
Com21 Cable Modem: 66 22.46
```

```
Netscape-Enterprise/4.1: 63 18.63
Apache/2.0.x: 60 15.23
EMWHTTPD/1.0: 59 14.19
dwhttpd (Sun Answerbook): 56 11.34
SMC Wireless Router 7004VWBR: 56 11.34
Agranat-EmWeb: 52  8.11
Microsoft-IIS/URLScan: 50  6.73
Oracle Servlet Engine: 48  5.49
Microsoft-IIS/6.0: 48  5.49
Netscape-Enterprise/3.6 SP2: 47  4.92
AOLserver/3.5.6: 46  4.39
TightVNC: 46  4.39
MiniServ/0.01 Webmin: 41  2.19
Netscape-Enterprise/3.5.1: 41  2.19
Microsoft-IIS/5.0 Virtual Host: 22  0.78
Orion/2.0x: 21  0.78
AOLserver/3.4.2-3.5.1: 21  0.78
Xerver_v3: 23  0.78
Zeus/4_2: 23  0.78
Domino-Go-Webserver/4.6.2.8: 24  0.76
Jana Server/1.45: 24  0.76
Zope/2.6.0 ZServer/1.1b1: 18  0.72
Hewlett Packard xjet: 25  0.72
thttpd: 36  0.69
fnord: 17  0.69
Zeus/4.1: 16  0.65
HP-ChaiServer/3.0: 27  0.62
Linksys AP2: 15  0.61
Cisco-HTTP: 15  0.61
Zeus/4.0: 15  0.61
Lotus-Domino/6.x: 28  0.55
Stronghold/2.4.2-Apache/1.3.x: 28  0.55
Oracle XML DB/Oracle9i: 13  0.51
WebLogic Server 8.1: 11  0.40
Netscape-Enterprise/3.6: 11  0.40
WebLogic Server 8.x: 11  0.40
Microsoft ISA Server: 11  0.40
NetWare-Enterprise-Web-Server/5.1: 11  0.40
CompaqHTTPServer/4.2: 30  0.35
Stronghold/4.0-Apache/1.3.x: 30  0.35
CompaqHTTPServer-SSL/4.2: 10  0.35
3Com/v1.0: 10  0.35
RemotelyAnywhere: 10  0.35
Linksys Print Server: 10  0.35
CompaqHTTPServer/1.0: 31  0.23
Netscape-Enterprise/3.5.1G: 31  0.23
Netscape-Enterprise/6.0: 33  0.08
Snap Appliances, Inc./3.x: 4  0.07
Lotus-Domino/5.x: 2  0.02
Netscape-Enterprise/4.1: 2  0.02
SunONE WebServer 6.0: 2  0.02
EHTTP/1.1: 1  0.00
Linksys Router: 0  0.00
Linksys AP1: 0  0.00
ServletExec: 0  0.00

---------------------------------------------------
Dumping results in html file..
```

Here, even though the server's responses were obfuscated by ServerMask, **httprint** still accurately identifies it as a Microsoft-IIS/5.x or 4.0 web server.

Given below is an example of five servers using a combination of the techniques discussed above, to disguise their HTTP server behaviour. **httprint** succeeds in identifying the correct web server platform.

A detailed analysis and validation of the above report can be found here

10. Accuracy Issues

httprint's accuracy depends on semantically correct responses delivered from the server, as well as the fact that the server being tested should be a part of the signatures file. There are certain situations where **httprint**'s accuracy is reduced. These are:

- Load balancers or inward proxy servers before the web server. Load balancers and inward proxy servers rewrite HTTP requests when they are forwarded to the web server being tested. In this situation, the original HTTP request gets obfuscated and so does the response.
- Default redirects. In many cases, servers tend to respond with an HTTP 301 or 302 redirect response for all HTTP requests. These are servers that either redirect domain names to newer ones, or are simply old and want to redirect users to new contents on new servers. Older versions of **httprint** used to suffer from accuracy problems when dealing with such servers. The latest version of **httprint** (version 202) has a feature to automatically follow server redirection to the new server and test it. A classic example of this is http://www.hotmail.com/, which redirects to http://login.passport.net/.
- Server signature not present. If a new server is being tested, and its signature is not present in the signature database, then **httprint** reports very low confidence ratings for all the servers it knows about. In this situation, the analyst must infer that the signature is not present, and a new signature should be added to the signature database.

11. Conclusion

This paper was meant to serve as an introduction to HTTP fingerprinting, and has provided an overview of some of the techniques used. HTTP fingerprinting can be also extended to various other areas of research, such as fingerprinting applications running on HTTP, embedded devices that run an HTTP interface, etc.

12. References

[1] nmap OS fingerprinting *http://www.insecure.org/nmap/nmap-fingerprinting-article.html*
[2] Section 6.2 of RFC 2616 *http://www.ietf.org/rfc/rfc2616.txt*
[3] Netcat for Unix *http://www.atstake.com/research/tools/network_utilities/nc110.tgz*,
Netcat for Windows *http://www.atstake.com/research/tools/network_utilities/nc11nt.zip*
[4] ServerMask *http://www.port80software.com/products/servermask/*
[5] HTTP/1.1 RFC 2616 *http://www.ietf.org/rfc/rfc2616.txt*
[6] **httprint** from Net Square *http://net-square.com/httprint/*
[7] **httprint** signatures *http://net-square.com/httprint/signatures.txt*
[8] HMAP web server fingerprinter *http://wwwcsif.cs.ucdavis.edu/~leed/hmap/*

revised 19/05/04 - saumil