

CMSC 426 Computer Security Lab 4

Name: Joshua Adewunmi

Assigned: 11/23/2021

Due: 12/9/2021 at 11:59pm

Total points: 100

Virtual Machine Setup

To set up the lab, you will need to import the lab VM into VirtualBox. You can download the VM from the following link:

❓ <https://drive.google.com/file/d/1pIPgPrPXyU22KfvK0C4zYMIInwusXwVjx/view?usp=sharing>

Import the VM into VirtualBox by using the menu option File -> Import Appliance and selecting the downloaded Lab4_VM.ova file.

Start the VM and log in. The password to the VM is `infected`. By default, the VM's network settings are configured to be in NAT mode, which allows it to connect to the internet.

Part 1: Basic Static Analysis (25 pts)

In this lab you will analyze CMSC426Lab4.exe, a malware sample which has been defanged so that it cannot communicate with its original command and control (C&C) infrastructure. It is located at `C:\users\student\CMSC426Lab4.exe` on the VM. **You should treat this file as if it were live malware.** Make sure to follow all instructions carefully.

UNDER NO CIRCUMSTANCES should you run the malware until instructed to do so. You will later be given directions to configure the network settings of the VM so that it is isolated on your VM and cannot reach your host machine or the internet. In this section of the lab you will be given directions to analyze the malware statically – you will obtain all answers for Part 1 **without running the file.** You may use any of the static analysis tools provided on this VM to answer the following questions about CMSC426Lab4.exe:

1. What is the MD5 hash of CMSC426Lab4.exe? (3 pts) The MD5 hash of CMSC426Lab4.exe is 73d88fe552e214ab57a008c42591f83e

2. In a few sentences, describe why file hashes (MD5, SHA-1, SHA-256, etc.) are useful to malware analysts. (6 pts) file hashes are useful to malware analysts because it keeps track of individual malware samples. They can share files with one another because the hashes in the files are somewhat like unique indicators or Ids for whatever file is being talked about, so a file with the exact same hash would be the exact same file.

3. Imagine an antivirus product that has a list of file hashes corresponding to every single malicious file that has been previously seen. The antivirus uses this list of file hashes to identify malware on the filesystem of the computer it is installed on. Describe some weaknesses of this approach. Make sure to give an example of how a malware author could ensure that their malware goes undetected. (8 pts) Some weaknesses of this approach is that since it is not a pattern based signature a malware author can take the original malware, change even a tiny bit of it and it will have a different hash which would make it hard for people to know what was done thereby ensuring that their malware goes undetected. Another weaknesses is that sometimes the antivirus won't have something to compare against so it might not be able to detect/ fight against that malware

4. How many DLL files does the malware import functions from? (4pts) 3

5. What is the compilation timestamp of this malware sample? (4pts) 2011-08-22 05:08:27

Part 2: Writing a YARA Rule (40 pts)

In Part 2 of the lab, you will write a YARA rule to detect CMSC426Lab4.exe and another malicious file that is very similar to it (which has been hidden somewhere on the VM). During this part of the lab, **do not run the malware sample**. You will be able to do the entire part using static analysis tools such as the `strings` command-line utility and a disassembler such as Ghidra or Ida Pro.

YARA is a pattern-matching tool that is widely used by malware analysts for writing malware signatures. It can detect files based on strings, bytes sequences, file metadata, and other patterns.

Documentation for YARA is linked below:

- 📄 <https://yara.readthedocs.io/en/stable/>
- 📄 <https://yara.readthedocs.io/en/stable/writingrules.html>

To scan a file with a YARA rule, open a command prompt on your VM and run `yara64.exe`. Here are a few example commands.

To scan the file `fileToScan.exe` with a YARA rule defined in `myRule.yar`, run:

```
📄 yara64.exe myRule.yar fileToScan.exe
```

To scan all the files in a folder named `folderToScan/` instead of a single file, run:

```
📄 yara64.exe -r myRule.yar folderToScan/
```

In this lab, you will write a YARA rule for `CMSC426Lab4.exe` containing three strings and a byte sequence. First, use the `strings` command-line utility to identify strings matching the following descriptions.

1. Provide the string `CMSC426Lab4.exe` that is a URL: (3 pts) _

`http://www.thisisthemalwareswebsite.com/start.htm`

2. Provide the string `CMSC426Lab4.exe` that is a User Agent: (3 pts) _____

`Mozilla/4.0 (ThisIsTheEvilUserAgent; MSIE 7.0; Windows NT 5.1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)`

3. Provide the string `CMSC426Lab4.exe` that is a file path: (3 pts) `C:\badfile.exe`

Next, we will identify a unique byte sequence in CMSC426Lab4.exe. Often, this can be done by selecting bytes from a unique function in the file (such as a custom encryption routine). In this case, you will select a sequence of bytes from one of the functions that handles much of the malware sample's networking capabilities. Instructions for this part of the lab will be provided for Ghidra, but you are welcome to use Ida Pro instead if you wish.

Run Ghidra and select `File -> New Project`. Create a non-shared project with a name of your choice. Next, select `File -> Import File`. Navigate to CMSC426Lab4.exe, select it, and hit `OK`.

Double click CMSC426Lab4.exe under your project to open it. Ghidra will begin analyzing it; you may choose all the default analysis options when prompted.

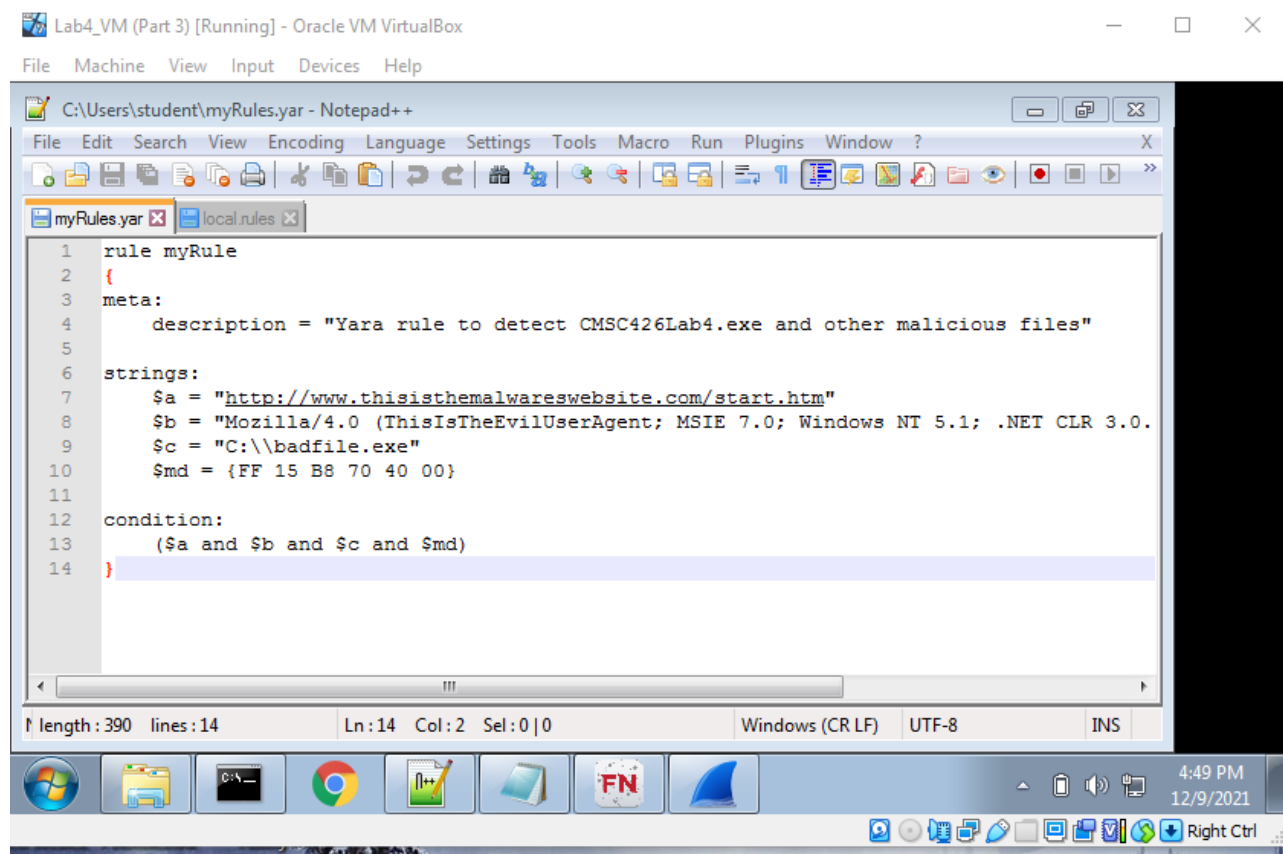
4. At what address is the Windows API function `InternetOpenA` called inside of CMSC426Lab4.exe? Ghidra and Ida Pro provide this address as a hexadecimal number – please leave your answer in this format. (4 pts) 004011f3

Switch to the listing view in Ghidra, showing the call to `InternetOpenA`. Find the bytes representing the machine code for the five assembly instructions starting with `CALL InternetOpenA`. This should be approximately two dozen bytes.

5. What are the bytes corresponding to these five instructions? Leave them in hexadecimal, separated by spaces (6 pts). Ff 15 b8 70 40 00

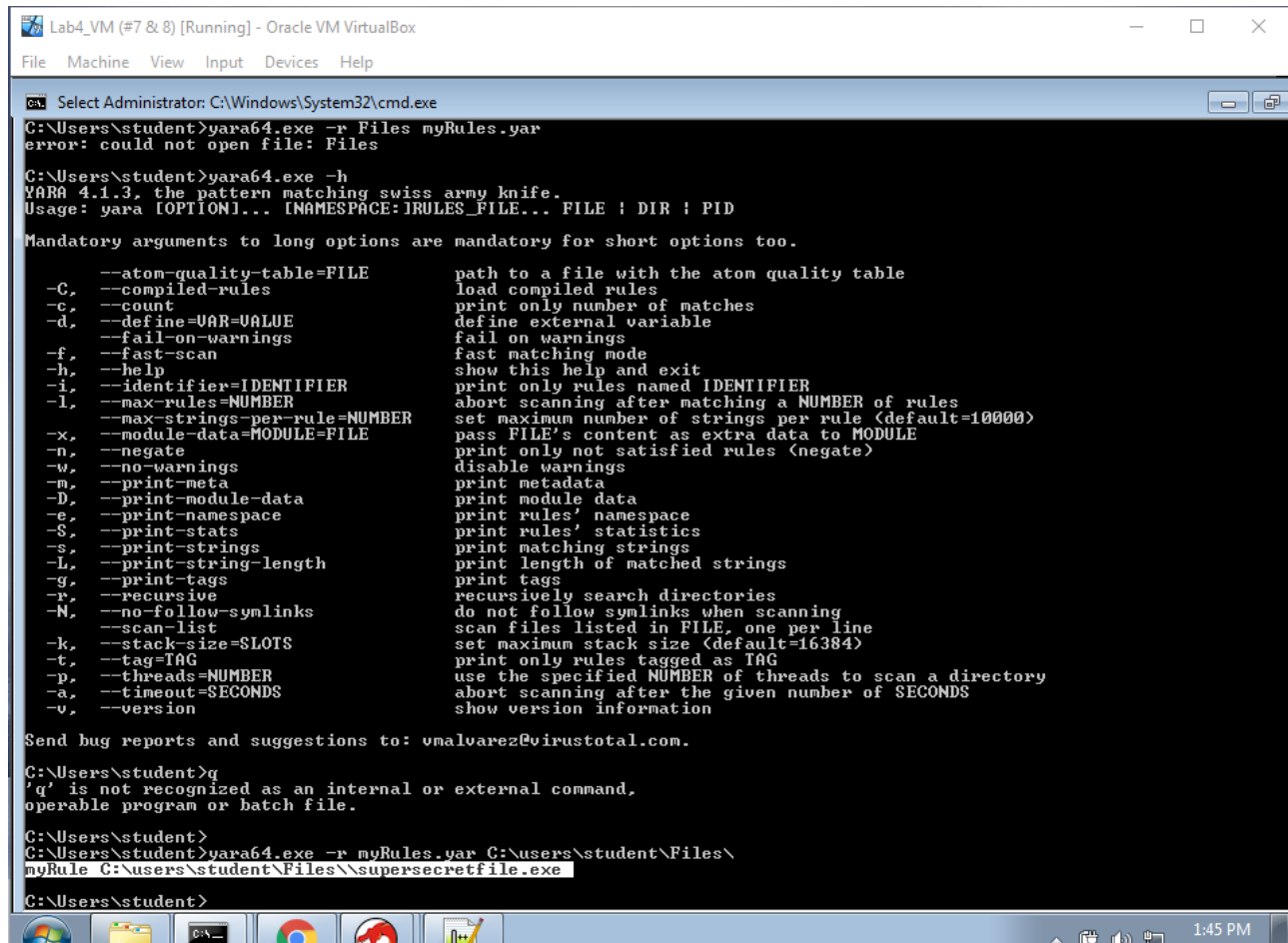
Write a YARA rule that matches files which contain the three strings from questions 1-3 and the byte sequence from question 5.

6. Provide a screenshot of your YARA rule below (10 pts)



Scan the folder `C:\users\student\Files\` using your YARA rule. One of the files in System32 is very similar to CMSC426Lab4.exe and it should be detected by your rule.

7. Provide a screenshot of your rule detecting the similar file in System32. It should not detect any other files. (8 pts)



```
C:\Users\student>yara64.exe -r Files myRules.yar
error: could not open file: Files

C:\Users\student>yara64.exe -h
YARA 4.1.3, the pattern matching swiss army knife.
Usage: yara [OPTION]... [NAMESPACE:RULES_FILE]... FILE ! DIR ! PID

Mandatory arguments to long options are mandatory for short options too.

  -C, --atom-quality-table=FILE      path to a file with the atom quality table
  -c, --compiled-rules              load compiled rules
  -e, --count                       print only number of matches
  -d, --define=VAR=VALUE            define external variable
  -f, --fail-on-warnings            fail on warnings
  -f, --fast-scan                   fast matching mode
  -h, --help                        show this help and exit
  -i, --identifier=IDENTIFIER       print only rules named IDENTIFIER
  -l, --max-rules=NUMBER            abort scanning after matching a NUMBER of rules
  -L, --max-strings-per-rule=NUMBER set maximum number of strings per rule (default=10000)
  -x, --module-data=MODULE=FILE     pass FILE's content as extra data to MODULE
  -n, --negate                      print only not satisfied rules (negate)
  -w, --no-warnings                disable warnings
  -m, --print-meta                 print metadata
  -D, --print-module-data          print module data
  -e, --print-namespace            print rules' namespace
  -s, --print-stats               print rules' statistics
  -S, --print-strings              print matching strings
  -L, --print-string-length        print length of matched strings
  -g, --print-tags                 print tags
  -r, --recursive                  recursively search directories
  -M, --no-follow-symlinks         do not follow symlinks when scanning
  -l, --scan-list                  scan files listed in FILE, one per line
  -k, --stack-size=SLOTS          set maximum stack size (default=16384)
  -t, --tag=TAG                   print only rules tagged as TAG
  -p, --threads=NUMBER            use the specified NUMBER of threads to scan a directory
  -a, --timeout=SECONDS           abort scanning after the given number of SECONDS
  -v, --version                   show version information

Send bug reports and suggestions to: vmalvarez@virustotal.com.

C:\Users\student>q
'q' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\student>
C:\Users\student>yara64.exe -r myRules.yar C:\users\student\Files\
myRule C:\users\student\Files\supersecretfile.exe
C:\Users\student>
```

8. What is the MD5 hash of the similar file that your rule detected in System32? (3 pts) CF1CE84CAFDF6465FC9AC0548716119E

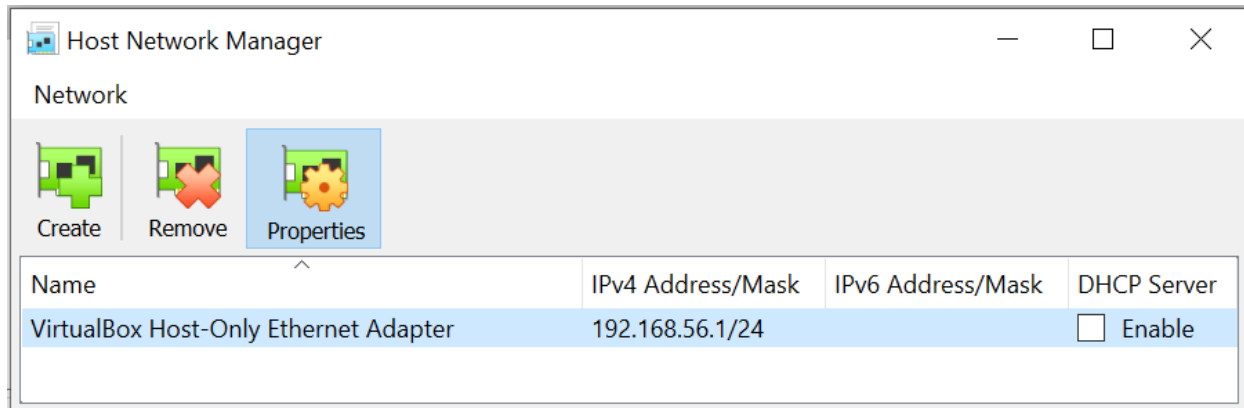
Setup for Dynamic Analysis

After completing Part 2, download this Snort configuration file:

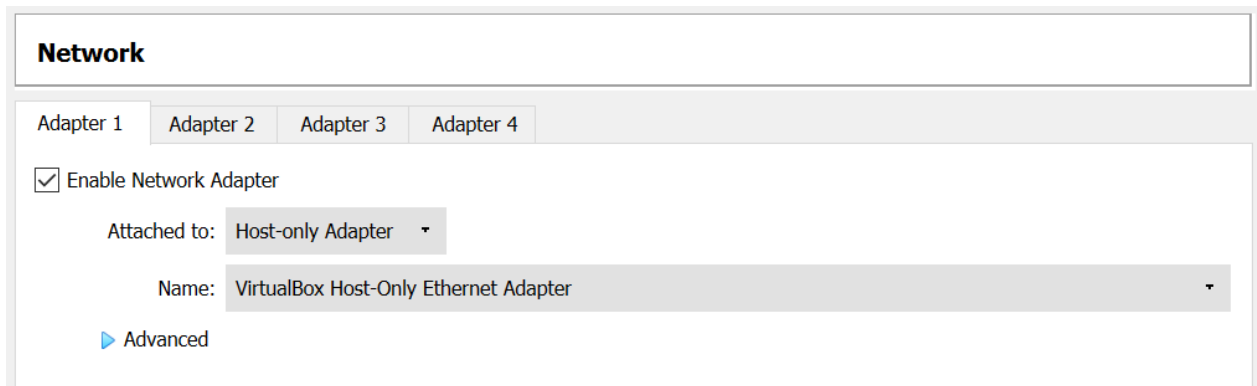
2 <https://drive.google.com/file/d/1S9858uywk0A9DkJ-KJmG1GFs4386T6fr/view?usp=sharing>

Replace C:\Snort\etc\snort.conf with the version that you downloaded.

Once you have done this, shut down your VM. You will need to configure its networking settings in VirtualBox to prepare for running the malware safely. First, select `File -> Host Network Manager` and edit the host networking interface that you have been using for Labs 2 and 3. Make sure that DHCP is **disabled**, then right-click and select `Properties` to edit the IPv4 address to be 192.168.56.1. (Once done, it should display as 192.168.56.1/24).

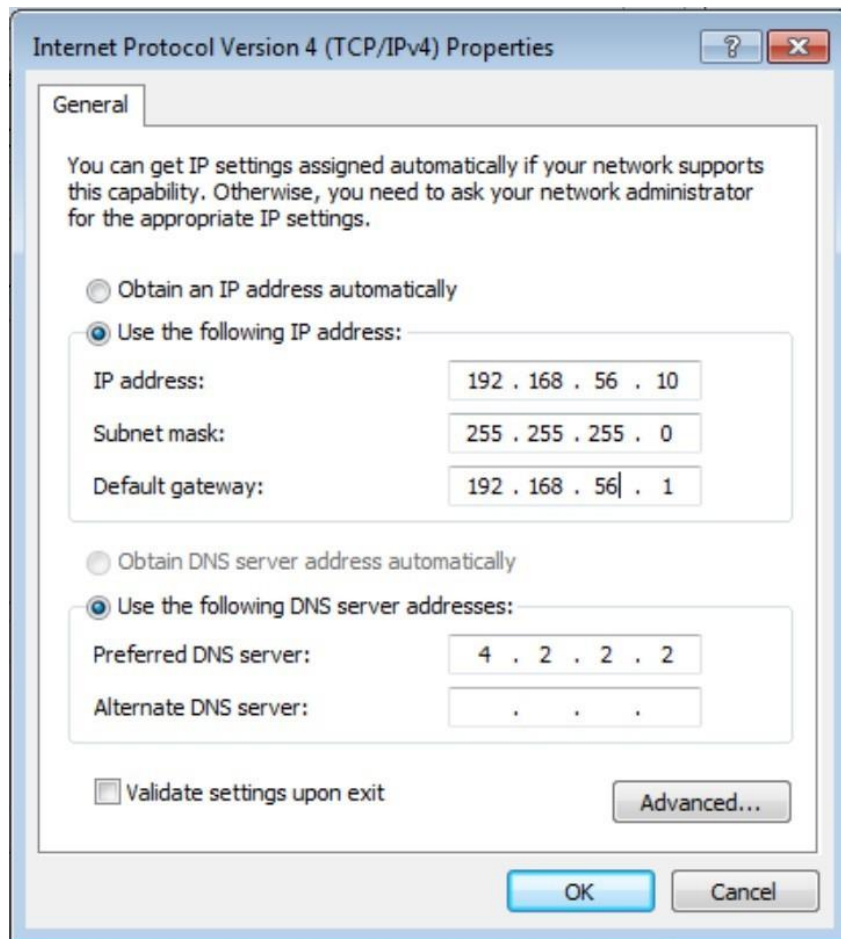


Next, go to the network settings for your VM (Settings -> Network). Make sure that Adapter 1 is enabled, that it is attached to Host-only Adapter, and that its name matches the name of the host network interface that you just created (e.g. VirtualBox Host-Only Ethernet Adapter)



Once you are finished configuring the network settings,

On your VM, go to Control Panel -> Network and Internet -> Network and Sharing Center -> Local Area Connection. Select Properties -> Internet Protocol Version 4 (TCP/IPv4) -> Properties. Set your VM's IP address and DNS server to the following values:



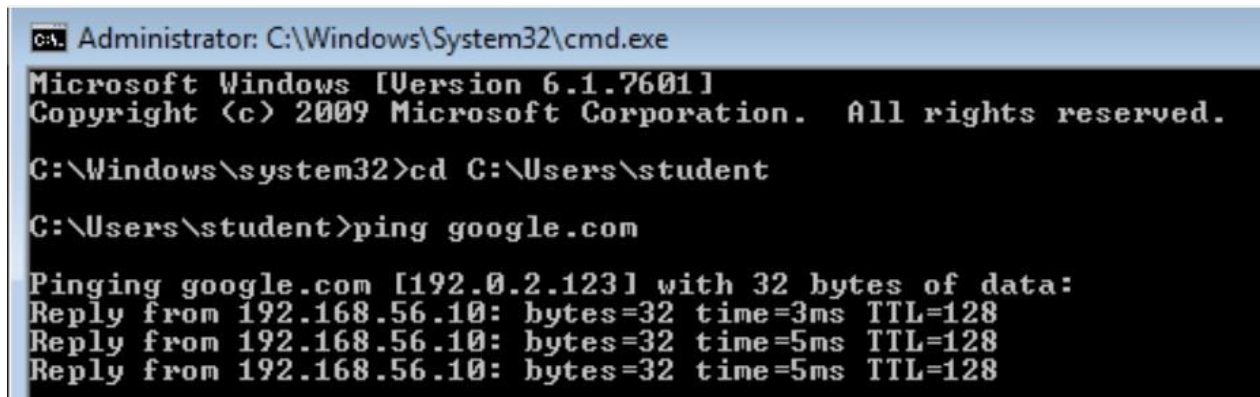
Once you have configured your VM's network settings, **take a snapshot of your VM in VirtualBox**. You need to do this to ensure that you can restore your VM to a state prior to when it was infected by the malware. This step is **NOT OPTIONAL**.

Part 3: Writing a Snort Rule (35 pts)

In Part 3 of the lab you will write a Snort rule to detect the network traffic of CMSC426Lab4.exe. Snort is a network-based intrusion detection system.

First, you will use FakeNet-NG and Wireshark to capture the malware's network traffic. FakeNet-NG is a tool that intercepts and redirects network traffic. This will prevent the malware from interacting with your host machine, and it will trick the malware into believing that it is connected to the internet. Since many types of malware expect to be connected to the internet in order to receive instructions from a C&C server, this tool allows analysts to observe further behavior from the malware than if the VM were fully disconnected from the network.

First, ensure that you have **taken a snapshot of your VM** with the network settings configured as specified by the earlier directions. You must take a snapshot prior to running FakeNet-NG. Next, run FakeNet-NG (and allow it through the firewall, if prompted). To test that it is working correctly, ping google.com from the command line. It should respond with the IP address 192.168.56.10, showing that FakeNet-NG is intercepting and redirecting the ICMP traffic.



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\student

C:\Users\student>ping google.com

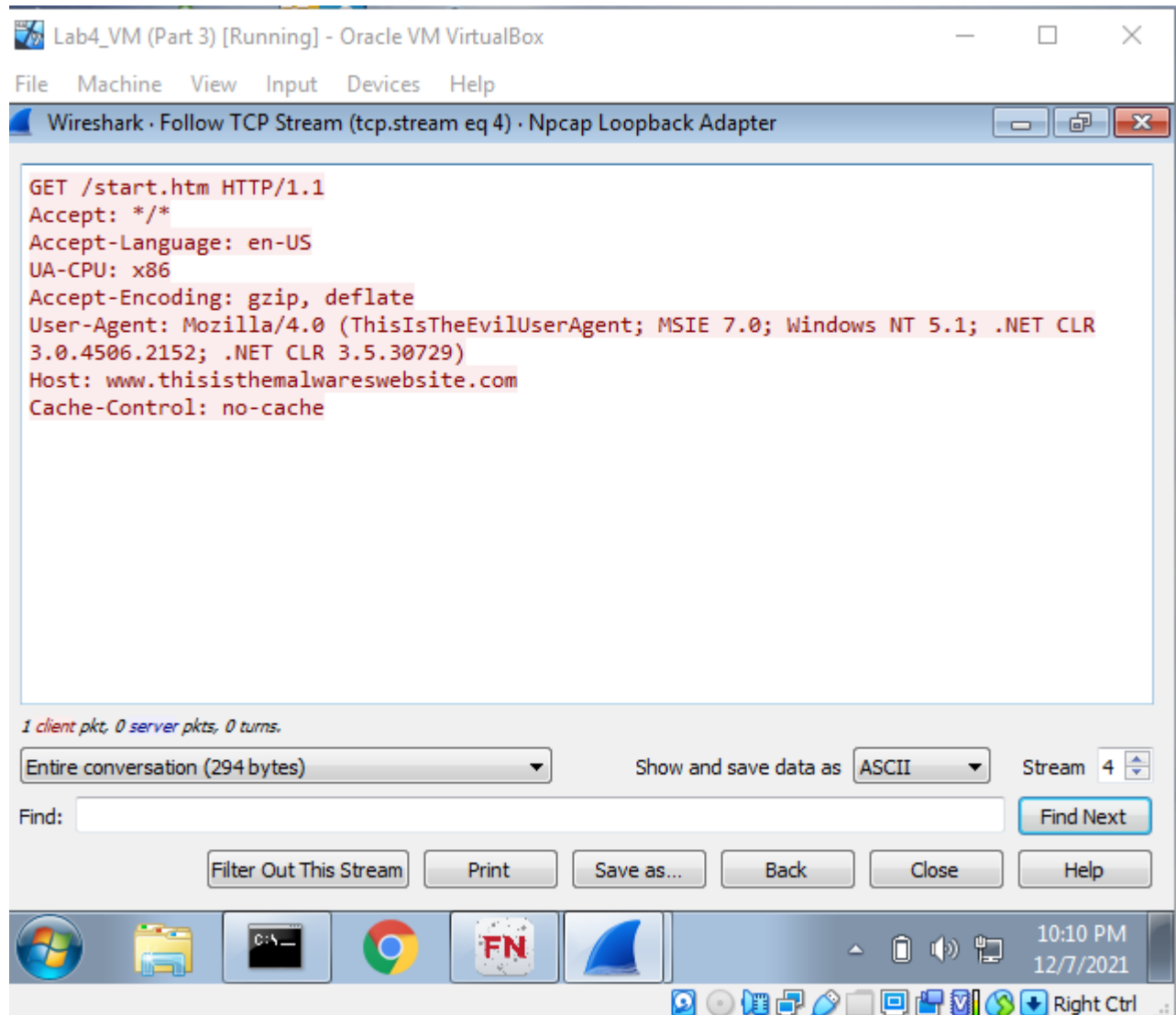
Pinging google.com [192.0.2.123] with 32 bytes of data:
Reply from 192.168.56.10: bytes=32 time=3ms TTL=128
Reply from 192.168.56.10: bytes=32 time=5ms TTL=128
Reply from 192.168.56.10: bytes=32 time=5ms TTL=128
```

Next, run Wireshark and begin capturing network traffic on the Npcap Loopback Adapter. Leave FakeNet-NG and Wireshark running so that you can capture the malware's network traffic.

With FakeNet-NG and Wireshark running, open an **Administrator** command prompt and run `cd C:\Users\student\` to change to the directory in which CMSC426Lab4.exe is located. Then run CMSC426Lab4.exe in the Administrator command prompt.

The malware should repeatedly produce the same network traffic at short intervals. In Wireshark, identify traffic sent by CMSC426Lab4.exe. Right click one of these packets and select Follow TCP Stream.

1 Provide a screenshot showing the output of the Follow TCP Stream in Wireshark. It should show the user agent string that you found in Part 2, and the Host field in the screenshot should be based on the URL you found in Part 2. (10 pts)



FakeNet-NG saves .pcap files containing the redirected network traffic to `C:\Users\student\Desktop\fakenet_logs\`. You will use Snort to detect network traffic in this .pcap that was produced by the malware sample.

Some resources for writing Snort rules are provided below:

- ❏ https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/596/original/Rules_Writers_Guide_to_Snort_3_Rules.pdf
- ❏ <https://www.rapid7.com/blog/post/2016/12/09/understanding-and-configuring-snort-rules/>
- ❏ <https://coralogix.com/blog/writing-effective-snort-rules-for-the-sta/>
- ❏ <https://resources.infosecinstitute.com/topic/snort-rules-workshop-part-one/>

Note that Snort 2.9.14.1 is installed on your VM, so some syntax for Snort 3 may not work.

Write a Snort rule for the malware's network traffic. Follow these instructions when writing your snort rule:

- ❏ Your rule should match HTTP traffic containing the user agent string from Part 2
 - You do not need to match the entire user agent string
 - Just up to (not including) the first semicolon
- ❏ Your rule should use the `flow` keyword to match only outbound traffic
- ❏ Choose a unique Snort ID (`sid`) for your rule that is greater than 1000000
- ❏ Your rule should also contain a message (`msg`) about the detected network traffic.
- ❏ For readability, please make a multi-line Snort rule, with each line ending with a `\` (see the Rapid7 resource linked above for more details)

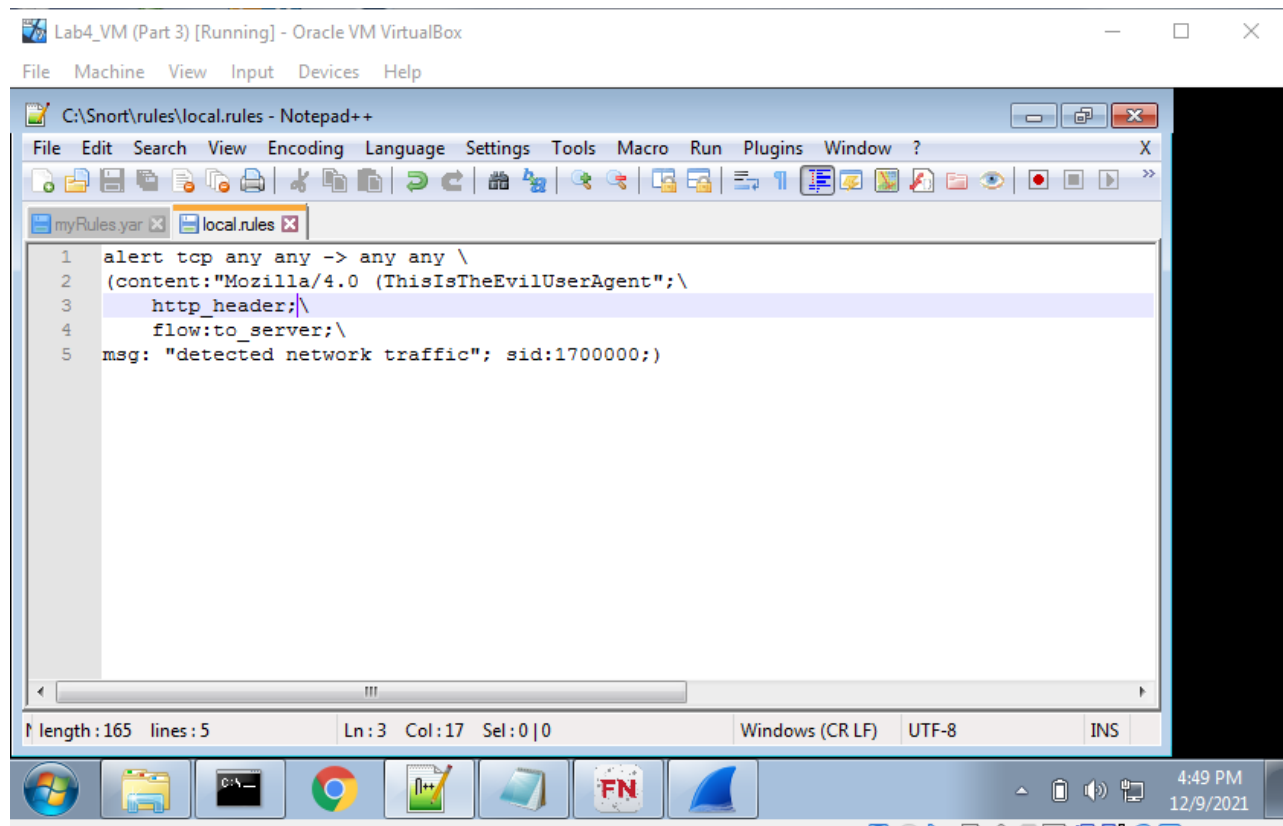
You should have already overwritten the old `snort.conf` file before configuring your VM's network settings. The Snort rule should be saved to `C:\Snort\rules\local.rules`

To run your Snort rule, use the following command in the command prompt:

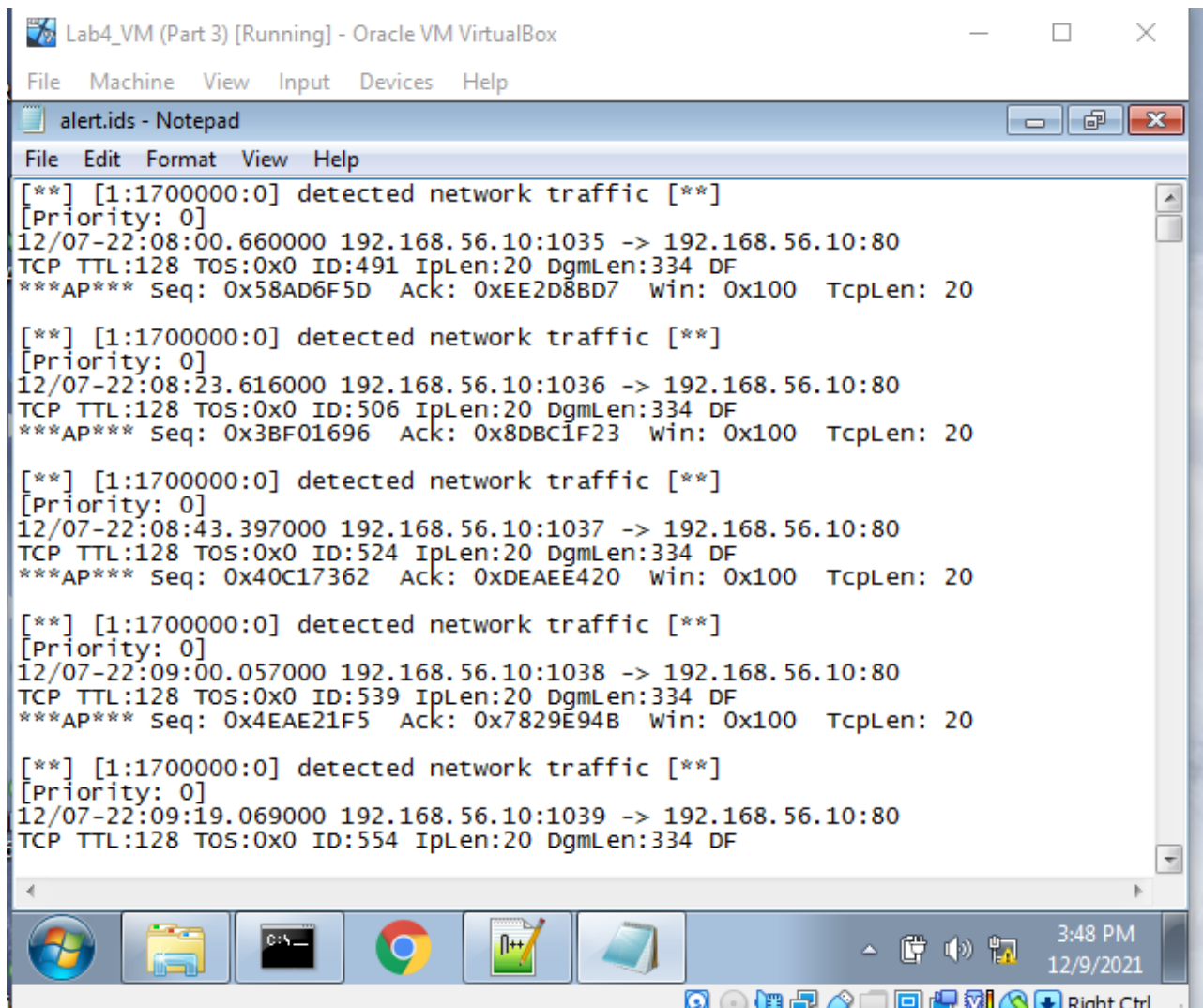
- ❏ `C:\Snort\bin\snort.exe -c C:\Snort\etc\snort.conf -r [path to .pcap file] -l C:\Snort\log`

Alerts from your rule will be written to `C:\Snort\log\alert.ids`

2 Provide a screenshot of your Snort rule below (15 pts)



3 Provide a screenshot of your `alert.ids` file, showing that it has matched traffic originating from the malware sample (10 pts)



```
Lab4_VM (Part 3) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

alert.ids - Notepad
File Edit Format View Help

[**] [1:1700000:0] detected network traffic [**]
[Priority: 0]
12/07-22:08:00.660000 192.168.56.10:1035 -> 192.168.56.10:80
TCP TTL:128 TOS:0x0 ID:491 IpLen:20 DgmLen:334 DF
***AP*** Seq: 0x58AD6F5D Ack: 0xEE2D8BD7 win: 0x100 TcpLen: 20

[**] [1:1700000:0] detected network traffic [**]
[Priority: 0]
12/07-22:08:23.616000 192.168.56.10:1036 -> 192.168.56.10:80
TCP TTL:128 TOS:0x0 ID:506 IpLen:20 DgmLen:334 DF
***AP*** Seq: 0x3BF01696 Ack: 0x8DBC1F23 win: 0x100 TcpLen: 20

[**] [1:1700000:0] detected network traffic [**]
[Priority: 0]
12/07-22:08:43.397000 192.168.56.10:1037 -> 192.168.56.10:80
TCP TTL:128 TOS:0x0 ID:524 IpLen:20 DgmLen:334 DF
***AP*** Seq: 0x40C17362 Ack: 0xDEAEE420 win: 0x100 TcpLen: 20

[**] [1:1700000:0] detected network traffic [**]
[Priority: 0]
12/07-22:09:00.057000 192.168.56.10:1038 -> 192.168.56.10:80
TCP TTL:128 TOS:0x0 ID:539 IpLen:20 DgmLen:334 DF
***AP*** Seq: 0x4EAE21F5 Ack: 0x7829E94B win: 0x100 TcpLen: 20

[**] [1:1700000:0] detected network traffic [**]
[Priority: 0]
12/07-22:09:19.069000 192.168.56.10:1039 -> 192.168.56.10:80
TCP TTL:128 TOS:0x0 ID:554 IpLen:20 DgmLen:334 DF
```