

# Scanning & Enumeration

## Installing Kioptrix

Kioptrix is a vulnerable VM that will be used for some practice. It's downloadable from the internet. This is the link <https://www.vulnhub.com/entry/kioptrix-level-1-1,22/>. Another useful resource can be found here: <https://www.abatchy.com/2017/02/oscp-like-vulnhub-vm/>.

When importing the Kioptrix Level 1 VM for the first time, I encountered an issue where I got a `Kernel Panic` error while booting. I solved this problem by changing the Storage interface from SCSI to IDE. The VM only works with IDE storage interface. This resource helped: <https://www.hypn.za.net/blog/2017/07/15/running-kioptrix-level-1-and-others-in-virtualbox/>.

I also encountered another issue where I could not ping IP addresses (e.g `1.1.1.1`). I fixed this issue by searching for the term `Bridge` in the `.vmx` config file and changing it to `nat` (lowercase). Then, I right-clicked on the config file and selected `Open with...` -> selected `VMware Fusion...`.

TIP: The login username is `john` and passwd is `TwoCows2`.

When using the Kioptrix VM, we cannot easily know our IP address using a command like `ifconfig`. A little trick to know our password is using the `ping` command. Here's an example:

```
[john@kioptrix john]$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) from 192.168.229.133 : 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=0 ttl=128 time=6.776 msec
64 bytes from 1.1.1.1: icmp_seq=1 ttl=128 time=9.043 msec
64 bytes from 1.1.1.1: icmp_seq=2 ttl=128 time=8.701 msec
64 bytes from 1.1.1.1: icmp_seq=3 ttl=128 time=8.694 msec

--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 6.776/8.303/9.043/0.897 ms
[john@kioptrix john]$
```

The highlighted IP address is the IP address of the Kioptrix machine.

## Scanning with Nmap

First of all let's find out the IP address of our Kali Linux machine using the `ifconfig` command shown below:

```

root@kali:~# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.229.132 netmask 255.255.255.0 broadcast 192.168.229.255
    inet6 fe80::20c:29ff:fe6:c5b3 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:f6:c5:b3 txqueuelen 1000 (Ethernet)
    RX packets 987104 bytes 476404069 (454.3 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 990264 bytes 213579673 (203.6 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1022 bytes 351103 (342.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1022 bytes 351103 (342.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~# █

```

Next, let's use a tool called `arp-scan` to identify the devices on our network as shown below:

```

root@kali:~# arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:f6:c5:b3, IPv4: 192.168.229.132
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.229.1 00:50:56:c0:00:08 VMware, Inc.
192.168.229.2 00:50:56:eb:29:6e VMware, Inc.
192.168.229.133 00:0c:29:b7:cd:07 VMware, Inc.
192.168.229.254 00:50:56:e9:d2:b6 VMware, Inc.

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.877 seconds (136.39 hosts/sec). 4 responded
root@kali:~# █

```

Kioptrix  
VM

Let's use a tool called `netdiscover` to find out more information about our network using ARP as shown below:

```

root@kali:~# netdiscover -r 192.168.229.0/24 █

```

In the picture above, the `-r` flag is used to specify the range and the `192.168.229.0/24` means that we are looking for machines with these three octets on the `/24` subnet. The results of the scan are shown below:

```

Currently scanning: Finished! | Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240

  IP                At MAC Address      Count    Len  MAC Vendor / Hostname
  ---                -
192.168.229.1      00:50:56:c0:00:08      1       60  VMware, Inc.
192.168.229.2      00:50:56:eb:29:6e      1       60  VMware, Inc.
192.168.229.133    00:0c:29:b7:cd:07      1       60  VMware, Inc.
192.168.229.254    00:50:56:e9:d2:b6      1       60  VMware, Inc.

root@kali:~# █

```

Nmap stands for Network Mapper. In our practice, we'll be using Nmap in stealth mode most of the time.

NOTE: In a secure network, stealth mode is very detectable. It's not that "stealth".

Nmap uses a modified three-way handshake when scanning in stealth mode:

Non-Stealth Mode (Three-way handshake): SYN -> SYN ACK -> ACK

Stealth Mode: SYN -> SYN ACK -> RST

Basically, the three-way handshake establishes a connection. On the other hand the stealth mode doesn't and that's because of the `RST` flag that resets the connection just as the connection is about to be established. Nmap stealth mode is default in some cases. Stealth mode can be used using the `-ss` switch (flag) as shown below:

```
root@kali:~# nmap -T4 -p- -A 192.168.229.133
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-10 00:04 EDT
```

As we can see in the above picture, the `-T4` flag means that on a scale of 1 to 5, with 1 being the slowest and 5 the fastest, we are scanning at a speed of 4 out of 5. With slower scans, detection is harder and we hardly miss vital information, but with faster scans, detection is easier and we can miss out on a lot of valuable information. The `-p-` flag means that we are scanning all ports on the target machine. In some cases where we are targeting a specific port such as port 80 (HTTP) we use the `-p` flag such that `-p 80` will scan just port 80. When we don't use the `-p-` or `-p` flags, our attack machine will scan the top 1000 most common ports. It is recommended to use the `-p-` because our target system might be running services on "uncommon" ports. `-A` gives us all possible information on a port.

These are the results of the scan below:

```

root@kali:~# nmap -T4 -p- -A 192.168.229.133
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-10 00:04 EDT
Nmap scan report for 192.168.229.133
Host is up (0.00048s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 2.9p2 (protocol 1.99)
|_ ssh-hostkey:
|   1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|_  1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
|_ _sshv1: Server supports SSHv1
80/tcp    open  http         Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
|_ http-methods:
|_ _ Potentially risky methods: TRACE
|_ _http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_ _http-title: Test Page for the Apache Web Server on Red Hat Linux
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd (workgroup: Cf2MYGROUP)
443/tcp   open  ssl/https    Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_ _http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_ _http-title: 400 Bad Request
|_ _ssl-date: 2020-07-10T04:07:39+00:00; +1m59s from scanner time.
|_ _sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_RC4_64_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|_ 1024/tcp open  status       1 (RPC #100024)
MAC Address: 00:0C:29:B7:CD:07 (VMware)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 1 hop

Host script results:
|_ _clock-skew: 1m58s
|_ _nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_ _smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT      ADDRESS
1   0.48 ms  192.168.229.133

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 131.74 seconds
root@kali:~#

```

When examining the results, we want to look at the open ports and the services running on those ports.

## Enumerating HTTP/HTTPS Part 1

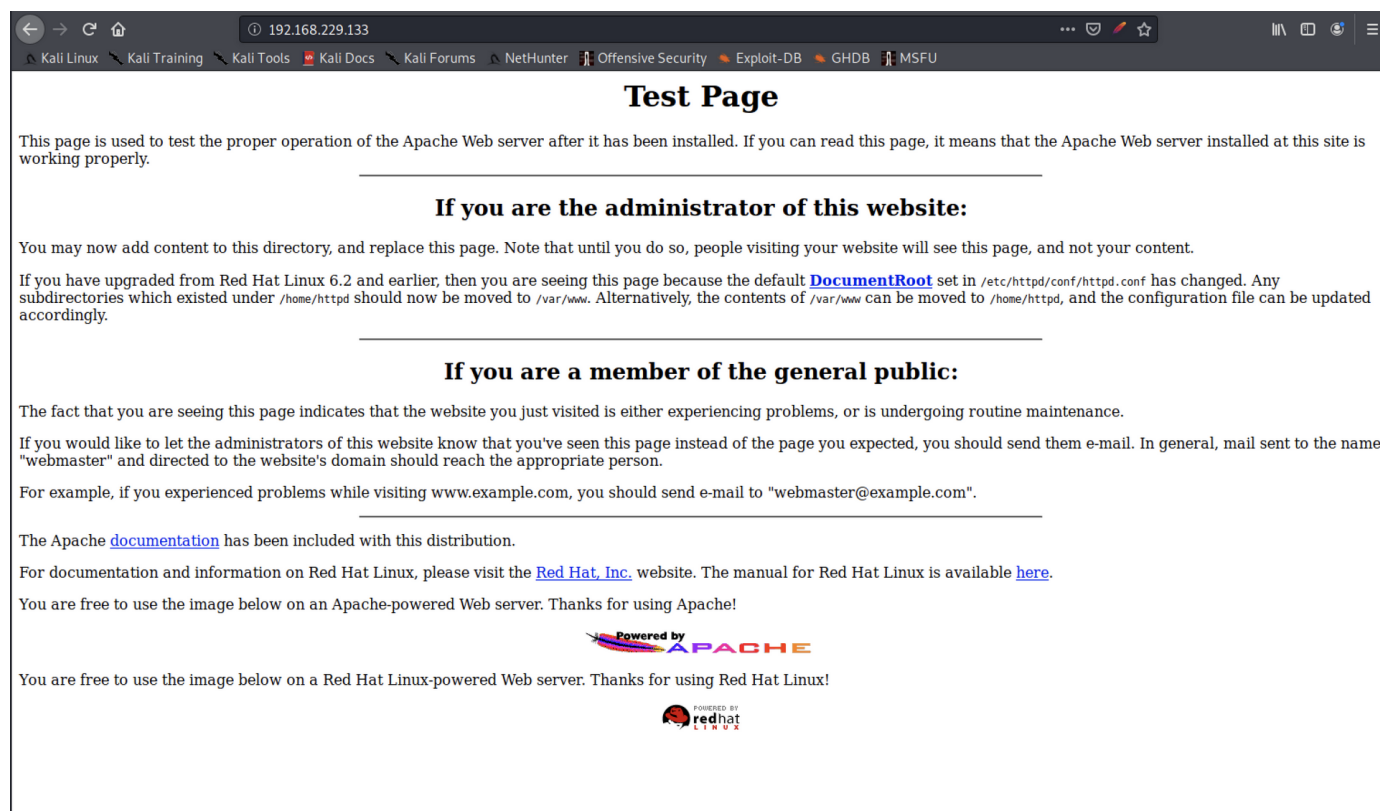
Before we start enumeration, we should notice some important ports such at:

1. 22 - SSH
2. 80 - HTTP (probably a Web Server)
3. 111 - RPC
4. 139 - Samba
5. 443 - HTTPS

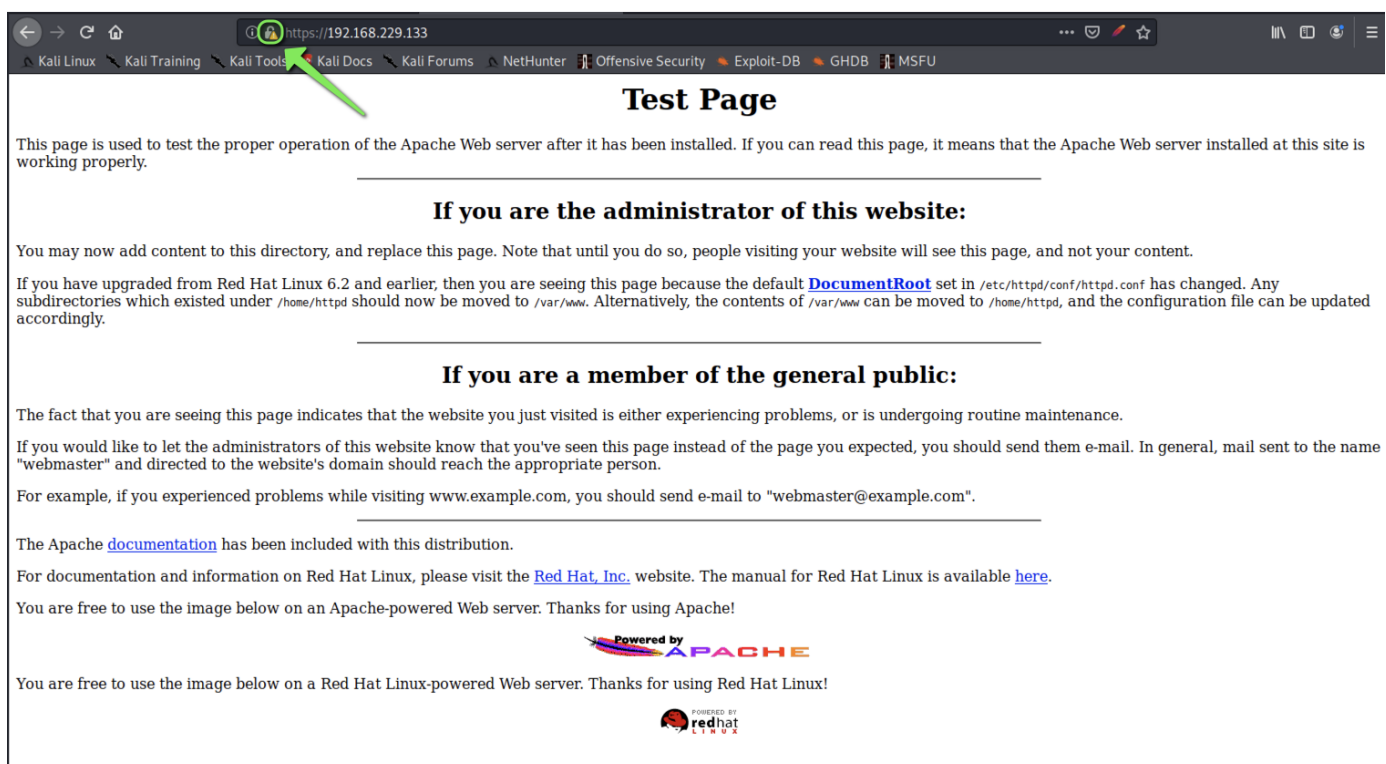
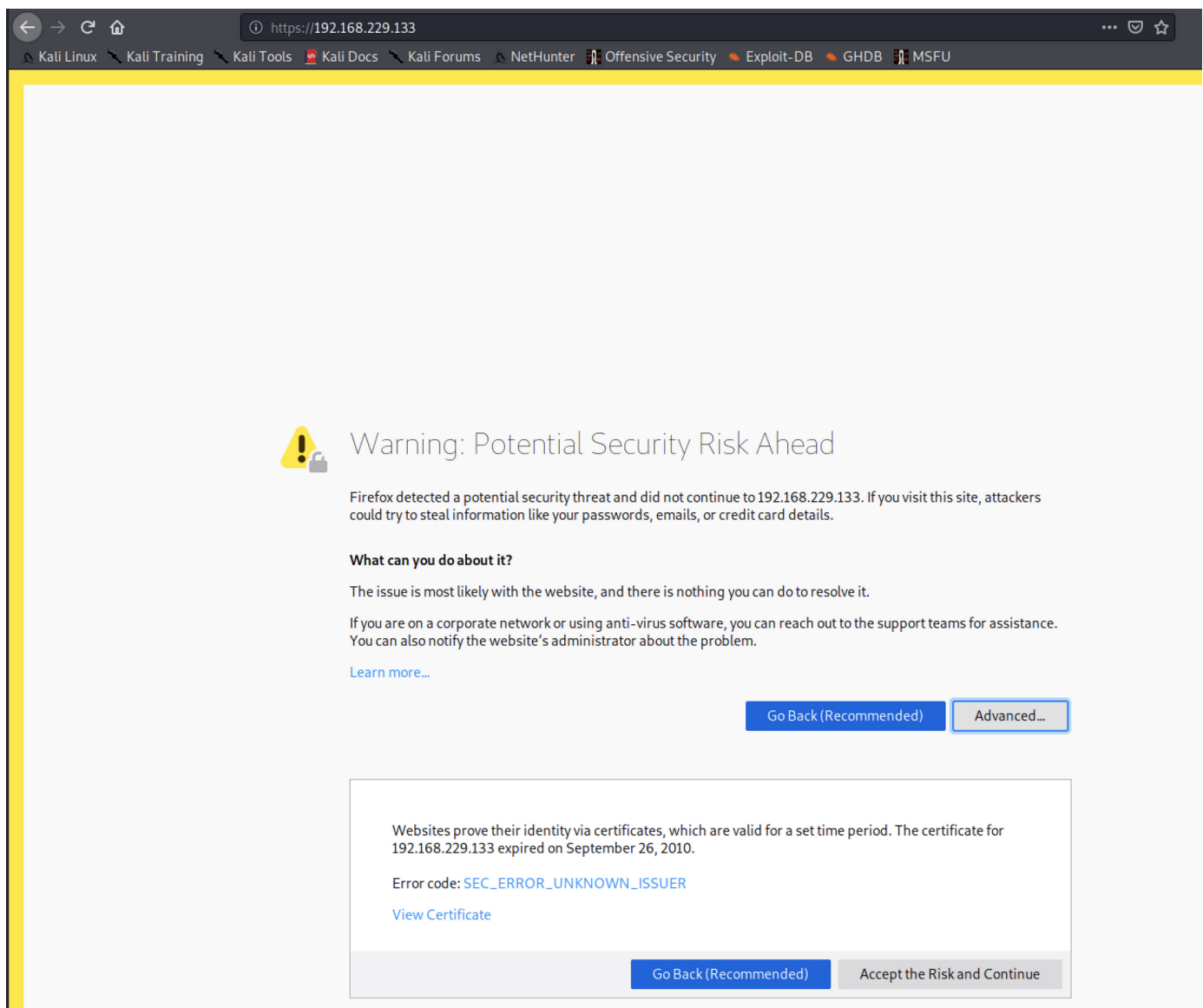
We also need to think of our "Point/Port of Attack". Usually we want to look at port 80, 111, 139, and 443. Port 22 attacks are uncommon and are not the easiest to attack.

Let's investigate ports 80 and 443. Usually, when we have those ports, we have a web server and a website. So the first step would be to go to the website on our browser to see what it's like as shown below:

Port 80 (HTTP): Simply type the IP address in the search bar as shown below:



Port 443 (HTTPS): Simply prefix the IP address with `https://` as shown below and you'll receive a warning page (accept it and you'll get the same page on the port 80 with a minor difference):



The webpage is not really exploitable. However, it gives us some information. For example the owner is running a version of Apache on a Red Hat Linux computer. Furthermore this webpage



is a default webpage for Aoache. When a website owner runs a default webpage, it usually implies that they have directories behind that website. We could then use a directory busting tool such as `dirbuster` to find those directories. On the other hand, it could also imply that the website owners just left ports 80 & 443 open which means that the website owner has poor security hygiene. If we click on a link such as `documentation` it takes us to a `Not Found...` page which is an error 404. This page, however, gives us some information too as shown below:

# Not Found

The requested URL `/manual/index.html` was not found on this server.

*Apache/1.3.20 Server at 127.0.0.1 Port 443*

We can see that it is running Apache version 1.3.20. In some cases, we could find the hostname of the server on this page too.

Now, we will use a tool called `nikto` to run a vulnerability scan on the target box as shown below:

NOTE: a secure network/website will auto-block `nikto` scans.

```
root@kali:~# nikto -h https://192.168.229.133
- Nikto v2.1.6

+ No web server found on 192.168.229.133:443
+ The requested URL /manual/index.html was not found on this server.

+ 0 host(s) tested
root@kali:~# nikto -h http://192.168.229.133
- Nikto v2.1.6

+ Target IP: 192.168.229.133
+ Target Hostname: 192.168.229.133
+ Target Port: 80
+ Start Time: 2020-07-10 00:45:40 (GMT-4)

+ Server: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
+ Server may leak inodes via ETags, header found with file /, inode: 34821, size: 2890, mtime: Wed Sep 5 23:12:46 2001
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Apache/1.3.20 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ mod_ssl/2.8.4 appears to be outdated (current is at least 2.8.31) (may depend on server version)
+ OpenSSL/0.9.6b appears to be outdated (current is at least 1.1.1). OpenSSL 1.0.0a and 0.9.8zc are also current.
+ OSVDB-27487: Apache is vulnerable to XSS via the Expect header
+ Allowed HTTP Methods: GET, HEAD, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ OSVDB-838: Apache/1.3.20 - Apache 1.x up to 1.2.34 are vulnerable to a remote DoS and possible code execution. CAN-2002-0392.
+ OSVDB-4552: Apache/1.3.20 - Apache 1.3 below 1.3.27 are vulnerable to a local buffer overflow which allows attackers to kill any process on the system. CAN-2002-0839.
+ OSVDB-2733: Apache/1.3.20 - Apache 1.3 below 1.3.29 are vulnerable to overflows in mod_rewrite and mod_cgi. CAN-2003-0542.
+ mod_ssl/2.8.4 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0082, OSVDB-756.
+ //etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ OSVDB-682: /usage/: Webalizer may be installed. Versions lower than 2.01-09 vulnerable to Cross Site Scripting (XSS).
+ OSVDB-3268: /manual/: Directory indexing found.
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-3092: /test.php: This might be interesting...
+ /wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpresswp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpresswp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file manager was found.
+ /login.cgi?cli=aa%27cat%20/etc/hosts: Some D-Link router remote command execution.
+ /shell?cat=/etc/hosts: A backdoor was identified.
+ 8724 requests: 0 error(s) and 30 item(s) reported on remote host
+ End Time: 2020-07-10 00:46:00 (GMT-4) (20 seconds)

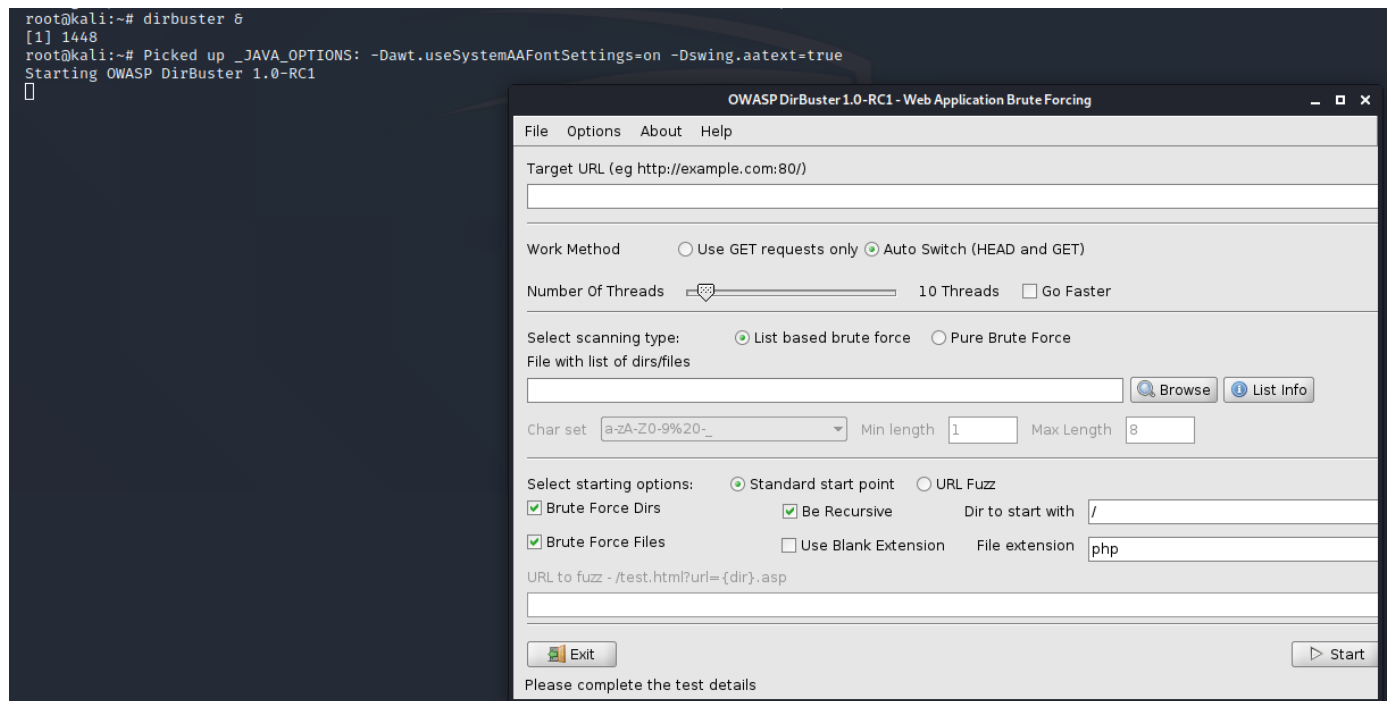
+ 1 host(s) tested
root@kali:~#
```

As we can see above, I used the `nikto -h` command where the `-h` flag identifies the host's (website's) address. In the first run, we targetted port 443 (HTTPS) with no results. In the second run, we targetted port 80 (HTTP) with some results. Looking through the results, we would be interested in `code execution`, `remote buffer overflow`, `HTTP TRACE`, etc.

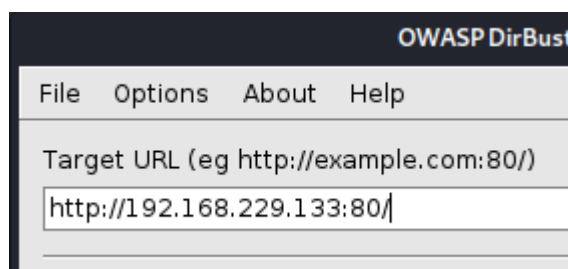
Next we'll use a tool called `dirbuster` to do some directory busting to find hidden directories behind the website.

## Enumerating HTTP/HTTPS Part 2

We can use tools such as `dirb`, `dirbuster`, and `gobuster`. Let's give `dirbuster` a try. These tools go to websites and search for directories. A common way is to brute force the directory search by using wordlists of common directory names. To run `dirbuster`, we can use the `dirbuster &` command to start the program and maintain shell access as shown below:



In the `Target URL` textbox, enter the address of the web server following the syntax `http://<IP_address or hostname>:80/`. The `:80` refers to the port 80 of the web server as shown below:



NOTE: The syntax is strict and must be followed.

Next, we select the `Go Faster` and `List based brute force` options. Then, we'll click `Browse` and go to the directory where wordlists are stored.

TIP: The parent directory where wordlists are stored is `/usr/share/wordlists/`.

These tools also look for specific file extension/file types. In our case we'll stick with `php` because Apache runs on PHP. In other cases, Microsoft websites run on ASP or ASPX. That's why it's important to know the services running on the web server before doing anything else.

We can also search for `txt`, `zip`, `rar` file types.

Now, we can click `start` to start the process.



While the process is running we can start a Burpsuite session, by going to browser and starting the proxy service and starting the Burpsuite program.

TIP: If this webserver was running an actual website, a good thing to also do is to look at the page source code (primarily the comments, information disclosures, keys, passwords, usernames, etc.)

In Burpsuite, let's try intercepting the website requests. This is what we get:

The screenshot shows the Burp Suite interface with the Proxy tab selected. A request to http://192.168.229.133:80 is intercepted. The 'Intercept' tab is active, showing the raw request details. The request is a GET / HTTP/1.1 from 192.168.229.133, using Mozilla/5.0 (X11; Linux x86\_64; rv:68.0) Gecko/20100101 Firefox/68.0 as the user agent. The 'Intercept is on' button is highlighted.

the proper operation of the Apache Web  
e is working properly.

**If you are the**

t to this directory, and replace this page

m Red Hat Linux 6.2 and earlier, then v  
actories which existed under /home/httpd  
uration file can be updated accordingly

**If you are a r**

ing this page indicates that the website

Let's try sending this to the Repeater in Burpsuite. We can do this by right-clicking and selecting **Send to Repeater**. We can then go over to the **Repeater** tab and view the requests and response in real-time. So if we click **Send**, we'll see the response in real-time as shown below:

The screenshot shows the Burp Suite Repeater tab. The request is loaded, and the 'Send' button is visible. The 'Response' tab is selected, showing the response from the server. The response is an HTTP/1.1 304 Not Modified status, indicating the resource has not changed since the last request.

Request

```
1 GET / HTTP/1.1
2 Host: 192.168.229.133
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 If-Modified-Since: Thu, 06 Sep 2001 03:12:46 GMT
10 If-None-Match: "8805-b4a-3b96e9ae"
11 Cache-Control: max-age=0
```

Response

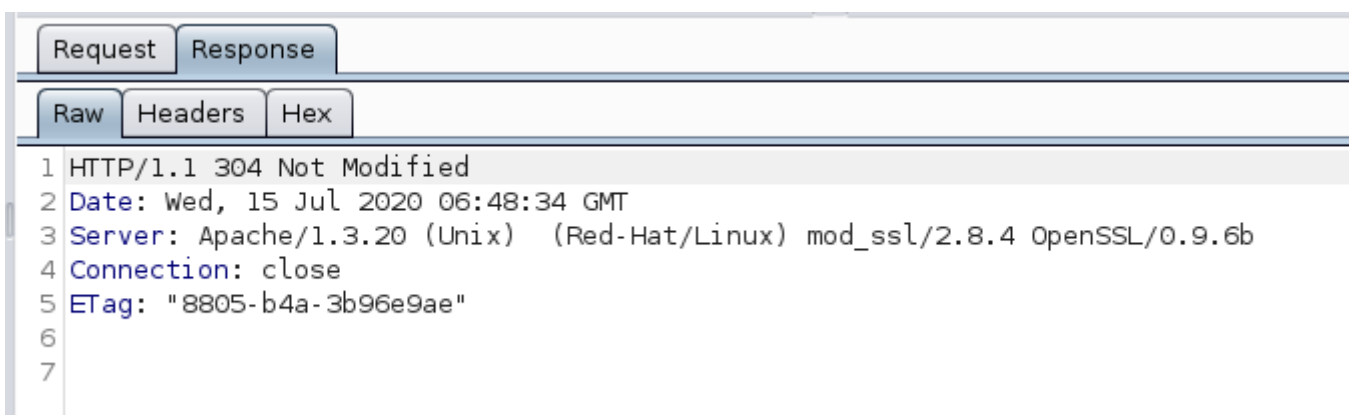
```
1 HTTP/1.1 304 Not Modified
2 Date: Wed, 15 Jul 2020 06:37:06 GMT
3 Server: Apache/1.3.20 (Unix) (Red-Hat/Linux)
  mod_ssl/2.8.4 OpenSSL/0.9.6b
4 Connection: close
5 ETag: "8805-b4a-3b96e9ae"
6
7
```

We can also edit/modify the requests. For example, we can change the **GET** request to a **POST** request and send it as shown below:



TIP: Before performing the operation below, turn off interception.

We can also set the scope of the scan by going to the **Target** tab, clicking on **Scope**, then clicking **Add** under **Include in scope** which will open the small window where we will enter the URL of the web server. A small dialogue box might ask a question regarding out-of-scope items, click **yes**. We'll go over to **Site map** and look at the **Raw** response from the web server as shown below:



We can see that we some information on the server. We ca tell that it's running Apache on Red Hat Linux with OpenSSL. This is an issue for the owners of the server because the server is disclosing information (Server headers with version information).

Looking back at the Dirbuster session we can see that we have gathered some information as shown below:

File Options About Help		
http://192.168.229.133:80/		
<span>Scan Information</span> <span>Results - List View: Dirs: 10 Files: 25</span> <span>Results - Tree View</span> <span>Errors: 0</span>		
Directory Structure	Response Code	Response Size
/	200	3267
cg-bin	403	231
icons	200	204
manual	200	204
doc	403	231
test.php	200	323
usage	200	4672
mrtg	200	18036

For **Response Code**, values in the 200's show that the directory is available, values in the 300's mean that there is a redirect, values in the 400's show that there is an error somewhere, values in the 500's show that there is a server error.

We can start looking through the directories and opening them by right-clicking and selecting **Open in browser**. In the **Usage** folder, we find some interesting information that discloses information on the Website statistics. We can also see that the information was generated using a tool/service called **Webalizer** - version 2.01.

---