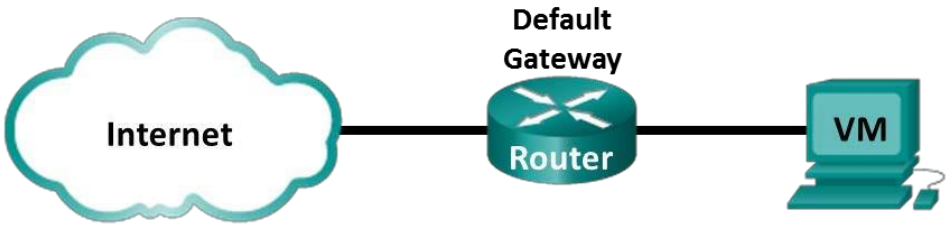
Lab 4.6.2.7 - Using Wireshark to Examine a UDP DNS Capture

This lab has been updated for use on NETLAB+.
www.netdevgroup.com

Topology



Objectives

Part 1: Record a PC's IP Configuration Information

Part 2: Analyze Captured DNS or UDP Packets

Background / Scenario

When you use the Internet, you use the Domain Name System (DNS). DNS is a distributed network of servers that translates user-friendly domain names like www.google.com to an IP address. W hen you type a website URL into your browser, your PC performs a DNS query to the DNS server's IP address. Your PC's DNS query and the DNS server's response make use of the User Datagram Protocol (UDP) as the transport layer protocol. UDP is connectionless and does not require a session setup as does TCP. DNS queries and responses are very small and do not require the overhead of TCP.

In this lab, you will communicate with a DNS server by sending a DNS query using the UDP transport protocol. You will use Wireshark to examine the DNS query and response exchanges with the same server.

Part 1: Record VM's IP Configuration Information

In *Part 1*, you will use commands on your *CyberOps Workstation* VM to find and record the MAC and IP addresses of your VM's virtual network interface card (NIC), the IP address of the specified default gateway, and the DNS server IP address specified for the PC. Record this information in the table provided. The information will be used in parts of this lab with packet analysis.

IP address	192.168.0.11
MAC address	08:00:27:23:b2:31
Default gateway IP address	209.165.200.235
DNS server IP address	192.168.0.1

a. Launch the **CyberOps** VM. Log in with username `analyst` and the password `cyberops`.

b. Open a **terminal** in the VM. Enter `ifconfig` at the prompt to display interface information.

```
[analyst@secOps ~]$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.11  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe23:b231  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:23:b2:31  txqueuelen 1000  (Ethernet)
        RX packets 1381  bytes 87320 (85.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 1857 (1.8 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 19  base 0xd000
<some output omitted>
```

c. At the terminal prompt, enter `cat /etc/resolv.conf` to determine the DNS server.

```
[analyst@secOps ~]$ cat /etc/resolv.conf
# Generated by resolvconf nameserver 209.165.200.235
```

d. At the terminal prompt, enter **netstat -r** to display the IP routing table to the default gateway IP address.

```
[analyst@secOps ~]$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.0.1     0.0.0.0         UG        0 0          0 ens33
192.168.0.0     0.0.0.0         255.255.255.0   U         0 0          0 ens33
```

Note: The DNS IP address and default gateway IP address are often the same, especially in small networks. However, in a business or school network, the addresses would most likely be different.

Part 2: Analyze Captured DNS or UDP Packets

In *Part 2*, you will examine the *UDP* packets that were generated when communicating with a *DNS* server for the IP addresses for www.google.com.

Step 1: Filter DNS packets.

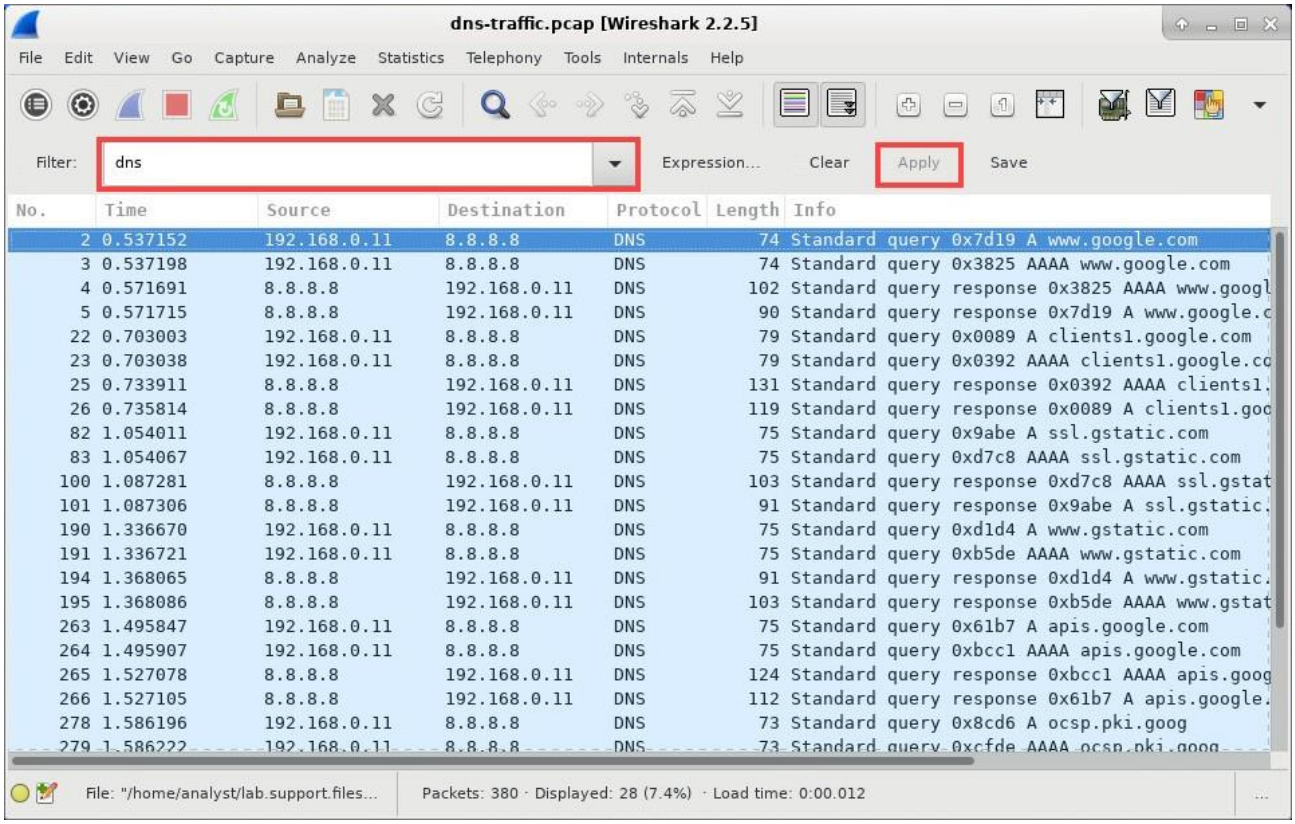
a. In the terminal window, start Wireshark and click **OK** when prompted.

```
[analyst@secOps ~]$ sudo wireshark-gtk
[sudo] password for analyst:


** (wireshark-gtk:950): WARNING **: Couldn't connect to accessibility bus:
Failed to connect to socket /tmp/dbus-REDRWOHelr: Connection refused
Gtk-Message: GtkDialog mapped without a transient parent. This is
discouraged.
```

b. In the *Wireshark* main window, navigate to **File > Open** and open the **dns-traffic.pcap** file from *~/lab.support.files/pcaps/*.

c. Filter the results by typing `dns` in the *Filter* field. Click **Apply**.



d. In the packet list pane (top section) of the main window, locate the packet that includes *Standard query* and *A www.google.com*. See frame 2 above as an example.

Step 2: Examine the fields in a DNS query packet.

The protocol fields, highlighted in gray, are displayed in the packet details pane (middle section) of the main window.

a. In the first line in the packet details pane, frame 2 had 74 bytes of data on the wire. This is the number of bytes it took to send a *DNS* query to a named server requesting the IP addresses of *www.google.com*. If you used a different web address, such as *www.cisco.com*, the byte count might be different.

b. The *Ethernet II* line displays the source and destination MAC addresses. The source MAC address is where the *DNS* query originated from. The destination MAC address is from the default gateway because this is the last stop before this query exits the local network.

Is the source MAC address the same as the one recorded from *Part 1* for the VM?

Yes

c. In the *Internet Protocol Version 4* line, the IP packet *Wireshark* capture indicates that the source IP address of this DNS query is 192.168.0.11 and the destination IP address is 8.8.8.8. In this example, the destination address is a *DNS* server.

Can you identify the IP and MAC addresses for the source device?

Device	IP Address	MAC Address
VM	192.168.0.1	00:50:56:f0:01:58

The *IP* packet and header encapsulates the *UDP* segment. The *UDP* segment contains the *DNS* query as the data.

d. Click the **arrow** next to **User Datagram Protocol** to view the details. A *UDP* header only has four fields: source port, destination port, length, and checksum. Each field in a *UDP* header is only 16 bits as depicted below.

UDP SEGMENT

0	16	31
UDP SOURCE PORT	UDP DESTINATION PORT	
UDP MESSAGE LENGTH	UDP CHECKSUM	
DATA		
DATA ...		

e. Notice that there are only four fields. The source port number in this example is 44188. The source port was randomly generated by the VM using port numbers that are not reserved. The destination port is 53. Port 53 is a well-known port reserved for use with *DNS*. *DNS* servers listen on port 53 for *DNS* queries from clients.

```
▸ Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▸ Ethernet II, Src: PcsCompu_23:b2:31 (08:00:27:23:b2:31), Dst: Vmware_f0:01:58 (00:50:56:f0:01:58)
▸ Internet Protocol Version 4, Src: 192.168.0.11, Dst: 8.8.8.8
▾ User Datagram Protocol, Src Port: 44188, Dst Port: 53
    Source Port: 44188
    Destination Port: 53
    Length: 40
    Checksum: 0xd0fc [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
▸ Domain Name System (query)

0010  00 3c 13 ab 40 00 40 11  56 43 c0 a8 00 0b 08 08   .<..@.@. VC......
0020  08 08 ac 9c 00 35 00 28  d0 fc 7d 19 01 00 00 01   .....5.( ..}.....
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   .......w ww.googl
0040  65 03 63 6f 6d 00 00 01  00 01                     e.com... ..
```

In this example, the length of the UDP segment is 40 bytes. The length of the UDP segment in your example may be different. Out of 40 bytes, 8 bytes are used as the header. The other 32 bytes are used by DNS query data. The 32 bytes of DNS query data is in the following illustration in the packet bytes pane (lower section) of the *Wireshark* main window.

```
▾ Domain Name System (query)
    [Response In: 5]
    Transaction ID: 0x7d19
  ▾ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ www.google.com: type A, class IN
        Name: www.google.com
        [Name Length: 14]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

The checksum is used to determine the integrity of the *UDP* header after it has traversed the Internet.

The *UDP* header has low overhead because *UDP* does not have fields that are associated with the three-way handshake in *TCP*. Any data transfer reliability issues that occur must be handled by the application layer.

Record your *Wireshark* results in the table below:

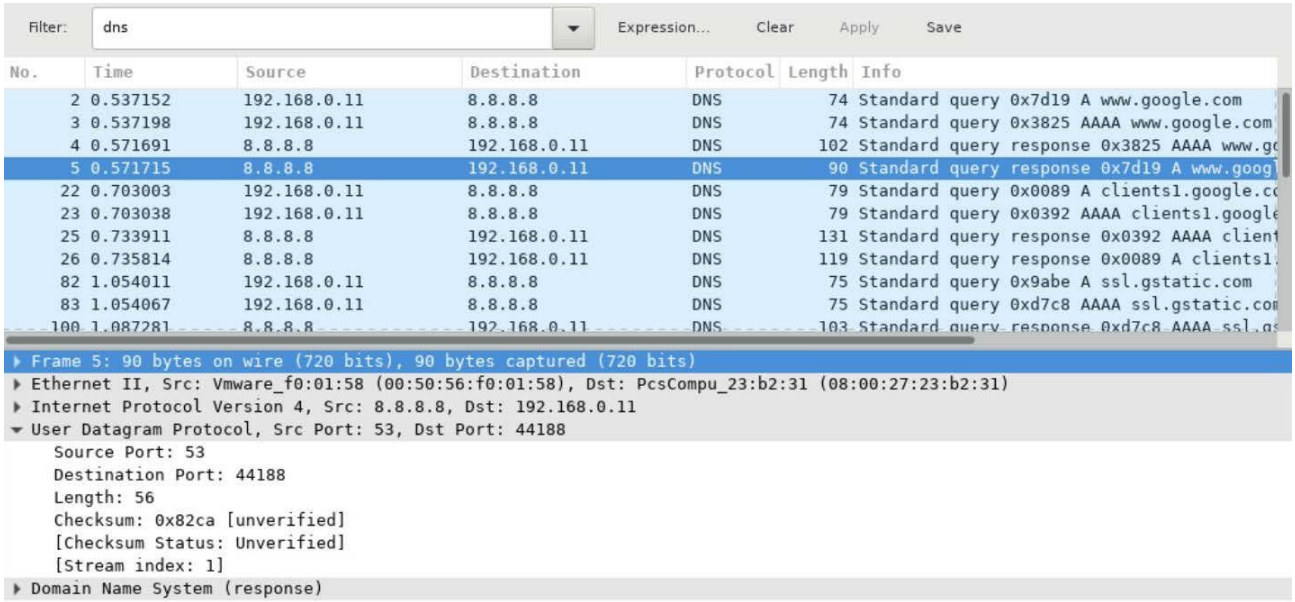
Frame size	64
Source MAC address	192.168.0.1
Destination MAC address	08:00:27:23:b2:31
Source IP address	192.168.0.11
Destination IP address	8.8.8.8
Source port	44188
Destination port	53

Is the source IP address the same as the local PC's IP address you recorded in *Part 1*? Yes

Step 3: Examine the fields in a DNS response packet.

In this step, you will examine the *DNS* response packet and verify that the *DNS* response packet also uses the *UDP*.

a. In this example, *Frame 5* is the corresponding *DNS* response packet. Notice the number of bytes on the wire is 102. It is a larger packet compared to the *DNS* query packet. This is because the *DNS* response packet will include a variety of information about the domain.



b. In the *Ethernet II* frame for the *DNS* response, what device is the destination MAC address?

Source MAC address is default gateway and destination MAC is VM.

c. Notice the source and destination IP addresses in the IP packet. What is the destination IP address? What is the source IP address?

Destination IP address: 192.168.0.11 Source IP address: 8.8.8.8

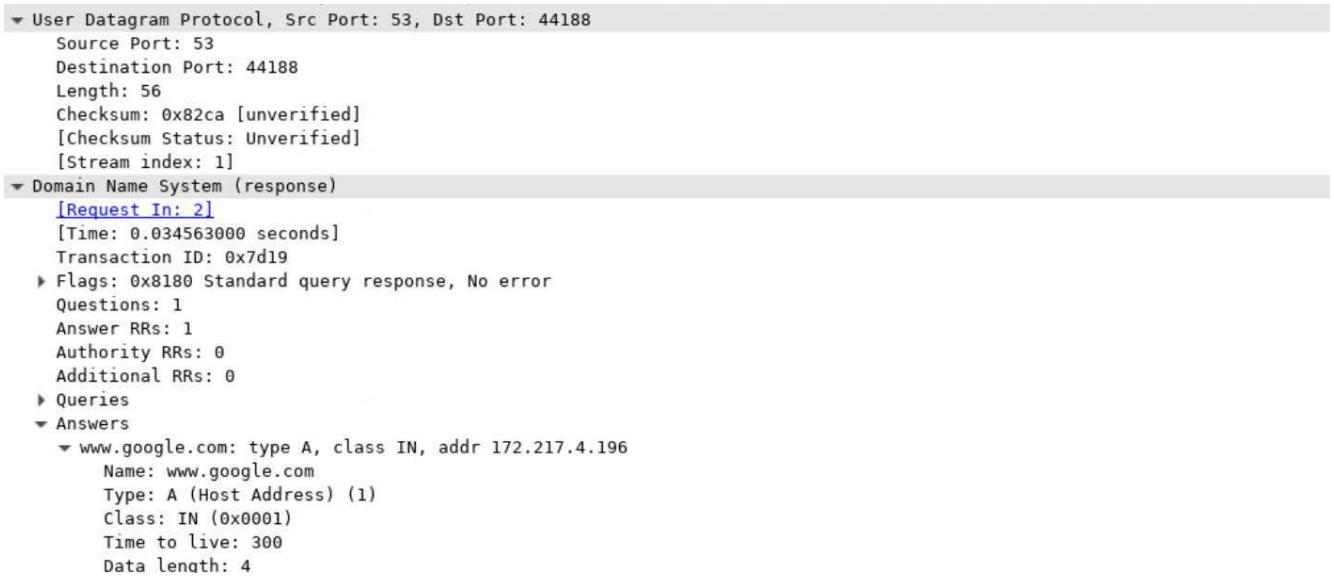
What happened to the roles of source and destination for the VM and DNS server?

VM and DNS server are reversed their roles in default.

d. In the UDP segment, the role of the port numbers has also reversed. The destination port number is 44188. Port number 44188 is the same port that was generated by the VM when the DNS query was sent to the DNS server. Your VM listens for a DNS response on this port.

The source port number is 53. The DNS server listens for a DNS query on port 53 and then sends a DNS response with a source port number of 53 back to the originator of the DNS query.

When the DNS response is expanded, notice the resolved IP addresses for www.google.com in the **Answers** section. Right-click on **Answers** and select **Expand Subtrees** to review.



Reflection

What are the benefits of using UDP instead of TCP as a transport protocol for DNS?

Domain Name System (DNS) is an important component of the Internet that convert domain names into IP addresses. When a DNS request is made, it is sent over a transport protocol such as User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). The advantages of using UDP instead of TCP as the transport protocol for DNS are: This allows you to easily scale your DNS server to handle large amounts of traffic. DNS queries are typically small, so the additional upward associated with TCP may not be necessary. Using UDP can improve DNS performance for small queries.