Lab 2

Capturing Network Traffic

1. Wireshark on Kali

Wireshark is already installed in Kali by default. At terminal type **wireshark** to run wireshark. You can see the number of interfaces in the initial screen. If you set your Kali properly in the last lab, you will have the interface eth0. Notice that eth0 is connected to the Internet through NATNetwork. Now, do the following:

1) Get IP address of `www.howtogeek.com`. (By now, you should know how to get it.)
2) Run the Wireshark tool by issuing wireshark command at the terminal.
3) Open a web browser → Type `www.howtogeek.com` in the URL bar. → Connect to `www.howtogeek.com` in the web browser.
4) Start to capture the traffic from eth0 on Wireshark. If loading the initial page was finished, stop capturing the traffic by pressing the red "stop" button on the wireshark manager.
5) Take a look at what you have captured. Can you see "DNS" in the "Protocol" section? What is the IP address of your DNS server?
6) Try to view the SSL traffic only between your Kali VM and UOW's web using *Follow* function.
    a. Highlight one piece of SSL communications between your Kali VM and the UOW website by identifying their IPs and "Protocol". (Here, the protocol value is "TLSv1.2")
    b. Right then, select "Follow" and "TLS Stream".
7) Can you check the SYN → SYN-ACK → ACK for TCP handshake? (As SSL/TLS provides security over TCP, TCP stream will be displayed too once you select the SSL stream.)
8) Can you check the SSL/TLS connection?
    - Identify the following traffic sequences:
        - Client Hello
        - Server Hello
        - Certificate, Server Key Exchange, Server Hello Done
        - Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

    - Find the following information:
        - CipherSuites that your browser support. (Hint: Your browser is client. Double-click on the "Client Hello" traffic.)

- The agreed CipherSuite used in this SSL/TLS connection. (Hint: Double-click on the "Server Hello" traffic.)

- Double-click on the "Certificate, Server Key Exchange, Server Hello Done" traffic.

9) Use the filter in Wireshark
   - From the wireshark manager, select "Analyze". Take a look at "Display Filters".
   - Try a few filters: In the filter bar on Wireshark, issue tcp, udp, ssl and etc.
   - Try to do more filtering activities using the **[Expression…]** which locates in "Analyze > Display Filter Expression…".
   - For example, search "tls" and click "TLS – Transport Layer Security" in the Field Name. Then scroll down to find an expression for finding all the certificates in the TLS traffic.

2. Packet Analysis Using **Scapy**.

Scapy is a Python program that enables the user to generate, modify, capture, dissect and transmit network packets. This capability allows construction of tools that can probe, scan or attack networks. Scapy is already installed on the Kali Linux. Start it by issuing `scapy` at the terminal.

1) Running scapy
   Open the terminal and type; sudo scapy
   (We need the root privilege to send a packet.)
2) Create packets using Scapy.
   - Generating IP packets

```
>>> a= IP()
>>> a.ttl
64
>>> a.ttl=10  (this is to change ttl value)
>>> a
<IP ttl=10 |>
>>> del a.ttl
>>> a
<IP |>
>>> a.src  (this is to display src (sender) address)
'127.0.0.1'
>>> a.dst= '10.0.2.5'
>>> a
```

```
<IP dst=10.0.2.5 |>
```

- Generate a stacked packet (TCP/IP packet) and add the payload: Here low-level protocol comes first. After putting "/", we add a higher-level protocol.

```
>>> b=IP()/TCP()
>>> b
<IP frag=0 proto=tcp | <TCP  |>>
>>> b=IP()/TCP()/"abcdef"
>>> hexdump(b)
```

  The result will be displayed here

- Another example

```
>>> c=IP(dst='188.184.100.182')/TCP(sport=5555,
dport=80)/'GET /index.html HTTP/1.0 \n\n'
```

- Generate sets of packets. Instead of creating one packet per variable, we can generate multiple packets.

```
>>> a= IP(dst='10.0.2.5/30')
>>> [adr for adr in a]
[<IP dst=10.0.2.4|>,
 <IP dst=10.0.2.5 |>,
 <IP dst=10.0.2.6 |>,
 <IP dst=10.0.2.7 |>]
>>> b=IP(ttl=[1,2,(5,9)])
>>> [adr_ttl for adr_ttl in b]
[<IP ttl=1 |>,
 <IP ttl=2 |>,
 <IP ttl=5 |>,
 <IP ttl=6 |>,
 <IP ttl=7 |>,
 <IP ttl=9 |>,
 <IP ttl=8 |>]
>>> c=TCP(dport=[80,443])
>>> [adr_p for adr_p in a/c]
[<IP  frag=0 proto=tcp dst=10.0.2.4 |<TCP  dport=http
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.4 |<TCP  dport=https
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.5 |<TCP  dport=http
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.5 |<TCP  dport=https
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.6 |<TCP  dport=http
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.6 |<TCP  dport=https
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.7 |<TCP  dport=http
|>>,
<IP  frag=0 proto=tcp dst=10.0.2.7 |<TCP  dport=https
|>>]
```

3) SYN-ACK Test: Now, let's try to do some fun things. The `sr()` function in Scapy is for sending a packet and receiving corresponding answers. Its variant `sr1()` returns *only one answer.*

- We can send a TCP handshake request and receive the response by issuing the following command:

```
>>> sr1(IP(dst='74.125.130.104')/TCP(sport=5555,
dport=80,flags='S'))
```

- Now, change `dport` to 443. Can you get the answer (response) from the destination? What value does `flags` have in the answer? What does that mean?
- Change `dport` to 22. Can you get the answer (response) from the destination?
- We can also set multiple ports, e.g. dport=[21,22,25,53,80,110, 443,3306]. In this case, we need to use `sr()` to capture multiple responses.

```
>>> res= sr(IP(dst='74.125.130.104')/TCP(sport=5555,
    dport=[21,22,25,53,80,110,443,3306],flags='S'))
>>> res
(<Results: TCP:2 UDP:0 ICMP:0 Other:0>,
<Unanswered: TCP:6 UDP:0 ICMP:0 Other:0>)
>>> res[0].summary()
IP / TCP 10.0.2.15:5555 > 74.125.130.104:https S ==> IP
/ TCP 74.125.130.104:https > 10.0.2.15:5555 SA /
Padding
IP / TCP 10.0.2.15:5555 > 74.125.130.104:http S ==> IP
/ TCP 74.125.130.104:http > 10.0.2.15:5555 SA / Padding
>>> res[1].summary()
```

(See what happens.)

- Turn on Metasploitabl VM and get its IP address. Then issue the following command:

```
>>> sr1(IP(dst='<Metasploitable IP>')/TCP(sport=5555,
dport=80,flags='S'))
```

- Try other destination port (`dport`) numbers, such as 21, 22, 53 and 3306. Can you see the difference? Observe how `sport` in the response packet changes depending on the value of `dport`.

4) Read a pcap file and analyse packets
   - Download lab2.pcap file from the Moodle site.
   - Read the file using the following command:
     `>>> a=rdpcap("/home/kali/Desktop/lab2.pcap")` (depending on where your file is located, you can change the path)
   - For example, you can display the hexadecimal dump of the 23rd packet using the following command:
     `>>> hexdump(a[23])`
   - For example, you can visualize the 23rd packet using the following command:

     ```
     >>> a[23].pdfdump()
     [Note: If you hit error regarding pyx, you can issue
     the following commands in Kali terminal]
     #sudo apt-get update
     #sudo pip install pyx
     #sudo apt-get install texlive
     ```

   - Try more commands to get the information of the packet using the table below: (Hint: Replace "pkt" with the packet you have read, e.g. a[23])

hexdump(pkt)	**have a hexadecimal dump**
ls(pkt)	have the list of fields values
pkt.summary()	for a one-line summary
pkt.show()	for a developed view of the packet
pkt.show2()	same as show but on the assembled packet (checksum is calculated, for instance)
pkt.command()	return a Scapy command that can generate the packet

3. Python Programming with Scapy
   Scapy can also be used as a python library. You can write your own packet analysis program using Scapy.
   1) The first task is to write a python program to display the hexdump of the packet a[23] from Q2-3). Your code should begin with

```
from scapy.all import *
a=rdpcap("lab2.pcap")
```

Compile your python program by issuing `python filename.py` on the terminal. (The default version of kali's python compiler is Python3.)

2) Your second task is to write a python program that extracts the packet containing a string `password` from the downloaded pcap file.

   To do this, create your python file analysis.py, which has the following skeleton code:

```
from scapy.all import *
a=rdpcap("lab2.pcap")
for packet in a:
#Your code snippet…
```

   (Hint: Use the python str() function to convert each packet into a string. Then, use find method to search "password". Also, use the Scapy command to display packet.)

4. ICMP probing using Scapy
   Note that an ICMP packet for the destination can be created using scapy as follows `icmp_pkt=IP(dst='some ip')/ICMP()`. Using this, write a simple program to check whether destination is online.
   Hint: Use `sr1` function with the packet and `timeout = 1` to `10` as input (not to wait forever). If the `sr1()` returns `None` (which indicates null value in python), we conclude that the destination is not online. Once you get your code working, try to improve it.