

Name: **Kamalesh Ram Chandran Govindaraj**

## Step 1: Setup VM Infrastructure

**Goal** - The goal of this activity is for you to understand how a host-only network is set up and how it works. You will go through a basic configuration. You will not be graded on this portion of the lab. Concepts you will learn in this activity:

- How to enable host-only on VMs
- How to create static IP addresses for machines
- How to enable communication between host-only VMs
- How to use ping to verify host-only communications between VMs is working

### Steps:

1. Grab the clean copy of the clean Windows FlareVM you exported and saved in the first lab. Import an instance of it for this lab.
2. You should also download the Kali Linux VM found here. Make certain to get the VirtualBox distribution: <https://www.kali.org/downloads/>
3. Double click the Kali Linux desktop image you just downloaded and import it into VirtualBox.
4. Double click the Windows FlareVM desktop image you just copied and import it into VirtualBox. You can set the number of CPUs and RAM to use during import based on your system.
5. Run the Kali and Windows Flare VMs. The UN/PWD for Kali is `kali/kali`.
6. On the Windows machine, open up PowerShell and type:  

```
> ping 192.168.56.109
```

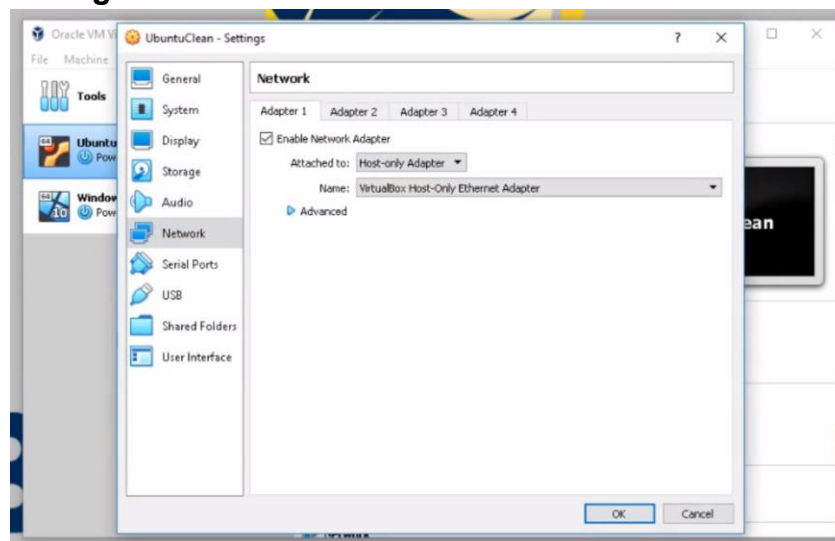
Nothing comes back. We will set this IP on the Kali machine and soon it will come back.
7. On the Kali machine, open up a terminal and type:  

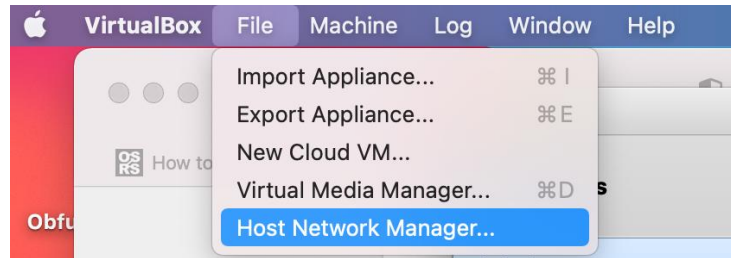
```
> ifconfig
```

You should see this, indicating the IP address of the machine is `192.168.0.4`. (at least that was the IP address assigned for the VM I downloaded and installed). We first start by setting that value to `192.168.56.109`

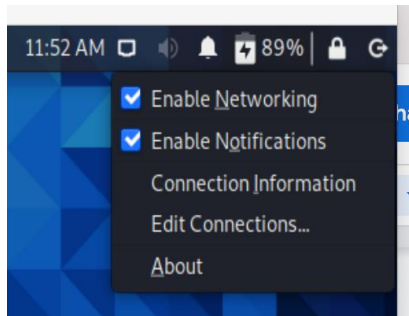
```
(kali@kali)-[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.4 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fee9:3015 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:e9:30:15 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 604 bytes 36452 (35.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

8. Shut down both machines.
9. We will start with the Kali machine first. Go to VirtualBox, find the Kali machine you just imported, and:
  - a. Right click on it and go to **Settings**
  - b. Select the **Network** property
  - c. Click on **Adapter 1**
  - d. Select **Host-only Adapter** like the image below. Click **Ok** to save it.Note, the name provided by VirtualBox may be **vboxnet0** or something like that. Also, it may not be there at all! In that case you will need to add one. Look at the two images below and add it through **Host Network Manager...**

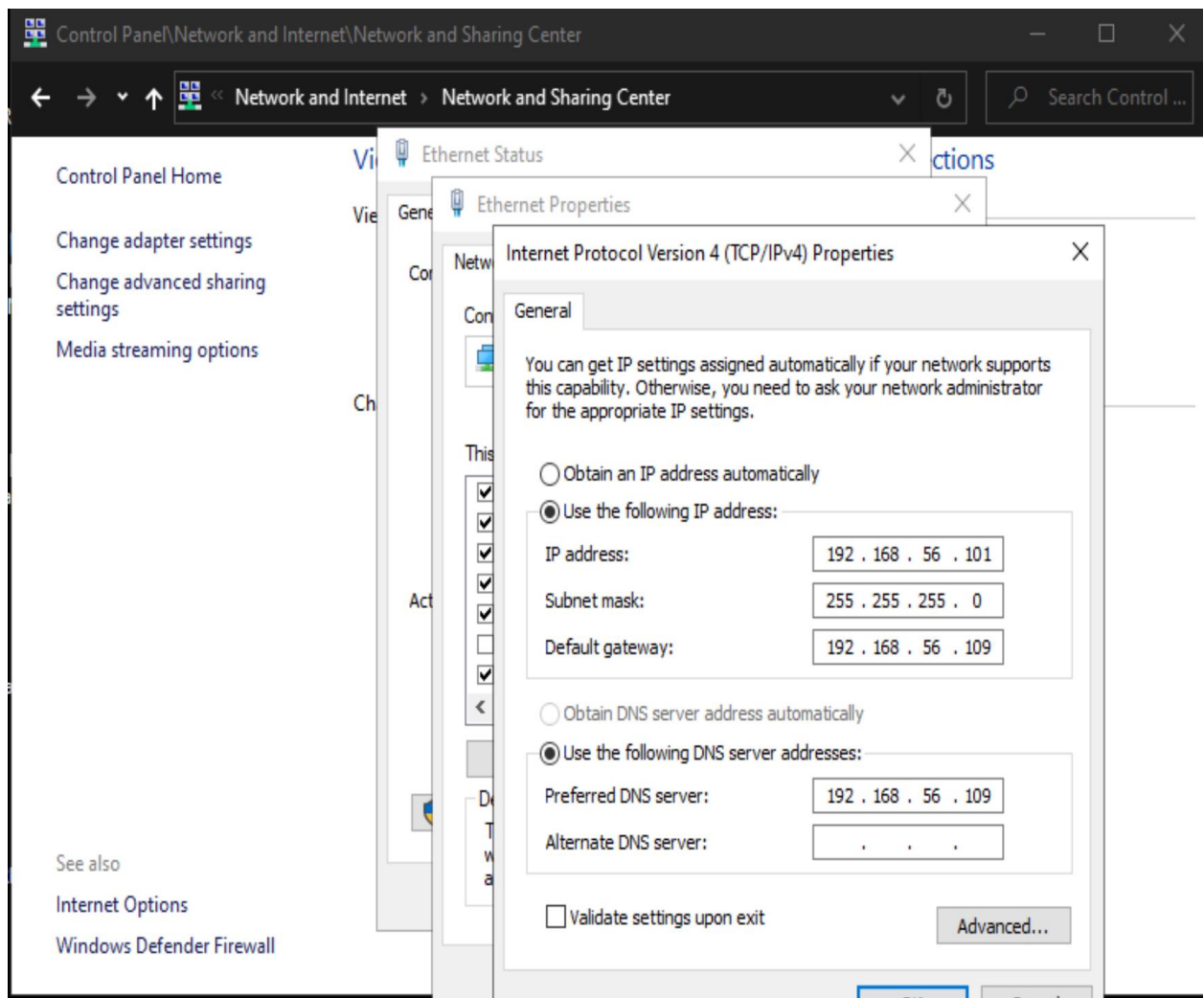




10. Startup the Kali VM. Open up a Web browser and go to a page you like. There is no Internet connection. We are halfway there.
11. You will need to change the static IP address for Kali. Follow these directions here: <https://miloserdov.org/?p=542> Note, make sure to right click on connections to edit the wired connection as shown in the image below. Set the address and gateway to 192.168.56.109.



12. Restart the Kali machine. Run ifconfig and verify the IP address is reset.
13. We really are half-way there! Now, onto the FlareVM machine.
14. Shut down the Kali VM.
15. From VirtualBox - right click on the Windows Flare VM and set it to **Host-only Adapter** like we did for the Ubuntu server in Step 10, setting it to **vboxnet0** just like we did for the Kali machine.
16. Start the FlareVM machine and login.
17. Open up a Web browser and go to a Web page. It should have no connection to the Internet.
18. Open up PowerShell and try to ping 192.168.56.109. No luck - nothing should come back.
19. Now, adjust the Windows network adapter from the Control Panel. Go to **Network and Internet** and then drill down to **Internet Protocol Version 4**. Edit it so it has the values in the image - and **exactly** like this. You are assigning your Windows machine a unique IP address and telling it to route **all** network traffic through the Kali VM:



20. Now shut down the Windows VM, restart it, and login.
21. Use PowerShell and ping 192.168.56.109. Still no luck.
22. Start the Kali VM and login.
23. Use PowerShell on Windows and ping 192.168.56.109. Now it should work!

For your reference, there are also tools that enable you to model more sophisticated network connections. An example is GNS3 - <https://www.gns3.com> - which would allow you to simulate hardware such as routers and switches for connectivity.

## Exercise 1: Capturing Malware Network Traffic

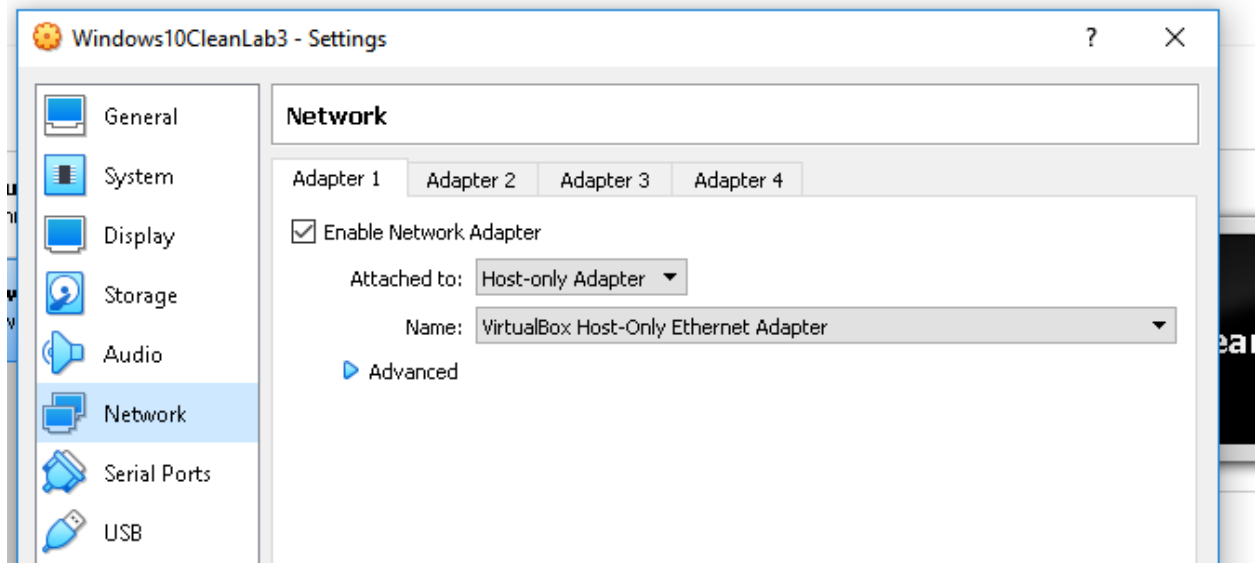
**What You Need:** The Windows FlareVM and Kali VMs configured earlier.

**Problem:** We will perform the first exercise from Chapter 3 in the book. This is the one that uses sales.exe. Note, this is fairly old malware and it has been run on Windows machines using

Windows VMs with no issues. We will set up the monitoring services, run the malware, and then collect results using screen captures.

Steps:

1. Make a clone of the Windows Flare VM you configured in case you make a mistake later.
2. First, let's do a sanity check. Right click on each and check the Network setting for Adapter 1. Make sure both are in host-only mode, like below. It might say **vboxnet** for the Name.



3. Right click on both and select **Settings**. Click on **General > Advanced**. Make sure the Drag and Drop option is enabled for **Host to Guest**.
4. Let's focus on the Kali machine. Login, open a terminal and run **ifconfig**. Make sure the IP address is **192.168.56.109**
5. Let's open Wireshark next on Kali. Open up a new terminal and issue the following command:  

```
> sudo wireshark
```
6. It opens the user interface. All you really need to know is that the fin icon starts collecting network traffic. You can give it a try, and then turn it off. When you turn it on it asks if you want to save the results - you'll have to do this in a second.
7. The Kali server is ready to go. Onto Windows.
8. Start the FlareVM and login.
9. Open PowerShell and run **ipconfig**. Verify the IP address of the machine is **192.168.56.101**
10. Let's use wireshark. Go to the Ubuntu machine and click the fin - start recording.
11. Go to the Windows machine and issue a ping command to ping the Kali machine. Like this - and let ping complete:  

```
> ping 192.168.56.109
```

12. Head back to the Ubuntu machine and go to Wireshark. Stop recording.
13. Next, sort the **Source** column. You should see the traffic from the Windows FlareVM machine. Take a screenshot and paste it below. Here is an easy way to do this - on the Windows machine running the VMs, open SnippingTool. Snip the part of the table that has the network traffic of interest. You can **ctrl-v** the image below:

Score: / 2pts

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.101	192.168.56.109	ICMP	74	Echo (ping) request id=0x0001, seq=5/1280, ttl=128 (reply in 2)
3	1.016318570	192.168.56.101	192.168.56.109	ICMP	74	Echo (ping) request id=0x0001, seq=6/1536, ttl=128 (reply in 4)
5	2.029393619	192.168.56.101	192.168.56.109	ICMP	74	Echo (ping) request id=0x0001, seq=7/1792, ttl=128 (reply in 6)
7	3.050394008	192.168.56.101	192.168.56.109	ICMP	74	Echo (ping) request id=0x0001, seq=8/2048, ttl=128 (reply in 8)
2	0.000014096	192.168.56.109	192.168.56.101	ICMP	74	Echo (ping) reply id=0x0001, seq=5/1280, ttl=64 (request in 1)
4	1.016331382	192.168.56.109	192.168.56.101	ICMP	74	Echo (ping) reply id=0x0001, seq=6/1536, ttl=64 (request in 3)
6	2.029408552	192.168.56.109	192.168.56.101	ICMP	74	Echo (ping) reply id=0x0001, seq=7/1792, ttl=64 (request in 5)
8	3.050408073	192.168.56.109	192.168.56.101	ICMP	74	Echo (ping) reply id=0x0001, seq=8/2048, ttl=64 (request in 7)
10	4.610149760	PcsCompu_0e:34:8d	PcsCompu_56:e1:21	ARP	42	192.168.56.109 is at 08:00:27:0e:34:8d
11	5.223471515	PcsCompu_0e:34:8d	PcsCompu_56:e1:21	ARP	42	Who has 192.168.56.101? Tell 192.168.56.109
9	4.610141878	PcsCompu_56:e1:21	PcsCompu_0e:34:8d	ARP	60	Who has 192.168.56.109? Tell 192.168.56.101
12	5.223732419	PcsCompu_56:e1:21	PcsCompu_0e:34:8d	ARP	60	192.168.56.101 is at 08:00:27:56:e1:21

14. Go back to the Windows machine. We need to learn about two more utilities you will use.
15. From the Windows icon, run Process Hacker, described in the book. Right click on the icon and run as **Administrator**. It lists all the running processes on the machine.
16. On the desktop is a folder named **noriben**. If you open the folder, you will see the python file to run Noriben and the procmon.exe file it needs to monitor processes.
17. Open up PowerShell as administrator.
18. Cd to the noriben folder on the desktop. It is:
 

```
>cd C:\Users\cse410\Desktop\noriben\noriben
```
19. Let's run noriben. To do this, open PowerShell as administrator and run:
 

```
>python.exe Noriben.py #the default python is 3.8
```
20. Noriben will start to run. Let it run for about 30 seconds. Then, as the instructions say, hit **ctrl-c**. This will kill the process. Note - you must wait until it fully returns and you are back at the command prompt. And hitting the **ctrl-c** took a couple tries for me before it worked.
21. Noriben will generate a text summary for you. Depending on what you did while it ran - you might see something in there.
22. Look in the noriben folder on the Desktop. You will see the Noriben files collected during the run - they will be dated. You are interested in the **Noriben...csv** file. The one with the .csv extension is a wrap-up of the collected information by time. Open up the csv and take a look at what was collected - it should make sense.
23. Delete all the files that Noriben generated for that run.
24. We are now ready to run the malware.
25. First, go to **Windows Security > Virus & threat protection > Virus & threat protection settings > manage settings** and ensure Windows Defender is off.
26. Download this zip from Piazza:
 

[VirusShare\\_51d9e2993d203bd43a502a2b1e1193da-2.zip](#)



Copy it to the FlareVM's desktop. Go to the desktop and unzip the file that starts with **VirusShare...** This has the malware in it. The password is **infected**

27. It will extract a single file that begins with: **33403...**
28. Rename this file with a **.exe** extension. Do not click it yet!
29. A sequence of things will happen when you run the malware you need to be aware of:
  - a. When you run it, a process with that name will start.
  - b. The process will create a new process - **ieplorer.exe** - and terminate the original process.
  - c. It will also **delete** the original file from the desktop and store ieplorer.exe in a different folder.
30. Here is what you need to do. First, start Noriben like we did in Step 19.
31. Bring **ProcessHacker** to the foreground.
32. Double click on the malware to run it.
33. Watch **ProcessHacker** - you might see the **33403...** process show up for a moment and then disappear. You will certainly see the **ieplorer.exe** process pop up. It will then turn red and **disappear**.
34. Wait another 10 seconds.
35. Stop Noriben on Windows FlareVM. Be sure to let it complete on its own and write the CSV file.

The book gives a detailed account of what happens beneath. I want you to do the following:

**For Noriben** - Go to the Noriben CSV file. Take two screenshots:

- 1.) Take a shot of the output where it creates the file ieplorer.exe. You need to know the directory where it writes the ieplorer.exe file to help answer a later question.

**Score: / 5 pts**

```
e", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", "Desired Access: Read Attributes, Read Control, Synchronize, Disposition: Open, Options: Synchro
rityFile", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "BUFFER OVERFLOW", "Information: Label"
rityFile", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", "Information: Label"
", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", ""
e", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", "Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/
cInformationFile", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", "CreationTime: 6/17/2021 10:18:41 AM, LastAccessTime: 6/17/2021 10:18:41 AM, Last
", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", ""
e", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", "Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/
cInformationFile", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", "CreationTime: 6/17/2021 10:18:41 AM, LastAccessTime: 6/17/2021 10:18:41 AM, Last
", "C:\Users\cse410\AppData\Roaming\ieplorer.exe", "SUCCESS", ""
e", "C:\Users\cse410\AppData\Roaming\ieplorer.exe:Zone.Identifier", "NAME NOT FOUND", "Desired Access: Read Attributes, Disposition: Open, Options: Open Repars
ey", "HKCU\Software\Microsoft\Internet Explorer\Main\FeatureControl", "SUCCESS", "Query: HandleTags, HandleTags: 0x400"
y", "HKCU\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_PROTOCOL_LOCKDOWN", "NAME NOT FOUND", "Desired Access: Query Value"
ey", "HKLM\SOFTWARE\Wow6432Node\Microsoft\Internet Explorer\Main\FeatureControl", "SUCCESS", "Query: HandleTags, HandleTags: 0x400"
```

- 2.) Take a shot of some registry key values being accessed by ieplorer.exe. It does this many times - I am not looking for any specific ones:

**Score: / 5 pts**

```

"10:18:30.5887428 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: HandleTags, HandleTags: 0x0"
"10:18:30.5887555 AM", "Explorer.EXE", "4032", "RegOpenKey", "HKCU\Software\Classes\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}\TreatAs", "NAME NOT FOUND", "Desir
"10:18:30.5887662 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: HandleTags, HandleTags: 0x0"
"10:18:30.5887741 AM", "Explorer.EXE", "4032", "RegOpenKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}\TreatAs", "NAME NOT FOUND", "Desired Access: Query
"10:18:30.5887829 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: Name"
"10:18:30.5887934 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: Name"
"10:18:30.5888010 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: HandleTags, HandleTags: 0x0"
"10:18:30.5888115 AM", "Explorer.EXE", "4032", "RegOpenKey", "HKCU\Software\Classes\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "NAME NOT FOUND", "Desired Acces
"10:18:30.5888228 AM", "Explorer.EXE", "4032", "RegQueryValue", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}\ActivateOnHostFlags", "NAME NOT FOUND", "Length:
"10:18:30.5888312 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: Name"
"10:18:30.5888395 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: HandleTags, HandleTags: 0x0"
"10:18:30.5888485 AM", "Explorer.EXE", "4032", "RegOpenKey", "HKCU\Software\Classes\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "NAME NOT FOUND", "Desired Acces
"10:18:30.5888579 AM", "Explorer.EXE", "4032", "RegQueryValue", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}\(Default)", "BUFFER OVERFLOW", "Length: 12"
"10:18:30.5888671 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: Name"
"10:18:30.5888767 AM", "Explorer.EXE", "4032", "RegQueryKey", "HKCR\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "SUCCESS", "Query: HandleTags, HandleTags: 0x0"
"10:18:30.5888908 AM", "Explorer.EXE", "4032", "RegOpenKey", "HKCU\Software\Classes\CLSID\{4db26476-6787-4046-b836-e8412a9e8a27}", "NAME NOT FOUND", "Desired Acces

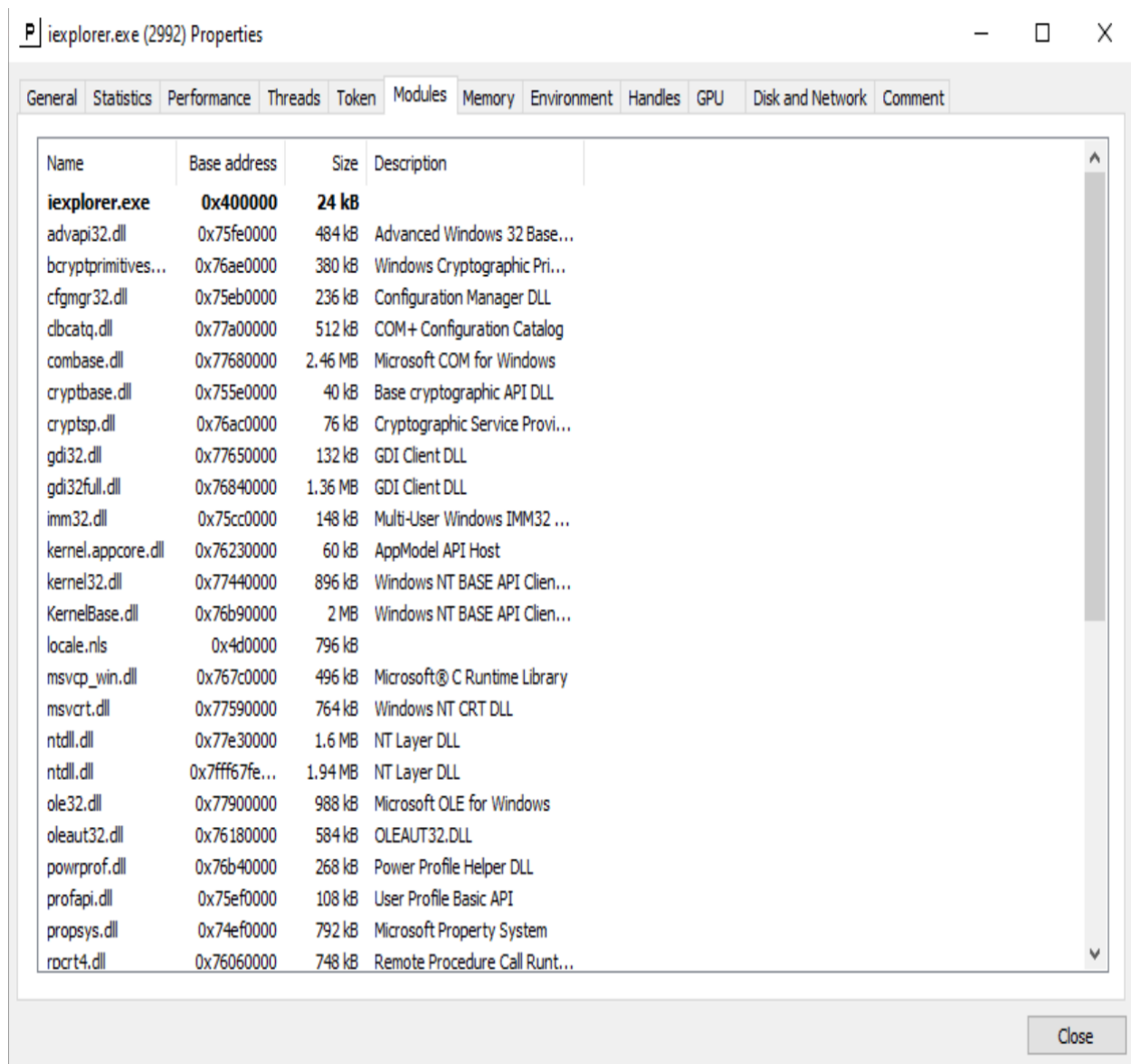
```

**For ProcessHacker 2:** Open PowerShell as admin. Go to the directory where iexplorer.exe is located - you know that from the screen shot above. Now, open up ProcessHacker right beside the PowerShell window. You are going to run iexplorer.exe and suspend it in ProcessHacker. Here is what to do.

- In ProcessHacker, right click on any running process. You will see there is a Suspend option. We are going to try and suspend iexplorer.exe but we need to be fast.
  - In the upper right of ProcessHacker is a search window. Type *iexplorer.exe* in that window. All the processes should disappear but when we do run iexplorer.exe - it will show up.
  - Go to PowerShell and run iexplorer.exe. Like this:  
**>. iexplorer.exe**
  - Immediately go ProcessHacker and right click and suspend iexplorer.exe when it appears.
  - Note - you will need to be fast and might have to repeat the part where you run iexplorer.exe from PowerShell - but you can do it!
  - Once suspended, double click iexplorer.exe in ProcessHacker to get its details. Click on the tab named Modules.
- 3.) Take a screen shot of the Modules tab. Be sure to show that it has loaded crypto related DLLS in this screenshot - they should be close to the top.

**Score: / 5 pts**





Later on, a couple of exercises down, you will open up iexplorer.exe in CFF Explorer and notice something. You can terminate the iexplorer.exe process from ProcessHacker.

## Exercise 2: Capture Content With WireShark

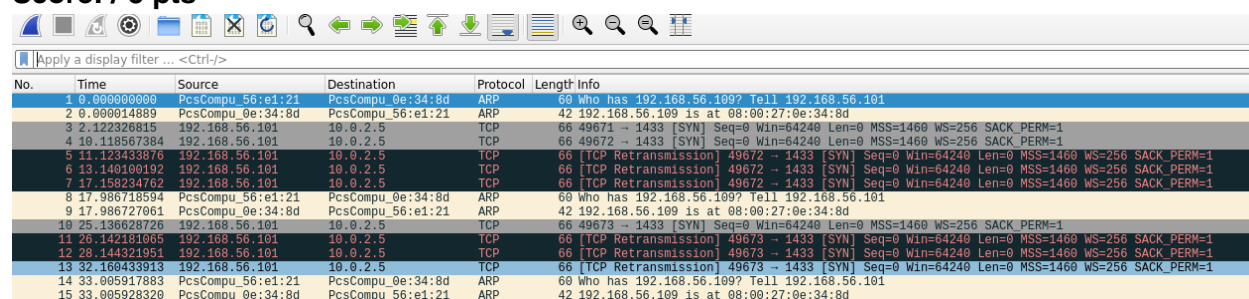
The Windows FlareVm and Kali machine are still connected in host only mode. We will now simulate network traffic from a command and control (C2) client. Download the following file which is a command and control client developed by MITRE to simulate real malware functionality of the FIN7 APT:

[c2fin7.exe](#)

This tool will certainly emit network traffic and it will be realistic malware traffic. For this exercise, do the following.

1. Copy c2fin7.exe to the Windows Flare VM desktop.
2. Make certain Kali and the Flare VM are both up and connected in host only mode.
3. Begin capture of network traffic on the Kali machine.
4. Double click c2fin7.exe to run the C2 client.
5. Watch the network traffic come in on the Kali machine.
6. Take a screen shot of a SYN request coming from the FlareVM (198.162.56.101) and one or maybe two TCP retransmissions

Score: / 5 pts



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_0e:34:8d	PcsCompu_0e:34:8d	ARP	60	Who has 192.168.56.109? Tell 192.168.56.101
2	0.000014889	PcsCompu_0e:34:8d	PcsCompu_56:e1:21	ARP	42	192.168.56.109 is at 08:00:27:0e:34:8d
3	2.122326815	192.168.56.101	10.0.2.5	TCP	66	49671 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
4	10.118567384	192.168.56.101	10.0.2.5	TCP	66	49672 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	11.123433876	192.168.56.101	10.0.2.5	TCP	66	[TCP Retransmission] 49672 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	13.140100192	192.168.56.101	10.0.2.5	TCP	66	[TCP Retransmission] 49672 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	17.158234762	192.168.56.101	10.0.2.5	TCP	66	[TCP Retransmission] 49672 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
8	17.986718594	PcsCompu_56:e1:21	PcsCompu_0e:34:8d	ARP	60	Who has 192.168.56.109? Tell 192.168.56.101
9	17.986727061	PcsCompu_0e:34:8d	PcsCompu_56:e1:21	ARP	42	192.168.56.109 is at 08:00:27:0e:34:8d
10	25.136828726	192.168.56.101	10.0.2.5	TCP	66	49673 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
11	26.142181065	192.168.56.101	10.0.2.5	TCP	66	[TCP Retransmission] 49673 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	28.144321951	192.168.56.101	10.0.2.5	TCP	66	[TCP Retransmission] 49673 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
13	32.160433913	192.168.56.101	10.0.2.5	TCP	66	[TCP Retransmission] 49673 → 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
14	33.005917883	PcsCompu_56:e1:21	PcsCompu_0e:34:8d	ARP	60	Who has 192.168.56.109? Tell 192.168.56.101
15	33.005928320	PcsCompu_0e:34:8d	PcsCompu_56:e1:21	ARP	42	192.168.56.109 is at 08:00:27:0e:34:8d

We talked about Inetsim in class. Had I had you run it and configure it to reply to TCP - you would have at least seen some [RST, ACK] coming back meaning the Kali machine realized a TCP SYN request came through but the IP address to field it could not be found.

### Exercise 3: DLL Analysis

This exercise will have you look at two DLL files and an EXE. Initially, I had wanted you to run the DLLs but had some difficulties getting rundll32.exe to do what I wanted it to. However, DLLs are important and you should have some hands-on experience with them. For this Exercise:

1. Import the FlareVM with the Host-Only forwarding to the Kali VM. If you reuse the FlareVM from exercise 1 you might not get the same results.
2. Download the following file with the malware DLLs from here:

[dlls.zip](#)

3. Unzip the file. Some things to know:
  1. The pwd for the zip file is always **Password123!**
  2. Be sure Windows Defender is off. If you do not - the DLL in the zip will be removed as soon as you extract it!
4. You will see these two files - they are both DLLs though neither has a DLL extension.

12311c40e7e642240ab77024c5bd0ab1f602fa9f83c43cd7cf7c462b7ac05e89  
4fb41362aa8ccd9550ba96e74f52687c72e6751029a00e2037080f3ce86d90a3

5. Go to the Windows icon in the lower left and search for “CFF Explorer”. You will see a tool with a red jalapeno pepper. It is a great resource to evaluate DLLs! Open that tool

***12311c40e7e642240ab77024c5bd0ab1f602fa9f83c43cd7cf7c462b7ac05e89 analysis***

First, what type of malware is this sample? You have the fingerprint for this malware:  
12311c40e7e642240ab77024c5bd0ab1f602fa9f83c43cd7cf7c462b7ac05e89

**Score: / 2 pts**

Trojan

Next, open this DLL with CFF Explorer. In the Imports section you will see the other DLL files this DLL imports. This list is critical as it provides access to the Windows API functionality the DLL uses and often explains what the malware is likely to do when it runs.

Look at the ADVAPI32.dll and the list of functions it uses from this DLL. Provide **two things** this malware likely does that is typical of malware running in a Windows environment as it attempts to gain persistence.

**Score: / 4 pts**

Creates a process and escalates its privileges.

Delete the original process

Modify and set new Reg Key values.

Look at the WININET.dll and the functions it uses from this DLL. What do you suspect this malware will do when it runs and has access to an Internet connection?

**Score: / 4 pts**

This malware tries to establish a connection on the internet and send a request. It reads a canonical Url and reads a file. Once done it closes the connection. Then it reads the url from that file to open another url.

***4fb41362aa8ccd9550ba96e74f52687c72e6751029a00e2037080f3ce86d90a3 analysis***

First, what type of malware is this sample? You have the fingerprint for this malware:  
4fb41362aa8ccd9550ba96e74f52687c72e6751029a00e2037080f3ce86d90a3

**Score: / 2 pts**

Trojan

Next, open this DLL with CFF Explorer. Look at the import list and identify a DLL that is used mostly by installers. Note, searching Microsoft documentation and answers might be a good place to start.

**Score: / 4 pts**

SHELL32.dll

Next, look at the KERNEL32.dll. There is a function from this DLL used by the malware that is typical malware behavior as it wants to understand if it is being watched. What is the name of that function?

**Score: / 2 pts**

IsWow64Process - Determines whether the specified process is running under WOW64 or an Intel64 or x64 processor.

Last part of this question. Go back to where iexplorer.exe is, the malware from Exercise 1, and open it in CFF Explorer. Look at its imports. Does it import any obvious crypto libraries? Yes or no.

**Score: / 2 pts**

NO

## Exercise 4 - YARA Rule

In this exercise you will create a single YARA rule. However, you will need to do some research to create it. Below is a link to a zip file that contains some classes.dex files from Android malware.

[dex\\_files.zip](#)

Classes.dex are the compiled Java code that Android runs. The files are only harmful on Android phones and in their current form are not a danger to your machine. Feel free to do this analysis wherever you would like, you will need Yara installed. Here is what you need to do. Create a single YARA rule that identifies when a classes.dex file likely (because we can't be 100% certain using basic techniques) changes the permission of a file on an Android phone **and** makes a directory. I believe 18 of the dex files in the zip do this.

**Score: / 4 pts**

```
rule file_permission
{
  meta:
    description = "YARA RULE TO DETECT CHANGE IN FILE PERMISSION AND CREATE
    DIRECTORY"

  strings:
    $a = "chmod"
    $b = "mkdir"
  condition:
    ($a and $b)
}
```