
CSG1105 Workshop Five

1 INTRODUCTION

This week we are going to explore aspects of TCP (Transmission Control Protocol) and Network Address Translation. These are all features of the Transport Layer. This layer is responsible for application multiplexing, segmentation of messages, reliable transfer and flow control.

2 EXERCISE: USER DATAGRAM PROTOCOL

The User Datagram Protocol is for short messages that will fit within a 516 byte limit. It is also used for protocols that manage their own connections, not relying on TCP to do so. In this exercise, we will use **Wireshark** to capture some traffic and then examine some UDP datagrams.

1. Start Wireshark capturing packets on your active network interface.
2. Use your browser to open a few websites
3. Stop the Wireshark capture
4. Enter "udp" into the **display filter** bar and click on the white with blue background arrow at the end of the bar to apply the filter
5. There will be many protocols displayed such as DNS and SNMP (simple network management protocol)
6. Select a **IPv4 DNS** entry. Wireshark will also show the DNS response. Note: This is **NOT** a connection, it is a query and a response sent and received in separate UDP datagrams.
7. Examine both the IP and UDP headers. The IP packet will contain the destination and source IP addresses. The UDP datagram will contain the destination and source **ports**. Recall that both of these are required to address a message to a server and for the server to reply to the client
8. Examine the DNS query and response, noting how much data was exchanged.

3 EXERCISE: TCP CONNECTIONS

This exercise and the next will be completed in Wireshark using packet capture files already generated. You will need to download from Blackboard two packet capture files:

NAT_Linux_inside.pcapng and NAT_Windows_outside.pcapng. The Linux PC is inside a NAT gateway using local addresses, the Windows PC is on a network outside the NAT

gateway using a different network. (In reality, there is another NAT gateway between the Windows PC and the Internet.)

1. Run Wireshark, but don't select the active network connection.
2. Use the file menu to load NAT_Linux_inside.pcapng.
3. It will display a set of frames captured using **tcpdump** on Linux. This may be configured to produce a Wireshark compatible frame capture file.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.75.129	23.215.164.221	TCP	74	45880 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK
2	0.036241	23.215.164.221	192.168.75.129	TCP	60	80 → 45880 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.036264	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=1 Ack=1 Win=29200 Len=0
4	0.036332	192.168.75.129	23.215.164.221	HTTP	195	GET / HTTP/1.1
5	0.036460	23.215.164.221	192.168.75.129	TCP	60	80 → 45880 [ACK] Seq=1 Ack=142 Win=64240 Len=0
6	0.078922	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=1 Ack=142 Win=64240 Len=1400
7	0.078937	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=1461 Win=32120 Len=0
8	0.079120	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=1461 Ack=142 Win=64240 Len=1400
9	0.079126	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=2921 Win=35040 Len=0
10	0.079470	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=2921 Ack=142 Win=64240 Len=1400
11	0.079485	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=4381 Win=37960 Len=0
12	0.080101	23.215.164.221	192.168.75.129	TCP	2974	80 → 45880 [PSH, ACK] Seq=4381 Ack=142 Win=64240 Len=2860
13	0.080106	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=7301 Win=43800 Len=0
14	0.080429	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=7301 Ack=142 Win=64240 Len=1400
15	0.080433	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=8761 Win=46720 Len=0
16	0.080808	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=8761 Ack=142 Win=64240 Len=1400
17	0.080812	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=10221 Win=49640 Len=0
18	0.081339	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=10221 Ack=142 Win=64240 Len=1400
19	0.081343	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=11681 Win=52560 Len=0
20	0.081701	23.215.164.221	192.168.75.129	TCP	1514	80 → 45880 [PSH, ACK] Seq=11681 Ack=142 Win=64240 Len=1400
21	0.081705	192.168.75.129	23.215.164.221	TCP	54	45880 → 80 [ACK] Seq=142 Ack=13141 Win=55480 Len=0
22	0.082035	23.215.164.221	192.168.75.129	TCP	2974	80 → 45880 [PSH, ACK] Seq=13141 Ack=142 Win=64240 Len=2860

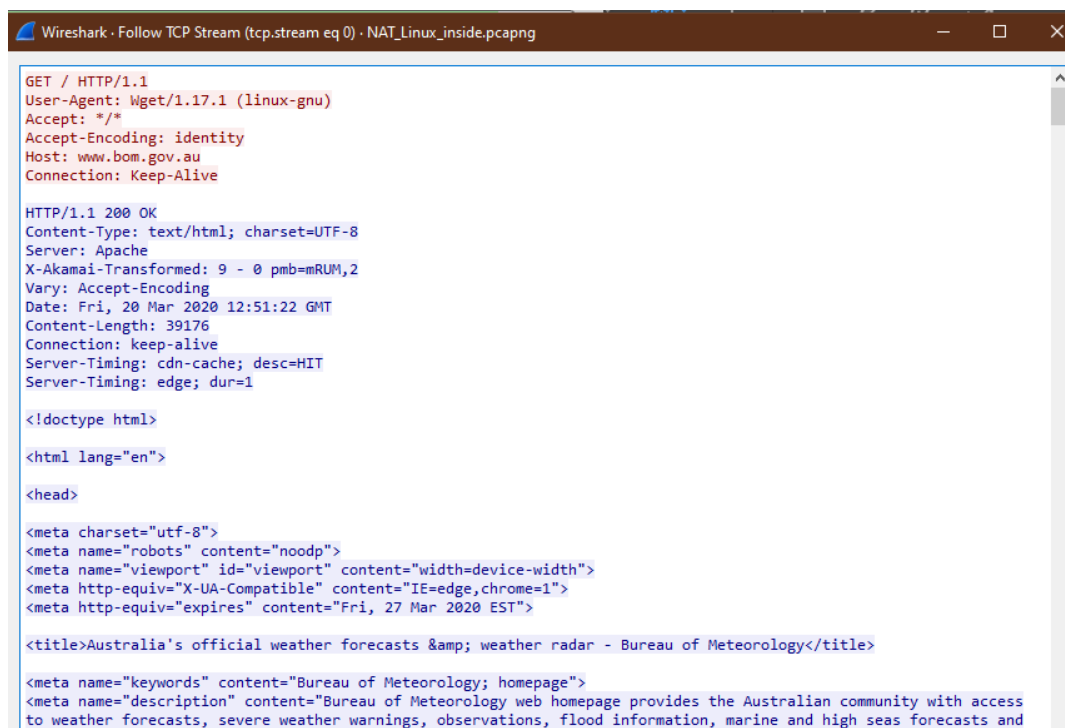
> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface unknown, id 0
 > Ethernet II, Src: VMware_ed:31:a3 (00:0c:29:ed:31:a3), Dst: VMware_f6:05:54 (00:50:56:f6:05:54)
 > Internet Protocol Version 4, Src: 192.168.75.129, Dst: 23.215.164.221
 > Transmission Control Protocol, Src Port: 45880, Dst Port: 80, Seq: 0, Len: 0

Source Port: 45880
 Destination Port: 80
 [Stream index: 0]
 [TCP Segment Len: 0]
 Sequence number: 0 (relative sequence number)
 Sequence number (raw): 2544597062
 [Next sequence number: 1 (relative sequence number)]
 Acknowledgment number: 0
 Acknowledgment number (raw): 0
 1010 ... = Header Length: 40 bytes (10)

Flags: 0x002 (SYN)
 000. = Reserved: Not set
 ...0 = Nonce: Not set
0... = Congestion Window Reduced (CWR): Not set
0... = ECN-Echo: Not set
0... = Urgent: Not set
0... = Acknowledgment: Not set
0... = Push: Not set
0... = Reset: Not set
0... = Syn: Set

4. Examine the first three segments in the TCP connection. Make sure to expand the flags section.
5. Note the use of the source and destination ports (45880 and 80). The destination port, 80, is the location of a Web Server listening for connections
6. The first segment is from the client to the server and will have the 'SYN' flag set (1) and a **relative** sequence number of zero (the actual sequence number is randomly assigned to assist in the prevention of session hijacking)
7. The second segment is from the server to the client and will have both the 'SYN' and 'ACK' flags set. The server will set a sequence number randomly (again with a relative value of zero) and acknowledge the SYN with an Acknowledgement of 1 (next expected sequence number)
8. The third segment is from the client to the server and has the 'ACK' flag set, a sequence number of 1 and acknowledges the next expected sequence number of 1 from the server.

9. These three segments are the **Three Way Handshake** that establishes a TCP connection.
10. Now that the TCP connection is established, the fourth segment is a HTTP get request from the client to the server
11. Frame number six is the beginning of the transfer of a web page from the server to the client. There is a jump in sequence numbers as some irrelevant frames have been filtered from the capture. Trace a number of frames and see the use of sequence numbers in both directions
12. With frame six selected, from the **analyze** menu, select **Follow TCP stream**. You will see a window pop up with the contents of the web page retrieved.



```

Wireshark - Follow TCP Stream (tcp.stream eq 0) - NAT_Linux_inside.pcapng

GET / HTTP/1.1
User-Agent: Wget/1.17.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: www.bom.gov.au
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Server: Apache
X-Akamai-Transformed: 9 - 0 pmb=mRUM,2
Vary: Accept-Encoding
Date: Fri, 20 Mar 2020 12:51:22 GMT
Content-Length: 39176
Connection: keep-alive
Server-Timing: cdn-cache; desc=HIT
Server-Timing: edge; dur=1

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">
<meta name="robots" content="noodp">
<meta name="viewport" id="viewport" content="width=device-width">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta http-equiv="expires" content="Fri, 27 Mar 2020 EST">

<title>Australia's official weather forecasts & weather radar - Bureau of Meteorology</title>

<meta name="keywords" content="Bureau of Meteorology; homepage">
<meta name="description" content="Bureau of Meteorology web homepage provides the Australian community with access to weather forecasts, severe weather warnings, observations, flood information, marine and high seas forecasts and
  
```

13. Close the pop up window and scroll to the end of the capture
14. Examine Segments 44 -> 47. This show the closing of a TCP connection. Segment 44 acknowledges the previous segment but also sets the **FIN** flag. The following two segments perform an Acknowledgement then a 'FIN' from the server end.
15. A final 'ACK' closes the connection.

4 NETWORK ADDRESS TRANSLATION

1. Use Wireshark to open the second capture file
2. Use the file menu to load NAT_Windows_outside.pcapng
3. This file was captured on Windows outside a NAT gateway
4. Arrange the two Wireshark windows side-by-side
5. Compare the contents of the first few frames in both windows

6. First, the Linux inside capture

1	0.000000	192.168.75.129	23.215.164.221	TCP	74 45880 → 80 [SYN] Seq=0 Win=29200
2	0.036241	23.215.164.221	192.168.75.129	TCP	60 80 → 45880 [SYN, ACK] Seq=0 Ack=1
3	0.036264	192.168.75.129	23.215.164.221	TCP	54 45880 → 80 [ACK] Seq=1 Ack=1 Win=
4	0.036332	192.168.75.129	23.215.164.221	HTTP	195 GET / HTTP/1.1
5	0.036460	23.215.164.221	192.168.75.129	TCP	60 80 → 45880 [ACK] Seq=1 Ack=142 Wi
6	0.078922	23.215.164.221	192.168.75.129	TCP	1514 80 → 45880 [PSH, ACK] Seq=1 Ack=1
7	0.078937	192.168.75.129	23.215.164.221	TCP	54 45880 → 80 [ACK] Seq=142 Ack=1461
8	0.079120	23.215.164.221	192.168.75.129	TCP	1514 80 → 45880 [PSH, ACK] Seq=1461 Acl
9	0.079126	192.168.75.129	23.215.164.221	TCP	54 45880 → 80 [ACK] Seq=142 Ack=2921
10	0.079470	23.215.164.221	192.168.75.129	TCP	1514 80 → 45880 [PSH, ACK] Seq=2921 Acl
11	0.079485	192.168.75.129	23.215.164.221	TCP	54 45880 → 80 [ACK] Seq=142 Ack=4381
12	0.080101	23.215.164.221	192.168.75.129	TCP	2974 80 → 45880 [PSH, ACK] Seq=4381 Acl

7. Next the Windows outside capture

1	0.000000	192.168.0.254	23.215.164.221	TCP	66 23936 → 80 [SYN] Seq=0 Win=64240
2	0.035856	23.215.164.221	192.168.0.254	TCP	66 80 → 23936 [SYN, ACK] Seq=0 Ack=1
3	0.035922	192.168.0.254	23.215.164.221	TCP	54 23936 → 80 [ACK] Seq=1 Ack=1 Win=1
4	0.036216	192.168.0.254	23.215.164.221	HTTP	195 GET / HTTP/1.1
5	0.073133	23.215.164.221	192.168.0.254	TCP	60 80 → 23936 [ACK] Seq=1 Ack=142 Wir
6	0.078326	23.215.164.221	192.168.0.254	TCP	1514 80 → 23936 [ACK] Seq=1 Ack=142 Wir
7	0.078802	23.215.164.221	192.168.0.254	TCP	1514 80 → 23936 [ACK] Seq=1461 Ack=142
8	0.078855	192.168.0.254	23.215.164.221	TCP	54 23936 → 80 [ACK] Seq=142 Ack=2921
9	0.079183	23.215.164.221	192.168.0.254	TCP	1514 80 → 23936 [ACK] Seq=2921 Ack=142
10	0.079787	23.215.164.221	192.168.0.254	TCP	1514 80 → 23936 [ACK] Seq=4381 Ack=142
11	0.079788	23.215.164.221	192.168.0.254	TCP	1514 80 → 23936 [ACK] Seq=5841 Ack=142
12	0.079814	192.168.0.254	23.215.164.221	TCP	54 23936 → 80 [ACK] Seq=142 Ack=7301

- Examine the IP addresses and port numbers expand the IP and TCP headers
- The inside IP is **192.168.75.129** and the the inside port is **45880**
- The outside IP is **192.168.0.254** and the outside port is **23936**. (Yes, this is a **Private IP address**, there's another NAT gateway out to the Internet. The Linux capture was performed in a VM with the hypervisor (Vmware) providing the NAT gateway. If you wanted to repeat the entire exercise yourself, install Wireshark on Ubuntu in a VM and perform simultaneous captures on the host OS as well as on Linux)
- Examine segments and packets pairwise between the two captures. You will see that other than the re-written IPs and port numbers, the contents are identical and the end user would be unaware that NAT was in action.
- The TCP sequence numbers are identical (although the TCP window size changes across the NAT gateway)
- To confirm this, use Analyse->Follow TCP Stream in both Wireshark windows. Compare and you will see identical HTTP streams (including the User_Agent field identifying Wget as the client)

5 SUMMARY

We have looked at a simple UDP datagram where the message is in a single packet and no setup is required. We have also looked at the establishment, operation and closing of a TCP connection to send a single HTML page. There is considerable overhead to establish a TCP connection, but it provides reliability and flow control for the connection. Lastly, we looked at the same TCP connection on either side of a NAT gateway and have seen that the contents of the connection are unaltered, only the IP and port numbers are rewritten.