HTTP: Advanced Assessment Techniques

Saumil Shah
Director of R&D, NT Objectives Inc.
Director, Net-Square
Author: “Web Hacking - Attacks and Defense”
BlackHat Windows Security 2003, Seattle
The Web Hacker’s playground
The Evolution of Web Hacking

- Classic “phf” bug
- Automated web vulnerability checking
- Input validation attacks
- Buffer overflows
- Source code disclosure
- Session Hijacking
- Lame attacks (the client side XSS attacks)
The Evolution of Web Defense

- Tight web server configuration.
- Web server plug-in filters.
- Secure coding (yee a rright)
- Security by obscurity.
Security by obscurity

• Who is running IIS? … Not me!
• Web server target acquisition:
  • largely by banner grabbing

$ nc 192.168.7.247 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://192.168.7.247/Default.htm
Date: Fri, 01 Jan 1999 20:09:05 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 01 Jan 1999 20:09:05 GMT
ETag: W/"e0d362a4c335be1:ae0"
Content-Length: 133
Security by obscurity

- Patch web server binaries to change server banner.
  - e.g. “Microsoft-IIS/5.0” rewritten to be “Apache/1.3.26”
- If source is available, recompile with different server banner.
  - e.g. “Apache/1.3.26” rewritten to be “WebSTAR”
- Works well in defeating certain automated attacks.
Security by obscurity

- Web server configuration rules / plug-ins to disguise the server header.
- Re-order HTTP header fields, change cookie names, filter certain responses, etc.

```
$ nc 192.168.7.247 80
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Fri, 01 Jan 1999 20:06:24 GMT
Server: Apache/1.3.19 (Unix) (Red-Hat/Linux) mod_ssl/2.8.1
OpenSSL/0.9.6 DAV/1.0.2 PHP/4.0.4pl1 mod_perl/1.24_01
Content-Length: 133
Content-Type: text/html

with ServerMask 2.0
```
HTTP Fingerprinting

- Objective: To accurately determine the underlying web server platform.
- Also attempt to uncover any plug-ins, app servers, etc.
- Based on implementation assumptions / peculiarities of the HTTP protocol spec.
HTTP Fingerprinting

- Fingerprinting logic
  - Decision-tree based methods
  - Statistical methods
  - Neural Network based methods

- Fingerprinting engine
  - Set of test cases, carefully chosen
  - Response-tree
  - Weight vectors
HTTP Fingerprinting Techniques

- Deviation from HTTP RFCs
- Behaviour not specified by the HTTP RFCs
- Default behaviour
- Header field order
- Implementation peculiarities
- Error analysis
- Cookie strings
- … similar to OS fingerprinting
HTTP Fingerprinting - Accuracy

- Choice of test cases
- Decision-trees are hard to scale
- Choice of result weights
- Scoring system
- Training input set (for neural networks)
HTTP Fingerprinting - example 1

REPORTED: Apache-AdvancedExtranetServer/1.3.19 (Linux-Mandrake/3mdk) mod_ssl/2.8.2 OpenSSL/0.9.6 PHP/4.0.4p11

Best Match: Apache/1.3.x

Microsoft-IIS/4.0: 23    Netscape-Enterprise/6.0: 24
Microsoft-IIS/5.0: 23    Netscape-FastTrack/4.1: 37
Microsoft-IIS/5.1: 22    Netscape-Enterprise/4.0: 10
Microsoft-IIS/6.0: 19    Netscape-Enterprise/4.1: 37
Microsoft-IIS/URLScan: 18 Netscape-Enterprise/3.6: 10

Apache/2.0.x: 70          Zeus/4.0: 29
Apache/1.3.27: 77         Zeus/4.1: 28
Apache/1.3.26: 76         Zeus/4_2: 23
Apache/1.3.x: 78          Lotus-Domino/5.0.x: 1
Apache/1.2.6: 73          AOLserver/3.4.2-3.5.1: 20
Stronghold/4.0-Apache/1.3.x: 68
Stronghold/2.4.2-Apache/1.3.x: 38

No obfuscation.
Verification of testing.
HTTP Fingerprinting - example 2

REPORTED: WebSTAR

Best Match: Apache/1.3.27 Apache/1.3.26

Microsoft-IIS/4.0: 29  Netscape-Enterprise/6.0: 26
Microsoft-IIS/5.0: 29  Netscape-FastTrack/4.1: 23
Microsoft-IIS/5.1: 29  Netscape-Enterprise/4.0: 14
Microsoft-IIS/6.0: 39  Netscape-Enterprise/4.1: 23
Microsoft-IIS/URLScan: 27  Netscape-Enterprise/3.6: 25

Apache/2.0.x: 56  Zeus/4.0: 10
**Apache/1.3.27: 59**  Zeus/4.1: 21
**Apache/1.3.26: 59**  Zeus/4_2: 27
Apache/1.3.x: 58  Lotus-Domino/5.0.x: 1
Apache/1.2.6: 43  AOLserver/3.4.2-3.5.1: 34
Stronghold/4.0-Apache/1.3.x: 51
Stronghold/2.4.2-Apache/1.3.x: 56

Recompiled Apache - banner patching.  Easy to tell
HTTP Fingerprinting - example 3

REPORTED: Apache/1.3.23 (Unix)

Best Match: Microsoft-IIS/4.0

<table>
<thead>
<tr>
<th>Server</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft-IIS/4.0</td>
<td>63</td>
</tr>
<tr>
<td>Microsoft-IIS/5.0</td>
<td>53</td>
</tr>
<tr>
<td>Microsoft-IIS/5.1</td>
<td>54</td>
</tr>
<tr>
<td>Microsoft-IIS/6.0</td>
<td>31</td>
</tr>
<tr>
<td>Microsoft-IIS/URLScan</td>
<td>50</td>
</tr>
<tr>
<td>Apache/2.0.x</td>
<td>40</td>
</tr>
<tr>
<td>Apache/1.3.27</td>
<td>49</td>
</tr>
<tr>
<td>Apache/1.3.26</td>
<td>48</td>
</tr>
<tr>
<td>Apache/1.3.x</td>
<td>48</td>
</tr>
<tr>
<td>Apache/1.2.6</td>
<td>48</td>
</tr>
<tr>
<td>Stronghold/4.0-Apache/1.3.x</td>
<td>35</td>
</tr>
<tr>
<td>Stronghold/2.4.2-Apache/1.3.x</td>
<td>33</td>
</tr>
<tr>
<td>Netscape-Enterprise/6.0</td>
<td>25</td>
</tr>
<tr>
<td>Netscape-FastTrack/4.1</td>
<td>28</td>
</tr>
<tr>
<td>Netscape-Enterprise/4.0</td>
<td>11</td>
</tr>
<tr>
<td>Netscape-Enterprise/4.1</td>
<td>28</td>
</tr>
<tr>
<td>Netscape-Enterprise/3.6</td>
<td>22</td>
</tr>
<tr>
<td>Zeus/4.0</td>
<td>15</td>
</tr>
<tr>
<td>Zeus/4.1</td>
<td>16</td>
</tr>
<tr>
<td>Zeus/4_2</td>
<td>23</td>
</tr>
<tr>
<td>Lotus-Domino/5.0.x</td>
<td>2</td>
</tr>
<tr>
<td>AOLserver/3.4.2-3.5.1</td>
<td>21</td>
</tr>
</tbody>
</table>

Servermask: Scores are close enough to one another. Bit harder to tell.
HTTP Response Codes

• Customised error pages.
• A non existent page should return an HTTP 404 code.
• Many servers return:
  • 301/302 - redirect to some starting page
  • 200 OK - to fool crawlers
  • …and other customised codes.
Page Signatures

- Objective: To accurately identify proper HTTP response codes.
- Minimize false positives.
- Greatly helps in automated testing.
- Can be extended beyond error detection
  - e.g. group similar pages together
Page Signatures

• Each HTTP response has a page signature.
• Content independent.
• Ability to overlook random content.
• Constant length.
• Computation time: $O(n)$
• Comparison time: $O(k)$
Normal error page

Not Found

The requested URL /junk was not found on this server.

Apache/1.3.26 Server at 192.168.7.70 Port 8222

$ nc 192.168.7.70 8222
GET /junk HTTP/1.0

HTTP/1.1 404 Not Found
Date: Tue, 04 Feb 2003 06:22:00 GMT
Server: Apache/1.3.26 (Unix) mod_perl/1.26 mod_ssl/2.8.9 OpenSSL/0.9.6e
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head></head><body><h1>Not Found</h1>The requested URL /junk was not found on this server.<p>
</p><hr/>
<address>Apache/1.3.26 Server at 192.168.7.70 Port 8222</address></body></html>
Sorry!

Random number: 1198361.67040379

The link you requested **http://192.168.7.2/junk** was not found

Please contact the site administrator at root@dev.null if you feel this is in error. Alternately, try searching with Google

In 1 minute, you will be refreshed back to the main page

$ nc 192.168.7.70 8222
GET /junk HTTP/1.0
HTTP/1.1 200 OK
Date: Tue, 04 Feb 2003 01:41:06 GMT
Server: Apache-AdvancedExtranetServer/1.3.19 (Linux-Mandrake/3mdk) mod_ssl/2.8.2 OpenSSL/0.9.6 PHP/4.0.4p11 Connection: close Content-Type: text/html; charset=ISO-8859-1

<html><body><H1>Sorry!</H1><p>Random number: 318405.070147527<p>The link you requested <b>http://192.168.7.2/junk</b> was not found
<p>Please contact the site administrator at root@dev.null if you feel this is in error. Alternately, try searching with Google
<p>In 1 minute, you will be refreshed back to the main page
<p><FORM method=GET action=http://www.google.com/search>
<IMG SRC=http://www.google.com/logos/Logo_40wht.gif border=0 ALT=Google align=absmiddle>
<INPUT TYPE=text name=q size=15 maxlength=255><INPUT type=submit name=btnG VALUE=Search>
</FORM></body></html>
Dealing with random content

- Page signatures are independent of content


- All of the above are 404 pages.
- Though their content may change, their signature doesn’t.
Reverse Proxy Servers

- Web proxy servers may work both ways!
- Typically meant to allow users from within a network to access external web sites.
- May end up proxying HTTP requests from the outside world to the internal network.
- e.g. Compaq Insight Manager
- Usually happens when the front end web server proxies requests to back end app servers.
Reverse Proxying

Web Client

10.0.0.1

GET http://10.0.0.3/ HTTP/1.0
Port Scanning through Proxies

- Issue multiple GET requests to the proxy:
  - GET http://10.0.0.3:21/ HTTP/1.0
  - GET http://10.0.0.3:25/ HTTP/1.0
  - GET http://10.0.0.3:135/ HTTP/1.0
  - GET http://10.0.0.3:139/ HTTP/1.0
- Use Page signatures to identify accurately if a port is open on an internal host.
Better CONNECTivity

• HTTP CONNECT can be used to open up a bi-directional TCP connection.
• Originally intended for SSL traffic.
• Often overlooked.
• Ability to tunnel arbitrary TCP data over an HTTP proxy connection.
• Once CONNECTed, the proxy simply passes the TCP data back and forth.
An Advanced HTTP attack example

- Scan an internal network through a proxy.
- Own a server on the internal network via “one-way” hacking techniques.
- Plant port redirectors through HTTP uploads.
- Set up a connection via HTTP CONNECT through the proxy to the internal redirector.
- … where the “Art of Attack and Penetration” leaves off!
Closing Thoughts

• “You can’t patch (or hide) carelessness”.
• Web Hacking: Attacks and Defense
  Saumil Shah,
  Shreeraj Shah,
  Stuart McClure
Thank you!

saumil@ntobjectives.com
saumil@net-square.com