

深圳大学实验报告

课程名称： 计算机网络及应用

实验项目名称： 数据包抓取与分析

学院： 计算机与软件学院

专业： 计算机科学与技术

指导教师： 李雪亮

报告人： 林宪亮 学号： 2022150130 班级： 国际班

实验时间： 2024 年 3 月 20 日

实验报告提交时间： 2024 年 3 月 28 日

教务处制

实验目的：

学习安装、使用协议分析软件，掌握基本的数据报捕获、过滤和协议的分析技巧，能对 IP 协议数据包进行分析。

实验环境：

使用 Windows 操作系统；Internet 连接
抓包软件 Wireshark。

实验内容：

协议分析软件的安装、使用、对 IP 协议数据包进行分析。

实验步骤：

（用文字描述实验过程，可用截图辅助说明）

1. 安装学习 Wireshark 软件

（1）Wireshark 软件的安装

Wireshark 的官网地址为 <http://www.wireshark.org/>。

该软件的使用手册为：http://www.wireshark.org/docs/wsug_html_chunked。

通过第一个网址访问软件官网，如图 1 所示。

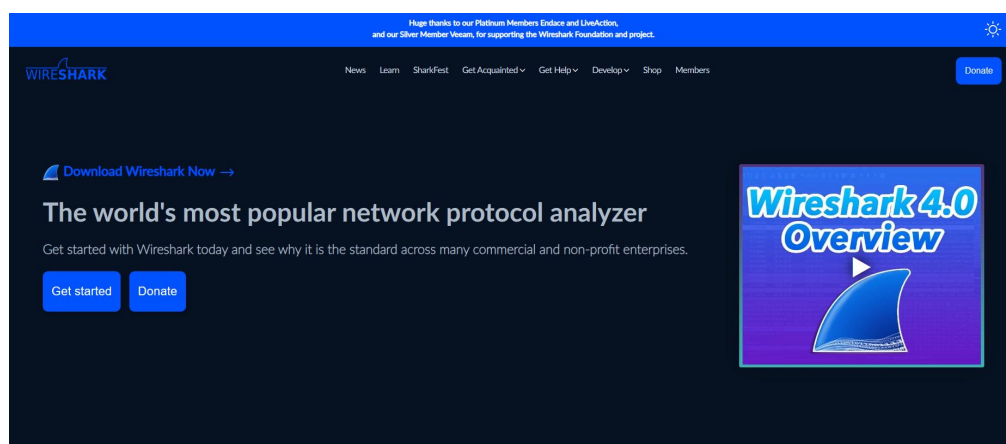


图 1 wireshark 软件官网

点击 “Get started” 之后下载对应的操作系统版本即可。

点击打开下载好的.exe 文件，选择安装地址，一直点击 “next” 即可。

(2) 运行 Wireshark，初始化界面如图 2 所示。

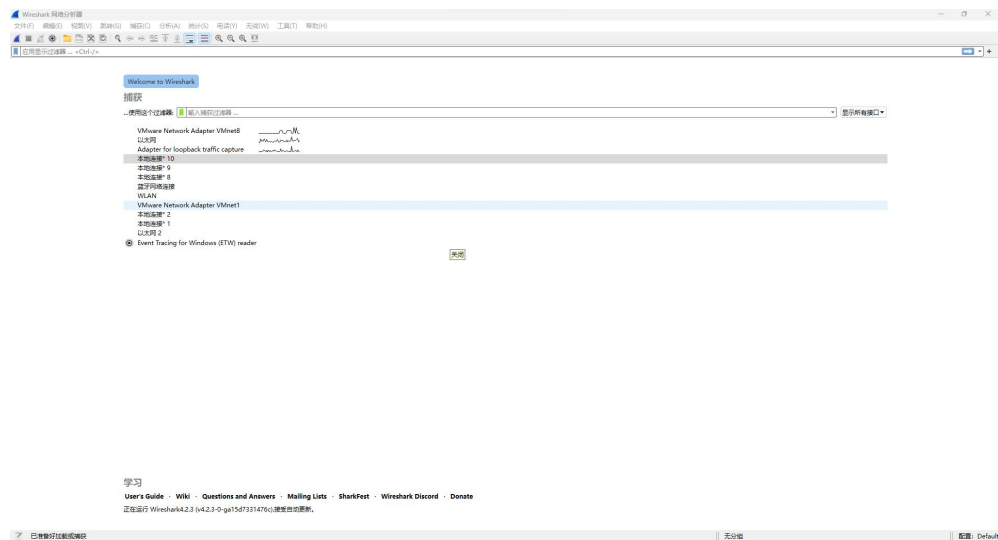


图 2 Wireshark 初始化界面

(3) 从接口列表选择要捕获的接口，双击即可开始捕获。

例如，双击 “以太网” 之后：

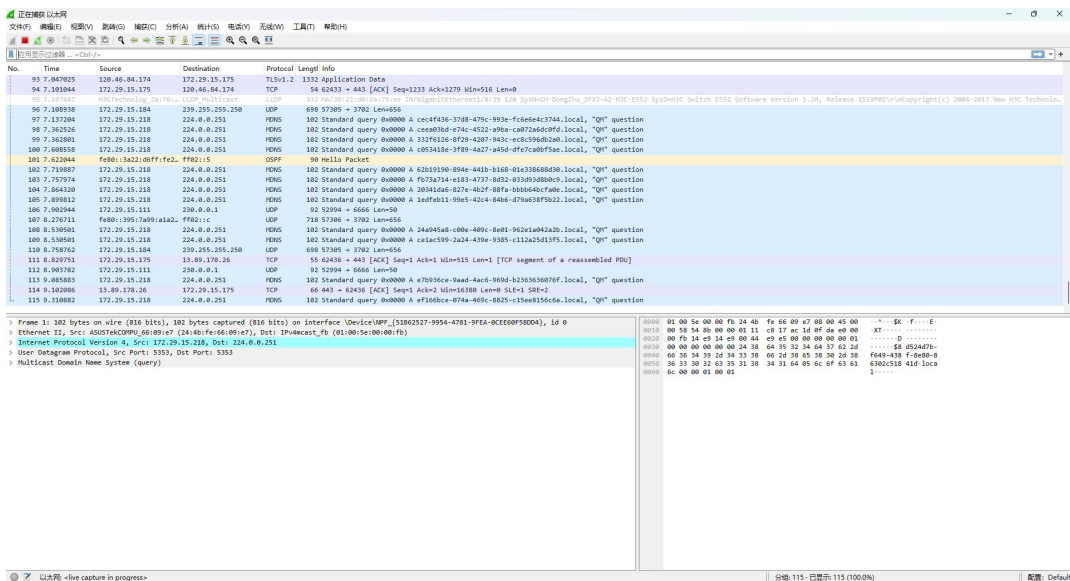


图 3 捕获界面

(4) WireShark 界面的主要组成

- Display Filter （显示过滤器）：用于过滤 如下图

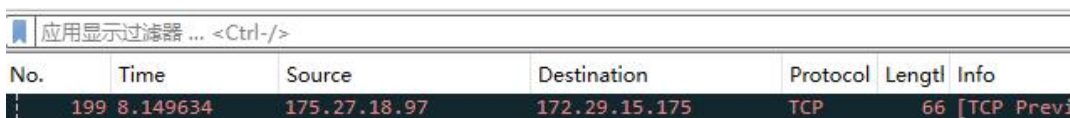


图 4 Display Filter

• **Packet List Pane (封包列表):** 显示捕获到的封包， 有源地址和目标地址，端口号。 颜色不同，代表不同的协议。如下图。

| No. | Time | Source | Destination | Protocol | Length | Info |
|------|------------|-----------------|-----------------|----------|--------|---|
| 3054 | 141.119667 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A 2c54594f-77e9-4e9e-8232-20ce583867ee.local, "QM" question |
| 3055 | 141.444410 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A 34f3866e-dd90-4e97-8fd7-74672680552c.local, "QM" question |
| 3056 | 141.696471 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A fa0383c4-25eb-428d-b919-43f825391b5b.local, "QM" question |
| 3057 | 141.696931 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A f993eefc-3fd7-45d2-86b4-d5a4c14e4684.local, "QM" question |
| 3058 | 141.717576 | 172.29.15.206 | 224.0.0.251 | MDNS | 82 | Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question |
| 3059 | 141.878943 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A bca7cae9-8cb8-4c18-ac97-9327cda939cb.local, "QM" question |
| 3060 | 141.882460 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A 5476e00b-3f92-4a81-80af-acab3b34391e.local, "QM" question |
| 3061 | 141.891144 | 172.29.15.175 | 14.17.27.114 | UDP | 69 | 1863 → 8017 Len=27 |
| 3062 | 141.891280 | 172.29.15.175 | 120.231.141.227 | UDP | 126 | 1863 → 9424 Len=84 |
| 3063 | 141.891309 | 172.29.15.175 | 120.235.155.163 | UDP | 126 | 1863 → 9778 Len=84 |
| 3064 | 141.894391 | 183.233.75.29 | 172.29.15.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 3065 | 141.904411 | 120.231.141.227 | 172.29.15.175 | UDP | 202 | 9424 → 1863 Len=160 |
| 3066 | 141.905002 | 120.235.155.163 | 172.29.15.175 | UDP | 202 | 9778 → 1863 Len=160 |
| 3067 | 141.927288 | 172.29.15.111 | 230.0.0.1 | UDP | 92 | 52994 → 6666 Len=50 |
| 3068 | 141.937597 | 172.29.15.175 | 175.27.18.97 | TCP | 55 | [TCP Keep-Alive] 65235 → 443 [ACK] Seq=0 Ack=1760 Win=516 Len=1 |
| 3069 | 141.956714 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | [TCP Keep-Alive ACK] 443 → 65235 [ACK] Seq=1760 Ack=1 Win=310 Len=0 SLE=0 SRE=1 |
| 3070 | 142.180752 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A ad8f28bc-7cca-479a-b447-0050fbf41d9c.local, "QM" question |
| 3071 | 142.214409 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A 21fd2f00-20ac-4566-b9c7-8d478294ca23.local, "QM" question |
| 3072 | 142.420947 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A 54b0d7e5-7b2c-4b5a-9ddf-a2559c2b40f9.local, "QM" question |
| 3073 | 142.450873 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A d4b6ce5b-b394-4aba-b53a-88660408a092.local, "QM" question |
| 3074 | 142.450991 | 172.29.15.218 | 224.0.0.251 | MDNS | 102 | Standard query 0x0000 A b984b972-2974-45ec-a023-46fa71a06679.local, "QM" question |
| 3075 | 142.495782 | 172.29.15.175 | 144.202.115.34 | TCP | 66 | 65308 → 8237 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 3076 | 142.730214 | 172.29.15.175 | 223.5.5.5 | TCP | 55 | [TCP Keep-Alive] 65289 → 443 [ACK] Seq=793 Ack=3607 Win=130304 Len=1 |

图 5 Packet List Pane

• **Packet Details Pane(封包详细信息):** 显示封包中的字段。如下图。

| |
|---|
| > Frame 3068: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF_{51862527-9954-4781-9FEA-0CEE60F58D04}, id 0 |
| > Ethernet II, Src: ASUSTekCOMPU_cd:35:cf (c8:7f:54:cd:35:cf), Dst: H3Technolog_2b:9a:ff (38:22:d6:2b:9a:ff) |
| > Internet Protocol Version 4, Src: 172.29.15.175, Dst: 175.27.18.97 |
| > Transmission Control Protocol, Src Port: 65235, Dst Port: 443, Seq: 0, Ack: 1760, Len: 1 |
| > Data (1 byte) |

图 6 Packet Details Pane

• **Dissector Pane(16 进制数据)**

| | | | |
|------|-------------------------|-------------------------|-----------------------------|
| 0000 | 38 22 d6 2b 9a ff c8 7f | 54 cd 35 cf 08 00 45 00 | 8" + . . . T 5 . . . E . |
| 0010 | 00 29 0f ff 40 00 80 06 | 00 00 ac 1d 0f af af 1b | .) . @ |
| 0020 | 12 61 fe d3 01 bb a5 53 | b4 4c b4 19 04 ab 50 10 | . a S . L . . . P . |
| 0030 | 02 04 7d 64 00 00 00 | | . . } d . . . |

图 7 Dissector Pane

• **Miscellaneous(地址栏，杂项)**

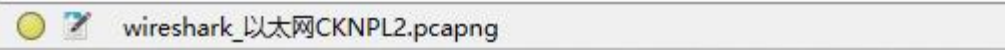


图 8 Miscellaneous

(5) 过滤器

过滤器会帮助我们在大量的数据中迅速找到我们需要的信息。

过滤器有两种，

一种是显示过滤器，就是主界面上那个，用来在捕获的记录中找到所需要的记录。

一种是捕获过滤器，用来过滤捕获的封包，以免捕获太多的记录。在 Capture -> Capture Filters 中设置。如下图所示。

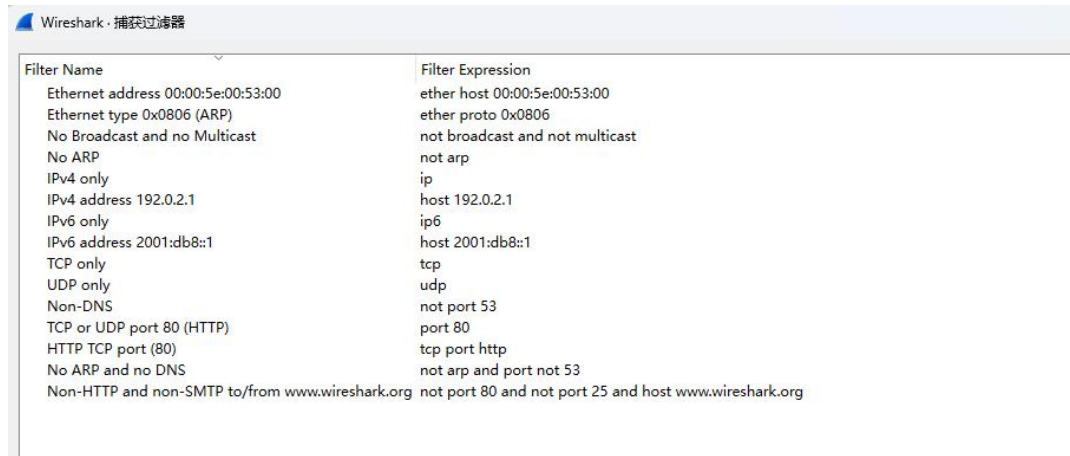


图 9 捕获过滤器

(6) 过滤表达式

- 协议过滤：比如 TCP，只显示 TCP 协议。

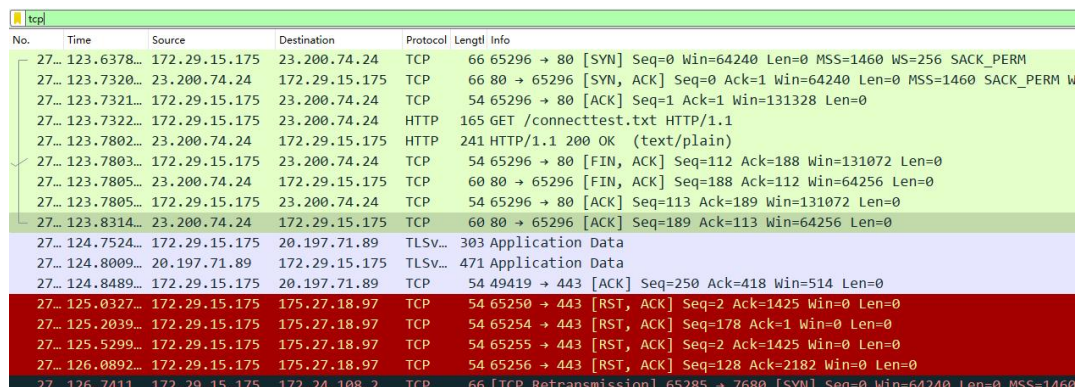


图 10 协议过滤

如图 10，使用了 TCP 进行过滤后，就只会显示与 TCP 协议相关的封包。

- IP 过滤：比如 ip.src==192.168.1.102 显示源地址为 192.168.1.102，ip.dst==192.168.1.102, 目标地址为 192.168.1.102。



图 11 IP 过滤

如图 11 所示，当我使用 ip.src==192.168.1.102 过滤后，所有的封包信息都不再显示了，

因为我的源地址不是 192.168.1.102。

- 端口过滤: tcp.srcport== 80, 只显示 TCP 协议的源端口为 80 的。

| tcp.srcport== 80 | | | | | | |
|------------------|-----------|----------------|---------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 67 | 2.897056 | 184.26.43.82 | 172.29.15.175 | TCP | 66 | 80 → 65258 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 |
| 70 | 2.959848 | 184.26.43.82 | 172.29.15.175 | TCP | 60 | 80 → 65258 [ACK] Seq=1 Ack=112 Win=64256 Len=0 |
| 71 | 2.960548 | 184.26.43.82 | 172.29.15.175 | HTTP | 241 | HTTP/1.1 200 OK (text/plain) |
| 73 | 2.961216 | 184.26.43.82 | 172.29.15.175 | TCP | 60 | 80 → 65258 [FIN, ACK] Seq=188 Ack=112 Win=64256 L |
| 77 | 3.023083 | 184.26.43.82 | 172.29.15.175 | TCP | 60 | 80 → 65258 [ACK] Seq=189 Ack=113 Win=64256 Len=0 |
| 148 | 6.902764 | 120.234.70.147 | 172.29.15.175 | TCP | 60 | 80 → 65237 [FIN, ACK] Seq=1 Ack=1 Win=4087 Len=0 |
| 158 | 7.201734 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | 80 → 65260 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 |
| 199 | 8.149634 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | [TCP Previous segment not captured] 80 → 65260 [A |
| 361 | 15.099650 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | 80 → 65263 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 |
| 367 | 15.355116 | 175.27.18.97 | 172.29.15.175 | HTTP | 496 | HTTP/1.1 200 OK (text/html) |
| 371 | 15.409901 | 175.27.18.97 | 172.29.15.175 | TCP | 60 | 80 → 65263 [FIN, ACK] Seq=443 Ack=1406 Win=68608 |
| 377 | 16.092818 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | 80 → 65259 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 |
| 459 | 19.315680 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | 80 → 65265 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 |
| 465 | 19.334679 | 175.27.18.97 | 172.29.15.175 | TCP | 60 | 80 → 65265 [ACK] Seq=1 Ack=2849 Win=71424 Len=0 |
| 474 | 19.582662 | 175.27.18.97 | 172.29.15.175 | HTTP | 325 | HTTP/1.1 200 OK (text/html) |
| 476 | 19.642773 | 175.27.18.97 | 172.29.15.175 | TCP | 60 | 80 → 65265 [FIN, ACK] Seq=272 Ack=3304 Win=74240 |
| 605 | 26.324534 | 175.27.18.97 | 172.29.15.175 | TCP | 66 | [TCP Previous segment not captured] 80 → 65260 [A |
| 731 | 33.003751 | 184.26.43.73 | 172.29.15.175 | TCP | 66 | 80 → 65268 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 |

图 12 端口过滤

如图, 使用 tcp.srcport== 80 过滤后, 只会显示 TCP 协议的源端口为 80 的封包。

- HTTP 模式过滤: http.request.method=="GET", 只显示 HTTP GET 方法的。

| http.request.method=="GET" | | | | | | |
|----------------------------|-------------|---------------|--------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 69 | 2.897309 | 172.29.15.175 | 184.26.43.82 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |
| 363 | 15.116989 | 172.29.15.175 | 175.27.18.97 | HTTP | 14... | GET /qqmusic/cgi-bin/qm_rpstopmus.fcgi?vers |
| 733 | 33.093977 | 172.29.15.175 | 184.26.43.73 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |
| 12... | 63.292174 | 172.29.15.175 | 23.200.74.24 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |
| 19... | 93.497248 | 172.29.15.175 | 184.26.43.82 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |
| 27... | 123.7322... | 172.29.15.175 | 23.200.74.24 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |

图 13 HTTP 模式过滤

- 逻辑运算符为 AND/ OR, 使用逻辑运算符可以组合不同的过滤表达式, 达成多样的过滤效果。

| http.request.method=="GET" and ip.dst==184.26.43.82 | | | | | | |
|---|-----------|---------------|--------------|----------|--------|-------------------------------|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 69 | 2.897309 | 172.29.15.175 | 184.26.43.82 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |
| 19... | 93.497248 | 172.29.15.175 | 184.26.43.82 | HTTP | 165 | GET /connecttest.txt HTTP/1.1 |

图 14 逻辑运算符过滤

如图, 我使用了 http.request.method=="GET" and ip.dst==184.26.43.82 表达式进行过滤, 这用到了 “and” 逻辑运算符。

(7) 封包详细信息 (Packet Details Pane)

```
> Frame 69: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits) on interface \Device\NPF_{51B62527-9954-4...}
> Ethernet II, Src: ASUSTekCOMPU_cd:35:cf (c8:7f:54:cd:35:cf), Dst: H3CTechnolog_2b:9a:ff (38:22:d6:2b:9a:ff)
> Internet Protocol Version 4, Src: 172.29.15.175, Dst: 184.26.43.82
> Transmission Control Protocol, Src Port: 65258, Dst Port: 80, Seq: 1, Ack: 1, Len: 111
> Hypertext Transfer Protocol
```

图 15 Packet Details Pane

如图 15, 这个面板是最重要的, 用来查看协议中的每一个字段。

Frame: 物理层的数据帧概况

Ethernet II: 数据链路层以太网帧头部信息

Internet Protocol Version 4: 互联网层 IP 包头部信息

Transmission Control Protocol: 传输层 T 的数据段头部信息, 此处是 TCP

Hypertext Transfer Protocol: 应用层的信息, 此处是 HTTP 协议

(8) TCP 包的具体内容

在 Transmission Control Protocol 中可以看到 wireshark 捕获到的 TCP 包中的每个字段。如下图, 可以查看端口号, 序号, 确认号等等信息。

```
Transmission Control Protocol, Src Port: 65258, Dst Port: 80, Seq: 1, Len: 111
  Source Port: 65258
  Destination Port: 80
  [Stream index: 10]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 111]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 62886980
  [Next Sequence Number: 112 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 155799967
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window: 513
  [Calculated window size: 131328]
  [Window size scaling factor: 256]
  Checksum: 0x9fc2 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (111 bytes)
```

图 16 TCP 包的具体内容

2. 示例分析 TCP 三次握手的过程

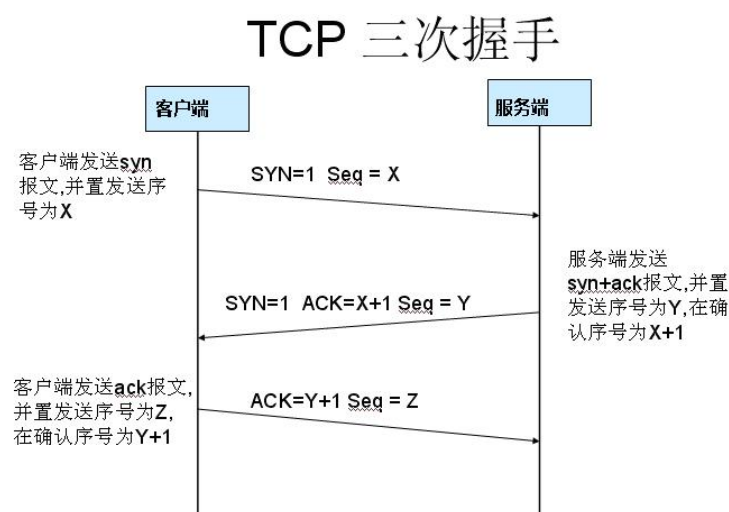


图 17 TCP 三次握手

- (1) 打开 wireshark, 开启抓包, 使用浏览器访问百度, 触发 TCP 三次握手。
- (2) 打开 cmd, 使用 “ping -4 www.baidu.com ” 查询百度的 IP 地址。
- (3) 在 wireshark 的过滤器中使用 “tcp and ip.addr==120.232.145.144”, 过滤得到期望的数据包。

| No. | Time | Source | Destination | Protocol | Length | Info |
|------|-----------|------------------|------------------|----------|--------|---|
| 25.. | 17.144078 | 120.232.145.1... | 172.29.15.175 | TCP | 60 | [TCP Previous segment not captured] 443 → 57712 [ACK] Seq=2 Ack=1991 Win=2712 Len=0 |
| 29.. | 17.227195 | 120.232.145.1... | 172.29.15.175 | TLSv... | 199 | Application Data |
| 32.. | 17.271705 | 172.29.15.175 | 120.232.145.1... | TCP | 54 | 57712 → 443 [ACK] Seq=1991 Ack=147 Win=514 Len=0 |
| 78.. | 18.352638 | 172.29.15.175 | 120.232.145.1... | TCP | 54 | 57712 → 443 [FIN, ACK] Seq=1991 Ack=147 Win=514 Len=0 |
| 78.. | 18.364368 | 120.232.145.1... | 172.29.15.175 | TCP | 60 | 443 → 57712 [ACK] Seq=147 Ack=1992 Win=2712 Len=0 |
| 78.. | 18.364368 | 120.232.145.1... | 172.29.15.175 | TLSv... | 85 | Encrypted Alert |
| 78.. | 18.364368 | 120.232.145.1... | 172.29.15.175 | TCP | 60 | 443 → 57712 [FIN, ACK] Seq=178 Ack=1992 Win=2712 Len=0 |
| 78.. | 18.364469 | 172.29.15.175 | 120.232.145.1... | TCP | 54 | 57712 → 443 [RST, ACK] Seq=1992 Ack=178 Win=0 Len=0 |
| 19.. | 32.435643 | 172.29.15.175 | 120.232.145.1... | TCP | 66 | 54909 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 19.. | 32.444945 | 120.232.145.1... | 172.29.15.175 | TCP | 66 | 443 → 54909 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM |
| 19.. | 32.444998 | 172.29.15.175 | 120.232.145.1... | TCP | 54 | 54909 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0 |
| 19.. | 32.445173 | 172.29.15.175 | 120.232.145.1... | TLSv... | 715 | Client Hello (SNI=sp0.baidu.com) |
| 19.. | 32.457085 | 120.232.145.1... | 172.29.15.175 | TCP | 60 | 443 → 54909 [ACK] Seq=1 Ack=662 Win=79744 Len=0 |

图 18 过滤后的数据包

- (4) 开始分析三个握手的包

| | | | | | | |
|------|-----------|------------------|------------------|-----|----|--|
| 19.. | 32.435643 | 172.29.15.175 | 120.232.145.1... | TCP | 66 | 54909 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 19.. | 32.444945 | 120.232.145.1... | 172.29.15.175 | TCP | 66 | 443 → 54909 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM |
| 19.. | 32.444998 | 172.29.15.175 | 120.232.145.1... | TCP | 54 | 54909 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0 |

图 19 三个握手的包

第一个数据包中, 源地址是本机地址, 目的地址是百度的地址, 在 INFO 中可以看到有一个 SYN。也就是我的电脑向百度服务器发送了一个 TCP 请求, SYN 表示这是第一个建立连接的请求, 这是第一次握手。

第二个数据包中, 源地址是百度的地址, 目的地址是我的本机地址, 在 INFO 字段中可以看到有一个 SYN 和 ACK。也就是代表百度的服务器接受了我的请求后, 向我的电脑发送了一个 TCP 请求, SYN+ACK 代表一个应答的请求, 这是第二次握手。

第三个数据包中, 源地址是我的电脑, 目的地址是百度的服务器, INFO 字段中有一个 ACK。也就是代表我的电脑在接收到百度服务器的相应请求后, 向百度的服务器发送了一个 TCP 请求, ACK 表示这是一个确认请求, 这是第三次握手。

发送完这个确认请求后, 三次握手就完成了, 就可以开始传输数据了。

3. 分析 DNS、HTTP、TCP、UDP 数据包

以访问深圳大学主页为例。

(1) UDP

UDP 是 User Datagram Protocol（用户数据协议）的简称，是一种无连接的协议，该协议工作在 OSI 模型中的第四层（传输层），处于 IP 协议的上一层。传输层的功能就是建立“端口到端口”的通信，UDP 提供面向事务的简单的不可靠信息传送服务。

UDP 协议是一种无连接的协议，该协议用来支撑那些需要在计算机之间传输数据的网络应用，包括网络视频会议系统在内的众多客户/服务器模式的网络应用。

UDP 协议的主要工作就是将网络数据流量压缩成数据包的形式。一个经典的数据包就是一个二进制数据的传输单位。每一个数据包的前 8 字节用来包含包头信息，剩余字节则用来包含具体的传输数据。

- 使用 wireshark 过滤器得到 UDP 报文。

| No. | udp udpcap udpencap udplite | Source | Destination | Protocol | Length | Info |
|--------|--------------------------------------|------------------|------------------|----------|--------|---------------------------|
| 960... | | 172.29.15.221 | 224.0.0.251 | MDNS | 169 | Standard query response |
| 960... | | fe80::cf4f:29... | ff02::fb | MDNS | 189 | Standard query response |
| 62... | 189.3807... | 172.29.15.29 | 239.255.255.2... | SSDP | 217 | M-SEARCH * HTTP/1.1 |
| 62... | 190.3853... | 172.29.15.29 | 239.255.255.2... | SSDP | 217 | M-SEARCH * HTTP/1.1 |
| 62... | 191.2671... | 172.29.15.6 | 14.22.9.195 | UDP | 69 | 1863 → 8011 Len=27 |
| 62... | 191.2697... | 183.233.75.33 | 172.29.15.6 | ICMP | 70 | Time-to-live exceeded (0) |
| 62... | 191.3947... | 172.29.15.29 | 239.255.255.2... | SSDP | 217 | M-SEARCH * HTTP/1.1 |
| 62... | 192.4005... | 172.29.15.29 | 239.255.255.2... | SSDP | 217 | M-SEARCH * HTTP/1.1 |
| 62... | 194.5326... | 172.29.15.6 | 14.22.9.195 | UDP | 69 | 1863 → 8011 Len=27 |
| 62... | 194.5369... | 183.233.75.33 | 172.29.15.6 | ICMP | 70 | Time-to-live exceeded (0) |
| 62... | 197.6893... | 172.29.15.6 | 112.53.47.99 | UDP | 260 | 1863 → 18888 Len=218 |
| 62... | 197.6952... | 112.53.47.99 | 172.29.15.6 | UDP | 332 | 18888 → 1863 Len=290 |
| 62... | 197.7973... | 172.29.15.6 | 14.22.9.195 | UDP | 69 | 1863 → 8011 Len=27 |
| 62... | 197.8008... | 183.233.75.33 | 172.29.15.6 | ICMP | 70 | Time-to-live exceeded (0) |
| 65... | 199.9340... | 172.29.15.6 | 112.53.47.99 | UDP | 337 | 1863 → 30013 Len=295 |

图 20 多条数据包

- 选择其中的一个 UDP 数据包进行分析。

```
▼ User Datagram Protocol, Src Port: 1863, Dst Port: 8011
  Source Port: 1863
  Destination Port: 8011
  Length: 35
  Checksum: 0xd330 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
  > [Timestamps]
    UDP payload (27 bytes)
```

图 21 UDP 数据包

如图 21，我们可以看到 UDP 数据包的源端口号（1863），目的端口号（8011），UDP 数据包的长度（35 字节），可以看看该数据包的检验和，用以大致确认数据传输的正确性。

(2) TCP

TCP (Transmission Control Protocol) 是一种面向连接的传输协议，它提供可靠的数据传输服务。TCP 通过序列号、确认和重传机制来确保数据的可靠传输，同时支持流量控制和拥塞控制。TCP 适用于对数据完整性和顺序性要求高的应用，如网页浏览、文件传输等。

- 过滤得到 TCP 协议数据包。

| tcp and ip.addr ==210.39.4.1 | | | | | | |
|------------------------------|-------------|-------------|-------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 84... | 1175.916... | 210.39.4.1 | 172.29.15.6 | TLSv... | 15... | Continuation Data |
| 84... | 1175.916... | 172.29.15.6 | 210.39.4.1 | TCP | 54 | 54173 → 443 [ACK] Seq=20522 Ack=470065876 |
| 84... | 1175.916... | 210.39.4.1 | 172.29.15.6 | TLSv... | 15... | Continuation Data |
| 84... | 1175.916... | 210.39.4.1 | 172.29.15.6 | TLSv... | 15... | Continuation Data |
| 84... | 1175.916... | 172.29.15.6 | 210.39.4.1 | TCP | 54 | 54173 → 443 [ACK] Seq=20522 Ack=470068788 |
| 84... | 1175.916... | 210.39.4.1 | 172.29.15.6 | TLSv... | 15... | Continuation Data |
| 84... | 1175.916... | 210.39.4.1 | 172.29.15.6 | TLSv... | 15... | Continuation Data |
| 84... | 1175.916... | 210.39.4.1 | 172.29.15.6 | TLSv... | 338 | Continuation Data |
| 84... | 1175.916... | 172.29.15.6 | 210.39.4.1 | TCP | 54 | 54173 → 443 [ACK] Seq=20522 Ack=470071984 |
| 84... | 1177.684... | 172.29.15.6 | 210.39.4.1 | TLSv... | 89 | Application Data |
| 84... | 1177.686... | 210.39.4.1 | 172.29.15.6 | TCP | 15... | 443 → 54173 [ACK] Seq=470071984 Ack=20557 |
| 84... | 1177.686... | 210.39.4.1 | 172.29.15.6 | TCP | 15... | 443 → 54173 [ACK] Seq=470073440 Ack=20557 |
| 84... | 1177.686... | 172.29.15.6 | 210.39.4.1 | TCP | 54 | 54173 → 443 [ACK] Seq=20557 Ack=470074896 |
| 84... | 1177.686... | 210.39.4.1 | 172.29.15.6 | TLSv... | 12... | Application Data |
| 84... | 1177.686... | 210.39.4.1 | 172.29.15.6 | TCP | 15... | 443 → 54173 [ACK] Seq=470076111 Ack=20557 |
| 84... | 1177.686... | 172.29.15.6 | 210.39.4.1 | TCP | 54 | 54173 → 443 [ACK] Seq=20557 Ack=470077567 |

图 22 过滤得到 TCP 协议数据包

- 选择其中一个数据包进行分析。

| |
|--|
| Transmission Control Protocol, Src Port: 443, Dst Port: 54331, Seq: 200785164, Ack: 5329, Len: 284 |
| Source Port: 443 |
| Destination Port: 54331 |
| [Stream index: 2059] |
| > [Conversation completeness: Incomplete, DATA (15)] |
| [TCP Segment Len: 284] |
| Sequence Number: 200785164 (relative sequence number) |
| Sequence Number (raw): 1736189111 |
| [Next Sequence Number: 200785448 (relative sequence number)] |
| Acknowledgment Number: 5329 (relative ack number) |
| Acknowledgment number (raw): 3618603130 |
| 0101 = Header Length: 20 bytes (5) |
| > Flags: 0x018 (PSH, ACK) |
| Window: 79 |
| [Calculated window size: 40448] |
| [Window size scaling factor: 512] |
| Checksum: 0xd879 [unverified] |
| [Checksum Status: Unverified] |
| Urgent Pointer: 0 |
| > [Timestamps] |
| > [SEQ/ACK analysis] |
| TCP payload (284 bytes) |

图 23 TCP 数据包具体信息

同样的，可以从中得到一些具体的信息，如源端口号为 443，目标端口号为 54331，发送序列号为 200785164，确认序列号，flag 同步序列号，窗口大小，校验和等信息。

(3) HTTP

HTTP (Hypertext Transfer Protocol) 是一种用于在 Web 浏览器和 Web 服务器之间传输信息的协议。它是一种无状态的协议,即每个 HTTP 请求都独立于之前的请求,服务器不会保存客户端的任何状态信息。HTTP 通常使用 TCP 作为传输协议,它基于请求-响应模型,客户端发送请求并等待服务器响应。

- 过滤抓包后可以得到两个数据包,分别是 http 请求和 http 响应,如下图。

| http and ip.addr ==210.39.4.1 | | | | | | |
|-------------------------------|-------------|-------------|-------------|----------|--------|--|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 33... | 1006.002... | 172.29.15.6 | 210.39.4.1 | HTTP | 536 | GET /board/ HTTP/1.1 |
| 33... | 1006.003... | 210.39.4.1 | 172.29.15.6 | HTTP | 387 | HTTP/1.1 302 Moved Temporarily (text/html) |

图 24 HTTP 数据包

```
> Frame 337625: 536 bytes on wire (4288 bits), 536 bytes captured (4288 bits) on interface \Device\NPF_{51B62527-9
> Ethernet II, Src: ASUSTekCOMPU_cd:35:cf (c8:7f:54:cd:35:cf), Dst: H3CTechnolog_2b:9a:ff (38:22:d6:2b:9a:ff)
> Internet Protocol Version 4, Src: 172.29.15.6, Dst: 210.39.4.1
> Transmission Control Protocol, Src Port: 54166, Dst Port: 80, Seq: 1, Ack: 1, Len: 482
< Hypertext Transfer Protocol
  < GET /board/ HTTP/1.1\r\n
    Host: www.szu.edu.cn\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appli
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
    \r\n
    [Full request URI: http://www.szu.edu.cn/board/]
    [HTTP request 1/1]
    [Response in frame: 337627]
```

图 25 HTTP 请求报文

```
< Hypertext Transfer Protocol
  < HTTP/1.1 302 Moved Temporarily\r\n
    Server: none\r\n
    Date: Thu, 28 Mar 2024 02:40:46 GMT\r\n
    Content-Type: text/html\r\n
  < Content-Length: 137\r\n
    Connection: keep-alive\r\n
    Location: https://www.szu.edu.cn/board/\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.001412000 seconds]
    [Request in frame: 337625]
    [Request URI: http://www.szu.edu.cn/board/]
    File Data: 137 bytes
  < Line-based text data: text/html (7 lines)
```

图 26 HTTP 响应报文

- 1) Accept: call 服务器,可以接收文件、网页和图片。
- 2) Accept-Charset: 所接收的字符编码。
- 3) Accept-Encoding: 可接收 () 压缩后的数据。
- 4) Accept-Language: Browser 支持中、英文。
- 5) Host: 要找的主机是。
- 6) If-Modified-Since: 告诉服务器我们的缓冲中有这个资源文件,该文件的时间是, ,
- 7) Referer: 告诉服务器,我来自哪里。

- 8) User-Agent: 告诉服务器, Browser 内核。
- 9) Cookie:
- 10) Connection: 保持连续发完信息后, 我不关闭连接。
- 11) Date: Browser 发送时间。

(4) DNS

DNS (Domain Name System) 是一种用于将域名转换为 IP 地址的网络协议。它充当互联网上域名和 IP 地址之间的映射服务。DNS 的主要作用是将易记的域名 (如 example.com) 转换为计算机可识别的 IP 地址 (如 192.0.2.1)。DNS 使用 UDP (User Datagram Protocol) 或 TCP (Transmission Control Protocol) 来进行通信。

```
> Frame 1548727: 269 bytes on wire (2152 bits), 269 bytes captured (2152 bits) on interface \Device\NPF_{51B62527-
> Ethernet II, Src: H3CTechnolog_2b:9a:ff (38:22:d6:2b:9a:ff), Dst: ASUSTekCOMPU_cd:35:cf (c8:7f:54:cd:35:cf)
> Internet Protocol Version 4, Src: 192.168.247.6, Dst: 172.29.15.6
✓ User Datagram Protocol, Src Port: 53, Dst Port: 60245
  Source Port: 53
  Destination Port: 60245
  Length: 235
  Checksum: 0xbb1a [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2114]
  > [Timestamps]
    UDP payload (227 bytes)
✓ Domain Name System (response)
  Transaction ID: 0x4edf
  > Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 3
  Authority RRs: 1
  Additional RRs: 0
  > Queries
  > Answers
  > Authoritative nameservers
    [Request In: 1548724]
    [Time: 0.000623000 seconds]
```

图 27 DNS 数据包

DNS 数据包主要分为四个部分, 数据包头部, 查询部分, 回答部分和授权资源记录部分。

- 数据包头部:

事务 ID (Transaction ID): 16 位字段, 用于标识每个 DNS 查询和响应的唯一事务。

标志位 (Flags): 包括查询/响应标志、操作码、授权回答标志、截断标志、递归查询标志等。

问题数 (Question Count): 指明查询部分包含的问题数目。

回答数 (Answer Count): 指明回答部分包含的资源记录数目。

授权资源记录数 (Authority Record Count): 指明授权资源记录部分的记录数目。

附加资源记录数 (Additional Record Count): 指明附加资源记录部分的记录数目。

- 查询部分 (Questions) :

域名 (Domain Name) : 需要解析的域名, 以标签 (label) 序列的方式表示, 例如 "example.com" 表示为 "example"、"com"。

查询类型 (Query Type) : 指明查询的类型, 比如 A 记录、AAAA 记录、MX 记录等。

查询类 (Query Class) : 通常是 IN (Internet) 类。

- 回答部分 (Answers) :

域名: 与查询部分中的域名对应。

类型 (Type) : 指明资源记录的类型, 比如 A 记录、AAAA 记录、CNAME 记录等。

类 (Class) : 通常是 IN (Internet) 类。

TTL (Time to Live) : 指明记录在缓存中的有效时间。

数据长度 (Data Length) : 指明数据字段的长度。

数据 (Data) : 具体的资源记录数据, 例如 IP 地址、CNAME 的目标域名等。

- 授权资源记录部分 (Authority Records) 和 附加资源记录部分 (Additional Records) :

同样包括域名、类型、类、TTL、数据长度和数据字段。

实验结果:

(此页附完成的实验结果、并给出个人对结果的分析、结论)

本次实验，我成功安装了协议分析软件 **wireshark**，熟悉了 **wireshark** 的界面以及其基本功能的使用。之后，我对 **TCP** 协议的三次握手进行分析，了解了 **TCP** 协议传输信息前的流程，三次握手的目的是确保客户端和服务端都能够正常发送和接收数据。通过这个过程，客户端和服务端之间建立了可靠的通信连接，并且在通信过程中可以进行序列号的同步，确保数据的有序传输和可靠性。最后，我对 **UDP**，**DNS**，**HTTP**，**TCP** 协议数据包进行分析，了解了它们的作用。**DNS**：将域名转换为 **IP** 地址的协议，使用 **UDP** 或 **TCP** 进行通信。**HTTP**：用于在 **Web** 浏览器和 **Web** 服务器之间传输信息的协议，基于 **TCP**，采用请求-响应模型。**UDP**：面向无连接的传输协议，快速但不可靠，适用于对传输速度要求高的应用。**TCP**：面向连接的传输协议，提供可靠的数据传输服务，适用于对数据完整性和顺序性要求高的应用。总的来说，本次实验成功完成。

实验小结：

（实验中出现问题解决方法，实验心得体会等）

本次实验的大部分内容都是需要自己去查找资料解决的，所以网上搜索的能力就是十分的重要。而且有很多网址，有时候可以捕获到信息，有时候又捕获不到，就十分考验耐心，很多时候也需要换着网址使用，才可以捕获到想要的数据包。

指导教师批阅意见：

成绩评定：

指导教师签字：

年 月 日

备注：