



PRIFYSGOL CYMRU
Y Drindod Dewi Sant
UNIVERSITY OF WALES
Trinity Saint David
SWANSEA - ABERTAWE

Applied Computing

Wales Institute of Science and Art (WISA)

Advanced Cybersecurity (ACCB6000)

Lab 10: Defend and Analyse



Nitheesh Murugan Kaliyamurthy – n.kaliyamurthy@uwtsd.ac.uk

Cisco | Networking Academy®
Mind Wide Open™



What we do?

1. *Analyzing an attack (MySQL DB)*
2. *Encrypt and Decrypt (Recover Encrypted) Files*
3. *Working with log files*
4. *Investigating an attack on a Windows Host*

Disclaimer

Do not try these in unsupervised environments/ real time environments. Attempting to use certain tools in real time environments or without proper written consents from the owners of the network is punishable offense. All our labs are only for educational purposes and to be practiced under supervised environments.

Takeaway

Maintenance (Manual) is now available to all students of this class to refer, add your experience and share your reflections. DON'T FORGET TO ADD YOUR NAME IN YOUR POST. Each poster will be awarded with for their work.

Link is shared in our Teams Channel!!!

Recommendations

You could use any of the following as per your convenience. Highlighted is used here.

- ✓ **VirtualBox**
- ✓ VMware Workstation Pro
- ✓ VMware Fusion Pro (for Mac),
- ✓ VMware ESXi
- ✓ Microsoft Hyper-V

It is strongly recommended to do this lab after completing the Lab 9.

What you need?

- CyberOps Workstation virtual machine. (Task 1, 2, 3)
- Security Onion virtual machine (Task 3, 4)
- Internet access (Task 1, 2, 3, 4)

Task 1

In this lab, you will view a PCAP file from a previous attack against a SQL database.

- 1: Open Wireshark and load the PCAP file.
- 2: View the SQL Injection Attack.
- 3: The SQL Injection Attack continues...
- 4: The SQL Injection Attack provides system information.
- 5: The SQL Injection Attack and Table Information
- 6: The SQL Injection Attack Concludes.

Background / Scenario

SQL injection attacks allow malicious hackers to type SQL statements in a web site and receive a response from the database. This allows attackers to tamper with current data in the database, spoof identities, and miscellaneous mischief.

A PCAP file has been created for you to view a previous attack against a SQL database. In this lab, you will view the SQL database attacks and answer the questions.

You will use Wireshark, a common network packet analyzer, to analyze network traffic. After starting Wireshark, you will open a previously saved network capture and view a step by step SQL injection attack against a SQL database.

Part 1: Open Wireshark and load the PCAP file.

The Wireshark application can be opened using a variety of methods on a Linux workstation.

- a. Start the CyberOps Workstation VM.
- b. Click **Applications > CyberOPS > Wireshark** on the desktop and browse to the Wireshark application.
- c. In the Wireshark application, click **Open** in the middle of the application under Files.
- d. Browse through the **/home/analyst/** directory and search for **lab.support.files**. In the **lab.support.files** directory and open the **SQL_Lab.pcap** file.

- e. The PCAP file opens within Wireshark and displays the captured network traffic. This capture file extends over an 8-minute (441 second) period, the duration of this SQL injection attack.

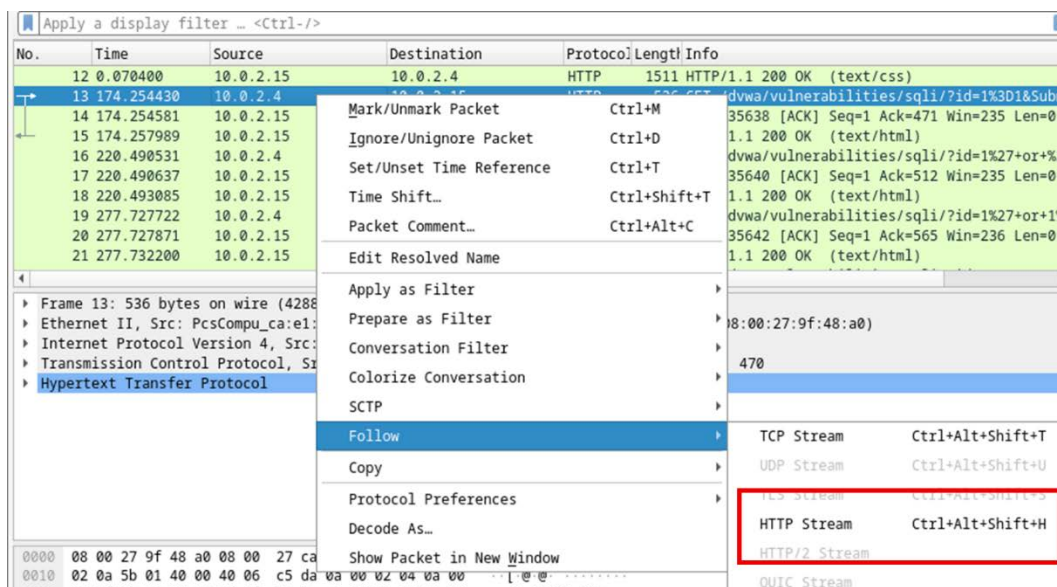
No.	Time	Source	Destination	Protocol	Length	Info
7	0.005700	10.0.2.15	10.0.2.15	HTTP	430	HTTP/1.1 302 Found
8	0.014383	10.0.2.4	10.0.2.15	TCP	66	35614->80 [ACK] Seq=589 Ack=365 Win=30336 Len=0 TSval=45840 TSecr=90
9	0.015485	10.0.2.15	10.0.2.4	HTTP	496	GET /dvwa/index.php HTTP/1.1
10	0.015485	10.0.2.4	10.0.2.15	TCP	3107	HTTP/1.1 200 OK (text/html)
11	0.068625	10.0.2.4	10.0.2.15	TCP	66	35614->80 [ACK] Seq=1019 Ack=3406 Win=36480 Len=0 TSval=45843 TSecr=90
12	0.070400	10.0.2.15	10.0.2.4	HTTP	429	GET /dvwa/dvwa/css/main.css HTTP/1.1
13	174.254430	10.0.2.4	10.0.2.15	HTTP	1511	HTTP/1.1 200 OK (text/css)
14	174.254581	10.0.2.15	10.0.2.4	TCP	536	GET /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit HTTP/1.1
15	174.257989	10.0.2.15	10.0.2.4	TCP	66	80->35638 [ACK] Seq=1 Ack=471 Win=235 Len=0 TSval=82101 TSecr=90
16	220.490531	10.0.2.4	10.0.2.15	HTTP	1861	HTTP/1.1 200 OK (text/html)
17	220.490637	10.0.2.15	10.0.2.4	TCP	577	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+%270%27%3D%270+&Submit=Submit HTTP/1.1
18	220.493085	10.0.2.15	10.0.2.4	TCP	66	80->35640 [ACK] Seq=1 Ack=512 Win=235 Len=0 TSval=93660 TSecr=10
19	277.727722	10.0.2.4	10.0.2.15	HTTP	1918	HTTP/1.1 200 OK (text/html)
20	277.727871	10.0.2.15	10.0.2.4	TCP	630	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+ HTTP/1.1
21	277.732200	10.0.2.15	10.0.2.4	TCP	66	80->35642 [ACK] Seq=1 Ack=565 Win=236 Len=0 TSval=107970 TSecr=10
22	313.710129	10.0.2.4	10.0.2.15	HTTP	1955	HTTP/1.1 200 OK (text/html)
23	313.710277	10.0.2.15	10.0.2.4	TCP	659	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+ HTTP/1.1
24	313.712414	10.0.2.15	10.0.2.4	TCP	66	80->35644 [ACK] Seq=1 Ack=594 Win=236 Len=0 TSval=116966 TSecr=10
25	383.277032	10.0.2.4	10.0.2.15	HTTP	1954	HTTP/1.1 200 OK (text/html)
26	383.277811	10.0.2.15	10.0.2.4	TCP	680	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+ HTTP/1.1
27	383.284289	10.0.2.15	10.0.2.4	TCP	66	80->35666 [ACK] Seq=1 Ack=615 Win=236 Len=0 TSval=134358 TSecr=10
28	441.804070	10.0.2.4	10.0.2.15	HTTP	4068	HTTP/1.1 200 OK (text/html)
29	441.804427	10.0.2.15	10.0.2.4	TCP	685	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+ HTTP/1.1
30	441.807206	10.0.2.15	10.0.2.4	TCP	66	80->35668 [ACK] Seq=1 Ack=620 Win=236 Len=0 TSval=148990 TSecr=10
31	441.807206	10.0.2.15	10.0.2.4	HTTP	2091	HTTP/1.1 200 OK (text/html)

What are the two IP addresses involved in this SQL injection attack based on the information displayed?

Part 2: View the SQL Injection Attack.

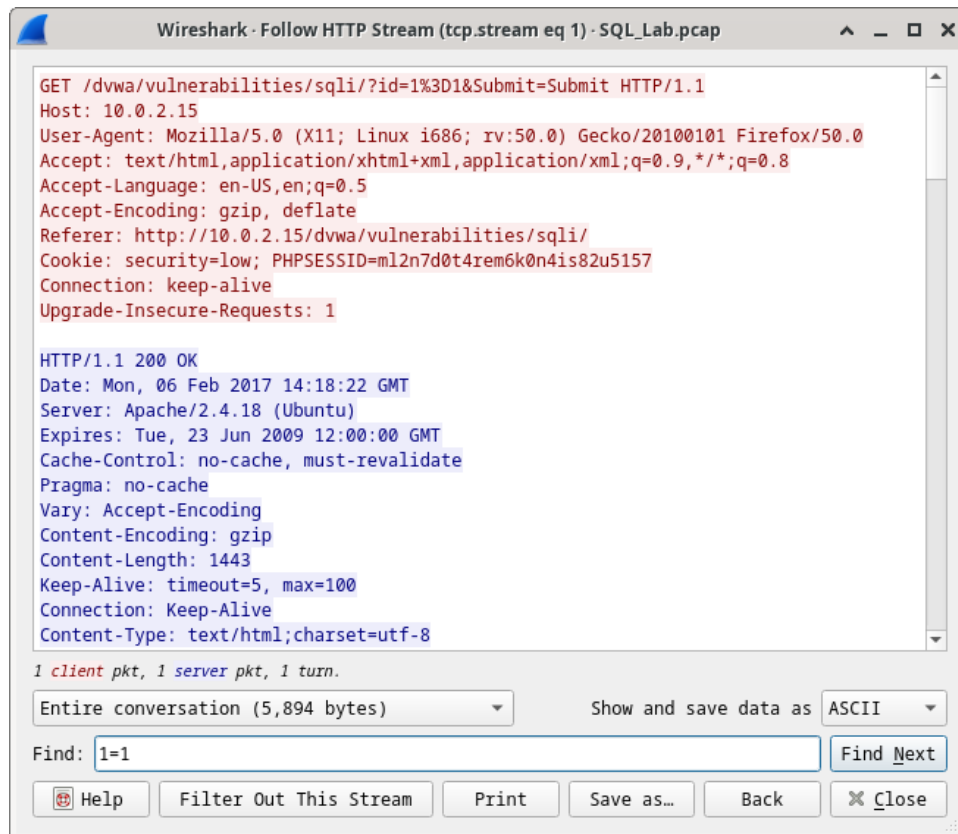
In this step, you will be viewing the beginning of an attack.

- a. Within the Wireshark capture, right-click line 13 and select **Follow > HTTP Stream**. Line 13 was chosen because it is a GET HTTP request. This will be very helpful in following the data stream as the application layers sees it and leads up to the query testing for the SQL injection.

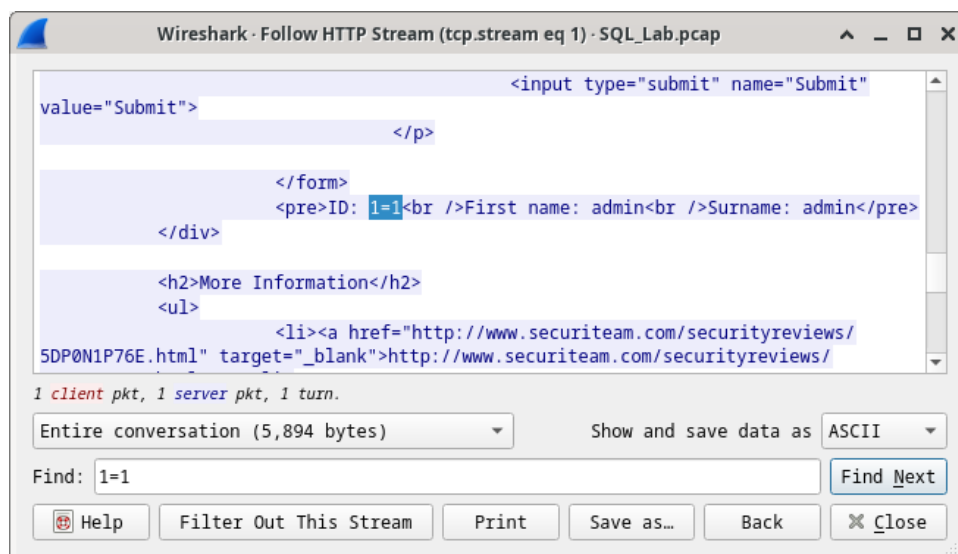


The source traffic is shown in red. The source has sent a GET request to host 10.0.2.15. In blue, the destination device is responding back to the source.

- b. In the **Find** field, enter **1=1**. Click **Find Next**.

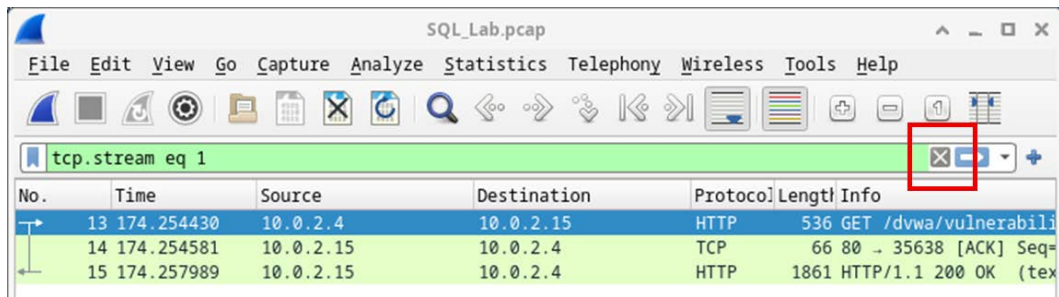


- c. The attacker has entered a query (1=1) into a UserID search box on the target 10.0.2.15 to see if the application is vulnerable to SQL injection. Instead of the application responding with a login failure message, it responded with a record from a database. The attacker has verified they can input an SQL command and the database will respond. The search string 1=1 creates an SQL statement that will be always true. In the example, it does not matter what is entered into the field, it will always be true.



- d. Close the Follow HTTP Stream window.

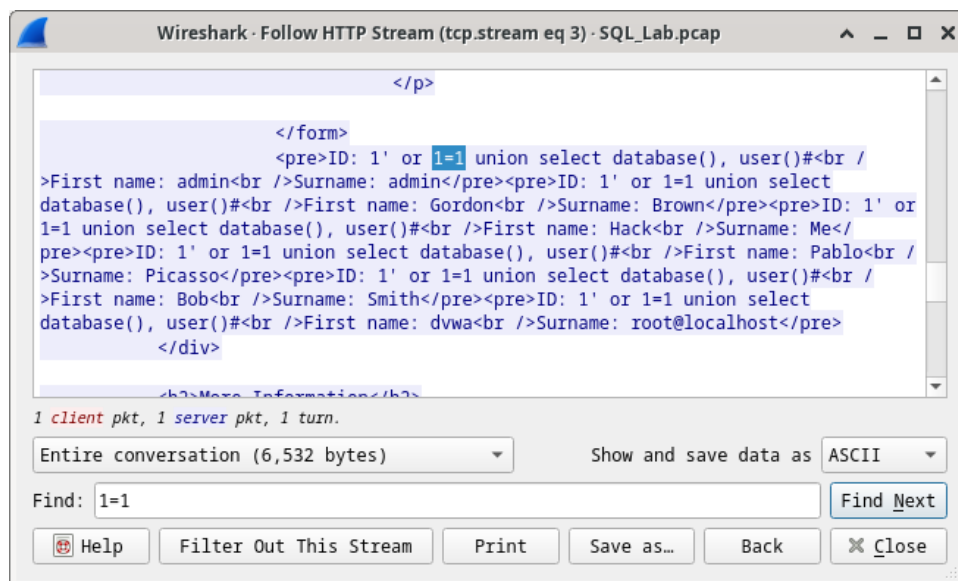
- e. Click **Clear display filter** to display the entire Wireshark conversation.



Part 3: The SQL Injection Attack continues...

In this step, you will be viewing the continuation of an attack.

- Within the Wireshark capture, right-click line 19, and click **Follow > HTTP Stream**.
- In the **Find** field, enter **1=1**. Click **Find Next**.
- The attacker has entered a query (1' or 1=1 union select database(), user()#) into a UserID search box on the target 10.0.2.15. Instead of the application responding with a login failure message, it responded with the following information:



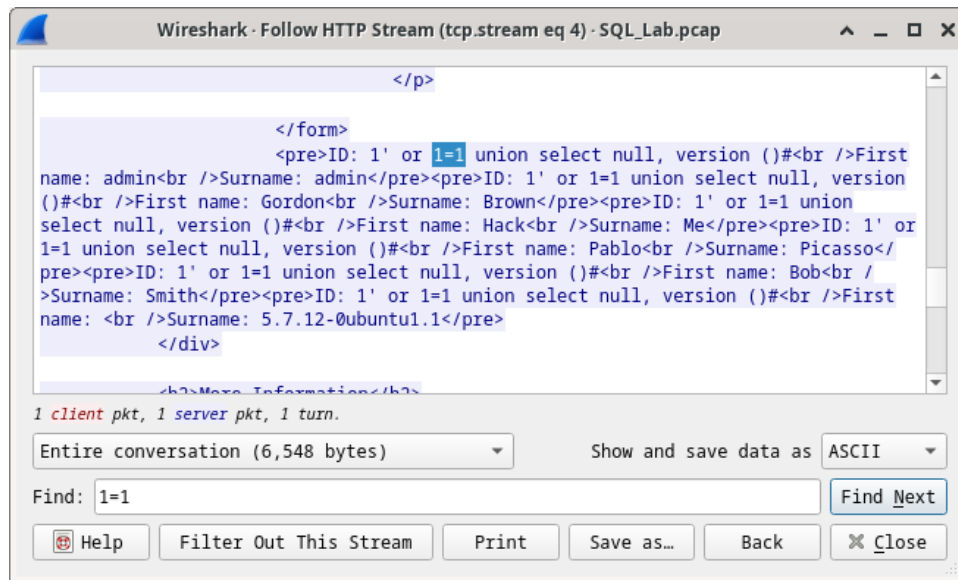
The database name is **dvwa** and the database user is **root@localhost**. There are also multiple user accounts being displayed.

- Close the Follow HTTP Stream window.
- Click **Clear display filter** to display the entire Wireshark conversation.

Part 4: The SQL Injection Attack provides system information.

The attacker continues and starts targeting more specific information.

- Within the Wireshark capture, right-click line 22 and select **Follow > HTTP Stream**. In red, the source traffic is shown and is sending the GET request to host 10.0.2.15. In blue, the destination device is responding back to the source.
- In the **Find** field, enter **1=1**. Click **Find Next**.
- The attacker has entered a query (1' or 1=1 union select null, version ()#) into a UserID search box on the target 10.0.2.15 to locate the version identifier. Notice how the version identifier is at the end of the output right before the `</pre></div>` closing HTML code.



What is the version?

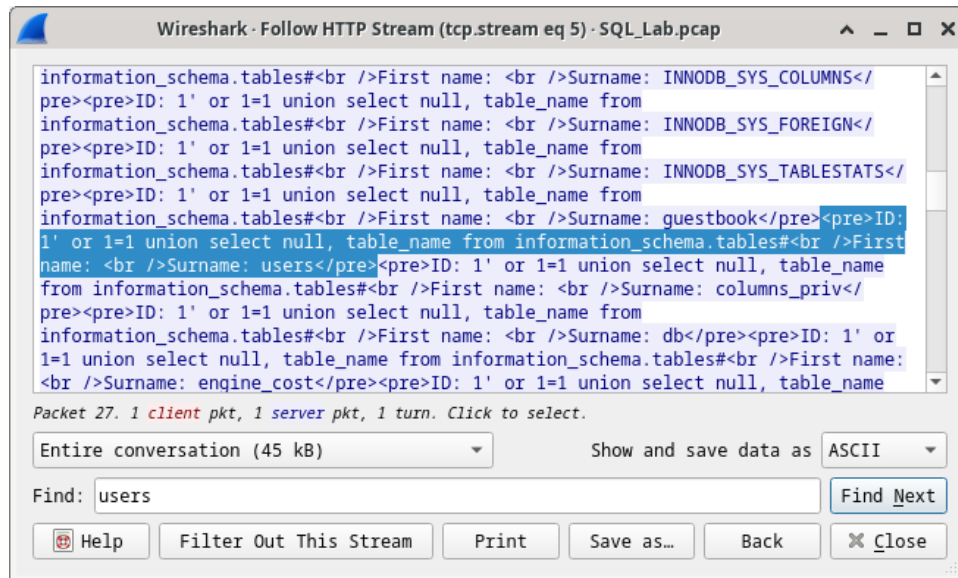
- Close the Follow HTTP Stream window.
- Click **Clear display filter** to display the entire Wireshark conversation.

Part 5: The SQL Injection Attack and Table Information.

The attacker knows that there is a large number of SQL tables that are full of information. The attacker attempts to find them.

- Within the Wireshark capture, right-click on line 25 and select **Follow > HTTP Stream**. The source is shown in red. It has sent a GET request to host 10.0.2.15. In blue, the destination device is responding back to the source.
- In the **Find** field, enter **users**. Click **Find Next**.
- The attacker has entered a query (1' or 1=1 union select null, table_name from information_schema.tables#) into a UserID search box on the target 10.0.2.15

to view all the tables in the database. This provides a huge output of many tables, as the attacker specified “null” without any further specifications.



What would the modified command of (1' OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users') do for the attacker?

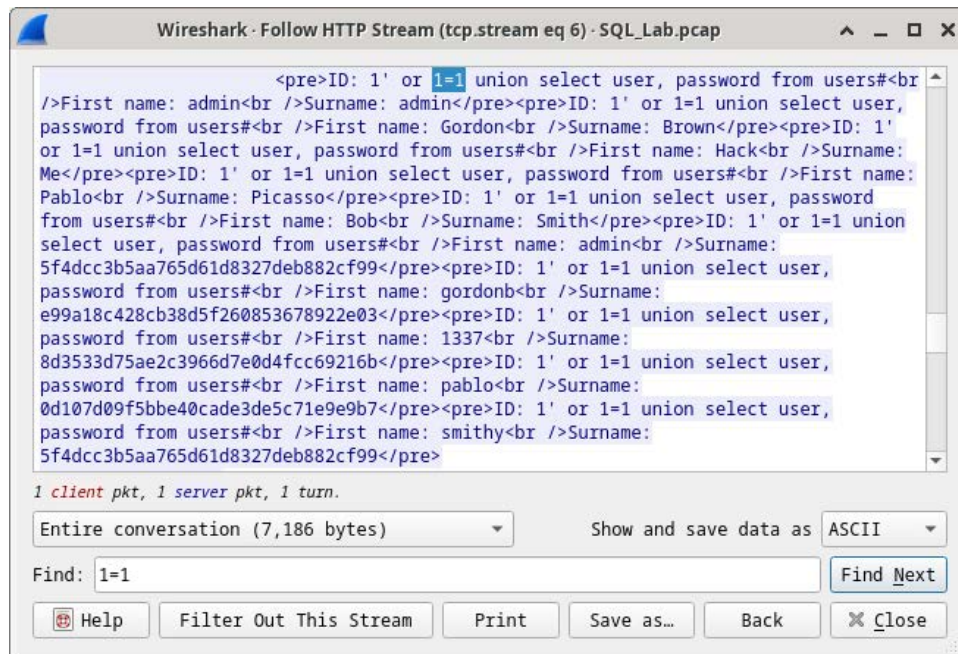
- d. Close the Follow HTTP Stream window.
- e. Click **Clear display filter** to display the entire Wireshark conversation.

Part 6: The SQL Injection Attack Concludes.

The attack ends with the best prize of all; password hashes.

- a. Within the Wireshark capture, right-click line 28 and select **Follow > HTTP Stream**. The source is shown in red. It has sent a GET request to host 10.0.2.15. In blue, the destination device is responding back to the source.
- b. Click **Find** and type in **1=1**. Search for this entry. When the text is located, click **Cancel** in the Find text search box.

The attacker has entered a query (1'or 1=1 union select user, password from users#) into a UserID search box on the target 10.0.2.15 to pull usernames and password hashes!



Which user has the password hash of 8d3533d75ae2c3966d7e0d4fcc69216b?

- c. Using a website such as <https://crackstation.net/>, copy the password hash into the password hash cracker and get cracking.

What is the plain-text password?

- d. Close the Follow HTTP Stream window. Close any open windows.

Record your reflection here

1. What is the risk of having platforms use the SQL language?
2. Browse the internet and perform a search on “prevent SQL injection attacks”. What are 2 methods or steps that can be taken to prevent SQL injection attacks?

Task 2

- 1: Create and Encrypt Files
- 2: Recover Encrypted Zip File Passwords

Background / Scenario

What if you work for a large corporation that had a corporate policy regarding removable media? Specifically, it states that only encrypted zipped documents can be copied to portable USB flash drives.

In this scenario, the Chief Financial Officer (CFO) is out-of-town on business and has contacted you in a panic with an emergency request for help. While out-of-town on business, he attempted to unzip important documents from an encrypted zip file on a USB drive. However, the password provided to open the zip file is invalid. The CFO contacted you to see if there was anything you could do.

Note: The provided scenario is simple and only serves as an example.

There may some tools available to recover lost passwords. This is especially true in situations such as this where the cybersecurity analyst could acquire pertinent information from the CFO. The pertinent information could be the length of the password and an idea of what it could be. Knowing pertinent information dramatically helps when attempting to recover passwords.

Examples of password recovery utilities and programs include hashcat, John the Ripper, Lophtrcrack, and others. In our scenario, we will use **fcrackzip** which is a simple Linux utility to recover the passwords of encrypted zip files.

Consider that these same tools can be used by cybercriminals to discover unknown passwords. Although they would not have access to some pertinent information, with time, it is possible to discover passwords to open encrypted zip files. The amount of time required depends on the password strength and the password length. Longer and more complex passwords (mix of different types of characters) are more secure.

In this lab, you will:

- Create and encrypt sample text files.
- Decrypt the encrypted zip file.

Note: This lab should be used for instructional purposes only. The methods presented here should NOT be used to secure truly sensitive data.

Part 1: Create and Encrypt Files

In this part, you will create a few text files that will be used to create encrypted zip files in the next step.

- 1) Start the CyberOps Workstation VM.
- 2) Open a terminal window. Verify that you are in the analyst home directory. Otherwise, enter **cd ~** at the terminal prompt.
- 3) Create a new folder called Zip-Files using the **mkdir Zip-Files** command.
- 4) Move into that directory using the **cd Zip-Files** command.
- 5) Enter the following to create three text files.

```
[analyst@secOps Zip-Files]$ echo This is a sample text file > sample-1.txt  
[analyst@secOps Zip-Files]$ echo This is a sample text file > sample-2.txt  
[analyst@secOps Zip-Files]$ echo This is a sample text file > sample-3.txt
```

- 6) Verify that the files have been created, using the **ls** command.

```
[analyst@secOps Zip-Files]$ ls -l  
total 12  
-rw-r--r-- 1 analyst analyst 27 May 13 10:58 sample-1.txt  
-rw-r--r-- 1 analyst analyst 27 May 13 10:58 sample-2.txt  
-rw-r--r-- 1 analyst analyst 27 May 13 10:58 sample-3.txt
```

Next, we will create several encrypted zipped files using varying password lengths. To do so, all three text files will be encrypted using the **zip** utility.

- 7) Create an encrypted zip file called **file-1.zip** containing the three text files using the following command:

```
[analyst@secOps Zip-Files]$ zip -e file-1.zip sample*
```

- 8) When prompted for a password, enter a one-character password of your choice. In the example, the letter **B** was entered. Enter the same letter when prompted to verify.

```
[analyst@secOps Zip-Files]$ zip -e file-1.zip sample-*
```

Enter password:

Verify password:

```
adding: sample-1.txt (stored 0%)  
adding: sample-2.txt (stored 0%)  
adding: sample-3.txt (stored 0%)
```

- 9) Repeat the procedure to create the following 4 other files
 - **file-2.zip** using a 2-character password of your choice. In our example, we used **R2**.
 - **file-3.zip** using a 3-character password of your choice. In our example, we used **0B1**.
 - **file-4.zip** using a 4-character password of your choice. In our example, we used **Y0Da**.

- **file-5.zip** using a 5-character password of your choice. In our example, we used **C-3P0**.

10) Verify that all zipped files have been created using the **ls -l f*** command.

```
[analyst@secOps Zip-Files]$ ls -l f*
-rw-r--r-- 1 analyst analyst 643 May 13 11:01 file-1.zip
-rw-r--r-- 1 analyst analyst 643 May 13 11:02 file-2.zip
-rw-r--r-- 1 analyst analyst 643 May 13 11:03 file-3.zip
-rw-r--r-- 1 analyst analyst 643 May 13 11:03 file-4.zip
-rw-r--r-- 1 analyst analyst 643 May 13 11:03 file-5.zip
```

11) Attempt to open a zip using an incorrect password as shown.

```
[analyst@secOps Zip-Files]$ unzip file-1.zip
Archive: file-1.zip
[file-1.zip] sample-1.txt password:
password incorrect--reenter:
password incorrect--reenter:
  skipping: sample-1.txt      incorrect password
[file-1.zip] sample-2.txt password:
password incorrect--reenter:
password incorrect--reenter:
  skipping: sample-2.txt      incorrect password
[file-1.zip] sample-3.txt password:
password incorrect--reenter:
password incorrect--reenter:
  skipping: sample-3.txt      incorrect password
```

Part 2: Recover Encrypted Zip File Passwords

In this part, you will use the **fcrackzip** utility to recover lost passwords from encrypted zipped files. Fcrackzip searches each zip file given for encrypted files and tries to guess the password using brute-force methods.

The reason we created zip files with varying password lengths was to see if password length influences the time it takes to discover a password.

From the terminal window, enter the **fcrackzip -h** command to see the associated command options.

In our examples, we will be using the **-v**, **-u**, and **-l** command options. The **-l** option will be listed last because it specifies the possible password length. Feel free to experiment with other options.

1) Now attempt to recover the password of the **file-1.zip** file. Recall, that a one-character password was used to encrypt the file. Therefore, use the following **fcrackzip** command:

```
[analyst@secOps Zip-Files]$ fcrackzip -vul 1-4 file-1.zip
found file 'sample-1.txt', (size cp/uc 39/ 27, flags 9, chk 5754)
```

```
found file 'sample-2.txt', (size cp/uc 39/ 27, flags 9, chk 5756)
found file 'sample-3.txt', (size cp/uc 39/ 27, flags 9, chk 5757)
```

PASSWORD FOUND!!!!: pw == B

Note: The password length could have been set to less than 1 – 4 characters.

How long does it take to discover the password?

- 2) Now attempt to recover the password of the **file-2.zip** file. Recall, that a two-character password was used to encrypt the file. Therefore, use the following **fcrackzip** command:

```
[analyst@secOps Zip-Files]$ fcrackzip -vul 1-4 file-2.zip
```

```
found file 'sample-1.txt', (size cp/uc 39/ 27, flags 9, chk 5754)
found file 'sample-2.txt', (size cp/uc 39/ 27, flags 9, chk 5756)
found file 'sample-3.txt', (size cp/uc 39/ 27, flags 9, chk 5757)
```

PASSWORD FOUND!!!!: pw == R2

How long does it take to discover the password?

- 3) Repeat the procedure and recover the password of the **file-3.zip** file. Recall, that a three-character password was used to encrypt the file. Time to see how long it takes to discover a 3-letter password. Use the following **fcrackzip** command:

```
[analyst@secOps Zip-Files]$ fcrackzip -vul 1-4 file-3.zip
```

```
found file 'sample-1.txt', (size cp/uc 39/ 27, flags 9, chk 5754)
found file 'sample-2.txt', (size cp/uc 39/ 27, flags 9, chk 5756)
found file 'sample-3.txt', (size cp/uc 39/ 27, flags 9, chk 5757)
```

PASSWORD FOUND!!!!: pw == 0B1

How long does it take to discover the password?

- 4) How long does it take to crack a password of four characters? Repeat the procedure and recover the password of the **file-4.zip** file. Time to see how long it takes to discover the password using the following **fcrackzip** command:

```
[analyst@secOps Zip-Files]$ fcrackzip -vul 1-4 file-4.zip
```

```
found file 'sample-1.txt', (size cp/uc 39/ 27, flags 9, chk 5754)
found file 'sample-2.txt', (size cp/uc 39/ 27, flags 9, chk 5756)
found file 'sample-3.txt', (size cp/uc 39/ 27, flags 9, chk 5757)
```

checking pw X9M~

PASSWORD FOUND!!!!: pw == Y0Da

How long does it take to discover the password?

- 5) How long does it take to crack a password of five characters? Repeat the procedure and recover the password of the **file-5.zip** file. The password length is five characters, so we need to set the **-l** command option to **1-5**. Again, time to see how long it takes to discover the password using the following **fcrackzip** command:

```
[analyst@secOps Zip-Files]$ fcrackzip -vul 1-5 file-5.zip
```

```
found file 'sample-1.txt', (size cp/uc 39/ 27, flags 9, chk 5754)
```

```
found file 'sample-2.txt', (size cp/uc 39/ 27, flags 9, chk 5756)
```

```
found file 'sample-3.txt', (size cp/uc 39/ 27, flags 9, chk 5757)
```

checking pw C-H*~

PASSWORD FOUND!!!!: pw == C-3P0

How long does it take to discover the password?

- 6) Recover a 6 Character Password using fcrackzip

It appears that longer passwords take more time to discover and therefore, they are more secure. However, a 6 character password would not deter a cybercriminal.

How long do you think it would take fcrackzip to discover a 6-character password?

*To answer that question, create a file called **file-6.zip** using a 6-character password of your choice. In our example, we used **JarJar**.*

```
[analyst@secOps Zip-Files]$ zip -e file-6.zip sample*
```

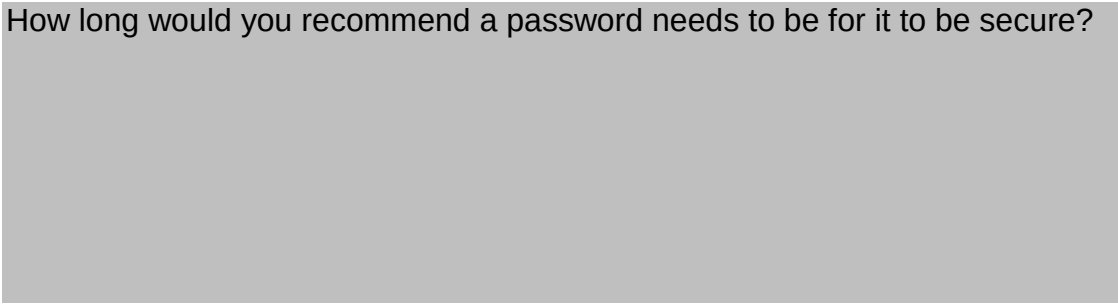
- 7) Repeat the procedure to recover the password of the **file-6.zip** file using the following **fcrackzip** command:

```
[analyst@secOps Zip-Files]$ fcrackzip -vul 1-6 file-6.zip
```

How long does it take fcrackzip to discover the password?

The simple truth is that longer passwords are more secure because they take longer to discover.

How long would you recommend a password needs to be for it to be secure?



Task 3

- 1: Normalize Timestamps in a Log File
- 2: Normalize Timestamps in an Apache Log File
- 3: Log File Preparation in Security Onion Virtual Machine

Background / Scenario

This lab will prepare you to learn where log files are located and how to manipulate and view log files. **Log entries** are generated by network devices, operating systems, applications, and various types of programmable devices. A file containing a time-sequenced stream of log entries is called a **log file**.

By nature, log files record events that are relevant to the source. The syntax and format of data within log messages are often defined by the application developer.

Therefore, the terminology used in the log entries often varies from source to source. For example, depending on the source, the terms login, logon, authentication event, and user connection, may all appear in log entries to describe a successful user authentication to a server.

It is often desirable to have a consistent and uniform terminology in logs generated by different sources. This is especially true when all log files are being collected by a centralized point.

The term **normalization** refers to the process of converting parts of a message, in this case a log entry, to a common format.

In this lab, you will use command line tools to manually normalize log entries. In Part 2, the timestamp field will be normalized. In Part 3, the IPv6 field will be normalized.

Note: While numerous plugins exist to perform log normalization, it is important to understand the basics behind the normalization process.

Part 1: Normalize Timestamps in a Log File

Timestamps are used in log entries to specify when the recorded event took place. While it is best practice to record timestamps in UTC, the format of the timestamp varies from log source to log source. There are two common timestamp formats, known as Unix Epoch and Human Readable.

Unix Epoch timestamps record time by measuring the number of seconds that have passed since January 1, 1970.

Human Readable timestamps record time by representing separate values for year, month, day, hour, minute, and second.

The Human Readable **Wed, 28 Jun 2017 13:27:19 GMT** timestamp is the same as **1498656439** in Unix Epoch.

From a programmability standpoint, it is much easier to work with Epoch as it allows for easier addition and subtraction operations. From an analysis perspective; however, Human Readable timestamps are much easier to interpret.

Converting Epoch to Human Readable Timestamps with AWK

AWK is a programming language designed to manipulate text files. It is very powerful and especially useful when handling text files where the lines contain multiple fields, separated by a delimiter character. Log files contain one entry per line and are formatted as delimiter-separated fields, making AWK a great tool for normalizing.

Consider the **applicationX_in_epoch.log** file below. The source of the log file is not relevant.

```
2|Z|1219071600|AF|0
3|N|1219158000|AF|89
4|N|1220799600|AS|12
1|Z|1220886000|AS|67
5|N|1220972400|EU|23
6|R|1221058800|OC|89
```

The log file above was generated by what we will call application X. The relevant aspects of the file are:

- o The columns are separated, or delimited, by the **|** character. Therefore, the data has five columns.
- o The third column contains timestamps in Unix Epoch.
- o The file has an extra line at the end. This will be important later in the lab.

Assume that a log analyst needs to convert the timestamps to a human-readable format. Follow the steps below to use AWK to easily perform the manual conversion:

- 1) Launch the **CyberOps Workstation VM** and then launch a terminal window.
- 2) Use the **cd** command to change to the **/home/analyst/lab.support.files/** directory. A copy of the file shown above is stored there.

```
[analyst@secOps ~]$ cd /home/analyst/lab.support.files/
```

```
[analyst@secOps lab.support.files]$ ls -l
```

```
total 580
```

```
-rw-r--r-- 1 analyst analyst 649 Jun 28 18:34 apache_in_epoch.log
```

```
-rw-r--r-- 1 analyst analyst 126 Jun 28 11:13 applicationX_in_epoch.log
```

```
drwxr-xr-x 4 analyst analyst 4096 Aug 7 15:29 attack_scripts
```

```
-rw-r--r-- 1 analyst analyst 102 Jul 20 09:37 confidential.txt
```

```
<output omitted>
```

```
[analyst@secOps lab.support.files]$
```

- 3) Issue the following AWK command to convert and print the result on the terminal:

Note: Up arrow can be used to edit the typing errors in the previous command entry.

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="|"}
{$3=strftime("%c",$3)} {print}' applicationX_in_epoch.log
2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0
3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89
4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12
1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
5|N|Tue 09 Sep 2008 11:00:00 AM EDT|EU|23
6|R|Wed 10 Sep 2008 11:00:00 AM EDT|OC|89
||Wed 31 Dec 1969 07:00:00 PM EST
[analyst@secOps lab.support.files]$
```

The command above is an AWK script. It may seem complicated. The main structure of the AWK script above is as follows:

- **awk** – This invokes the AWK interpreter.
- **'BEGIN** – This defines the beginning of the script.
- **{ }** – This defines actions to be taken in each line of the input text file. An AWK script can have several actions.
- **FS = OFS = "|"** – This defines the field separator (i.e., delimiter) as the bar (|) symbol. Different text files may use different delimiting characters to separate fields. This operator allows the user to define what character is used as the field separator in the current text file.
- **\$3** – This refers to the value in the third column of the current line. In the **applicationX_in_epoch.log**, the third column contains the timestamp in epoch to be converted.
- **strftime** - This is an AWK internal function designed to work with time. The %c and \$3 in between parenthesis are the parameters passed to **strftime**.
- **applicationX_in_epoch.log** – This is the input text file to be loaded and used. Because you are already in the **lab.support.files** directory, you do not need to add path information, **/home/analyst/lab.support.files/applicationX_in_epoch.log**.

The first script action that defined in the first set of curly brackets is to define the field separator character as the "|". Then, in the second set of curly brackets, it rewrites the third column of each line with the result of the execution of the **strftime()** function. **strftime()** is an internal AWK function created to handle time conversion. Notice that the script tells the function to use the contents of the third column of each line before the change (**\$3**) and to format the output (**%c**).

Were the Unix Epoch timestamps converted to Human Readable format? Were the other fields modified? Explain.

Compare the contents of the file and the printed output. Why is there the line, **||Wed 31 Dec 1969 07:00:00 PM EST**?

- 4) Use **nano** (or your favorite text editor) to remove the extra empty line at the end of the file and run the **AWK** script again by using the up-arrow to find it in the command history buffer.

```
[analyst@secOps lab.support.files]$ nano applicationX_in_epoch.log
```

Is the output correct now? Explain.

- 5) While printing the result on the screen is useful for troubleshooting the script, analysts will likely need to save the output in a text file. Redirect the output of the script above to a file named **applicationX_in_human.log** to save it to a file:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="|"}
{$3=strftime("%c",$3)} {print}' applicationX_in_epoch.log >
applicationX_in_human.log
```

```
[analyst@secOps lab.support.files]$
```

What was printed by the command above? Is this expected?

- 6) Use **cat** to view the **applicationX_in_human.log**. Notice that the extra line is now removed and the timestamps for the log entries have been converted to human readable format.

```
[analyst@secOps lab.support.files]$ cat applicationX_in_human.log
```

```
2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0
```

```
3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89
```

```
4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12
```

```
1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
```

```
5|N|Tue 09 Sep 2008 11:00:00 AM EDT|EU|23
```

6|R|Wed 10 Sep 2008 11:00:00 AM EDT|OC|89
[analyst@secOps lab.support.files]\$

Part 2: Normalize Timestamps in an Apache Log File

Similar to what was done with the **applicationX_in_epoch.log** file, Apache web server log files can also be normalized. Follow the steps below to convert Unix Epoch to Human Readable timestamps. Consider the following Apache log file, **apache_in_epoch.log**:

```
[analyst@secOps lab.support.files]$ cat apache_in_epoch.log
198.51.100.213 - - [1219071600] "GET
/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationV
ariables HTTP/1.1" 401 12846
198.51.100.213 - - [1219158000] "GET
/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1" 200
4523
198.51.100.213 - - [1220799600] "GET /mailman/listinfo/hsdivision HTTP/1.1"
200 6291
198.51.100.213 - - [1220886000] "GET /twiki/bin/view/TWiki/WikiSyntax
HTTP/1.1" 200 7352
198.51.100.213 - - [1220972400] "GET /twiki/bin/view/Main/DCCAndPostFix
HTTP/1.1" 200 5253
198.51.100.213 - - [1221058800] "GET
/twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore&m1=1.12&m2
=1.12 HTTP/1.1" 200 11382
```

The Apache Log file above contains six entries which record events related to the Apache web server. Each entry has seven fields. The fields are delimited by a space:

- The first column contains the IPv4 address, **198.51.100.213**, of the web client placing the request.
- The second and third columns are not used and a "-" character is used to represent no value.
- The fourth column contains the timestamp in Unix Epoch time, for example **[1219071600]**.
- The fifth column contains text with details about the event, including URLs and web request parameters. All six entries are HTTP GET messages. Because these messages include spaces, the entire field is enclosed with quotes.
- The sixth column contains the HTTP status code, for example **401**.
- The seventh column contains the size of the response to the client (in bytes), for example **12846**.

As in Part 1, a script will be created to convert the timestamp from Epoch to Human Readable.

- 1) First, answer the questions below. They are crucial for the construction of the script.

In the context of timestamp conversion, what character would work as a good delimiter character for the Apache log file above?

How many columns does the Apache log file above contain?

In the Apache log file above, what column contains the Unix Epoch Timestamp?

- 2) In the **CyberOps Workstation VM** terminal, a copy of the Apache log file, `apache_in_epoch.log`, is stored in the `/home/analyst/lab.support.files`.
- 3) Use an **awk** script to convert the timestamp field to a human readable format. Notice that the command contains the same script used previously, but with a few adjustments for the delimiter, timestamp field, and file name.

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}{
$4=strftime("%c",$4)} {print}' apache_in_epoch.log
```

Was the script able to properly convert the timestamps? Describe the output.

- 4) Before moving forward, think about the output of the script.

Can you guess what caused the incorrect output? Is the script incorrect? What are the relevant differences between the **applicationX_in_epoch.log** and **apache_in_epoch.log**?

- 5) To fix the problem, the square brackets must be removed from the timestamp field before the conversion takes place. Adjust the script by adding two actions before the conversion, as shown below:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}{
gsub(/\[|\]/,"",$4)}{print}' $4=strftime("%c",$4)}{print}'
apache_in_epoch.log
```

Notice after specifying space as the delimiter with **{FS=OFS=" "}**, there is a regular expression action to match and replace the square brackets with an empty string, effectively removing the square brackets that appear in the timestamp field. The second action prints the updated line so the conversion action can be performed.

- **gsub()** – This is an internal AWK function used to locate and substitute strings. In the script above, **gsub()** received three comma-separated parameters, described below.
- **/[[]]/** – This is a regular expression passed to **gsub()** as the first parameter. The regular expression should be read as ‘find “[OR “]’’. Below is the breakdown of the expression:
 - o The first and last “/” character marks the beginning and end of the search block. Anything between the first “/” and the second “/” are related to the search. The “\” character is used to escape the following “[”. Escaping is necessary because “[” can also be used by an operator in regular expressions. By escaping the “[” with a leading “\”, we tell the interpreter that the “[” is part of the content and not an operator. The “|” character is the OR operator. Notice that the “|” is not escaped and will therefore, be seen as an operator. Lastly, the regular expression escapes the closing square bracket with “\]”, as done before.
- **""** – This represents no characters, or an empty string. This parameter tells **gsub()** what to replace the “[” and “]” with, when found. By replacing the “[” and “]” with “”, **gsub()** effectively removes the “[” and “]” characters.
- **\$4** – This tells **gsub()** to work only on the fourth column of the current line, the timestamp column.

6) In a CyberOps Workstation VM terminal, execute the adjusted script, as follows:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}
{gsub(/[[]]/,"",$4){print}{$4=strftime("%c",$4){print}}
apache_in_epoch.log'
```

Was the script able to properly convert the timestamps this time? Describe the output.

7) Shut down CyberOps Workstation VM if desired.

Part 3: Log File Preparation in Security Onion Virtual Machine

Because log file normalization is important, log analysis tools often include log normalization features. Tools that do not include such features often rely on plugins for log normalization and preparation. The goal of these plugins is to allow log analysis tools to normalize and prepare the received log files for tool consumption.

The Security Onion appliance relies on a number of tools to provide log analysis services. **ELK**, **Zeek**, **Snort** and **SGUIL** are arguably the most used tools.

ELK (Elasticsearch, Logstash, and Kibana) is a solution to achieve the following:

- Normalize, store, and index logs at unlimited volumes and rates.
- Provide a simple and clean search interface and API.
- Provide an infrastructure for alerting, reporting and sharing logs.
- Plugin system for taking actions with logs.
- Exist as a completely free and open-source project.

Zeek (formerly called Bro) is a framework designed to analyze network traffic passively and generate event logs based on it. Upon network traffic analysis, Zeek creates logs describing events such as the following:

- TCP/UDP/ICMP network connections
- DNS activity
- FTP activity
- HTTPS requests and replies
- SSL/TLS handshakes

Snort and SGUIL

Snort is an IDS that relies on pre-defined rules to flag potentially harmful traffic. Snort looks into all portions of network packets (headers and payload), looking for patterns defined in its rules. When found, Snort takes the action defined in the same rule.

SGUIL provides a graphical interface for Snort logs and alerts, allowing a security analyst to pivot from SGUIL into other tools for more information. For example, if a potentially malicious packet is sent to the organization web server and Snort raised an alert about it, SGUIL will list that alert. The analyst can then right-click that alert to search the ELSA or Bro databases for a better understanding of the event.

Note: The directory listing maybe different than the sample output shown below.

Launch the **Security Onion VM** from VirtualBox's Dashboard (username: **analyst** / password: **cyberops**).

- 1) Open a terminal window in the Security Onion VM. Right-click the Desktop. In the pop-up menu, select **Open Terminal**.
- 2) Zeek logs are stored at **/nsm/bro/logs/**. As usual with Linux systems, log files are rotated based on the date, renamed and stored on the disk. The current log files can be found under the current directory. From the terminal window, change directory using the following command.

```
analyst@SecOnion:~$ cd /nsm/bro/logs/current
analyst@SecOnion:/nsm/logs/current$
```

- 3) Use the **ls -l** command to see the log files generated by Zeek:

Note: Depends on the state of the virtual machine, there may not be any log files yet.

- 4) Snort logs can be found at **/nsm/sensor_data/**. Change directory as follows.

```
analyst@SecOnion:/nsm/bro/logs/current$ cd /nsm/sensor_data
analyst@SecOnion:/nsm/sensor_data$
```

- 5) Use the **ls -l** command to see all the log files generated by Snort.

```
analyst@SecOnion:/nsm/sensor_data$ ls -l
total 12
drwxrwxr-x 7 sguil sguil 4096 Jun 19 18:09 seconion-eth0
drwxrwxr-x 5 sguil sguil 4096 Jun 19 18:09 seconion-eth1
drwxrwxr-x 7 sguil sguil 4096 Jun 19 18:32 seconion-import
```

- 6) Notice that Security Onion separates files based on the interface. Because the **Security Onion VM** image has two interfaces configured as sensors and a special folder for imported data, three directories are kept. Use the **ls -l seconion-eth0** command to see the files generated by the eth0 interface.

```
analyst@SecOnion:/nsm/sensor_data$ ls -l seconion-eth0
total 28
drwxrwxr-x 2 sguil sguil 4096 Jun 19 18:09 argus
drwxrwxr-x 3 sguil sguil 4096 Jun 19 18:09 dailylogs
drwxrwxr-x 2 sguil sguil 4096 Jun 19 18:09 portscans
drwxrwxr-x 2 sguil sguil 4096 Jun 19 18:09 sancp
drwxr-xr-x 2 sguil sguil 4096 Jun 19 18:24 snort-1
-rw-r--r-- 1 sguil sguil 5594 Jun 19 18:31 snort-1.stats
-rw-r--r-- 1 root root 0 Jun 19 18:09 snort.stats
```

- 7) While the **/nsm/** directory stores some log files, more specific log files can be found under **/var/log/nsm/**. Change directory and use the **ls** command to see all the log files in the directory.

```
analyst@SecOnion:/nsm/sensor_data$ cd /var/log/nsm/
analyst@SecOnion:/var/log/nsm$ ls
eth0-packets.log          sid_changes.log
netsniff-sync.log        so-elastic-configure-kibana-dashboards.log
ossec_agent.log           so-elasticsearch-pipelines.log
pulledpork.log            so-sensor-backup-config.log
seconion-eth0             so-server-backup-config.log
seconion-import           sosetup.log
securityonion             so-zeek-cron.log
sensor-clean.log          squert-ip2c-5min.log
sensor-clean.log.1.gz     squert-ip2c.log
sensor-clean.log.2.gz     squert_update.log
sensor-newday-argus.log    watchdog.log
sensor-newday-http-agent.log watchdog.log.1.gz
sensor-newday-pcap.log     watchdog.log.2.gz
sguil-db-purge.log
```

Notice that the directory shown above also contains logs used by secondary tools such as **OSSEC** and **Squert**.

- 8) ELK logs can be found in the **/var/log** directory. Change directory and use the **ls** command to list the files and directories.

```
analyst@SecOnion:/var/log/nsm$ cd ..
```

```
analyst@SecOnion:/var/log$ ls
```

```
alternatives.log  debug          kern.log.1      samba
alternatives.log.1 debug.1        kern.log.2.gz  sguild
apache2          debug.2.gz    kibana         so-boot.log
apt             dmesg        lastlog        syslog
auth.log        domain_stats  lightdm        syslog.1
auth.log.1      dpkg.log     logstash       syslog.2.gz
auth.log.2.gz   dpkg.log.1   lpr.log        syslog.3.gz
boot           elastalert    mail.err       syslog.4.gz
boot.log       elasticsearch mail.info      unattended-upgrades
bootstrap.log   error        mail.log       user.log
btmtp          error.1      mail.warn      user.log.1
btmtp.1        error.2.gz   messages       user.log.2.gz
cron.log       faillog      messages.1     wtmp
cron.log.1     freq_server  messages.2.gz  wtmp.1
cron.log.2.gz  freq_server_dns mysql          Xorg.0.log
curator        fsck        nsm            Xorg.0.log.old
daemon.log     gpu-manager.log ntpstats
daemon.log.1   installer    redis
daemon.log.2.gz kern.log     salt
```

- 9) Take some time to Google these secondary tools and answer the questions below:

For each one of the tools listed above, describe the function, importance, and placement in the security analyst workflow.

Takeaway

Log normalization is important and depends on the deployed environment.

Popular tools include their own normalization features, but log normalization can also be done manually.

When manually normalizing and preparing log files, double-check scripts to ensure the desired result is achieved. A poorly written normalization script may modify the data, directly impacting the analyst's work.

Task 4

In this lab you will:

- 1: Investigate the Attack with Sguil
- 2: Use Kibana to Investigate Alerts

This lab is based on an exercise from the website malware-traffic-analysis.net which is an excellent resource for learning how to analyze network and host attacks. Thanks to brad@malware-traffic-analysis.net for permission to use materials from his site.

Note: This lab requires a host computer that can access the internet.

Background / Scenario

In March 2019, network security monitoring tools alerted that a Windows computer on the network was infected with malware. In this task, you are to investigate the alerts and answer the following questions:

- What was the specific time of the attack on 2019-03-19?
- Which Windows host computer was infected? Who was the user?
- What was the computer infected with?

Part 1: Investigate the Attack with Sguil

In Part 1, you will use Sguil to check the IDS alerts and gather more information about the series of events related to an attack on 3-19-2019.

Note: The alert IDs used in this lab are for example only. The alert IDs on your VM may be different..

- a. Login to Security Onion VM with the **analyst** username and **cyberops** password.
- b. Launch Sguil from the desktop. Login with username **analyst** and password **cyberops**. Click **Select All** and **Start Sguil** to view all the alerts generated by the network sensors.
- c. Locate the group of alerts from 19 March 2019.

According to Sguil, what are the timestamps for the first and last of the alerts that occurred on 3-19-2019? What is interesting about the timestamps of all the alerts on 3-19-2019?

- d. In Sguil, click the first of the alerts on 3-19-2019 (Alert ID 5.439). Make sure to check the **Show Packet Data** and **Show Rule** checkboxes to examine the packet header information and the IDS signature rule related to the alert. Right on the **Alert ID** and pivot to Wireshark. Based on the information derived from this initial alert answer the following questions:

What was the source IP address and port number and destination IP address and port number?

What type of protocol and request or response was involved?

What is the IDS alert and message?

Do you think this alert was the result of an IDS misconfiguration or a legitimate suspicious communication?

What is the hostname, domain name, and IP address of the source host in the DNS update?

- e. In Sguil, select the second of the alerts on 3-19-2019. Right click the Alert ID 5.440 and select **Transcript**.

```

File
Sensor Name: seconion-import-1
Timestamp: 2019-03-19 01:47:04
Connection ID: .seconion-import-1_411
Src IP: 10.0.90.215
Dst IP: 209.141.34.8
Src Port: 49204
Dst Port: 80
OS Fingerprint: 10.0.90.215:49204 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]
OS Fingerprint: Signature: [8192:128:1:52:M1460,N,W8,N,N,S:::Windows:7]
OS Fingerprint: -> 209.141.34.8:80 (distance 0, link: ethernet/modem)

SRC: GET /test1.exe HTTP/1.1
SRC: Accept: */*
SRC: Accept-Encoding: gzip, deflate
SRC: User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
SRC: Host: 209.141.34.8
SRC: Connection: Keep-Alive
SRC:
SRC:
DST: HTTP/1.1 200 OK
DST: Date: Tue, 19 Mar 2019 01:45:55 GMT
DST: Server: Apache/2.4.6 (CentOS)
DST: Last-Modified: Mon, 18 Mar 2019 22:00:46 GMT
DST: ETag: "c6200-584658544df80"
DST: Accept-Ranges: bytes
DST: Content-Length: 811520
DST: Keep-Alive: timeout=5, max=100
DST: Connection: Keep-Alive

Search Abort Close
Debug Messages

DST: Connection: Keep-Alive
DST: Content-Type: application/octet-stream
DST:
DST: MZ.....@.....!..L!This program cannot be run in DOS
mode.
DST:
DST:
$.g.F.4.F.4.F.4..5.F.4..5.F.4..5.F.4..5.F.4.F.43F.4..5.F.4..4.F.4..5.F.4Rich.F.4.....
PE.L...IZ.....k.....@.....
DST: ...
DST: ...
DST: ...@.....
.....0.....8.....@.....
..text...e...f.....
..data.H...j.....@..idata.n.....i.....@..@.rsrc.....~.....@..@.relo
c.....
DST:
..X.....@..B.....
.....@.P.@.....
DST:
@.0.@.....@.....5.....k@.....j@..p@.....1..@1..2..4..
5..B..J..J..pK..L..pL..M..M..M..PP..@d..d..Ph..j..k..k..
n..@p..p..pg..t..Ot.....advapi32.dll..CheckTokenMembership...
....INF....[...].Reboot..AdvancedINF.Version.setupx.dll..setupapi.dll....BAT....SeShutdownPrivilege.a
dypack.dll.DelNodeRunDLL32*.....wininit.ini.%lu.Software\Microsoft\Windows\CurrentVersion\App
Paths\...K.e.r.n.e.i.3.2...d.i.l...HeapSetInformation..TITLE...EXTRACTOPT..INSTANCECHECK...VE
RCHECK...DecryptFileA...LICENSE.<None>..REBOOT..SHOWWINDOW..ADMQCMD.USRQCMD.R
UNPROGRAM..POSTRUNPROGRAM..FINISHMSG...LoadString() Error. Could not load string
resource....CABINET.FILESIZES...PACKINSTSPACE...UPROMPT.IXP%03d.TMP.IXP.i386....mips..

```

From the transcript answer the following questions:

What is the source and destination IP address and port numbers?

Looking at the request (blue) what was the request for?

Looking at the reply (red) many files will reveal their file signature in the initial few characters of the file when viewed as text. File signatures help identify the type of file that is represented. Use a web browser to search for a list of common file signatures.

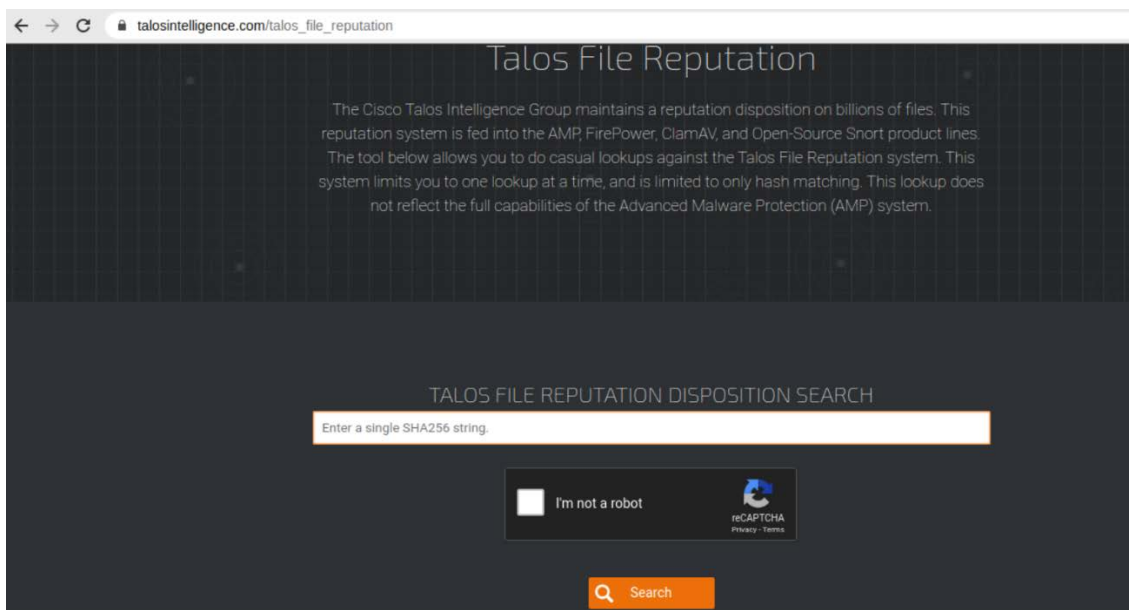
What is the initial few characters of the file file. Search for this file signature to find out what type of file was downloaded in the data?

- f. Close the transcript. Use Wireshark to export the executable file for malware analysis (**File > Export Objects > HTTP...**). Save the file to the analyst's home folder.
- g. Open a terminal in Security Onion VM and create a SHA256 hash from the exported file. Use the following command:

```
analyst@SecOnion:~$ sha256sum test1.exe
```

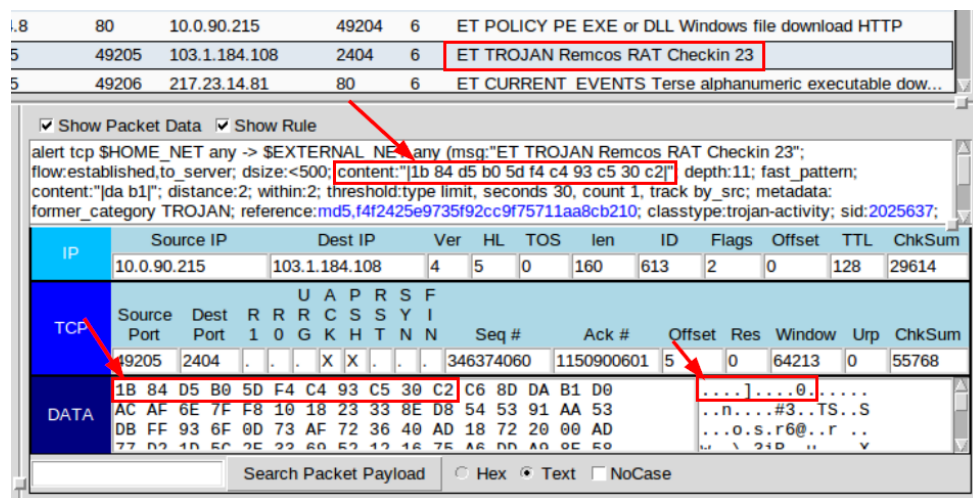
```
2a9b0ed40f1f0bc0c13ff35d304689e9cadd633781cbcad1c2d2b92ced3f1c85 test1.exe
```

- h. Copy the file hash and submit it to the Cisco Talos file reputation center at https://talosintelligence.com/talos_file_reputation.



Did Talos recognize the file hash and identify it as malware? If so, what kind of malware?

- i. In Sguil select the alert with **Alert ID 5.480** and the **Event Message** Remcos RAT Checkin 23. Notice that the IDS signature has detected the Remcos RAT based on the binary hex codes at the beginning of communication.



- j. Right click the Alert ID and select **Transcript**. Scroll through the transcript and answer the following questions:

What is the destination port of the communication? Is it a well-known port?

Is the communication readable or is it encrypted?

Do some online research on Remcos RAT Checkin 23. What does Remcos stand for?

What type of communication do you think was being transmitted?

What type of encryption and obfuscation was used to bypass detection?

- k. Using Sguil and the remaining alerts from 3-19-2019, locate the second executable file that was downloaded and check to see if it is known malware.

What Alert IDs alert to a second executable file being downloaded?

From which server IP address and port number was the file downloaded from?

What is the name of the file that was downloaded?

Create a SHA256 hash of the file and submit the hash online at Cisco Talos File Reputation Center to see if it matches known malware.

Is the executable file known malware and if so, what type? What is the AMP DETECTION NAME?

- l. Examine the remaining three alerts from 3-19-2019 by looking at the header information in Show Packet Data, the IDS signature in Show Rule, and the Alert ID Transcripts.

How are all three alerts related?

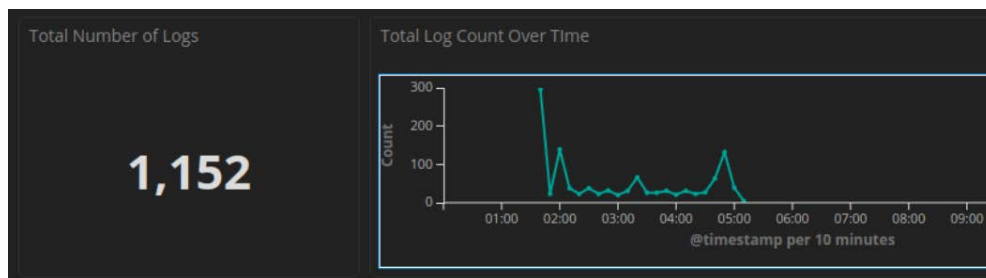
- m. Even though you have examined all the alerts in Sguil related to an attack on a Windows host on 3-19-2019, there may be additional related information available in Kibana. Close Sguil and launch Kibana from the desktop.

Part 2: Use Kibana to Investigate Alerts

In Part 2, use Kibana to further investigate the attack on 3-19-2019.

- Login to Kibana with the **analyst** username and **cyberops** password.
- Open Kibana (username **analyst** and password **cyberops**), click **Last 24 Hours** and the **Absolute** time range tab to change the time range to March 1, 2019 to March 31, 2019.

- c. The **Total Log Count Over Time** timeline will show an event on March 19. Click that event to narrow the focus to the specific time range of the attack.



- d. In the Kibana dashboard scroll down to the **All Sensors - Log Type** visualization. Review both pages and note the variety of log types related to this attack.

All Sensors - Log Type

Log Type(s) ▾	Count ▾
snort	541
bro_conn	271
bro_dns	85
bro_dce_rpc	51
bro_kerberos	50
bro_files	35
bro_smb_mapping	29
bro_ssl	29
bro_x509	25
bro_dhcp	8

Export: Raw Formatted

1 2 »

All Sensors - Log Type

Log Type(s) ▾	Count ▾
bro_weird	8
bro_notice	7
bro_smb_files	7
bro_http	4
bro_pe	2

Export: Raw Formatted

« 1 2

- e. Scroll down and notice that the NIDS Alert Summary in Kibana has many of the same IDS alerts as listed in Sguil. Click the magnifier to filter on the second

alert ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detected (Dridex) from Source IP Address 31.22.4.176.

NIDS - Alert Summary

Alert	Source IP Address	Destination IP Address	Count
ET TROJAN Remcos RAT Checkin 23	10.0.90.215	103.1.184.108	404
ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detected (Dridex)	31.22.4.176	10.0.90.215	16
ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detected (Dridex)	203.45.1.75	10.0.90.215	13
ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detected (Dridex)	115.112.43.81	10.0.90.215	3
ET CURRENT_EVENTS Likely Evil EXE download from dotted Quad by MSXMLHTTP M2	209.141.34.8	10.0.90.215	12
ET CURRENT_EVENTS Likely Evil EXE download from dotted Quad by MSXMLHTTP M2	217.23.14.81	10.0.90.215	12
ET CURRENT_EVENTS DRIVEBY Likely Evil EXE with no referer from HFS webserver (used by Unknown EK)	217.23.14.81	10.0.90.215	12
ET INFO EXE - Served Attached HTTP	217.23.14.81	10.0.90.215	12

- f. Scroll down to All Logs and click the arrow to expand the first log in the list with source IP address 31.22.4.176.

All Logs

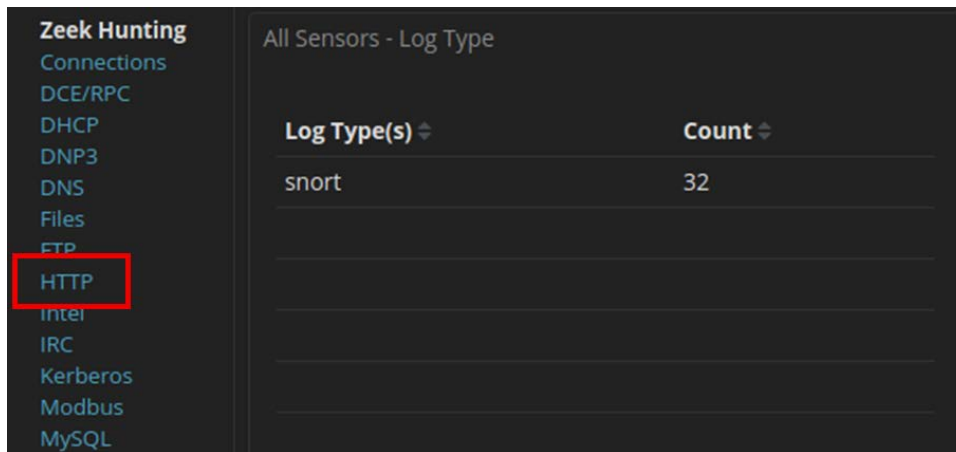
Limited to 10 results

Time	source_ip	source_port	destination_ip	destination_port
March 19th 2019, 04:55:13.000	115.112.43.81	443	10.0.90.215	49298
March 19th 2019, 04:54:57.000	115.112.43.81	443	10.0.90.215	49295
March 19th 2019, 04:54:34.000	115.112.43.81	443	10.0.90.215	49289
March 19th 2019, 04:50:21.000	31.22.4.176	3389	10.0.90.215	49281
March 19th 2019, 04:50:21.000	31.22.4.176	3389	10.0.90.215	49281
March 19th 2019, 04:50:15.000	31.22.4.176	3389	10.0.90.215	49280
March 19th 2019, 04:50:15.000	31.22.4.176	3389	10.0.90.215	49280

What is the geo country and city location for this alert?

What is the geo country and city for the alert from 115.112.43.81?

- g. Scroll back to the top of the page and click the Home link under Navigation.
- h. Earlier we noted log types like bro_http listed in the Home dashboard. You can filter for the various log type but the built-in dashboards will probably have more information. Scroll back to the top of the page and click **HTTP** in dashboard link under Zeek Hunting in Navigation.



All Sensors - Log Type	
Log Type(s) ▾	Count ▾
snort	32

- i. Scroll through the HTTP dashboard taking notice of the information presented and answer the following questions:

What is the Log Count in the HTTP dashboard? From what countries?

What are the URIs for the files that were downloaded?

- j. Match the **HTTP - URIs** to the **HTTP - Sites** on the dashboard.

What are the CSPCA.crl and ncsi.txt files related to? Use a web browser and a search engine for additional information.

- k. Scroll back to the top of the web page and under Navigation - Zeek Hunting click **DNS**. Scroll to the DNS Queries visualization. Notice page 1 and page 3 of the DNS queries.

DNS - Queries		DNS - Queries	
Query	Count	Query	Count
WPAD	27	isatap.localdomain	1
LITTLETIGERS	8	toptoptop1.online	1
dns.msftncsl.com	6	www.msftncsl.com	1
wpad	6		
littletigers-dc.littletigers.info	5		
_ldap._tcp.default-first-site-name._sites.littletigers-dc.littletigers.info	4		
_ldap._tcp.littletigers-dc.littletigers.info	4		
wpad.littletigers.info	3		
9.90.0.10.in-addr.arpa	2		
bobby-tiger-pc	2		
Export: Raw Formatted		Export: Raw Formatted	
1 2 3 »		« 1 2 3	

Do any of the domains seem potentially unsafe? Try submitting the URL toptoptop1.online to virustotal.com. What is the result?

Tasks to try

For further investigation and curiosity, try examining the following Zeek Hunting dashboards:

- ✓ DCE/RPC - for information about the Windows network remote procedures and resources involved
- ✓ Kerberos – for information on the hostnames, and domain names that were used
- ✓ PE – for information on the portable executables
- ✓ SSL and x.509 – for information on the security certificate names and countries that were used
- ✓ SMB – for more information on the SMB shares on the littletigers network
- ✓ Weird – for protocol and service anomalies and malformed communications

Record your reflection here. (Word Limit inclusive of all the tools: Minimum 500 Words Maximum 1000 Words)

As a cybersecurity analyst, look at the below flowchart (which you have done so far) and record their pro's, con's and detailed classification (which scenario is best suitable for the tool to be used).

The configurations/ ip addressing below listed is based on this topology. Make necessary changes as you work on.

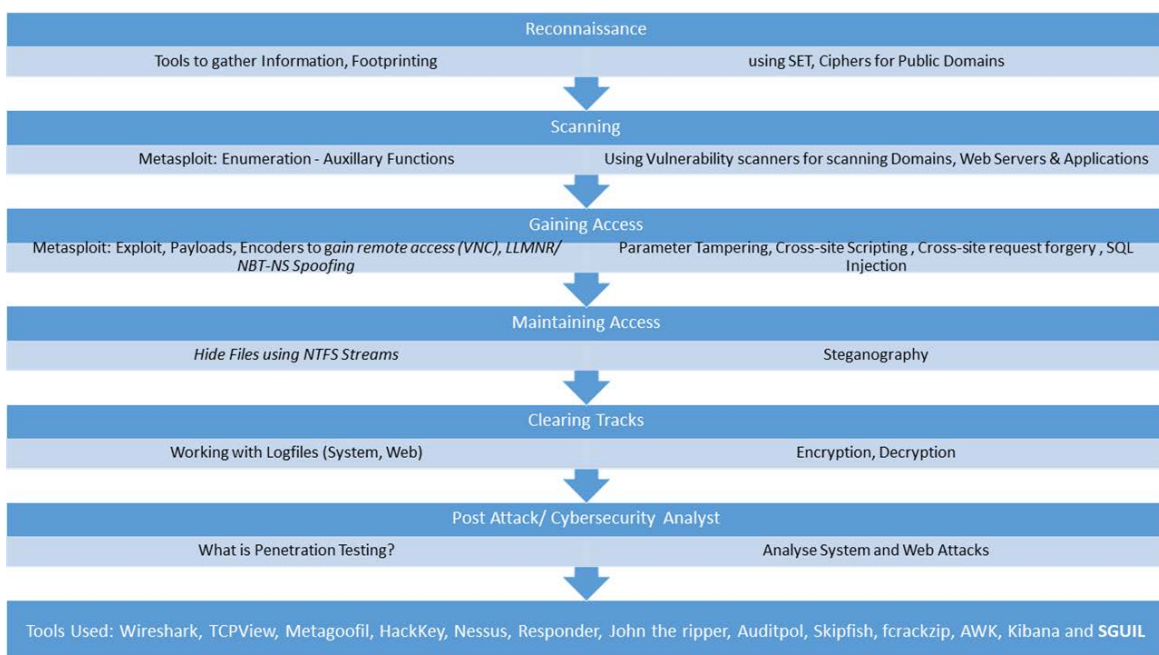


<u>Win 7 (Opt)</u>	<u>Win Sr 2012</u>	<u>Win Sr 2016</u>	<u>Win 10</u>
<u>10.0.2.7</u>	<u>10.0.2.10</u>	<u>10.0.2.9</u>	<u>10.0.2.8</u>
<u>10.0.2.1</u>	<u>10.0.2.1</u>	<u>10.0.2.1</u>	<u>10.0.2.1</u>

<u>Kali</u>	<u>Ubuntu</u>	<u>XP (Opt)</u>	<u>Win 8</u>
<u>10.0.2.15</u>	<u>10.0.2.14</u>	<u>10.0.2.6</u>	<u>10.0.2.11</u>
<u>10.0.2.1</u>	<u>10.0.2.1</u>	<u>10.0.2.1</u>	<u>10.0.2.1</u>

<u>Standalone Machines and Tools</u>
<u>Cyberops Workstation</u>
<u>Security Onion virtual machine</u>

- **Network Settings:** Nat Network; allow all VM's
- **Recommendations:** Use the latest tools/ softwares (if available)
- **If any errors, troubleshoot and record it.**



Start your reflections here!!!



Just the end of our lab practices. Wish you would use these skills to work in a real time security entity and achieve laurels. All the very best... As I would say to all... It is not the End, it is the Dawn!!!