

OLD DOMINION UNIVERSITY

CYSE 301: CYBERSECURITY TECHNIQUES AND OPERATIONS

FALL 2017

# Module I

## Traffic Tracing and Analysis

Topic 4 Use tShark to Capture Network Traffic

## 1. INTRODUCTION

In this module, we are going to learn about the basic network structures and the way of simple network defense and countermeasures. As a network administrator, if we want to set up and maintain a simple functionality and security network, we are not only need to master the fundamental knowledge of the network but also need to operate and configure the network facility such as the switch, router and firewall in the field and track the trace of the package through the network to deeply understand how to do cyber defense from the very beginning. Also, as a network administrator, one essential ability is to capture and analyze network traffic. This can be important to identify the cause of bottleneck, determining who is responsible for certain intrusion.

## 2. OBJECTIVE

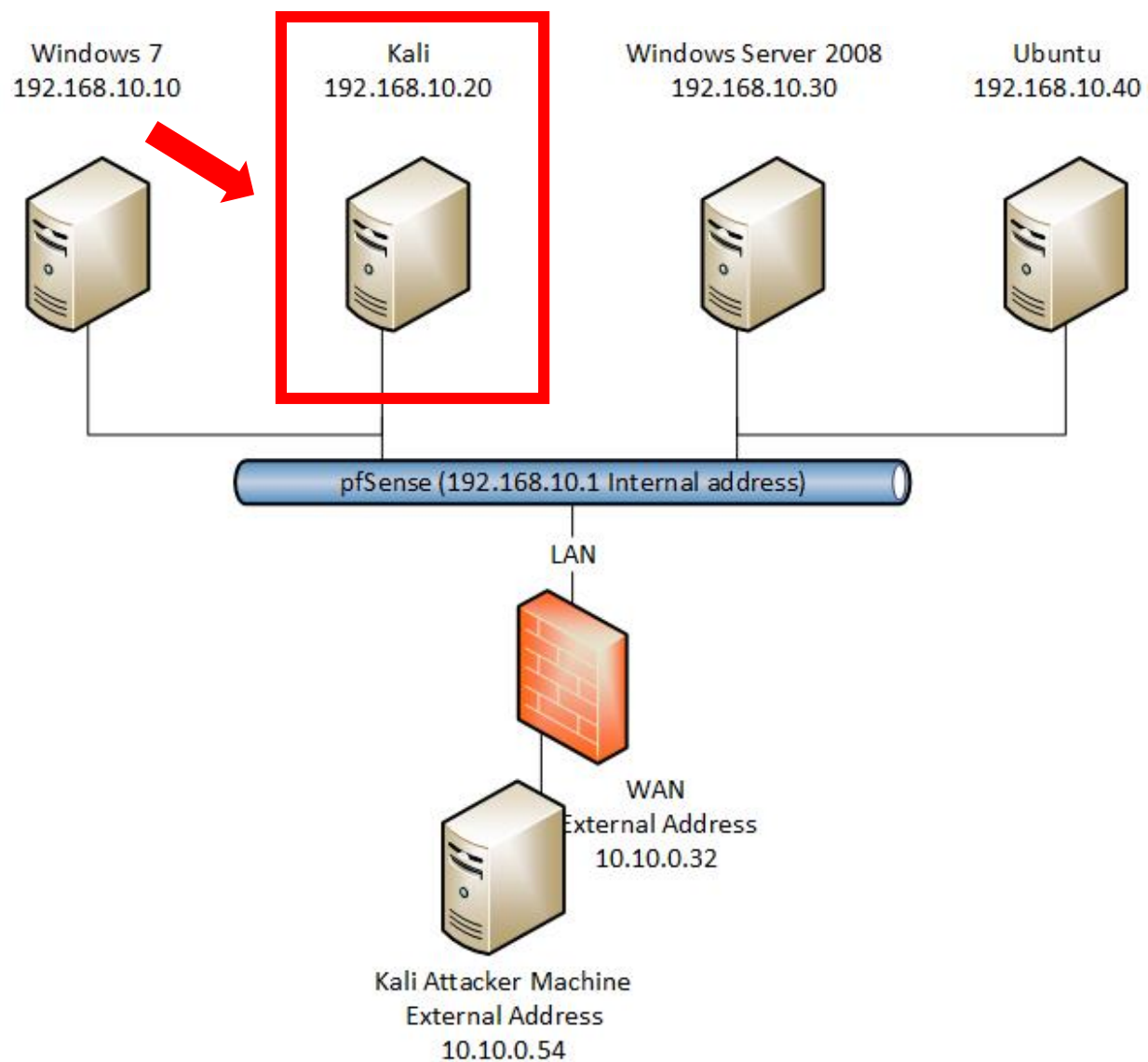
The objective of this topic is to master the command line version of Wireshark, tShark, to capture and display the packets when an interactive user interface isn't necessary or available.

### **WARNING:**

Listening, sniffing, eavesdropping on networks to which you do not have legal access is unethical and may even constitute a crime in your area.

## NETWORK TOPOLOGY

In this lab, you need to login Kali VM to complete all the assignments.



### 3. GETTING STARTED WITH TSHARK

#### 3.1. What is tShark

**tShark** is a network protocol analyzer. **tShark** can do anything Wireshark can do, provided that it does not require a GUI.

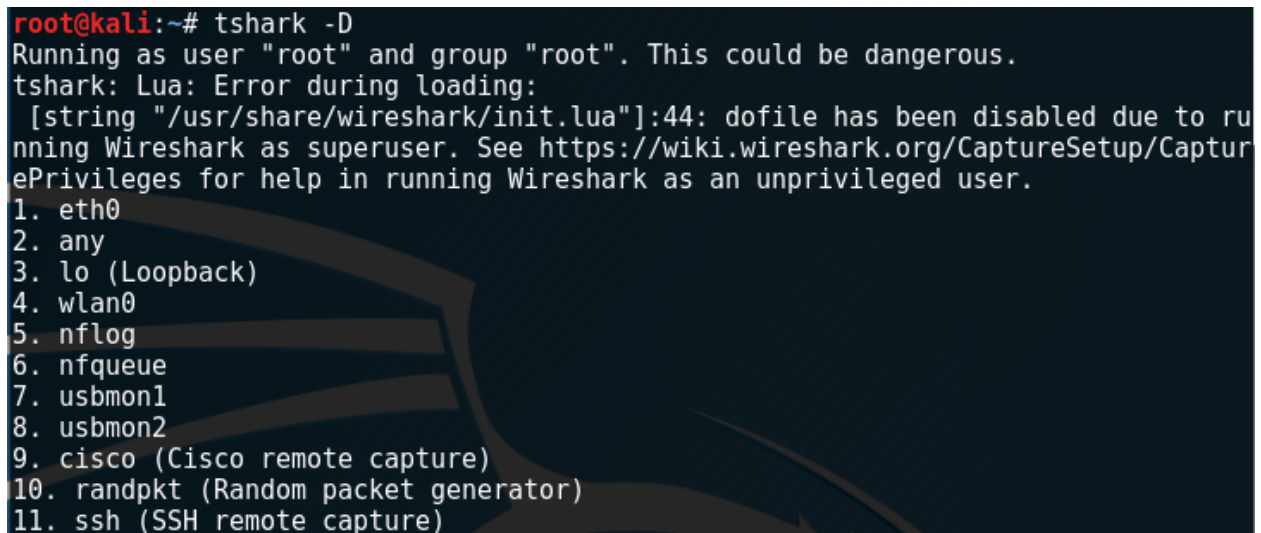
It lets you capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file. **tShark**'s native capture file format is **pcap** format, which is also the format used by **tcpdump** and various other tools

Without any options set, **tShark** will work much like **tcpdump**, which is an industry standard for network data capturing. It will use the pcap library to capture traffic from the first available network interface and displays a summary line on stdout for each received packet.

#### 3.2. Capturing Network Traffic Using tShark

The first command to run is **tshark -D** to get the list of the available network interfaces:

**#tshark -D**



```
root@kali:~# tshark -D
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled due to ru
nning Wireshark as superuser. See https://wiki.wireshark.org/CaptureSetup/Captur
ePrivileges for help in running Wireshark as an unprivileged user.
1. eth0
2. any
3. lo (Loopback)
4. wlan0
5. nflog
6. nfqueue
7. usbmon1
8. usbmon2
9. cisco (Cisco remote capture)
10. randpkt (Random packet generator)
11. ssh (SSH remote capture)
```

Figure 1

The simplest way of capturing data is by running **tshark** without any parameters, which will display all data on screen. You can stop data capturing by pressing **Ctrl-C**.

## #tshark

```
root@kali:~# tshark
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled due to running Wireshark as
superuser. See https://wiki.wireshark.org/CaptureSetup/CapturePrivileges for help in running Wiresh
ark as an unprivileged user.
Capturing on 'eth0'
  1 0.0000000000 Vmware_c0:00:08 → Broadcast      ARP 60 Who has 192.168.174.2? Tell 192.168.174.1
  2 1.161915135 Vmware_c0:00:08 → Broadcast      ARP 60 Who has 192.168.174.2? Tell 192.168.174.1
  3 2.000127513 Vmware_c0:00:08 → Broadcast      ARP 60 Who has 192.168.174.2? Tell 192.168.174.1
  4 2.197870617 192.168.174.129 → 54.230.19.124 TLSv1.2 85 Encrypted Alert
  5 2.198053820 192.168.174.129 → 54.230.19.124 TCP 54 42700 → 443 [FIN, ACK] Seq=32 Ack=1 Win=37
960 Len=0
  6 2.198345213 54.230.19.124 → 192.168.174.129 TCP 60 443 → 42700 [ACK] Seq=1 Ack=32 Win=64240 L
en=0
  7 2.198522360 54.230.19.124 → 192.168.174.129 TCP 60 443 → 42700 [ACK] Seq=1 Ack=33 Win=64239 L
en=0
  8 2.211067956 54.230.19.124 → 192.168.174.129 TCP 60 443 → 42700 [FIN, PSH, ACK] Seq=1 Ack=33 W
in=64239 Len=0
  9 2.211107204 192.168.174.129 → 54.230.19.124 TCP 54 42700 → 443 [ACK] Seq=33 Ack=2 Win=37960 L
en=0
 10 2.999391823 Vmware_c0:00:08 → Broadcast      ARP 60 Who has 192.168.174.2? Tell 192.168.174.1
 11 3.323237414 192.168.174.129 → 172.217.5.238 TCP 54 51008 → 80 [ACK] Seq=1 Ack=1 Win=32078 Len
=0
 12 3.323556972 172.217.5.238 → 192.168.174.129 TCP 60 [TCP ACKed unseen segment] 80 → 51008 [ACK
] Seq=1 Ack=2 Win=64240 Len=0
^C12 packets captured
```

Figure 2

If you don't specify any network interface then it will use **eth0** by default. To capture traffic on a specific interface, use command-line parameter **-i**

## #tshark -i wlan0

```
root@kali:~# tshark -i wlan0
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled due to running Wireshark as
superuser. See https://wiki.wireshark.org/CaptureSetup/CapturePrivileges for help in running Wiresh
ark as an unprivileged user.
Capturing on 'wlan0'
```

Figure 3

### 3.3. Saving and Reading Network Data Using Files

The single-most useful command-line parameter is **-w**, followed by a filename. This parameter allows you to save network data to a file in order to process it later. The following tshark command captures 1000 network packets (**-c 1000**) and saves them into a file called test.pcap (**-w test.pcap**):

```
#tshark -i eth0 -c 1000 -w test.pcap
```

*This part is intentionally left blank. Please screenshot the result and attach it in your lab report.*

### 3.4. Capture Filters

Capture filters are filters that are applied during data capturing; therefore, they make tshark discard network traffic that does not match the filter criteria and avoids the creation of huge capture files. This can be done using the **-f** command-line parameter, followed by a filter in **double quotes**. For example,

```
#tshark -i wlan0 -f "src port 53" -n -T fields -e dns.qry.name -e dns.a
```

*This part is intentionally left blank. Please screenshot the result and attach it in your lab report.*

Here is an example that extracts both the DNS query and the response address, “src port 53” indicates listen on port 53 for any DNS request and response. Command-line parameter **-e** means to Add a field to the list of fields to display if **-T ek/fields/json/pdml** is selected. This option can be used multiple times on the command line. At least one field must be provided if the **-T fields** option is selected.

The most important TCP-related Field Names used in capture filters are *tcp.port* (which is for filtering the source or the destination TCP port), *tcp.srcport* (which is for checking the TCP source port) and *tcp.dstport* (which is for checking the destination port).

Generally speaking, applying a filter after data capturing is considered more practical and versatile than filtering during the capture stage, because most of the time, you do not know in advance what you want to inspect. Nevertheless, if you really know what you're doing, using capture filters can save you time and disk space, and that is the main reason for using them.

### 3.5. Display Filters

Display filters are filters that are applied after packet capturing; therefore, they just "hide" network traffic without deleting it. You always can remove the effects of a display filter and get all your data back.

Display Filters support comparison and logical operators. The *http.response.code == 404 && ip.addr == 192.168.10.1* display filter shows the traffic that either comes from the 192.168.10.1 IP address or goes to the 192.168.10.1 IP address that also has the 404 (Not Found) HTTP response code in it. The *!bootp && !ip* filter excludes BOOTP and IP traffic from the output. The *eth.addr == 01:23:45:67:89:ab && tcp.port == 25* filter displays the traffic to or from the network device with the 01:23:45:67:89:ab MAC address that uses TCP port 25 for its incoming or outgoing connections.

When defining rules, remember that the *ip.addr != 192.168.1.5* expression does not mean that none of the ip.addr fields can contain the 192.168.1.5 IP address. It means that one of the ip.addr fields should not contain the 192.168.1.5 IP address! Therefore, the other ip.addr field value can be equal to 192.168.1.5! You can think of it as "*there exists one ip.addr field that is not 192.168.1.5*". The correct way of expressing it is by typing *!(ip.addr == 192.168.1.5)*. This is a common misconception with display filters.

Also remember that MAC addresses are truly useful when you want to track a given machine on your LAN, because the IP of a machine can change if it uses DHCP, but its MAC address is more difficult to change.

### 3.6. Usage example

In the following example, you can see that we extract data from any HTTP requests that are seen. Using the **-T** we specify that we want to extract fields and with the **-e** options we identify which fields we want to extract.

```
#tshark -i wlan0 -Y http.request -T fields -e http.host -e http.user_agent
```

*This part is intentionally left blank. Please screenshot the result and attach it in your lab report.*

Add time and source / destination IP addresses **-e frame.time -e ip.src -e ip.dst** to your output.

```
#tshark -i wlan0 -f "src port 53" -n -T fields -e frame.time -e ip.src -e ip.dst -e dns.qry.name -e dns.a
```

*This part is intentionally left blank. Please screenshot the result and attach it in your lab report.*



Topic IV How to use tShark to capture network traffic

Appendix: tShark Cheat sheet

Please check supplement material to get more tShark display filter usage.