实验 - 使用 Wireshark 检查 UDP DNS 捕获

拓扑

目标

第 1 部分：记录 PC 的 IP 配置信息

第 2 部分：使用 Wireshark 捕获 DNS 查询和响应

第 3 部分：分析捕获的 DNS 或 UDP 数据包

背景/场景

如果您曾经使用过互联网，那么您已经使用了域名系统 (DNS)。DNS 是服务器的分布式网络，将人性化的域名（比如 www.google.com）转换为 IP 地址。当您将网站 URL 键入浏览器时，PC 将会对 DNS 服务器的 IP 地址执行 DNS 查询。PC 的 DNS 服务器查询和 DNS 服务器的响应使用用户数据报协议 (UDP) 作为传输层协议。UDP 是无连接协议，不要求像 TCP 那样建立会话。DNS 查询和响应开销非常小，而且不需要 TCP 开销。

在本实验中，您将通过使用 UDP 传输协议发送 DNS 查询与 DNS 服务器进行通信。您将使用 Wireshark 来检查同一台服务器的 DNS 查询和响应交换。

注意：本实验不能使用 Netlab 来完成。本实验假设您可以访问互联网。

所需资源

1 台 PC（采用 Windows 7 或 8 且可以访问命令提示符和互联网，并且已安装 Wireshark）

第 1 部分：记录 PC 的 IP 配置信息

在第 1 部分，您将在您的本地 PC 上使用 **ipconfig /all** 命令来查找并记录 PC 网络接口卡 (NIC) 的 MAC 和 IP 地址、指定默认网关的 IP 地址和为 PC 指定的 DNS 服务器 IP 地址。在所提供的表中记录此信息。在实验的某些部分，将使用该信息进行数据包分析。

IP 地址	
MAC 地址	
默认网关 IP 地址	
DNS 服务器 IP 地址	

第 2 部分：使用 Wireshark 捕获 DNS 查询和响应

在第 2 部分，您将设置 Wireshark 来捕获 DNS 查询和响应数据包，以便演示在与 DNS 服务器通信时 UDP 传输协议的使用。

a. 单击 Windows **开始**按钮并导航到 Wireshark 程序。

b. 为 Wireshark 选择一个接口用于捕获数据包。使用 **Interface List**（接口列表）选择与第 1 部分所记录的 PC 的 IP 和 MAC 地址相关联的接口。

c. 选择了所需接口后，单击 **Start**（开始）来捕获数据包。

d. 打开 Web 浏览器并键入 **www.google.com**。按 **Enter** 键继续。

e. 当您看到 Google 主页时，单击 **Stop**（停止）以停止 Wireshark 捕获。

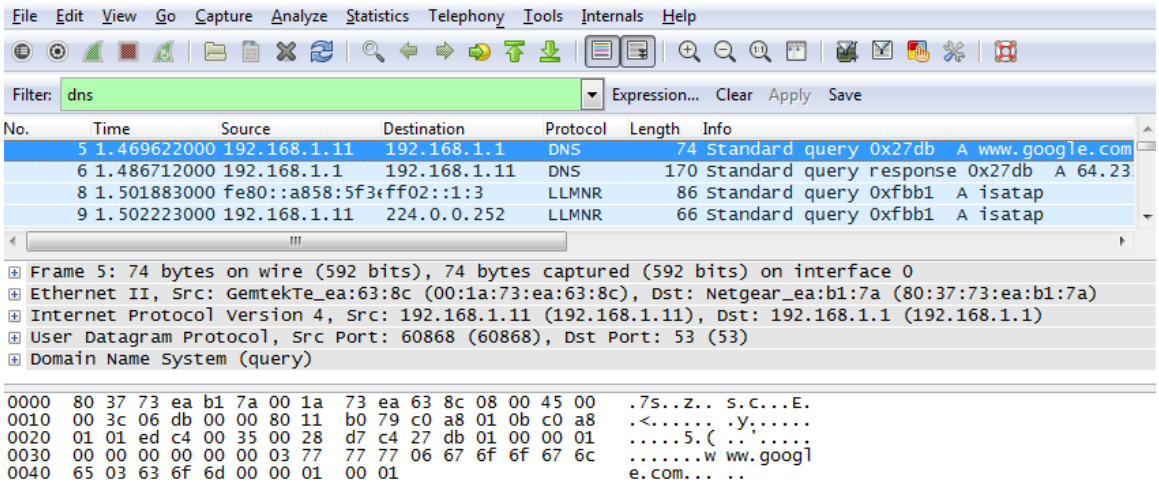
第 3 部分：分析捕获的 DNS 或 UDP 数据包

在第 3 部分，您将研究为了获取 www.google.com 的 IP 地址而与 DNS 服务器通信时生成的 UDP 数据包。

第 1 步：过滤 DNS 数据包。

a. 在 Wireshark 主窗口中，在 **Filter**（过滤器）工具栏的输入区键入 **dns**。单击 **Apply**（应用）或按 **Enter** 键。

注意：应用 DNS 过滤器后如未看到任何结果，请关闭 Web 浏览器。在命令提示符窗口中，键入 **ipconfig /flushdns** 删除之前所有的 DNS 结果。重新启动 Wireshark 捕获，并重复第 2b 到第 2e 部分的指令。如果这样不能解决问题，则您可以在命令提示符窗口输入 **nslookup www.google.com** 来代替 Web 浏览器。



b. 在主窗口的数据包列表窗格（顶部）中，找到包含“**standard query**”和“**A www.google.com**”的数据包。例如，参见帧 5。

第 2 步: 使用 DNS 查询检查 UDP 数据段。

使用通过 Wireshark 捕获的 www.google.com 的 DNS 查询检查 UDP。在本示例中，选择数据包列表窗格中 Wireshark 捕获帧 5 进行分析。此查询中的协议显示在主窗口的数据包详细信息窗格（中间）中。协议条目以灰色突出显示。

```
⊞ Frame 5: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
⊞ Ethernet II, Src: GemtekTe_ea:63:8c (00:1a:73:ea:63:8c), Dst: Netgear_ea:b1:7a (80:37:73:ea:b1:7a)
⊞ Internet Protocol Version 4, Src: 192.168.1.11 (192.168.1.11), Dst: 192.168.1.1 (192.168.1.1)
⊟ User Datagram Protocol, Src Port: 60868 (60868), Dst Port: 53 (53)
    Source Port: 60868 (60868)
    Destination Port: 53 (53)
    Length: 40
  ⊞ Checksum: 0xd7c4 [validation disabled]
    [Stream index: 2]
⊞ Domain Name System (query)
```

a. 在数据包详细信息窗格的第 1 行中，帧 5 在线路上有 74 个字节的数据。这是向指定服务器发送 DNS 查询以请求 www.google.com IP 地址的字节数。

b. “以太网 II”行显示源和目的 MAC 地址。源 MAC 地址来自您的本地 PC，因为是本地 PC 发起的 DNS 查询。目的 MAC 地址来自默认网关，因为这是此查询退出本地网络之前的最后一站。

源 MAC 地址是否与在第 1 部分为本地 PC 记录的相同？

c. 在“Internet Protocol Version 4”行中，IP 数据包 Wireshark 捕获表明此 DNS 查询的源 IP 地址为 192.168.1.11，目的 IP 地址为 192.168.1.1。在本示例中，目的地址是默认网关。路由器是该网络中的默认网关。

您能识别源设备和目的设备的 IP 和 MAC 地址吗？

设备	IP 地址	MAC 地址
本地 PC		
默认网关		

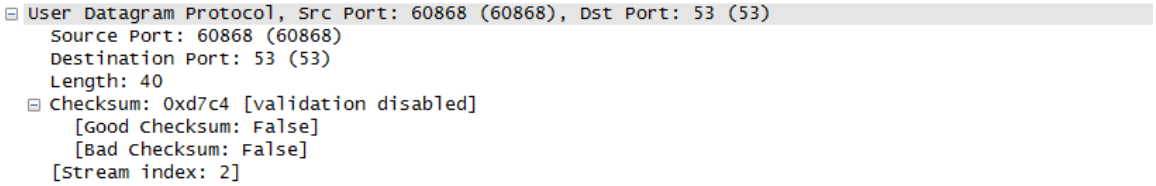
IP 数据包和报头封装 UDP 数据段。UDP 数据段将 DNS 查询作为数据包含在内。

d. UDP 报头只有四个字段：源端口、目的端口、长度和校验和。如下所示，UDP 报头的每个字段只有 16 位。

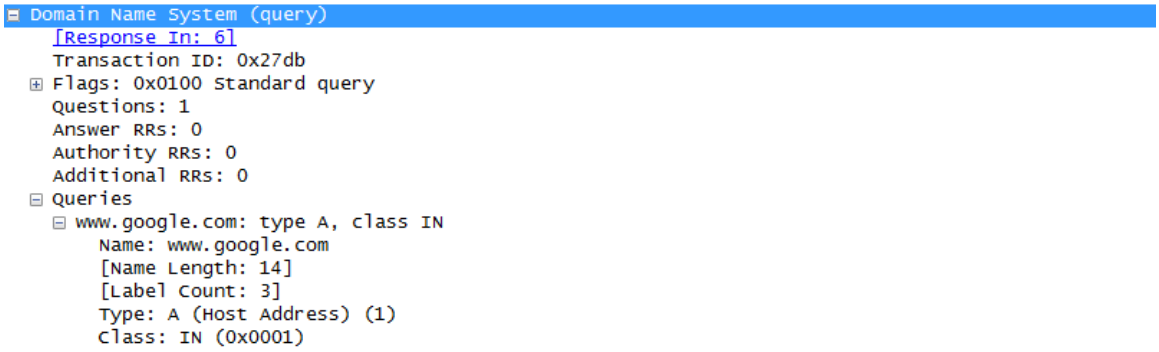
UDP 网段

0	16	31
UDP 源端口	UDP 目的端口	
UDP 报文长度	UDP 校验和	
数据		
数据. . .		

单击加号 (+) 展开数据包详细信息窗格中的用户数据报协议。注意只有四个字段。本示例中的源端口号为 60868。源端口是由本地 PC 使用不作为预留端口号的端口号随机生成的。目的端口是 53。端口 53 是公认端口，为 DNS 的使用保留。DNS 服务器在端口 53 上侦听来自客户端的 DNS 查询。



在本示例中，UDP 网段的长度为 40 个字节。在 40 个字节中，8 个字节用作报头。其他 32 个字节供 DNS 查询数据使用。在以下 Wireshark 主窗口的数据包字节窗格（底部）的图示中突出显示了 DNS 查询数据的 32 个字节。



校验和用于在数据包遍历互联网之后确定其完整性。

UDP 报头的开销很低，因为 UDP 没有与 TCP 三次握手相关的字段。出现的任何数据传输可靠性问题都必须由应用层处理。

在下表中记录您的 Wireshark 结果：

帧大小	
源 MAC 地址	
目的 MAC 地址	
源 IP 地址	
目的 IP 地址	
源端口	
目的端口	

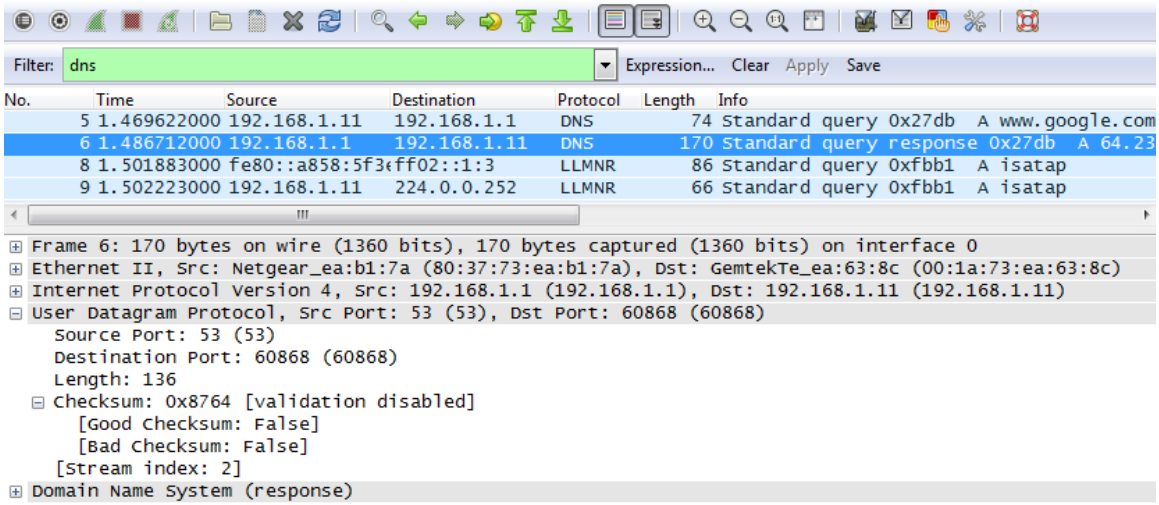
源 IP 地址是否与您在第 1 部分中记录的本地 PC 的 IP 地址相同？

目的 IP 地址是否与第 1 部分中记录的默认网关相同？

第 3 步： 使用 DNS 响应检查 UDP。

在此步骤中，您将检查 DNS 响应数据包并验证该 DNS 响应数据包是否也使用了 UDP。

a. 在本示例中，帧 6 是相应的 DNS 响应数据包。注意，线路上的字节数为 170 个字节。与 DNS 查询数据包相比，它是一个更大的数据包。



b. 在作为 DNS 响应的以太网 II 帧中，源 MAC 地址来自哪个设备？目的 MAC 地址来自哪个设备？

c. 注意 IP 数据包中的源 IP 地址和目的 IP 地址。目的 IP 地址是什么？ 源 IP 地址是什么？

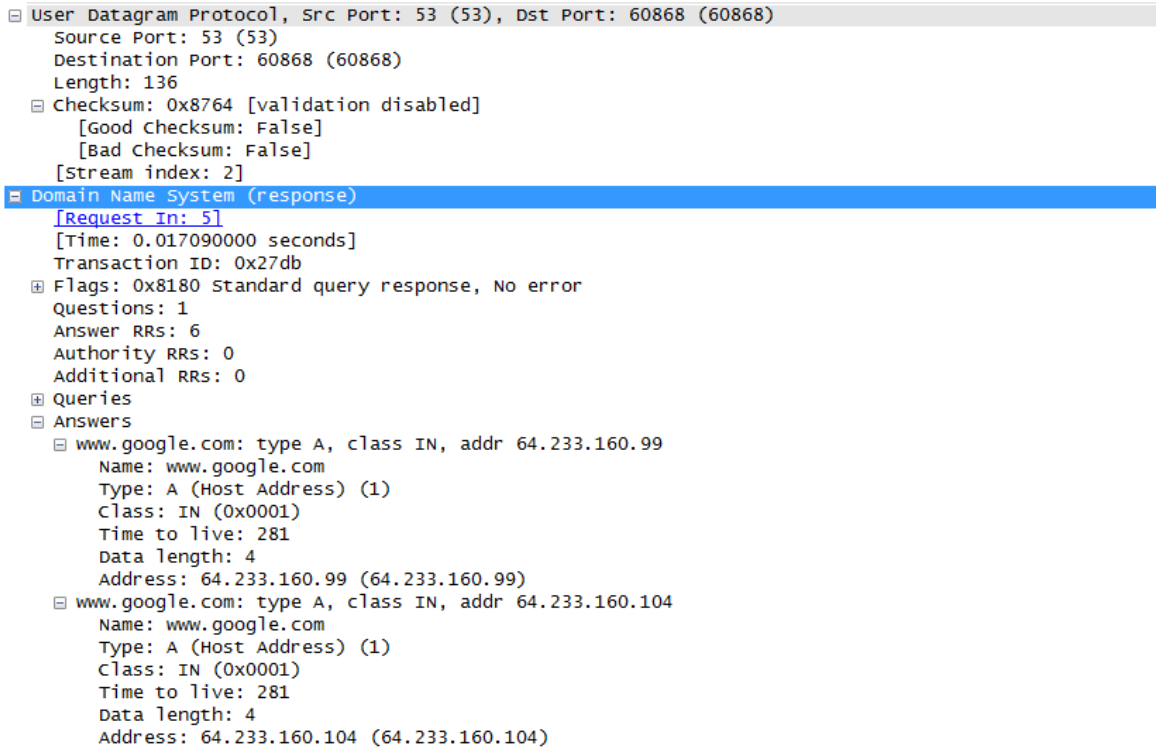
目的 IP 地址： 源 IP 地址：

本地主机和默认网关的源和目的的角色发生了什么变化？

d. 在 UDP 数据段中，端口号的角色也发生了互换。目的端口号为 60868。端口号 60868 与在将 DNS 查询发送到 DNS 服务器时本地 PC 所生成的端口相同。您的本地 PC 在此端口上侦听 DNS 响应。

源端口号为 53。DNS 服务器在端口 53 上侦听 DNS 查询，然后将带有源端口号 53 的 DNS 响应发送回 DNS 查询的发送方。

当 DNS 响应展开时，请注意 **Answers** 部分为 www.google.com 解析的 IP 地址。



思考

使用 UDP 而不是 TCP 作为 DNS 的传输协议的优势是什么？