



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
“Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ
И КОМПЬЮТЕРНОЙ ТЕХНИКИ



Учебно-исследовательская работа №4

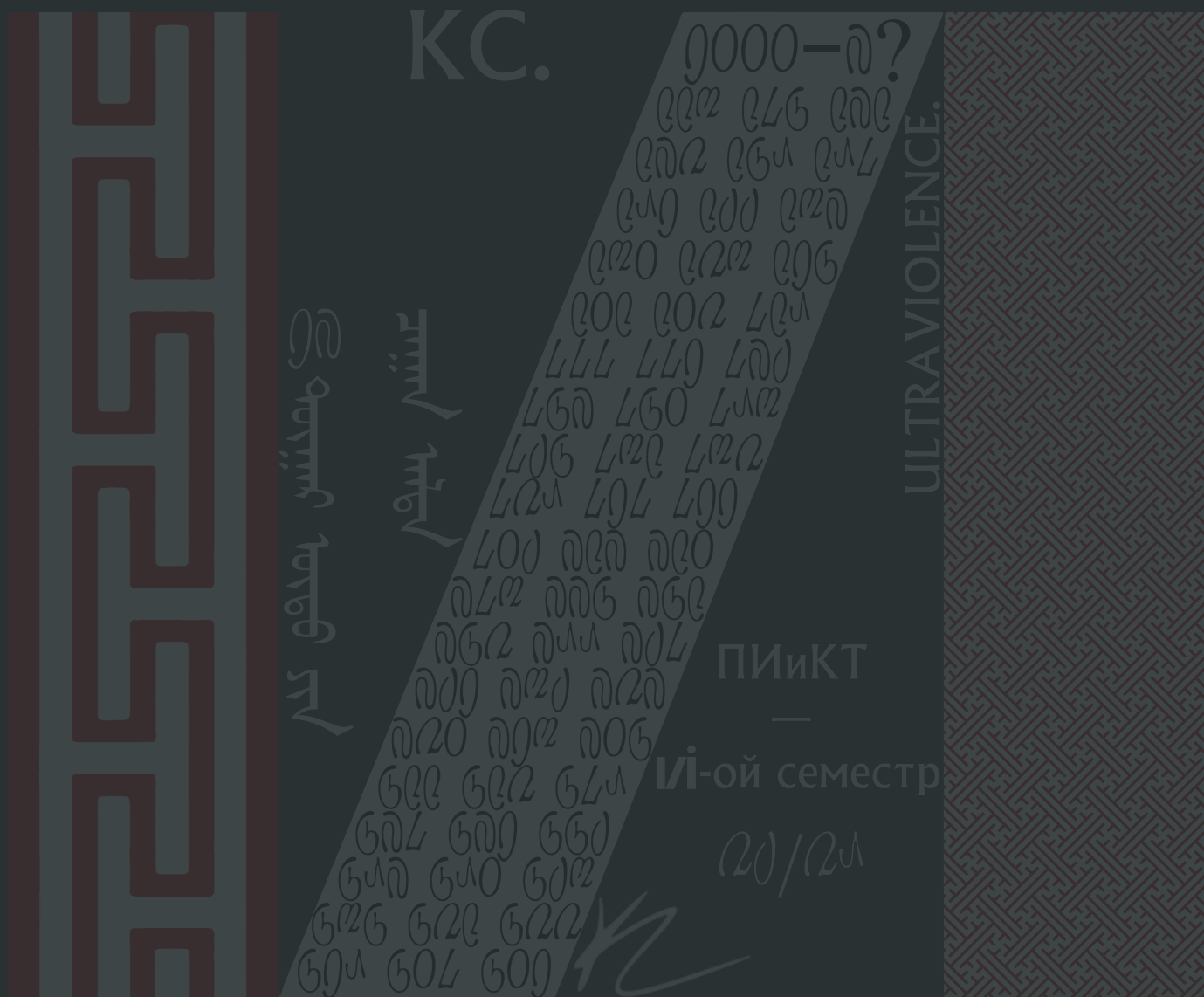
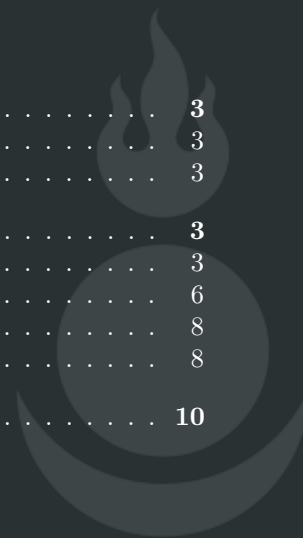
«Анализ трафика компьютерных сетей с помощью утилиты
Wireshark»
по дисциплине
“Компьютерные сети”

Работу выполнил:
Студент группы Р3311
Болорболд Аригуун

Лектор:
Алиев Тауфик Измаилович, д.т.н.
Практик:
Тропченко Андрей Александрович

Содержимое

1	Цель и описание работы	3
1.1	Цель работы	3
1.2	Краткое описание работы	3
2	Выполнение работы	3
2.1	Этап 1. Анализ трафика утилиты ping.	3
2.2	Этап 2. Анализ трафика утилиты tracert (traceroute).	6
2.3	Этап 3. Анализ HTTP-трафика.	8
2.4	Этап 4. Анализ ARP-трафика.	8
3	Вывод	10



1 Цель и описание работы

1.1 Цель работы

Цель работы – изучить структуру протокольных блоков данных, анализируя реальный трафик на компьютере студента с помощью бесплатно распространяемой утилиты Wireshark.

1.2 Краткое описание работы

В процессе выполнения домашнего задания выполняются наблюдения за передаваемым трафиком с компьютера пользователя в Интернет и в обратном направлении. Применение специализированной утилиты Wireshark позволяет наблюдать структуру передаваемых кадров, пакетов и сегментов данных различных сетевых протоколов. При выполнении УИР рекомендуется выполнить анализ последовательности команд и определить назначение служебных данных, используемых для организации обмена данными в протоколах: ARP, DNS, FTP, HTTP, DHCP.

2 Выполнение работы

Используемый веб-сайт: <https://www.nba.com>

2.1 Этап 1. Анализ трафика утилиты ping.

```
PS C:\Users\Admin> ping -l 10000 www.nba.com

Pinging e8017.dscb.akamaiedge.net [2.21.189.37] with 10000 bytes of data:
Reply from 2.21.189.37: bytes=10000 time=17ms TTL=54
Reply from 2.21.189.37: bytes=10000 time=49ms TTL=54
Reply from 2.21.189.37: bytes=10000 time=31ms TTL=54
Reply from 2.21.189.37: bytes=10000 time=37ms TTL=54

Ping statistics for 2.21.189.37:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 17ms, Maximum = 49ms, Average = 33ms
PS C:\Users\Admin> |
```

Анализ полученных пакетов

207 0.338283	172.28.21.50	2.21.189.37	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=4731) [Reassembled in #208]
208 0.338283	172.28.21.50	2.21.189.37	ICMP	562 Echo (ping) request id=0x0001, seq=48/12288, ttl=128 (reply in 228)
227 0.369099	2.21.189.37	172.28.21.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=7026) [Reassembled in #228]
228 0.369099	2.21.189.37	172.28.21.50	ICMP	562 Echo (ping) reply id=0x0001, seq=48/12288, ttl=54 (request in 208)
648 1.347321	172.28.21.50	2.21.189.37	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=4732) [Reassembled in #649]
649 1.347321	172.28.21.50	2.21.189.37	ICMP	562 Echo (ping) request id=0x0001, seq=50/12800, ttl=128 (reply in 657)
656 1.361732	2.21.189.37	172.28.21.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=7083) [Reassembled in #657]
657 1.361732	2.21.189.37	172.28.21.50	ICMP	562 Echo (ping) reply id=0x0001, seq=50/12800, ttl=54 (request in 649)
1313 2.356081	172.28.21.50	2.21.189.37	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=4733) [Reassembled in #1314]
1314 2.356081	172.28.21.50	2.21.189.37	ICMP	562 Echo (ping) request id=0x0001, seq=51/13056, ttl=128 (reply in 1326)
1325 2.373774	2.21.189.37	172.28.21.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=70c1) [Reassembled in #1326]
1326 2.373774	2.21.189.37	172.28.21.50	ICMP	562 Echo (ping) reply id=0x0001, seq=51/13056, ttl=54 (request in 1314)
2086 3.366515	172.28.21.50	2.21.189.37	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=4734) [Reassembled in #2087]
2087 3.366515	172.28.21.50	2.21.189.37	ICMP	562 Echo (ping) request id=0x0001, seq=52/13312, ttl=128 (reply in 2103)
2102 3.381558	2.21.189.37	172.28.21.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=72ed) [Reassembled in #2103]
2103 3.381558	2.21.189.37	172.28.21.50	ICMP	562 Echo (ping) reply id=0x0001, seq=52/13312, ttl=54 (request in 2087)

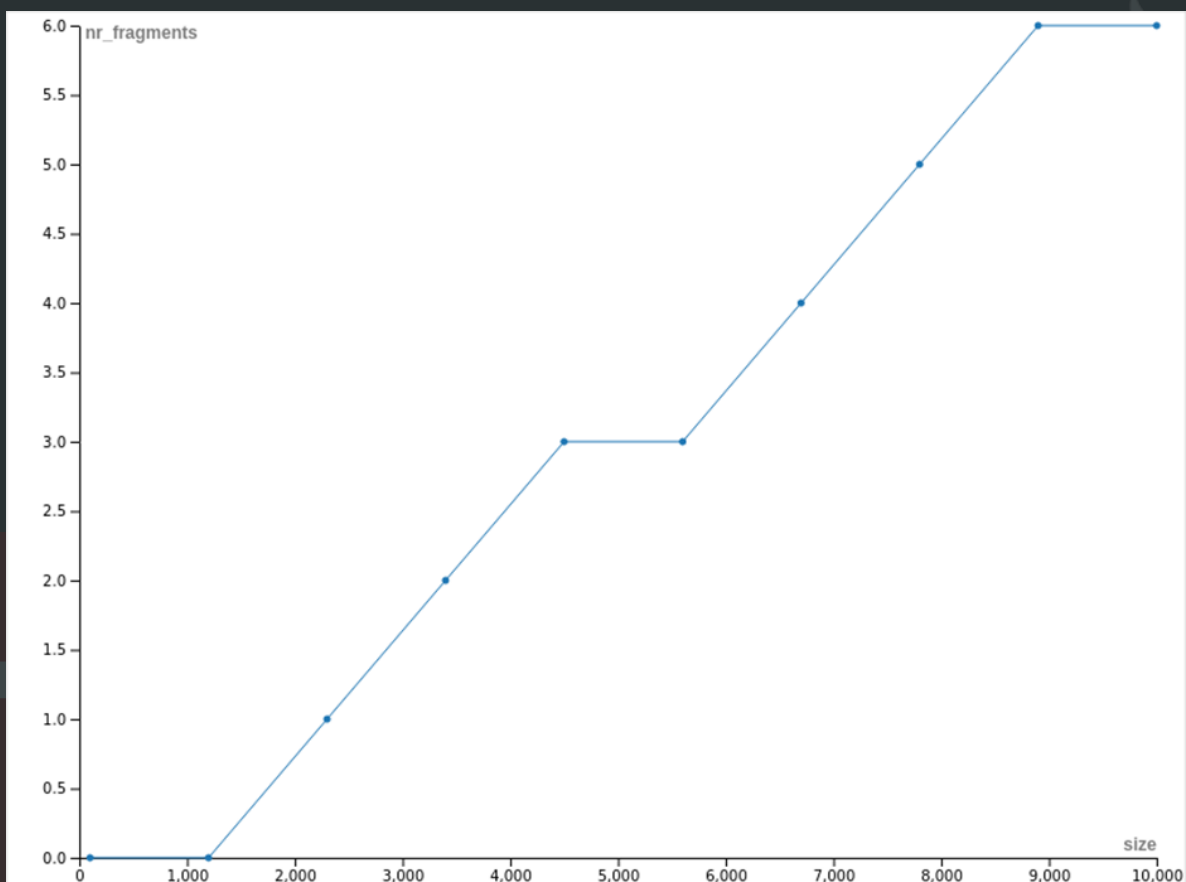
```

▼ Internet Protocol Version 4, Src: 172.28.21.50, Dst: 2.21.189.37
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x4731 (18225)
  ▼ 001. .... = Flags: 0x1, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x4d67 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.28.21.50
    Destination Address: 2.21.189.37
    [Reassembled IPv4 in frame: 208]
    [Stream index: 101]
  ► Data (1480 bytes)

```

1. Да, имеет место фрагментация пакета. Указано в поле MF IPv4 пакета.
2. Для промежуточных пакетов поле MF=1, для последнего MF=0
3. Ping передает данные по 32 байта, так что фрагментации для них нет, т.е. 0 фрагментов

График



1. Использовать флаг -i <ttl>
2. Символы английского алфавита, иногда может быть отметка времени и Sequence Number ICMP-пакета.


```

▼ Internet Protocol Version 4, Src: 172.28.21.50, Dst: 2.21.189.37
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x474b (18251)
  ▼ 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
  ▶ Time to Live: 3
    Protocol: ICMP (1)
    Header Checksum: 0xefcd [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.28.21.50
    Destination Address: 2.21.189.37
    [Stream index: 47]

```

1. Заголовок: 20 байт, данные: 92.
2. Увеличивается на 1 каждые 3 пакета, чтобы выявить расстояние в хопх до хоста

```

▼ Internet Protocol Version 4, Src: 172.28.21.50, Dst: 2.21.189.37
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x474e (18254)
  ▼ 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
  ▶ Time to Live: 4
    Protocol: ICMP (1)
    Header Checksum: 0xeeca [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.28.21.50
    Destination Address: 2.21.189.37
    [Stream index: 47]

```

1. В ping-пакетах есть отметка времени, дефолтное значение TTL сильно выше, не меняется со следующим пакетом (в силу специфики утилиты).
2. Различные значения в поле Type.
3. Будут также слаться DNS запросы, чтобы разрешить IP-адреса в доменные имена.

2.3 Этап 3. Анализ HTTP-трафика.

Запустим анализ в Wireshark и перейдём на сайт www.nba.com. К сожалению, сайт, который подходит нам по варианту, мало того, что не обладает возможностью принимать условные GET-запросы, так ещё и запрещает обращения по HTTP вместо HTTPS. Сколько раз не обновляй мы не можем получить ответ 304. Поэтому воспользуемся сайтом, который точно обладает такой возможностью, а именно сайтом example.com

Сначала просто зайдём на сайт example.com.

No.	Time	Source	Destination	Protocol	Length	Info
16381	44.231726	96.7.128.198	172.28.21.50	TLSv1.3	325	Application Data
16382	44.231726	96.7.128.198	172.28.21.50	TLSv1.3	325	Application Data
16384	44.231990	96.7.128.198	172.28.21.50	TCP	1514	443 → 57185 [ACK] Seq=3266 Ack=697 Win=64128 Len=1460 [TCP PDU reassembled in 16385]
16385	44.231990	96.7.128.198	172.28.21.50	TLSv1.3	238	Application Data
32779	82.249469	96.7.128.198	172.28.21.50	TCP	66	80 → 57192 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
32856	82.456001	96.7.128.198	172.28.21.50	TCP	56	80 → 57192 [ACK] Seq=1 Ack=157 Win=64128 Len=0
32857	82.457525	96.7.128.198	172.28.21.50	TCP	1514	80 → 57192 [ACK] Seq=1 Ack=157 Win=64128 Len=1460 [TCP PDU reassembled in 32858]
32858	82.457525	96.7.128.198	172.28.21.50	HTTP	112	HTTP/1.1 200 OK (text/html)
38160	93.873641	96.7.128.198	172.28.21.50	QUIC	1292	Initial, SCID=04862262d3402926, PKN: 1, ACK, PADDING
38162	93.874076	96.7.128.198	172.28.21.50	QUIC	1292	Initial, SCID=04862262d3402926, PKN: 2, CRYPTO, PADDING
38163	93.874876	96.7.128.198	172.28.21.50	QUIC	287	Handshake, SCID=04862262d3402926
38235	94.081814	96.7.128.198	172.28.21.50	QUIC	329	Protected Payload (KP0)
38236	94.081814	96.7.128.198	172.28.21.50	QUIC	103	Protected Payload (KP0)
38237	94.081814	96.7.128.198	172.28.21.50	QUIC	63	Protected Payload (KP0)
38238	94.081814	96.7.128.198	172.28.21.50	QUIC	112	Protected Payload (KP0)
38357	94.287107	96.7.128.198	172.28.21.50	QUIC	65	Protected Payload (KP0)
48953	114.080908	96.7.128.198	172.28.21.50	QUIC	73	Protected Payload (KP0)
49554	115.123547	96.7.128.198	172.28.21.50	QUIC	66	Protected Payload (KP0)
53975	124.680442	96.7.128.198	172.28.21.50	QUIC	100	Protected Payload (KP0)

Frame 32858: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface \Device\NPF_{9F4B4F81-7238-4090-B000-000000000000} on interface \Device\NPF_{9F4B4F81-7238-4090-B000-000000000000}

Ethernet II, Src: Routerboardc_21:4c:a5 (48:8f:5a:21:4c:a5), Dst: CloudNetwork_31:48:d7 (38:d5:7a:31:48:d7)

Internet Protocol Version 4, Src: 96.7.128.198, Dst: 172.28.21.50

Transmission Control Protocol, Src Port: 80, Dst Port: 57192, Seq: 1461, Ack: 157, Len: 58

[2 Reassembled TCP Segments (1518 bytes): #32857(1460), #32858(58)]

Hypertext Transfer Protocol

Line-based text data: text/html (46 lines)

Затем, что все отработало как надо, мы получаем ответ 200. Теперь попробуем обновить страницу и посмотрим, что будет.

2211	3.954360	172.28.21.50	23.192.228.84	HTTP	210	GET / HTTP/1.1
2393	4.151886	23.192.228.84	172.28.21.50	HTTP	112	HTTP/1.1 200 OK (text/html)
3126	5.282490	172.28.21.50	23.192.228.84	HTTP	186	GET / HTTP/1.1
3159	5.478514	23.192.228.84	172.28.21.50	HTTP	112	HTTP/1.1 200 OK (text/html)
3899	6.664571	172.28.21.50	23.192.228.84	HTTP	186	GET / HTTP/1.1
4006	6.862524	23.192.228.84	172.28.21.50	HTTP	112	HTTP/1.1 200 OK (text/html)
4727	7.927699	172.28.21.50	23.192.228.84	HTTP	186	GET / HTTP/1.1
4877	8.128736	23.192.228.84	172.28.21.50	HTTP	112	HTTP/1.1 200 OK (text/html)
5425	9.079313	172.28.21.50	23.192.228.84	HTTP	186	GET / HTTP/1.1
5553	9.275317	23.192.228.84	172.28.21.50	HTTP	112	HTTP/1.1 200 OK (text/html)
11716	22.026355	172.28.21.50	96.7.128.175	HTTP	667	GET / HTTP/1.1
11768	22.232387	96.7.128.175	172.28.21.50	HTTP	304	HTTP/1.1 304 Not Modified

Frame 11768: 304 bytes on wire (2432 bits), 304 bytes captured (2432 bits) on interface \Device\NPF_{9F4B4F81-7238-4090-B000-000000000000} on interface \Device\NPF_{9F4B4F81-7238-4090-B000-000000000000}

Ethernet II, Src: Routerboardc_21:4c:a5 (48:8f:5a:21:4c:a5), Dst: CloudNetwork_31:48:d7 (38:d5:7a:31:48:d7)

Internet Protocol Version 4, Src: 96.7.128.175, Dst: 172.28.21.50

Transmission Control Protocol, Src Port: 80, Dst Port: 57231, Seq: 1, Ack: 614, Len: 250

Hypertext Transfer Protocol

Content-Type: text/html\r\n

Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT\r\n

Заметим, что мы получаем совсем другую ситуацию. Здесь у нас получилось отправить условный GET-запрос. И мы получаем ответ 304 от сервера, которого можно понять по появившимся полям Last-Modified и If-Modified-Since.

2.4 Этап 4. Анализ ARP-трафика.

Для начала очистим ARP-таблицу с помощью команды:

```
netsh interface ip delete arpccache
```

Получим вот такую ARP-таблицу:

```
PS C:\WINDOWS\system32> netsh interface ip delete arpccache
Ok.
```

```
PS C:\WINDOWS\system32>
```

После удаления кэша браузера отправимся на сайт www.nba.com и увидим новую запись в ARP-

таблице.

```
PS C:\WINDOWS\system32> arp -a

Interface: 172.28.21.50 --- 0xc
    Internet Address      Physical Address
    172.28.16.1            48-8f-5a-21-4c-a5
    172.28.16.11           14-13-33-36-a6-55
    172.28.16.12           08-9d-f4-f4-ce-41
    172.28.19.59           bc-03-58-7e-32-54
    172.28.20.126          f0-a6-54-27-29-e9
    172.28.21.138          48-51-c5-6d-a9-b6
    172.28.21.252          f0-20-ff-dc-d0-3e
    172.28.22.49           e4-aa-ea-64-35-65
    172.28.22.60           f4-6a-dd-43-44-c7
    172.28.22.62           68-34-21-c8-98-f3
    172.28.23.43           6c-2f-80-f4-5b-64
    172.28.24.81           28-6b-35-a0-dc-8c
    172.28.24.175          6c-94-66-1c-92-ec
    172.28.26.239          6c-2f-80-20-fe-ae
    224.0.0.2              01-00-5e-00-00-02
    224.0.0.22             01-00-5e-00-00-16
    224.0.0.253            01-00-5e-00-00-fd
    239.192.152.143        01-00-5e-40-98-8f
    239.255.102.18         01-00-5e-7f-66-12
    239.255.188.58         01-00-5e-7f-bc-3a
    255.255.255.255        ff-ff-ff-ff-ff-ff

PS C:\WINDOWS\system32>
```

Заметим, что это вообще не похоже на IP-адрес сайта, на который мы перешли. А всё, потому что MAC-адреса используется только в локальной сети. Мы не сможем увидеть ARP-запрос, который узнаёт MAC-адрес нашего сайта, так как его и вовсе нет. Но мы видим IP-адрес 192.168.1.1 —

скорее всего это IP-адрес нашего маршрутизатора, который как раз таки и взялся в дальнейшем уже за поиск того сайта, на который мы перешли.

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000000	Apple_b4:f0:1a	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.147? Tell 172.28.23.70
20	0.055966	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.23.45? Tell 172.28.16.1
22	0.044076	f6:00:97:39:36:f8	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.20.155
26	0.051808	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.23.58? Tell 172.28.16.1
30	0.060755	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.23.144? Tell 172.28.16.1
31	0.060755	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.21.125? Tell 172.28.16.1
32	0.076122	ae:76:fb:c4:dd:c6	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.23.114
59	0.101568	ee:d0:a5:30:b0:85	CloudNetwork_31:48::	ARP	56	Who has 172.28.29.6? Tell 172.28.28.132
62	0.107793	Apple_17:34:de	CloudNetwork_31:48::	ARP	56	Who has 172.28.16.1? Tell 172.28.28.108
69	0.119602	62:51:97:7d:2b:44	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.18.255
72	0.131964	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.77? Tell 172.28.16.1
88	0.149647	Apple_1b:14:c8	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.23.92
97	0.173889	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.21.28? Tell 172.28.16.1
110	0.192062	9a:28:3f:22:14:7d	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.27.150
112	0.191666	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.174? Tell 172.28.16.1

1. В ARP-пакетах мы увидим 2 типа MAC-адресов:

- MAC-адрес отправителя запроса — адрес нашего компьютера. Он используется в поле Sender MAC-address.
- MAC-адрес искомого устройства:
 - В ARP-запросе (who-has) поле Target MAC Address будет заполнено нулями, потому что он ещё известен.
 - В ARP-ответе (is-at) это будет MAC-адрес шлюза/маршрутизатора, провайдера или другого узла локальной сети, связанного с IP, на который отправляется запрос.

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000000	Apple_b4:f0:1a	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.23.70
20	0.055966	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.23.45? Tell 172.28.16.1
22	0.044076	f6:00:97:39:36:f8	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.20.155
26	0.051808	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.23.58? Tell 172.28.16.1
30	0.060755	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.23.144? Tell 172.28.16.1
31	0.060755	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.21.125? Tell 172.28.16.1
32	0.076122	ae:76:fb:c4:dd:c6	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.23.114
59	0.101568	ee:d0:a5:30:b0:85	CloudNetwork_31:48::	ARP	56	Who has 172.28.29.6? Tell 172.28.28.132
62	0.107793	Apple_17:34:de	CloudNetwork_31:48::	ARP	56	Who has 172.28.16.1? Tell 172.28.28.108
69	0.119602	62:51:97:7d:2b:44	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.18.255
72	0.131964	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.77? Tell 172.28.16.1
88	0.149647	Apple_1b:14:c8	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.23.92
97	0.173889	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.21.28? Tell 172.28.16.1
110	0.192062	9a:28:3f:22:14:7d	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.181? Tell 172.28.27.150
112	0.191666	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.22.174? Tell 172.28.16.1
113	0.191666	Routerboardc_21:4c::	CloudNetwork_31:48::	ARP	56	Who has 172.28.31.204? Tell 172.28.16.1
119	0.203057	92:fa:3f:1e:5e:91	CloudNetwork_31:48::	ARP	56	ARP Announcement for 172.28.23.1
121	0.207599	c2:d6:35:e7:3e:00	CloudNetwork_31:48::	ARP	56	Who has 172.28.20.220? Tell 172.28.20.158
122	0.207793	Intel_fc:9e:f0	CloudNetwork_31:48::	ARP	56	ARP Announcement for 169.254.247.211

Frame 30: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{9F4B4F81-7238-4973-AE-0000} 38 d5 7a 31 48 d7 48 8f 5a 21 4c a5 08 06 00 01 8 :IH H ZIL ...

Ethernet II, Src: Routerboardc_21:4c:a5 (48:8f:5a:21:4c:a5), Dst: CloudNetwork_31:48:d7 (38:d5:7a:31:48:d7) 0010 00 06 04 00 01 48 8f 5a 21 4c a5 ac 1c 18 01H ZIL ...

Destination: CloudNetwork_31:48:d7 (38:d5:7a:31:48:d7) 0020 00 00 00 00 00 ac 1c 17 90 00 00 00 00 00

Source: Routerboardc_21:4c:a5 (48:8f:5a:21:4c:a5) 0030 00 00 00 00 00 00 00 00 00 00 00 00 00

Type: ARP (0x0806)

[Stream index: 9]

Trailer: 00000000000000000000000000000000

Address Resolution Protocol (request)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (1)

Sender MAC address: Routerboardc_21:4c:a5 (48:8f:5a:21:4c:a5)

Sender IP address: 172.28.16.1

Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)

Target IP address: 172.28.23.144

2.

HTTP работает поверх TCP/IP в Ethernet. В Ethernet-заголовке каждого HTTP-пакета указывается:

- MAC-адрес источника — это MAC-адрес нашего компьютера
- MAC-адрес назначения — это обычно MAC-адрес ближайшего маршрутизатора/шлюза, через который трафик пойдет в интернет.

MAC-адреса веб-сайта, на который мы заходим, мы не увидим, потому что MAC-адреса используются только внутри локальной сети.

3. ARP-запросы содержит IP-адреса источника, чтобы:

- Получатель запроса (тот, чей IP-адрес запрашивается) мог записать в свою ARP-таблицу соответствие, и тем самым сократить количество ARP-запросов в будущем. Получатель понимал, кто запрашивает — это нужно для формирования ARP-запроса/ответа.

IP-адрес источника нужен для обратной связи и корректного построения локальной маршрутизации.

3 Вывод

Во время выполнения лабораторной работы мы познакомились с работой различных протоколов передачи данных, проанализировали переданные пакеты с помощью программы Wireshark и протестировали соединения через разные утилиты.