

Creating Normal Network Traffic

The purpose of this lab is to introduce you to how the TCP and UDP protocols operate. This lab focuses on the fundamentals of analyzing the protocol behavior for security professionals.

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are core functions for how packets function and allowing applications to communicate effectively. You are going to start by generating some standard web traffic to get a better understanding on how these transport protocols function when used normally.

This first challenge will be using the Windows endpoint.



Important Note: We are building a custom environment just for you! This may take a few minutes depending on its complexity.

- On the desktop, there is a shortcut called "STATUS". Double-clicking this will give you an overview on applications installing and/or lab files being downloaded. This does not update automatically - you will need to close and reopen to get an updated progress status. For most applications, an icon will appear on the desktop once the installation is complete!

While you wait, check out some relevant resources:

- [Transport Layer](#)
- [Transmission Control Protocol \(TCP\)](#)
- [User Datagram Protocol \(UDP\)](#)

1. You'll want to start by opening Wireshark via the shortcut on the desktop and double clicking the "*Ethernet*" interface.

Note: This will allow you to capture live traffic you are about to generate. Leave this up and running in the background.

- Next, open up Firefox and browse to: <https://www.globomantics.com>. If you encounter any certificate errors, click "Show Advanced" and "Proceed to www.globomantics.com".
- Return to Wireshark and stop the capture so you can inspect the traffic.

Analysis: How many packets do you think you generated with that single request? A simple web connection may generate more packets than initially anticipated.

- Since you have an active RDP session to the machine, you'll need to filter that traffic out. Add a filter for `not tcp.port == 3389`.

The screenshot shows the Wireshark interface with the filter `not tcp.port == 3389` applied. The packet list shows a DNS query (No. 599) and a DNS response (No. 601) for `www.globomantics.com`. The packet details pane for the selected DNS response shows the following information:

- > Frame 599: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface \Device\NPF_{54B31D7E-36BF-4BBE-9AB2-106A93...}
- > Ethernet II, Src: 06:39:dc:7b:95:f7 (06:39:dc:7b:95:f7), Dst: 06:e9:c8:8f:e6:93 (06:e9:c8:8f:e6:93)
- > Internet Protocol Version 4, Src: 172.31.24.5, Dst: 172.31.0.2
- > User Datagram Protocol, Src Port: 55450, Dst Port: 53
- > Domain Name System (query)

- What's left over? You'll notice some packets near the top for "DNS". By default, DNS uses UDP for transport, so click one of those packets.

Note: When you browse to a domain (ex. www.globomantics.com), the domain must first be translated into an IP address. The protocol that allows for this translation to happen is DNS, or Domain Name System.

- Expand the "User Datagram Protocol" (UDP) section in Wireshark to see the individual fields. Here you can see there are only four fields: source port, destination port, length, and checksum. UDP is considered a "connectionless" transport protocol, meaning there are no built-in methods for confirming packets successfully reached their destination. This allows for the overhead on packets to be much smaller which is perfect for applications like DNS.

- > Frame 599: 80 bytes on wire (640 bits), 80 bytes captured
- > Ethernet II, Src: 06:39:dc:7b:95:f7 (06:39:dc:7b:95:f7), D
- > Internet Protocol Version 4, Src: 172.31.24.5, Dst: 172.31
- ▼ User Datagram Protocol, Src Port: 55450, Dst Port: 53
 - Source Port: 55450
 - Destination Port: 53
 - Length: 46
 - Checksum: 0x7085 [unverified]
 - [Checksum Status: Unverified]
 - [Stream index: 0]
 - > [Timestamps]
 - UDP payload (38 bytes)
- > Domain Name System (query)

Analysis: It is important to understand what protocols use TCP and UDP by default. This will help you find anomalies throughout your network traffic. For example, DNS can use TCP, but it only does so when sending large payloads. When used legitimately, this might be a DNS Zone transfer. When use maliciously, this is how you might catch DNS tunneling or exfiltration.

7. In contrast, the web connection will be using TCP for transport. Scroll down in Wireshark until you see TCP packets with a destination port of 443. This should be right after the DNS packets.
8. Taking a quick glance at this list will reveal three TCP packets followed by packets marked as "TLSv1.3" mixed with additional TCP packets.

No.	Time	Source	Destination	Protocol	Length	Info
1822	38.406750	52.43.167.167	172.31.24.5	TLSv1.2	92	Application Data
1823	38.406768	172.31.24.5	52.43.167.167	TCP	54	50064 → 443 [ACK] Seq=1271 Ack=6359 Win=2101248 Len=0
1824	38.410201	172.31.24.5	172.31.0.2	DNS	79	Standard query 0x7b42 A www.pluralsight.com
1827	38.435957	172.31.24.5	172.31.0.2	DNS	79	Standard query 0x7b42 A www.pluralsight.com
1830	38.481750	172.31.0.2	172.31.24.5	DNS	163	Standard query response 0x7b42 A www.pluralsight.com CNAME www.pluralsight.com.cdn
1831	38.481927	172.31.0.2	172.31.24.5	DNS	163	Standard query response 0x7b42 A www.pluralsight.com CNAME www.pluralsight.com.cdn
1832	38.482403	172.31.24.5	104.19.161.127	TCP	66	50065 → 443 [SYN, ECN, CWR] Seq=0 Win=62727 Len=0 MSS=8961 WS=256 SACK_PERM=1
1833	38.488080	104.19.161.127	172.31.24.5	TCP	66	443 → 50065 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024
1834	38.488156	172.31.24.5	104.19.161.127	TCP	54	50065 → 443 [ACK] Seq=1 Ack=1 Win=2105600 Len=0
1835	38.488433	172.31.24.5	104.19.161.127	TLSv1.3	571	Client Hello
1838	38.493988	104.19.161.127	172.31.24.5	TCP	54	443 → 50065 [ACK] Seq=1 Ack=518 Win=68608 Len=0
1839	38.497500	104.19.161.127	172.31.24.5	TLSv1.3	1514	Server Hello, Change Cipher Spec
1840	38.497501	104.19.161.127	172.31.24.5	TLSv1.3	723	Application Data

9. Click on one of the TCP packets and expand the details pane to reveal all the fields. Compared to UDP, there are a lot more. Near the top you can see fields for sequence and acknowledgement numbers. TCP is considered a "connection-oriented" transport protocol and it uses these fields to guarantee every packet is received and the data is pieced together in the correct order. This is a necessary overhead to have when ensuring the delivery of a web page.

```

> Frame 1832: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{54B3:
> Ethernet II, Src: 06:39:dc:7b:95:f7 (06:39:dc:7b:95:f7), Dst: 06:e9:c8:8f:e6:93 (06:e9:c8:8f:e6:93)
> Internet Protocol Version 4, Src: 172.31.24.5, Dst: 104.19.161.127
▼ Transmission Control Protocol, Src Port: 50065, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 50065
    Destination Port: 443
    [Stream index: 35]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0 (relative sequence number)
    Sequence Number (raw): 1320262682
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
> Flags: 0x0c2 (SYN, ECN, CWR)
    Window: 62727
    [Calculated window size: 62727]
    Checksum: 0xcddd [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), I
> [Timestamps]

```

Now that you understand the fundamental difference between these two transport protocols, you can move on to understanding the how specific fields may be used for malicious purposes or how you might detect anomalies when these protocols don't behave as expected.

It is important to note, while many packets may be marked as "TLSv1.3" or some other application protocol, TCP or UDP will still be used for transport and the data will still exist for all relevant packets.

```

> Frame 1835: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface \Device\NPF_{5
> Ethernet II, Src: 06:39:dc:7b:95:f7 (06:39:dc:7b:95:f7), Dst: 06:e9:c8:8f:e6:93 (06:e9:c8:8f:e6:93)
> Internet Protocol Version 4, Src: 172.31.24.5, Dst: 104.19.161.127
> Transmission Control Protocol, Src Port: 50065, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
> Transport Layer Security

```