HTTP/HTTPs Service Enumeration

Related PCAP File(s):

- `basic_fuzzing.pcapng`

Many times, we might notice strange traffic to our web servers. In one of these cases, we might see that one host is generating excessive traffic with HTTP or HTTPs. Attackers like to abuse the transport layer many times, as the applications running on our servers might be vulnerable to different attacks. As such, we need to understand how to recognize the steps an attacker will take to gather information, exploit, and abuse our web servers.

Generally speaking, we can detect and identify fuzzing attempts through the following

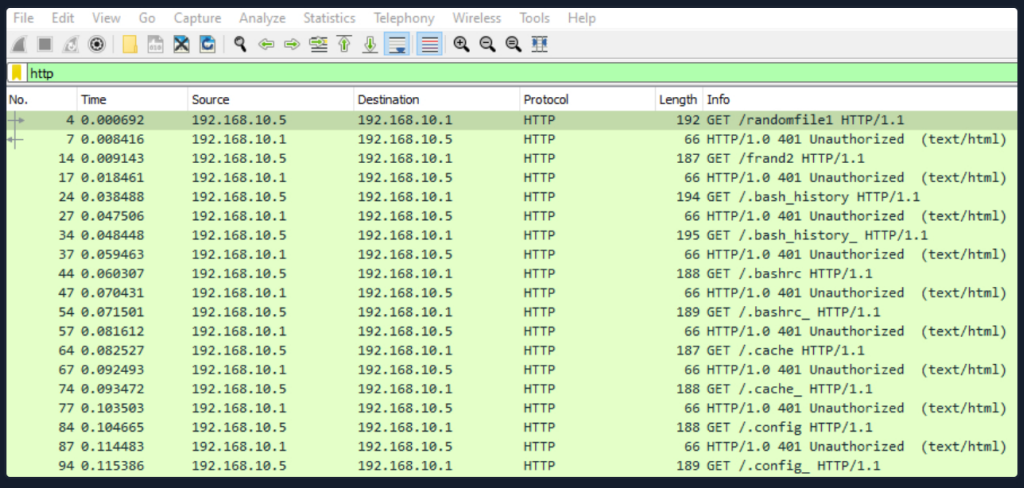
1. `Excessive HTTP/HTTPs traffic from one host`
2. `Referencing our web server's access logs for the same behavior`

Primarily, attackers will attempt to fuzz our server to gather information before attempting to launch an attack. We might already have a `Web Application Firewall` in place to prevent this, however, in some cases we might not, especially if this server is internal.

Finding Directory Fuzzing

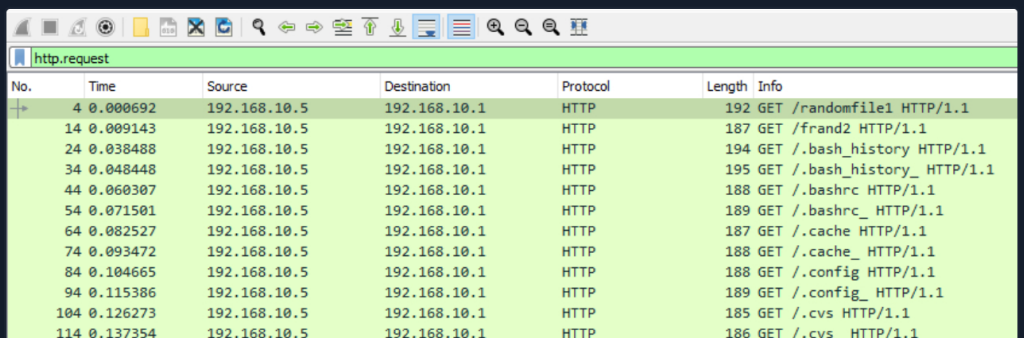
Directory fuzzing is used by attackers to find all possible web pages and locations in our web applications. We can find this during our traffic analysis by limiting our Wireshark view to only http traffic.

- `http`

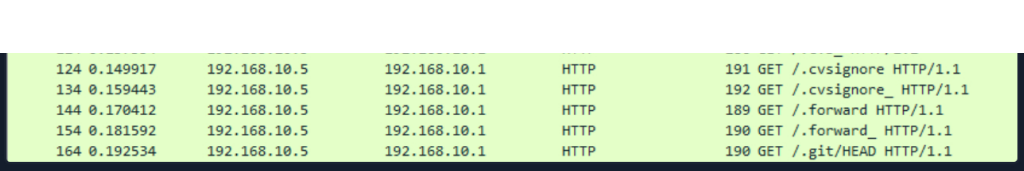


No.	Time	Source	Destination	Protocol	Length	Info
4	0.000692	192.168.10.5	192.168.10.1	HTTP	192	GET /randomfile1 HTTP/1.1
7	0.008416	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
14	0.009143	192.168.10.5	192.168.10.1	HTTP	187	GET /frand2 HTTP/1.1
17	0.018461	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
24	0.038488	192.168.10.5	192.168.10.1	HTTP	194	GET /.bash_history HTTP/1.1
27	0.047506	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
34	0.048448	192.168.10.5	192.168.10.1	HTTP	195	GET /.bash_history_ HTTP/1.1
37	0.059463	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
44	0.060307	192.168.10.5	192.168.10.1	HTTP	188	GET /.bashrc HTTP/1.1
47	0.070431	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
54	0.071501	192.168.10.5	192.168.10.1	HTTP	189	GET /.bashrc_ HTTP/1.1
57	0.081612	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
64	0.082527	192.168.10.5	192.168.10.1	HTTP	187	GET /.cache HTTP/1.1
67	0.092493	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
74	0.093472	192.168.10.5	192.168.10.1	HTTP	188	GET /.cache_ HTTP/1.1
77	0.103503	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
84	0.104665	192.168.10.5	192.168.10.1	HTTP	188	GET /.config HTTP/1.1
87	0.114483	192.168.10.1	192.168.10.5	HTTP	66	HTTP/1.0 401 Unauthorized (text/html)
94	0.115386	192.168.10.5	192.168.10.1	HTTP	189	GET /.config_ HTTP/1.1

Secondarily, if we wanted to remove the responses from our server, we could simply specify `http.request`



No.	Time	Source	Destination	Protocol	Length	Info
4	0.000692	192.168.10.5	192.168.10.1	HTTP	192	GET /randomfile1 HTTP/1.1
14	0.009143	192.168.10.5	192.168.10.1	HTTP	187	GET /frand2 HTTP/1.1
24	0.038488	192.168.10.5	192.168.10.1	HTTP	194	GET /.bash_history HTTP/1.1
34	0.048448	192.168.10.5	192.168.10.1	HTTP	195	GET /.bash_history_ HTTP/1.1
44	0.060307	192.168.10.5	192.168.10.1	HTTP	188	GET /.bashrc HTTP/1.1
54	0.071501	192.168.10.5	192.168.10.1	HTTP	189	GET /.bashrc_ HTTP/1.1
64	0.082527	192.168.10.5	192.168.10.1	HTTP	187	GET /.cache HTTP/1.1
74	0.093472	192.168.10.5	192.168.10.1	HTTP	188	GET /.cache_ HTTP/1.1
84	0.104665	192.168.10.5	192.168.10.1	HTTP	188	GET /.config HTTP/1.1
94	0.115386	192.168.10.5	192.168.10.1	HTTP	189	GET /.config_ HTTP/1.1
104	0.126273	192.168.10.5	192.168.10.1	HTTP	185	GET /.cvs HTTP/1.1
114	0.137354	192.168.10.5	192.168.10.1	HTTP	186	GET /.cvs_ HTTP/1.1



124	0.149917	192.168.10.5	192.168.10.1	HTTP	191 GET /.cvsignore HTTP/1.1
134	0.159443	192.168.10.5	192.168.10.1	HTTP	192 GET /.cvsignore_ HTTP/1.1
144	0.170412	192.168.10.5	192.168.10.1	HTTP	189 GET /.forward HTTP/1.1
154	0.181592	192.168.10.5	192.168.10.1	HTTP	190 GET /.forward_ HTTP/1.1
164	0.192534	192.168.10.5	192.168.10.1	HTTP	190 GET /.git/HEAD HTTP/1.1

Directory fuzzing is quite simple to detect, as it will in most cases show the following signs

1. A host will repeatedly attempt to access files on our web server which do not exist (response 404).
2. A host will send these in rapid succession.

We can also always reference this traffic within our access logs on our web server. For Apache this would look like the following two examples. To use grep, we could filter like so:

HTTP/HTTPs Service Enumeration

```
MisaelMacias@htb[/htb]$ cat access.log | grep "192.168.10.5"

192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /randomfile1 HTTP/1.1" 404 435 "-" "Mozilla/4.0
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /frand2 HTTP/1.1" 404 435 "-" "Mozilla/4.0 (comp
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.bash_history HTTP/1.1" 404 435 "-" "Mozilla/4.
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.bashrc HTTP/1.1" 404 435 "-" "Mozilla/4.0 (com
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.cache HTTP/1.1" 404 435 "-" "Mozilla/4.0 (comp
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.config HTTP/1.1" 404 435 "-" "Mozilla/4.0 (com
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.cvs HTTP/1.1" 404 435 "-" "Mozilla/4.0 (compat
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.cvsignore HTTP/1.1" 404 435 "-" "Mozilla/4.0 (
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.forward HTTP/1.1" 404 435 "-" "Mozilla/4.0 (co
...SNIP...
```

And to use awk, we could do the following

HTTP/HTTPs Service Enumeration

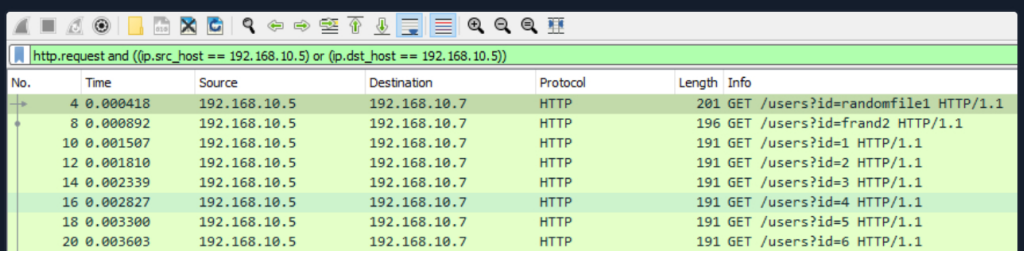
```
MisaelMacias@htb[/htb]$ cat access.log | awk '$1 == "192.168.10.5"'

192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /randomfile1 HTTP/1.1" 404 435 "-" "Mozilla/4.0
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /frand2 HTTP/1.1" 404 435 "-" "Mozilla/4.0 (comp
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.bash_history HTTP/1.1" 404 435 "-" "Mozilla/4.
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.bashrc HTTP/1.1" 404 435 "-" "Mozilla/4.0 (com
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.cache HTTP/1.1" 404 435 "-" "Mozilla/4.0 (comp
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.config HTTP/1.1" 404 435 "-" "Mozilla/4.0 (com
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.cvs HTTP/1.1" 404 435 "-" "Mozilla/4.0 (compat
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.cvsignore HTTP/1.1" 404 435 "-" "Mozilla/4.0 (
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.forward HTTP/1.1" 404 435 "-" "Mozilla/4.0 (co
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.git/HEAD HTTP/1.1" 404 435 "-" "Mozilla/4.0 (c
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.history HTTP/1.1" 404 435 "-" "Mozilla/4.0 (co
192.168.10.5 - - [18/Jul/2023:12:58:07 -0600] "GET /.hta HTTP/1.1" 403 438 "-" "Mozilla/4.0 (compat
...SNIP...
```

Finding Other Fuzzing Techniques

However, there are other types of fuzzing which attackers might employ against our web servers. Some of these could include fuzzing dynamic or static elements of our web pages such as id fields. Or in some other cases, the attacker might look for IDOR vulnerabilities in our site, especially if we are handling json parsing (changing `return=max` to `return=min`).

To limit traffic to just one host we can employ the following filter:

- `http.request and ((ip.src_host == <suspected IP>) or (ip.dst_host == <suspected IP>))`



http.request and ((ip.src_host == 192.168.10.5) or (ip.dst_host == 192.168.10.5))

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000418	192.168.10.5	192.168.10.7	HTTP	201	GET /users?id=randomfile1 HTTP/1.1
8	0.000892	192.168.10.5	192.168.10.7	HTTP	196	GET /users?id=frand2 HTTP/1.1
10	0.001507	192.168.10.5	192.168.10.7	HTTP	191	GET /users?id=1 HTTP/1.1
12	0.001810	192.168.10.5	192.168.10.7	HTTP	191	GET /users?id=2 HTTP/1.1
14	0.002339	192.168.10.5	192.168.10.7	HTTP	191	GET /users?id=3 HTTP/1.1
16	0.002827	192.168.10.5	192.168.10.7	HTTP	191	GET /users?id=4 HTTP/1.1
18	0.003300	192.168.10.5	192.168.10.7	HTTP	191	GET /users?id=5 HTTP/1.1
20	0.003603	192.168.10.5	192.168.10.7	HTTP	191	GET /users?id=6 HTTP/1.1

```
    22 0.003888      192.168.10.5          192.168.10.7          HTTP             191 GET /users?id=7 HTTP/1.1
    24 0.004178      192.168.10.5          192.168.10.7          HTTP             191 GET /users?id=8 HTTP/1.1
    26 0.004456      192.168.10.5          192.168.10.7          HTTP             191 GET /users?id=9 HTTP/1.1
    28 0.004737      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=10 HTTP/1.1
    30 0.005017      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=11 HTTP/1.1
    32 0.005317      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=12 HTTP/1.1
    34 0.005595      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=13 HTTP/1.1
    36 0.005875      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=14 HTTP/1.1
    38 0.006198      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=15 HTTP/1.1
    40 0.006481      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=16 HTTP/1.1
    42 0.006757      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=17 HTTP/1.1
    44 0.007031      192.168.10.5          192.168.10.7          HTTP             192 GET /users?id=18 HTTP/1.1
```

Secondarily, we can always build an overall picture by right clicking any of these requests, going to follow, and follow HTTP stream.

Wireshark · Follow HTTP Stream (tcp.stream eq 0) · number_fuzzing.pcapng

```
HTTP/1.1 404 Not Found
Date: Tue, 18 Jul 2023 19:02:35 GMT
Server: Apache/2.4.57 (Debian)
Content-Length: 274
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.57 (Debian) Server at 192.168.10.7 Port 80</address>
</body></html>
GET /users?id=8 HTTP/1.1
Host: 192.168.10.7
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Accept: */*

HTTP/1.1 404 Not Found
Date: Tue, 18 Jul 2023 19:02:35 GMT
Server: Apache/2.4.57 (Debian)
Content-Length: 274
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.57 (Debian) Server at 192.168.10.7 Port 80</address>
</body></html>
GET /users?id=9 HTTP/1.1
Host: 192.168.10.7
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Accept: */*
```

Suppose we notice that a lot of requests were sent in rapid succession, this would indicate a fuzzing attempt, and we should carry out additional investigative efforts against the host in question.

However sometimes attackers will do the following to prevent detection

1. Stagger these responses across a longer period of time.
2. Send these responses from multiple hosts or source addresses.

Preventing Fuzzing Attempts

We can aim to prevent fuzzing attempts from adversaries by conducting the following actions.

1. Maintain our virtualhost or web access configurations to return the proper response codes to throw off these scanners.
2. Establish rules to prohibit these IP addresses from accessing our server through our web

application firewall.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 137ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

+ 1 Inspect the basic_fuzzing.pcapng file, part of this module's resources, and enter the total number of HTTP packets that are related to GET requests against port 80 as your answer.

204

Submit

Previous Next

Mark Complete & Next