SUNY Polytechnic Institute

NCS 450

Network Security

Wireshark Lab

Author:
Jess Yanarella
Tony Perez

Professor:
Ronny Bull

April 2015

1 Introduction

In this Lab we used Wireshark and tcpdump as our networking monitor tools. We set up a span port on the switch to monitor the traffic being sent and received through the gateway. A Kali Laptop is used on the span port to view all the traffic in our network.

2 Network Diagram:

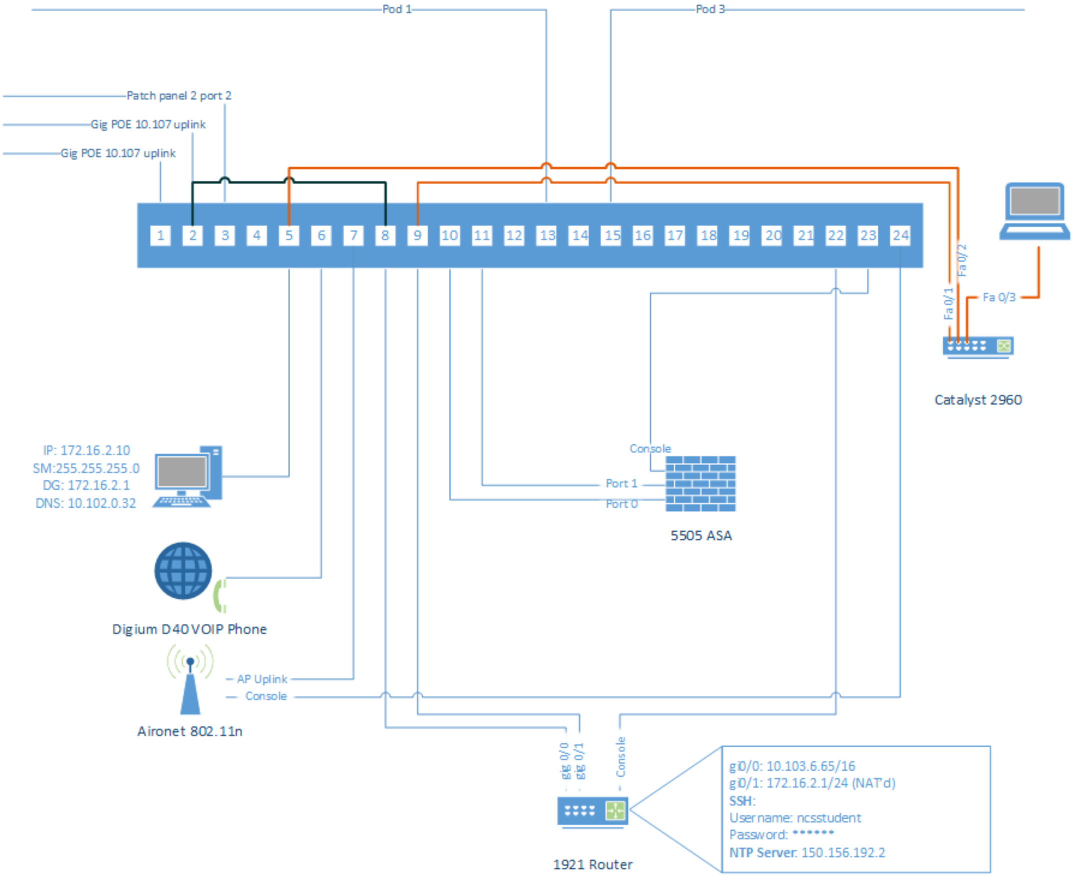


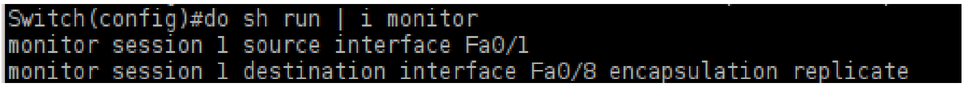
Figure 1: Network Setup

3 Span Port:

The span port is set up to receive a copy of a packets that the host sends. Our host is set up on Fa0/1 because it is the default gateway. Our sniffer machine is the Kali Laptop and connected to the span port.
In this lab, we need to first create a span port which is done on our switch. After we brought back our configuration from previous labs, we could go ahead and begin setting up the new

port. Our Span port we configured on the interface Fast Ethernet 0/8. All traffic is being sent through interface 0/1 and is being captured during our scans because it is the default gateway.

```
(config)# monitor session 1 source interface fa0/1
(config)# monitor session 1 destination interface fa0/8 encapsulation replicate
```



```
Switch(config)#do sh run | i monitor
monitor session 1 source interface Fa0/1
monitor session 1 destination interface Fa0/8 encapsulation replicate
```

Figure 2: Monitor Session

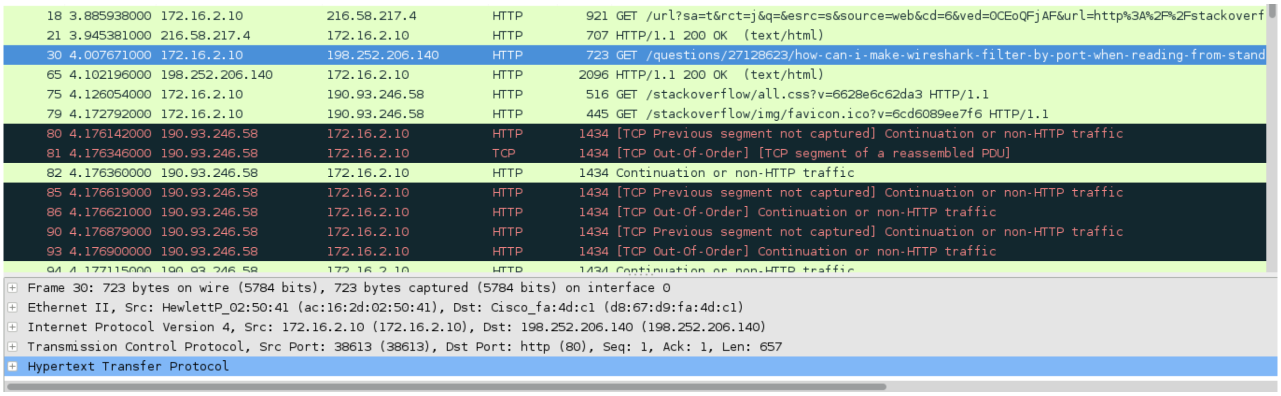
4 Kali Laptop:

Now we can take the Kali laptop and connected it to our newly created span port. We need to assign it an IP address so it is in the same network as our switch and Pod PC. We gave it an IP of 172.16.2.240 and default gateway of 172.16.2.1.

5 Wireshark:

Since our Kali laptop is now set up on the span port, we can begin creating traffic to be viewed in Wireshark. For our first tests, we began pinging different IP address to verify Kali is successfully set up.

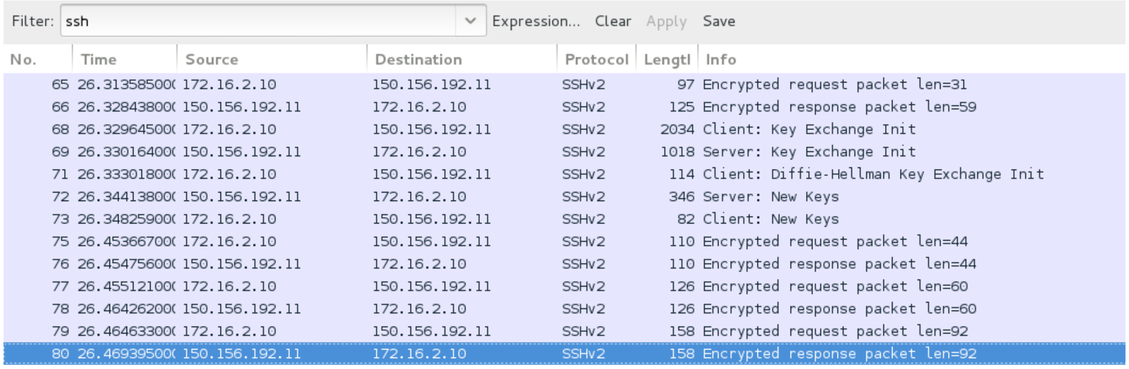
To generate some traffic, we connected to different websites on the Kali Laptop and performed different Google searches. In our Wireshark capture, we can sort by HTTP to filter our results. We were able to see the HTTP and GET requests. Highlighted in the following capture, we can clearly see the Google search we performed "how can i make Wireshark filter by port".



```
   18 3.885938000 172.16.2.10          216.58.217.4          HTTP        921 GET /url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&ved=0CEoQFjAF&url=http%3A%2F%2Fstackoverf
   21 3.945381000 216.58.217.4         172.16.2.10           HTTP        707 HTTP/1.1 200 OK  (text/html)
   30 4.007671000 172.16.2.10          198.252.206.140       HTTP        723 GET /questions/27128623/how-can-i-make-wireshark-filter-by-port-when-reading-from-stand
   65 4.102196000 198.252.206.140      172.16.2.10           HTTP       2096 HTTP/1.1 200 OK  (text/html)
   75 4.126054000 172.16.2.10          190.93.246.58         HTTP        516 GET /stackoverflow/all.css?v=6628e6c62da3 HTTP/1.1
   79 4.172792000 172.16.2.10          190.93.246.58         HTTP        445 GET /stackoverflow/img/favicon.ico?v=6cd6089ee7f6 HTTP/1.1
   80 4.176142000 190.93.246.58        172.16.2.10           HTTP       1434 [TCP Previous segment not captured] Continuation or non-HTTP traffic
   81 4.176346000 190.93.246.58        172.16.2.10           TCP        1434 [TCP Out-Of-Order] [TCP segment of a reassembled PDU]
   82 4.176360000 190.93.246.58        172.16.2.10           HTTP       1434 Continuation or non-HTTP traffic
   85 4.176619000 190.93.246.58        172.16.2.10           HTTP       1434 [TCP Previous segment not captured] Continuation or non-HTTP traffic
   86 4.176621000 190.93.246.58        172.16.2.10           HTTP       1434 [TCP Out-Of-Order] Continuation or non-HTTP traffic
   90 4.176879000 190.93.246.58        172.16.2.10           HTTP       1434 [TCP Previous segment not captured] Continuation or non-HTTP traffic
   93 4.176900000 190.93.246.58        172.16.2.10           HTTP       1434 [TCP Out-Of-Order] Continuation or non-HTTP traffic
   94 4.177115000 190.93.246.58        172.16.2.10           HTTP       1434 Continuation or non-HTTP traffic
⊞ Frame 30: 723 bytes on wire (5784 bits), 723 bytes captured (5784 bits) on interface 0
⊞ Ethernet II, Src: HewlettP_02:50:41 (ac:16:2d:02:50:41), Dst: Cisco_fa:4d:c1 (d8:67:d9:fa:4d:c1)
⊞ Internet Protocol Version 4, Src: 172.16.2.10 (172.16.2.10), Dst: 198.252.206.140 (198.252.206.140)
⊞ Transmission Control Protocol, Src Port: 38613 (38613), Dst Port: http (80), Seq: 1, Ack: 1, Len: 657
⊞ Hypertext Transfer Protocol
```

Figure 3: Wireshark Capture

On our Pod PC, we can ssh to the Kali Laptop set up on the span port. During this ssh connection, packets are being transmitted through our Wireshark scan. We can see that the source is from our PC and the destination is the IP address of the laptop. The packets being transmitted are Encryption request/response and the server/client keys being made.

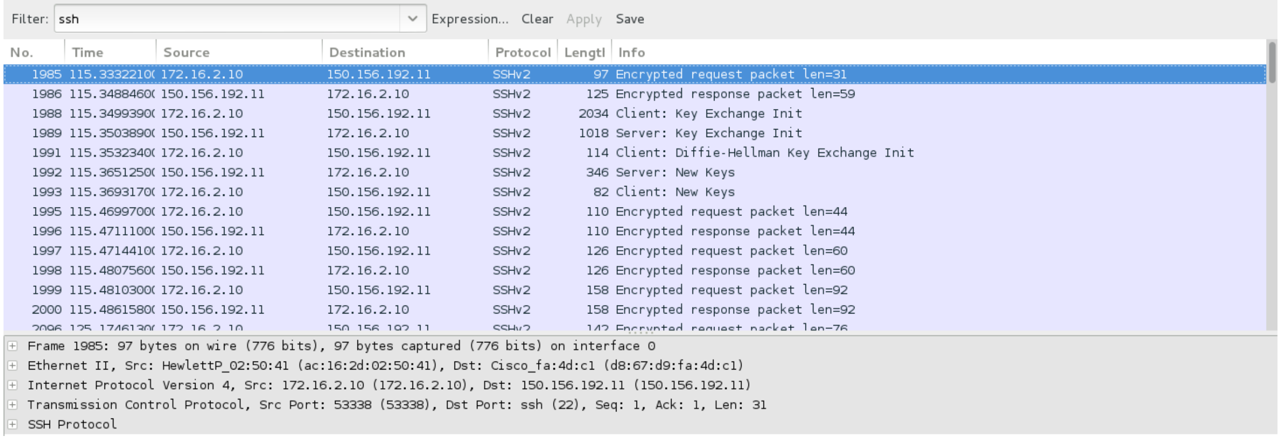


Filter: ssh

No.	Time	Source	Destination	Protocol	Lengtl	Info
65	26.31358500(	172.16.2.10	150.156.192.11	SSHv2	97	Encrypted request packet len=31
66	26.32843800(	150.156.192.11	172.16.2.10	SSHv2	125	Encrypted response packet len=59
68	26.32964500(	172.16.2.10	150.156.192.11	SSHv2	2034	Client: Key Exchange Init
69	26.33016400(	150.156.192.11	172.16.2.10	SSHv2	1018	Server: Key Exchange Init
71	26.33301800(	172.16.2.10	150.156.192.11	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
72	26.34413800(	150.156.192.11	172.16.2.10	SSHv2	346	Server: New Keys
73	26.34825900(	172.16.2.10	150.156.192.11	SSHv2	82	Client: New Keys
75	26.45366700(	172.16.2.10	150.156.192.11	SSHv2	110	Encrypted request packet len=44
76	26.45475600(	150.156.192.11	172.16.2.10	SSHv2	110	Encrypted response packet len=44
77	26.45512100(	172.16.2.10	150.156.192.11	SSHv2	126	Encrypted request packet len=60
78	26.46426200(	150.156.192.11	172.16.2.10	SSHv2	126	Encrypted response packet len=60
79	26.46463300(	172.16.2.10	150.156.192.11	SSHv2	158	Encrypted request packet len=92
80	26.46939500(	150.156.192.11	172.16.2.10	SSHv2	158	Encrypted response packet len=92

Figure 4: Wireshark Capture PC to Kali

Next, we can do the same process but this time we will ssh from the Kali Laptop to the POD PC. In this case, the Laptop will be the source and PC is the destination.



Filter: ssh

No.	Time	Source	Destination	Protocol	Lengtl	Info
1985	115.3332210(	172.16.2.10	150.156.192.11	SSHv2	97	Encrypted request packet len=31
1986	115.3488460(	150.156.192.11	172.16.2.10	SSHv2	125	Encrypted response packet len=59
1988	115.3499390(	172.16.2.10	150.156.192.11	SSHv2	2034	Client: Key Exchange Init
1989	115.3503890(	150.156.192.11	172.16.2.10	SSHv2	1018	Server: Key Exchange Init
1991	115.3532340(	172.16.2.10	150.156.192.11	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
1992	115.3651250(	150.156.192.11	172.16.2.10	SSHv2	346	Server: New Keys
1993	115.3693170(	172.16.2.10	150.156.192.11	SSHv2	82	Client: New Keys
1995	115.4699700(	172.16.2.10	150.156.192.11	SSHv2	110	Encrypted request packet len=44
1996	115.4711100(	150.156.192.11	172.16.2.10	SSHv2	110	Encrypted response packet len=44
1997	115.4714410(	172.16.2.10	150.156.192.11	SSHv2	126	Encrypted request packet len=60
1998	115.4807560(	150.156.192.11	172.16.2.10	SSHv2	126	Encrypted response packet len=60
1999	115.4810300(	172.16.2.10	150.156.192.11	SSHv2	158	Encrypted request packet len=92
2000	115.4861580(	150.156.192.11	172.16.2.10	SSHv2	158	Encrypted response packet len=92
2096	125.1746130(	172.16.2.10	150.156.192.11	SSHv2	142	Encrypted request packet len=76

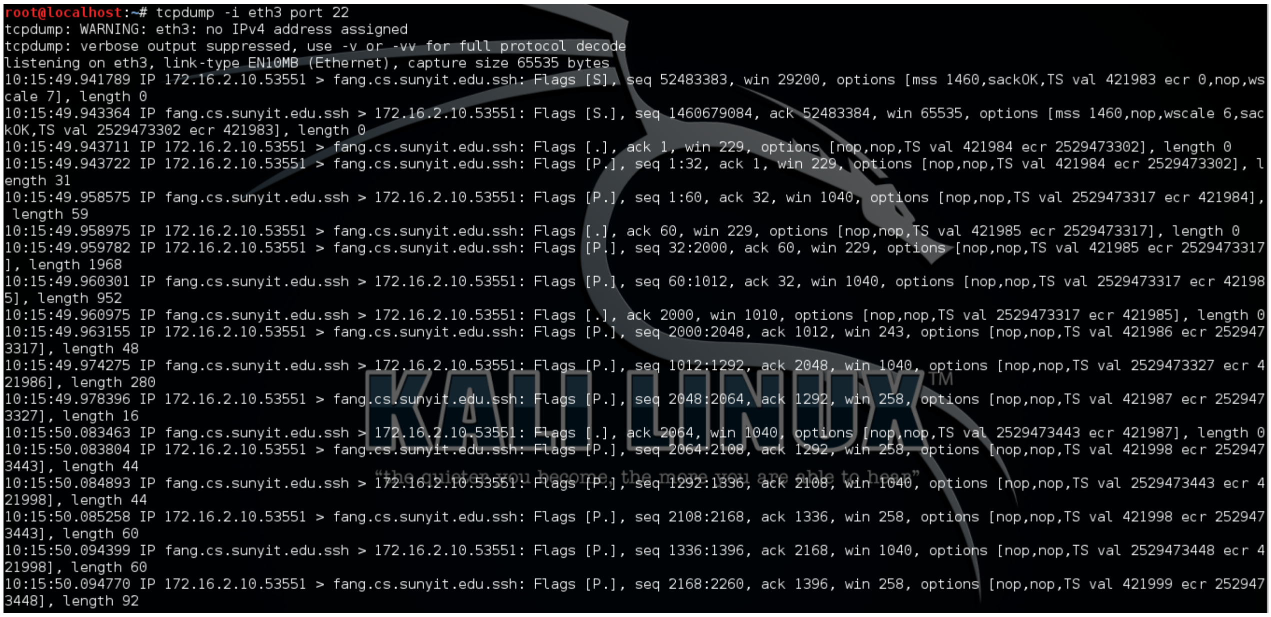
```
⊞ Frame 1985: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface 0
⊞ Ethernet II, Src: HewlettP_02:50:41 (ac:16:2d:02:50:41), Dst: Cisco_fa:4d:c1 (d8:67:d9:fa:4d:c1)
⊞ Internet Protocol Version 4, Src: 172.16.2.10 (172.16.2.10), Dst: 150.156.192.11 (150.156.192.11)
⊞ Transmission Control Protocol, Src Port: 53338 (53338), Dst Port: ssh (22), Seq: 1, Ack: 1, Len: 31
⊞ SSH Protocol
```

Figure 5: Wireshark Capture Kali to PC

6 TCPDump:

Tcpdump is another type of network analysis tool on Kali instead of using Wireshark. This tool is done in the command line not with a GUI interface. For this scan, we ssh to fang on the POD PC. In the output of the tcpdump tool, we clearly see the connection of the PC IP address 172.16.2.10 to the server `fang.cs.sunyit.edu.ssh`. Within each packet it outputs, it shows us the following information for seq, ack, and win. At the end, any Flags there were found are outputed in brackets.

```
# tcpdump -i eth3 port 22
```



```
root@localhost:~# tcpdump -i eth3 port 22
tcpdump: WARNING: eth3: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth3, link-type EN10MB (Ethernet), capture size 65535 bytes
10:15:49.941789 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [S], seq 52483383, win 29200, options [mss 1460,sackOK,TS val 421983 ecr 0,nop,ws
cale 7], length 0
10:15:49.943364 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [S.], seq 1460679084, ack 52483384, win 65535, options [mss 1460,nop,wscale 6,sac
kOK,TS val 2529473302 ecr 421983], length 0
10:15:49.943711 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [.], ack 1, win 229, options [nop,nop,TS val 421984 ecr 2529473302], length 0
10:15:49.943722 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 1:32, ack 1, win 229, options [nop,nop,TS val 421984 ecr 2529473302], l
ength 31
10:15:49.958575 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [P.], seq 1:60, ack 32, win 1040, options [nop,nop,TS val 2529473317 ecr 421984],
 length 59
10:15:49.958975 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [.], ack 60, win 229, options [nop,nop,TS val 421985 ecr 2529473317], length 0
10:15:49.959782 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 32:2000, ack 60, win 229, options [nop,nop,TS val 421985 ecr 2529473317
], length 1968
10:15:49.960301 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [P.], seq 60:1012, ack 32, win 1040, options [nop,nop,TS val 2529473317 ecr 42198
5], length 952
10:15:49.960975 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [.], ack 2000, win 1010, options [nop,nop,TS val 2529473317 ecr 421985], length 0
10:15:49.963155 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 2000:2048, ack 1012, win 243, options [nop,nop,TS val 421986 ecr 252947
3317], length 48
10:15:49.974275 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [P.], seq 1012:1292, ack 2048, win 1040, options [nop,nop,TS val 2529473327 ecr 4
21986], length 280
10:15:49.978396 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 2048:2064, ack 1292, win 258, options [nop,nop,TS val 421987 ecr 252947
3327], length 16
10:15:50.083463 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [.], ack 2064, win 1040, options [nop,nop,TS val 2529473443 ecr 421987], length 0
10:15:50.083804 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 2064:2108, ack 1292, win 258, options [nop,nop,TS val 421998 ecr 252947
3443], length 44
10:15:50.084893 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [P.], seq 1292:1336, ack 2108, win 1040, options [nop,nop,TS val 2529473443 ecr 4
21998], length 44
10:15:50.085258 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 2108:2168, ack 1336, win 258, options [nop,nop,TS val 421998 ecr 252947
3443], length 60
10:15:50.094399 IP fang.cs.sunyit.edu.ssh > 172.16.2.10.53551: Flags [P.], seq 1336:1396, ack 2168, win 1040, options [nop,nop,TS val 2529473448 ecr 4
21998], length 60
10:15:50.094770 IP 172.16.2.10.53551 > fang.cs.sunyit.edu.ssh: Flags [P.], seq 2168:2260, ack 1396, win 258, options [nop,nop,TS val 421999 ecr 252947
3448], length 92
```

Figure 6: TCPDump SSH

7 Conclusion:

In this lab we learned how to configure a span port on our switch to monitor the traffic going in the interface. We used the traffic from the default gateway interface that goes to the router. We only used this port because all traffic would go through this port and so there was no need to mirror traffic from the other interfaces. We were able to see the traffic from our PC and Laptop and verify the connections we made, ssh and web surfing.

References

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

http://www.tecmint.com/12-tcpdump-commands-a-network-sniffer-tool/