패킷 분석 보고서

Ver. 1.0

백 정 이

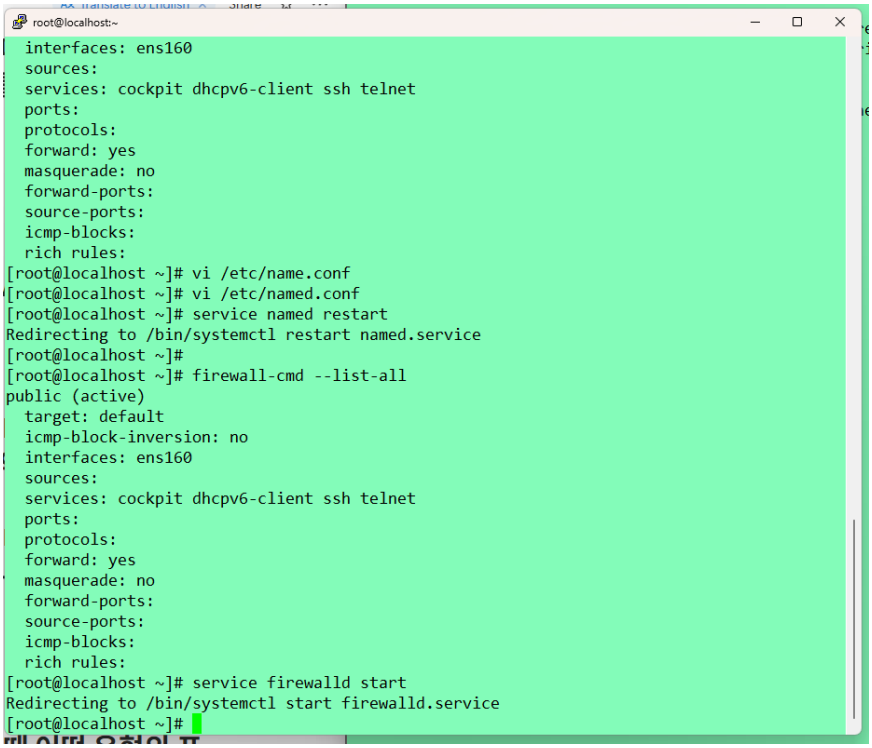
1. 포트스캔 결과 분석

1.1. 패킷 분석 목적

- 타겟의 포트를 nmap을 이용하여 TCP, UDP 스캔한 뒤 방화벽이 열렸을 때와 닫혔을 때의 wireshark 결과 차이 확인
- SYN 스캔과 FIN 스캔 실행 시 패킷 흐름 조사

1.2. 타겟 IP(10.10.10.10)의 1~100번 포트를 TCP SYN 스캔한 뒤 Wireshark 패킷 분석

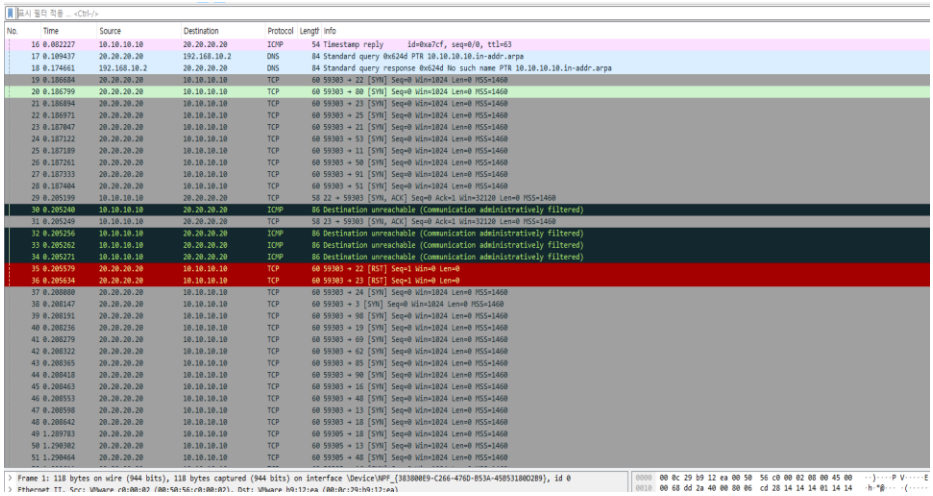
- **방화벽 작동 시작**

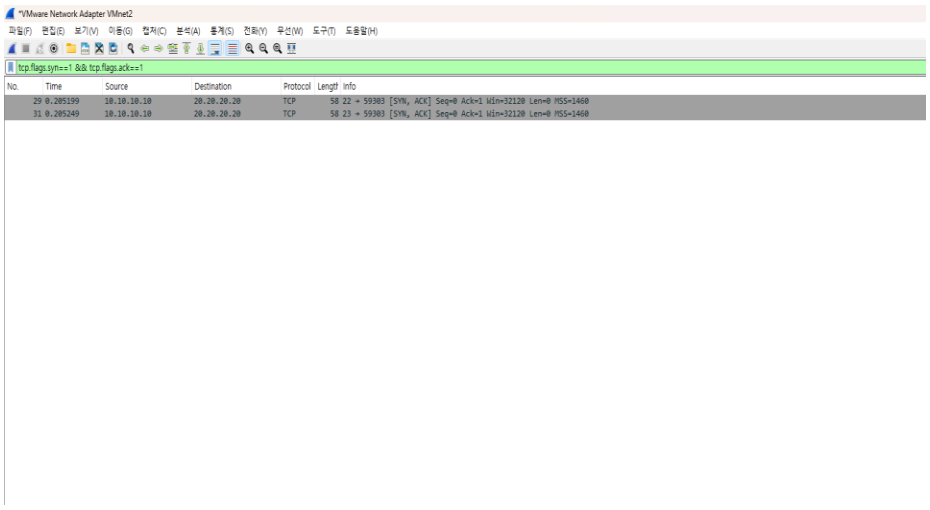


- **nmap으로 포트 스캔 시작**

```
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
┌──(root㉿kali)-[~]
└─# nmap -sS -p 1-100 10.10.10.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-24 02:09 KST
Nmap scan report for 10.10.10.10
Host is up (0.023s latency).
Not shown: 92 filtered tcp ports (no-response), 6 filtered tcp ports (admin-prohibited)
PORT   STATE SERVICE
22/tcp open  ssh
23/tcp open  telnet

Nmap done: 1 IP address (1 host up) scanned in 2.42 seconds

┌──(root㉿kali)-[~]
└─#
```

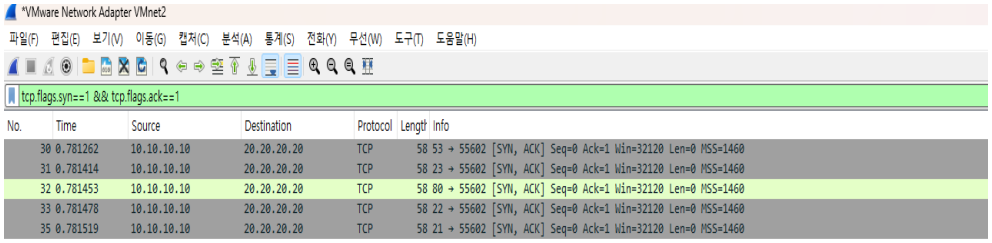
- **방화벽이 켜져있을 때 패킷 확인**





칼리 리눅스(20.20.20.20)에서 록키 리눅스(10.10.10.10)로 1 부터 100 번 포트에 syn 패킷을 보내고 반대로 syn + ack 응답은 방화벽에서 허용한 22, 23 번 포트에서 옴

- **방화벽이 꺼져있을 때 패킷 확인**

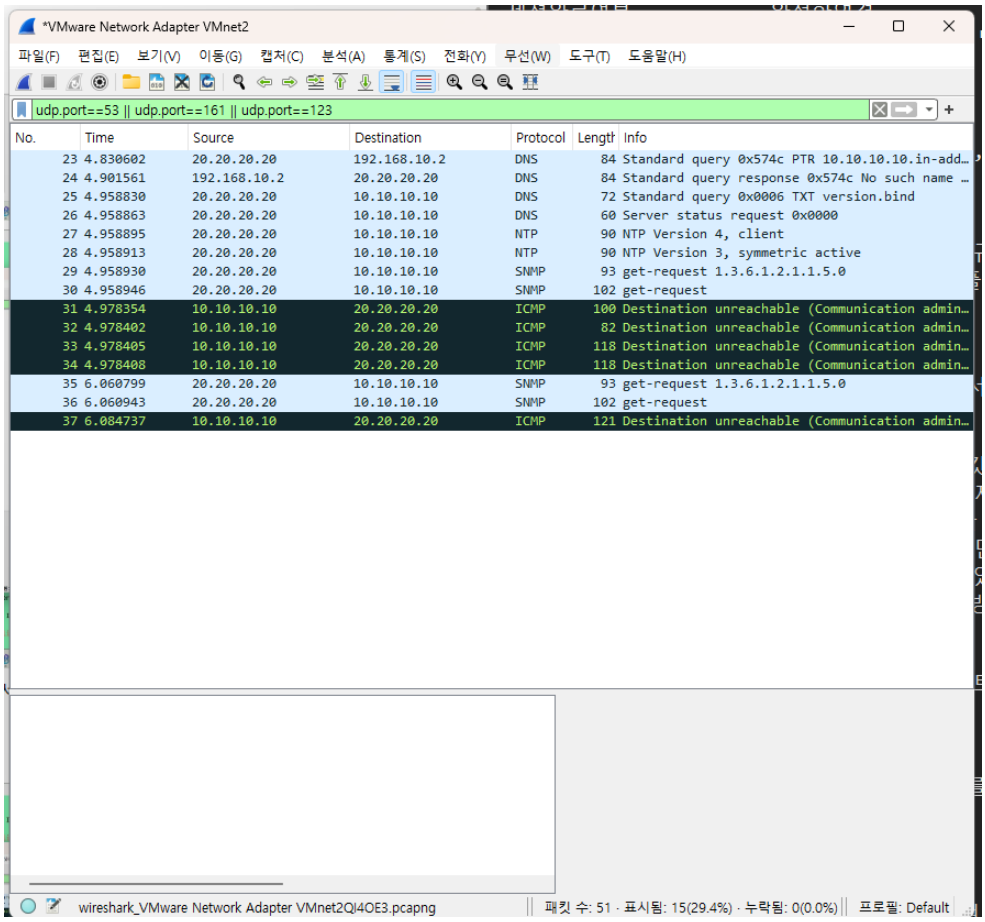


서비스 가동중인 포트인 21, 22, 23, 53, 80 번 포트에서 syn+ack 응답이 옴

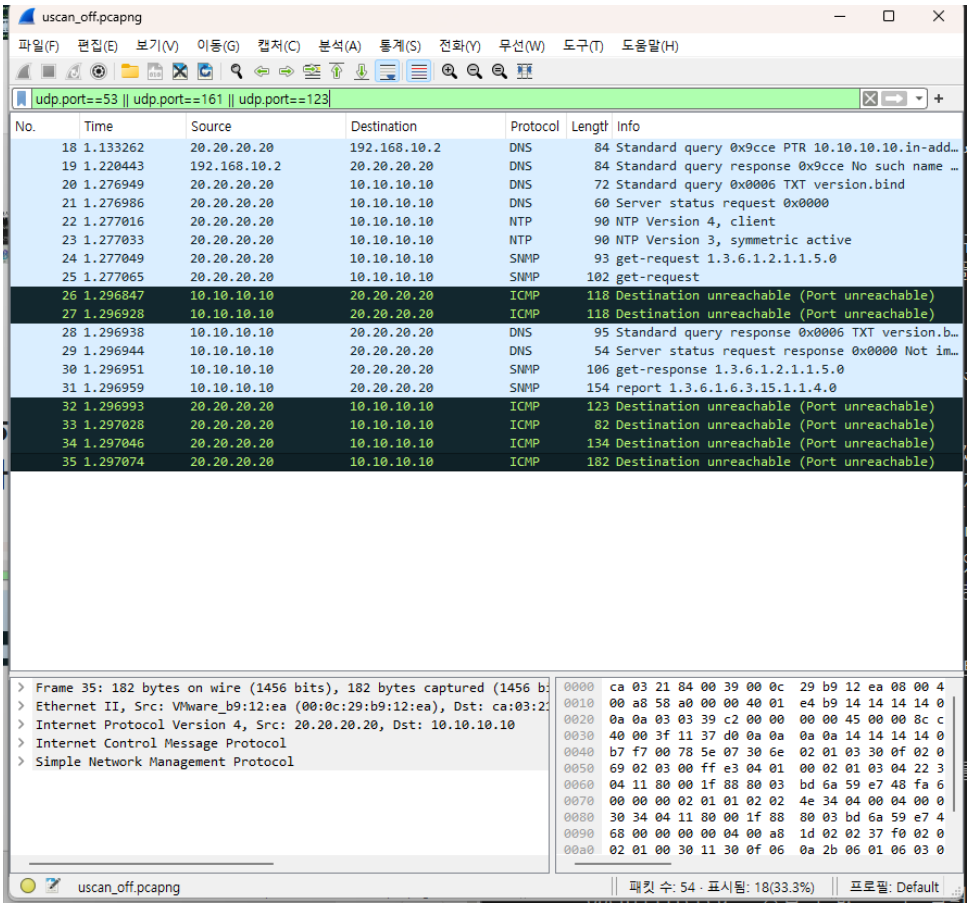
⇒ 열려 있는 포트에 syn 을 보내면 syn + ack 로 응답하고 닫혀있으면 응답하지 않음

1.3. UDP 포트 53, 161, 123을 스캔한 뒤 Wireshark 패킷 분석

- **방화벽이 꺼져있을 때 패킷 확인**



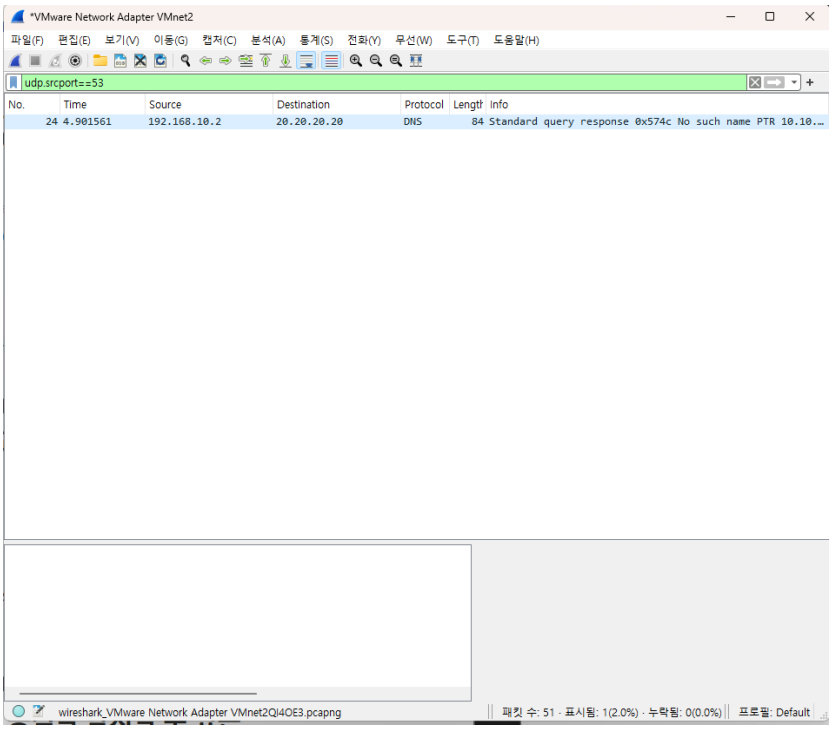
- **방화벽이 꺼져있을 때 패킷 확인**



⇒ 방화벽이 켜져있을 때 Destination Unreachable 메시지가 오지만 꺼져있을 때 10.10.10.10 으로부터 DNS(53), SNMP(161) 패킷이 전송됨

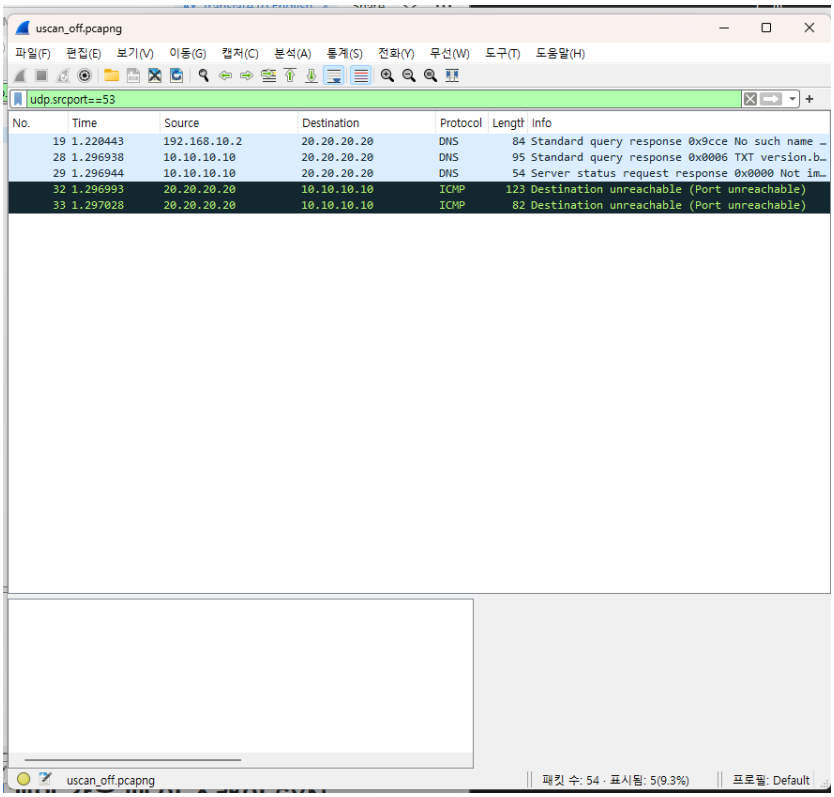
1.4. 출발지 포트를 53번으로 설정하여 DNS 요청으로 보이는 패킷을 Wireshark에서 분석

- **방화벽이 켜져있을 때 패킷 확인**



dns(53) 포트에서 패킷이 전송되지 않음

- **방화벽이 꺼져있을 때 패킷 확인**



DNS(53) 포트에서 패킷 전송됨

⇒ DNS 가 서비스되고있지만 방화벽에 의해 차단됨

1.5. SYN 스캔의 wireshark 패킷 흐름과 특징

[공격자] → [서버] SYN

[서버] → [공격자] SYN+ACK

[공격자] → [서버] RST

- 연결 수립하지 않고 RST 를 보내므로 SYN SCAN 임
- 연결 수립하지 않으므로 탐지 회피가 쉬움

1.6. FIN 스캔의 wireshark 패킷 흐름과 wireshark 탐지 필터

[공격자] → [서버] FIN

[서버] → [공격자] (응답 없음)

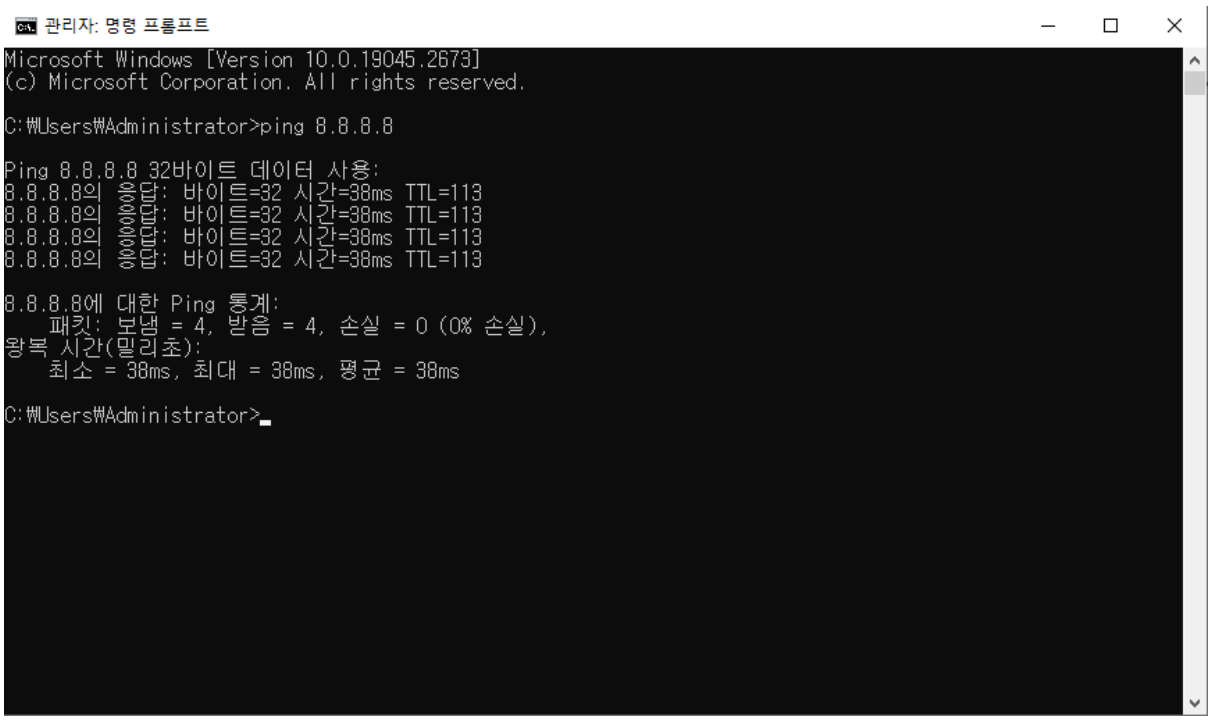
[공격자] → [서버] FIN

[서버] → [공격자] RST

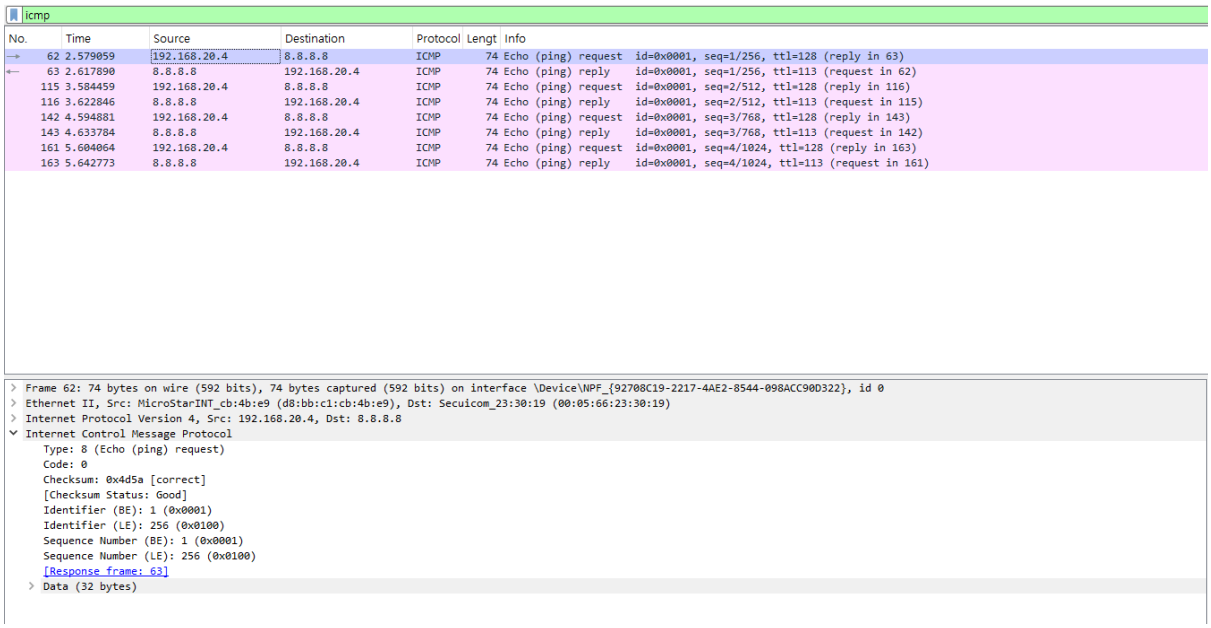
- FIN 을 보냈을 때 포트가 열려있으면 응답이 없고, 닫혀있으면 RST + ACK 를 보내는 것을 통해 열린 포트를 확인할 수 있음
- 와이어샤크 탐지 필터 : tcp.flags.fin

2. ICMP 패킷 분석

- **8.8.8.8로 ping 보내기**

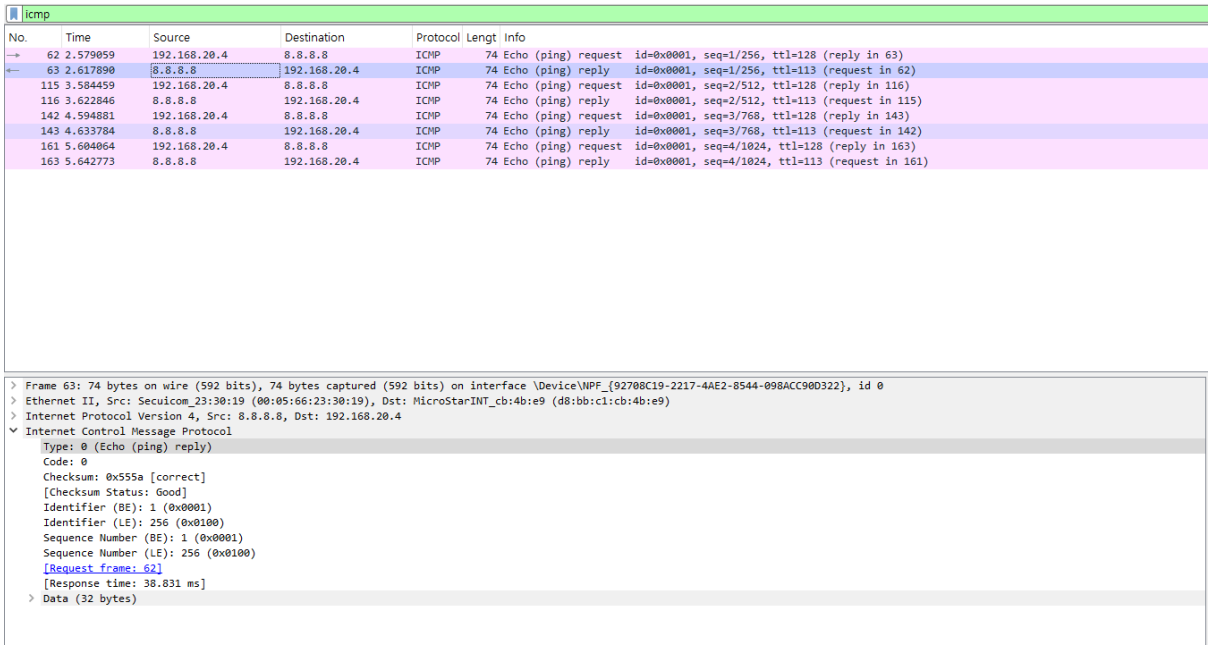


- **ICMP 필터링 결과 첫번째 패킷**



- ICMP 헤더에 type : 8 이 표시되고, IPv4 헤더에 Src : 192.168.20.4, Dst : 8.8.8.8 가 표시됨
- 목적지를 google DNS(8.8.8.8)으로 하여 ping 요청(ICMP Echo Request)을 보내는 것을 알 수 있음

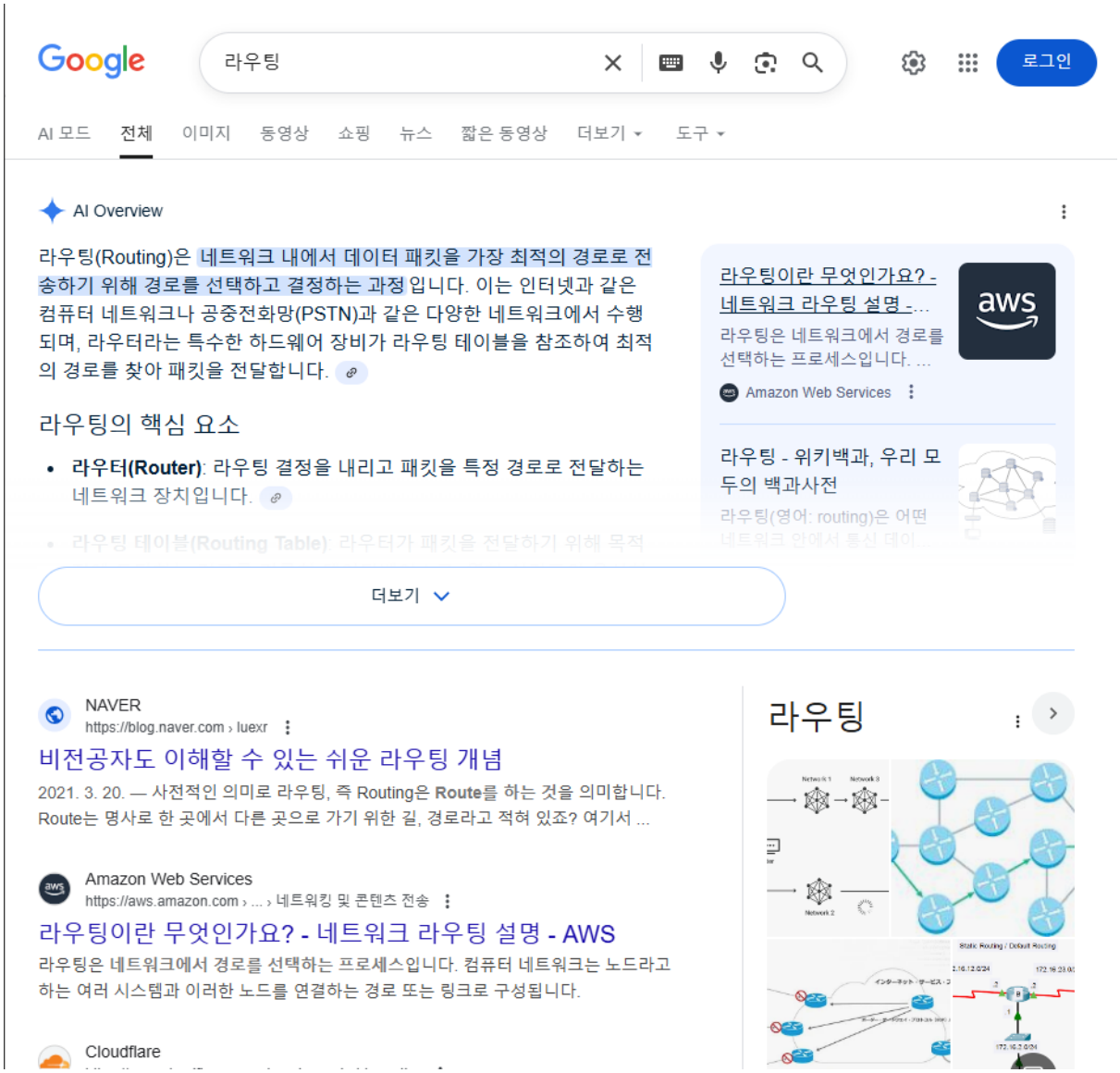
- **ICMP 필터링 결과 두번째 패킷**



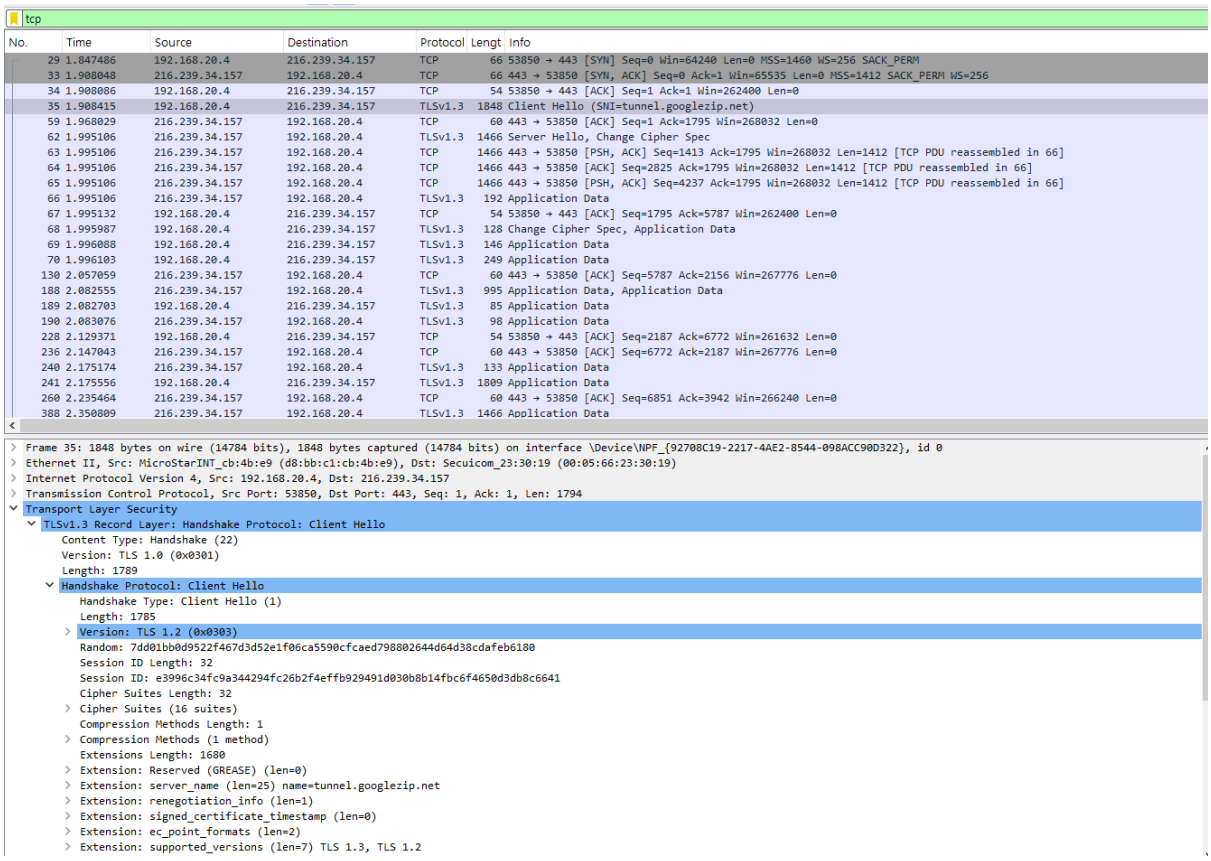
- ICMP 헤더에 typ : 0, IPv4 헤더에 Src : 8.8.8.8, Dst : 192.168.20.4 가 표시됨
- google DNS(8.8.8.8)으로부터 목적지 호스트(192.168.20.4)로 ping 응답(ICMP Echo Reply)을 보내는 것을 알 수 있음

3. TCP 패킷 분석

- **google chrome 브라우저에서 "라우팅" 검색**



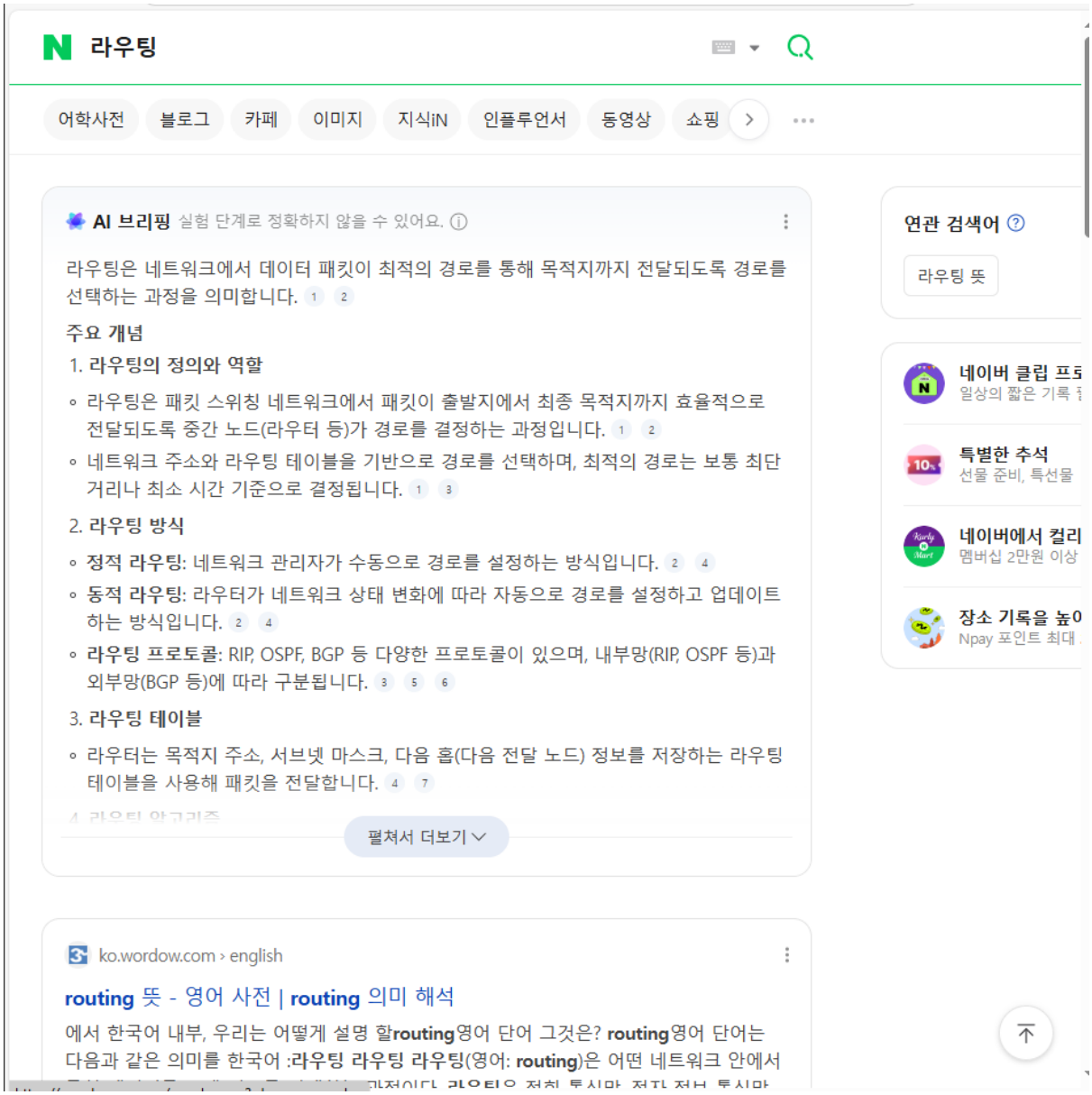
- **google chrome 브라우저에서 “라우팅” 검색 시 캡처한 TCP 패킷**



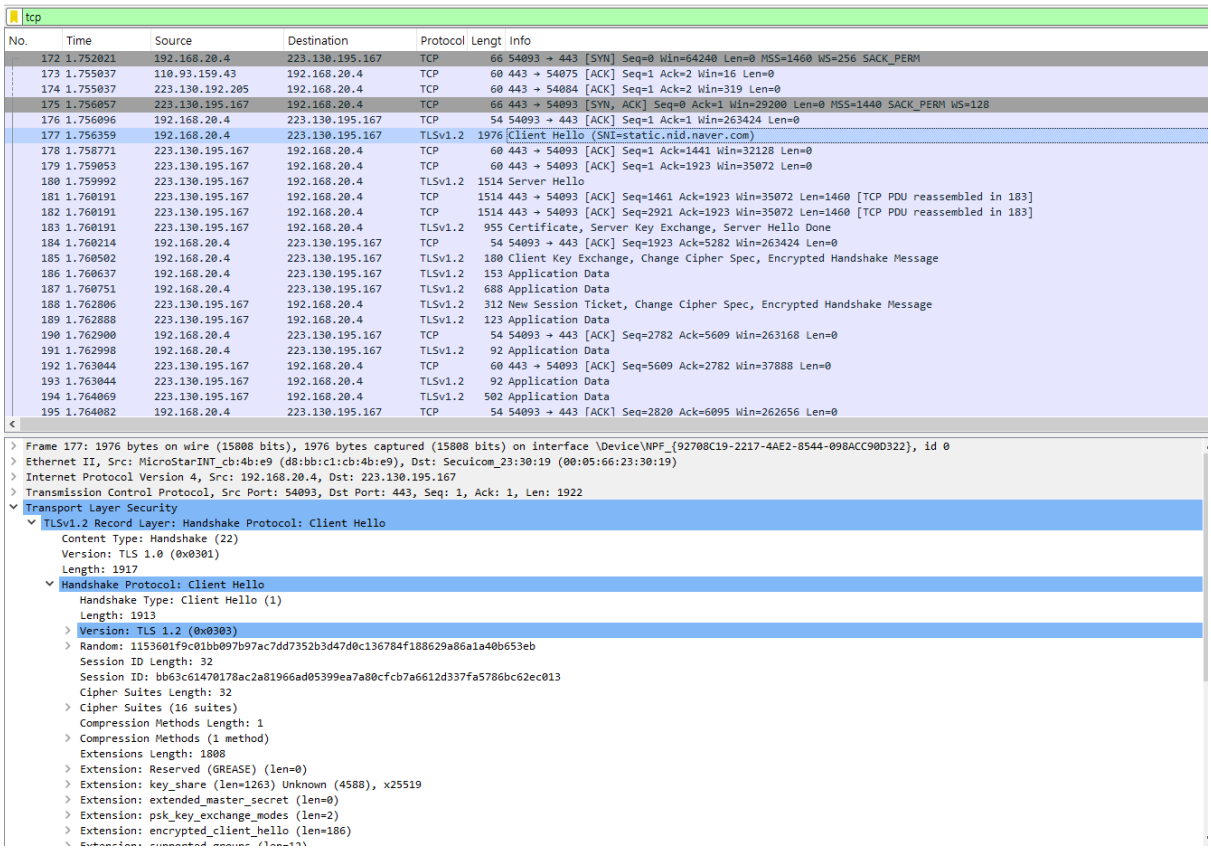
```
tcp
No.  Time      Source          Destination     Protocol Lengt Info
  29 1.847486  192.168.20.4    216.239.34.157  TCP        66 53850 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  33 1.908048  216.239.34.157  192.168.20.4    TCP        66 443 → 53850 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
  34 1.908086  192.168.20.4    216.239.34.157  TCP        54 53850 → 443 [ACK] Seq=1 Ack=1 Win=262400 Len=0
  35 1.908415  192.168.20.4    216.239.34.157  TLSv1.3  1848 Client Hello (SNI=tunnel.googlezip.net)
  59 1.968029  216.239.34.157  192.168.20.4    TCP        60 443 → 53850 [ACK] Seq=1 Ack=1795 Win=268032 Len=0
  62 1.995106  216.239.34.157  192.168.20.4    TLSv1.3  1466 Server Hello, Change Cipher Spec
  63 1.995106  216.239.34.157  192.168.20.4    TCP      1466 443 → 53850 [PSH, ACK] Seq=1413 Ack=1795 Win=268032 Len=1412 [TCP PDU reassembled in 66]
  64 1.995106  216.239.34.157  192.168.20.4    TCP      1466 443 → 53850 [ACK] Seq=2825 Ack=1795 Win=268032 Len=1412 [TCP PDU reassembled in 66]
  65 1.995106  216.239.34.157  192.168.20.4    TCP      1466 443 → 53850 [PSH, ACK] Seq=4237 Ack=1795 Win=268032 Len=1412 [TCP PDU reassembled in 66]
  66 1.995106  216.239.34.157  192.168.20.4    TLSv1.3   192 Application Data
  67 1.995132  192.168.20.4    216.239.34.157  TCP        54 53850 → 443 [ACK] Seq=1795 Ack=5787 Win=262400 Len=0
  68 1.995987  192.168.20.4    216.239.34.157  TLSv1.3   128 Change Cipher Spec, Application Data
  69 1.996088  192.168.20.4    216.239.34.157  TLSv1.3   146 Application Data
  70 1.996103  192.168.20.4    216.239.34.157  TLSv1.3   249 Application Data
 130 2.057059  216.239.34.157  192.168.20.4    TCP        60 443 → 53850 [ACK] Seq=5787 Ack=2156 Win=267776 Len=0
 188 2.082555  216.239.34.157  192.168.20.4    TLSv1.3   995 Application Data, Application Data
 189 2.082703  192.168.20.4    216.239.34.157  TLSv1.3    85 Application Data
 190 2.083076  216.239.34.157  192.168.20.4    TLSv1.3    98 Application Data
 228 2.129371  192.168.20.4    216.239.34.157  TCP        54 53850 → 443 [ACK] Seq=2187 Ack=6772 Win=261632 Len=0
 236 2.147043  216.239.34.157  192.168.20.4    TCP        60 443 → 53850 [ACK] Seq=6772 Ack=2187 Win=267776 Len=0
 240 2.175174  216.239.34.157  192.168.20.4    TLSv1.3   133 Application Data
 241 2.175556  192.168.20.4    216.239.34.157  TLSv1.3  1809 Application Data
 260 2.235464  216.239.34.157  192.168.20.4    TCP        60 443 → 53850 [ACK] Seq=6851 Ack=3942 Win=266240 Len=0
 388 2.350809  216.239.34.157  192.168.20.4    TLSv1.3  1466 Application Data

> Frame 35: 1848 bytes on wire (14784 bits), 1848 bytes captured (14784 bits) on interface \Device\NPF_{92708C19-2217-4AE2-8544-098ACC90D322}, id 0
> Ethernet II, Src: MicroStarINT_cb:4b:e9 (d8:bb:c1:cb:4b:e9), Dst: Secuicom_23:30:19 (00:05:66:23:30:19)
> Internet Protocol Version 4, Src: 192.168.20.4, Dst: 216.239.34.157
> Transmission Control Protocol, Src Port: 53850, Dst Port: 443, Seq: 1, Ack: 1, Len: 1794
v Transport Layer Security
  v TLSv1.3 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 1789
    v Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 1785
      > Version: TLS 1.2 (0x0303)
        Random: 7dd01bb0d9522f467d3d52e1f06ca5590cfcaed798802644d64d38cdafeb6180
        Session ID Length: 32
        Session ID: e3996c34fc9a344294fc26b2f4effb929491d030b8b14fbc6f4650d3db8c6641
        Cipher Suites Length: 32
      > Cipher Suites (16 suites)
        Compression Methods Length: 1
      > Compression Methods (1 method)
        Extensions Length: 1680
      > Extension: Reserved (GREASE) (len=0)
      > Extension: server_name (len=25) name=tunnel.googlezip.net
      > Extension: renegotiation_info (len=1)
      > Extension: signed_certificate_timestamp (len=0)
      > Extension: ec_point_formats (len=2)
      > Extension: supported_versions (len=7) TLS 1.3, TLS 1.2
```

- chrome 사용시 TLS1.3 버전 프로토콜을 사용
- 목적지 서버의 포트번호는 443 번, 클라이언트의 포트번호는 53850 번임
- 처음 연결 수립시 TCP 3 way handshake 과정을 거친 후 client hello, server hello 를 주고받음
- 처음 클라이언트는 Client Hello, Change Cipher Spec 패킷(seq : 1, ack : 1)을 보내고, 이에 대한 응답으로 서버는 Server Hello(seq : 1, ack : 1795)를 전송
- 서버가 클라이언트에게 Application Data(seq : 5649, ack : 1795) 를 한 번 보낸 뒤 클라이언트가 서버에게 Change Cipher Spec 패킷(seq : 1795, ack : 5787) 전송(Change Cipher Spec : 클라이언트와 서버가 협상한 암호 명세를 이후부터 적용 혹은 변경함을 알림)
- 그 이후 Application Data 를 서버와 클라이언트가 주고받음

- **edge 브라우저에서 네이버에서 “라우팅” 검색**



- **edge 브라우저에서 네이버에서 "라우팅" 검색 시 캡처한 TCP 패킷**

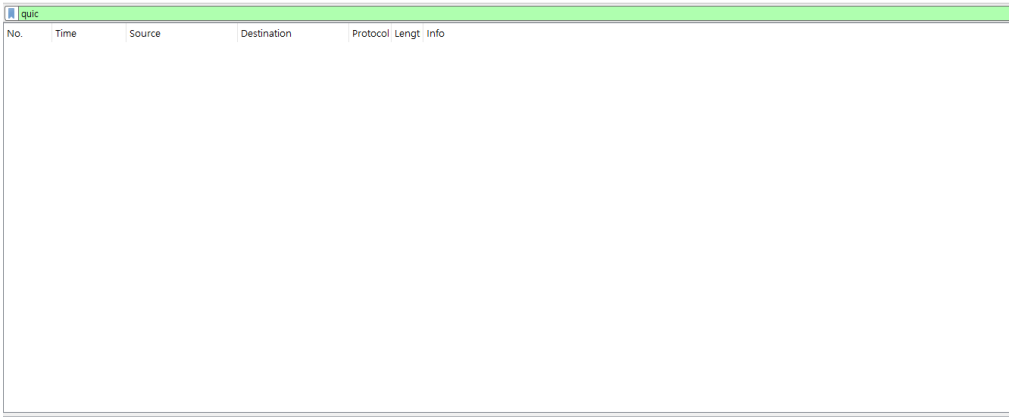


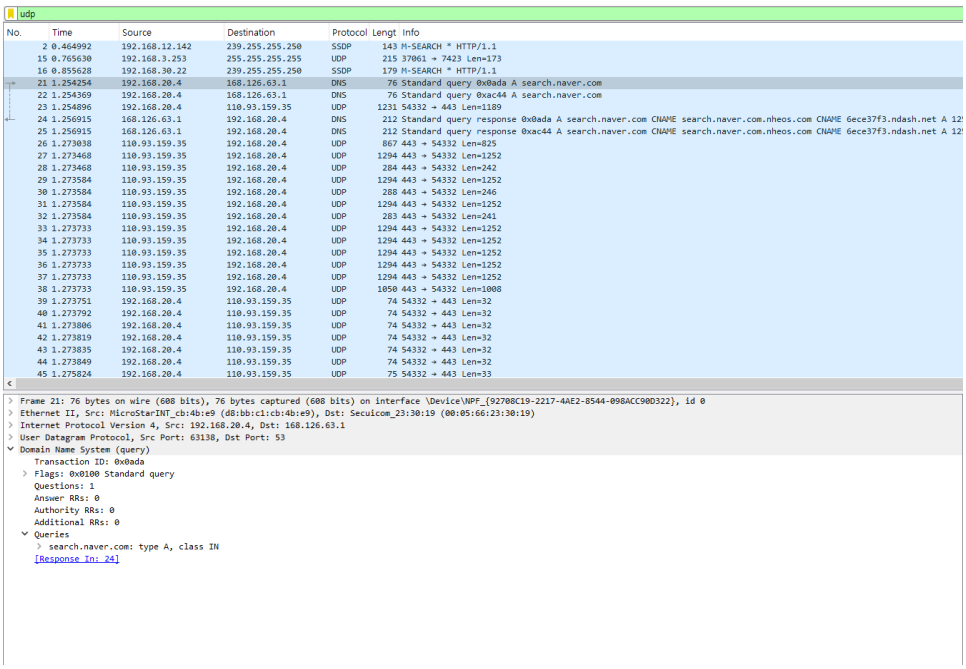
- 네이버에서는 TLS1.2 버전 프로토콜을 사용
- 클라이언트(192.168.20.4)와 서버(223.130.195.167)가 TCP 3-way handshake 수행 후 데이터를 주고받음
- 목적지 서버의 포트번호는 443 번, 클라이언트의 포트번호는 54093 번임
- 3 way hanshake 과정에서 처음 SYN 을 보낼 때는 seq 번호가 0 임
- 이에 대한 응답으로 서버가 클라이언트에게 syn + ack 를 보낼 때는 seq 번호가 0, ack 번호가 1 임
- 다시 클라이언트가 서버에게 ack 를 보낼 때는 seq 번호가 1, ack 번호가 1 임
- 구글에서 검색했을 때와 마찬가지로 3 way handhake 과정을 거친 후 TLS handshake 과정을 거침
- client hello 패킷에 접속하려는 도메인 이름(static.nid.naver.com)이 표시됨
- client hello, server hello 패킷을 주고받은 후 서버가 클라이언트에게 Certificate, Server Key Excahnge, Server Hello Done(seq : 4381, ack : 1923) 전송
    - Certificate : 서버의 인증서와 서버 인증서에 서명한 인증기관들의 인증서 목록을 클라이언트에게 전달
    - Server Key Exchange : 키 교환에 필요한 정보를 전달, ex) Diffie-Hellman 매개변수

    - Server Hello Done : Server Hello 과정을 종료함을 알림
- 클라이언트는 서버에게 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message(seq : 1923, ack : 5282) 전송
    - Client Key Exchange : 서버의 인증서를 검증한 후 앞으로 사용할 세션키 생성을 위한 pre-master secret 을 서버 인증서의 공개키로 암호화하여 전송, 혹은 Diffie-Hellman 공개키를 생성하고 서버에 전송한 후 클라이언트와 서버가 각각 Diffie-Hellman 연산을 통해 공통의 pre-master secret 생성
    - Encrypted Handshake Message : 클라이언트 측의 협상을 종료

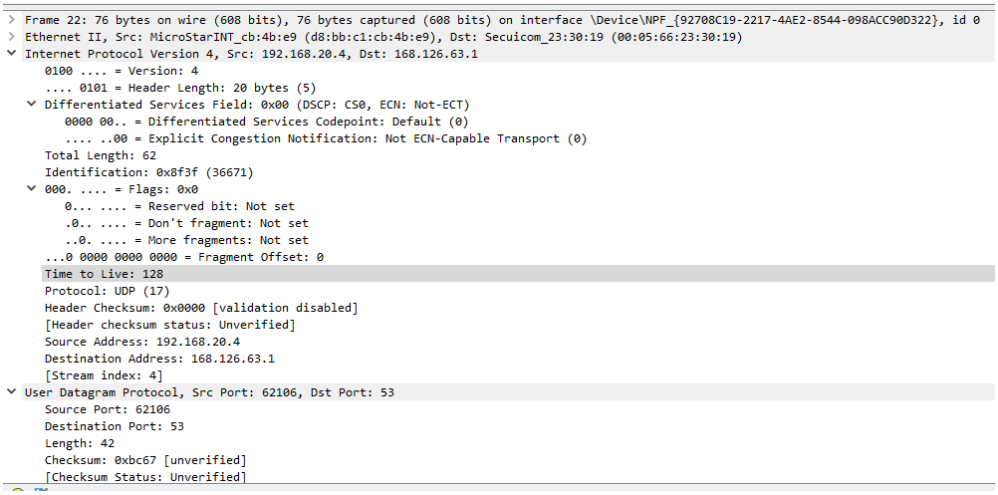
4. UDP 패킷 분석

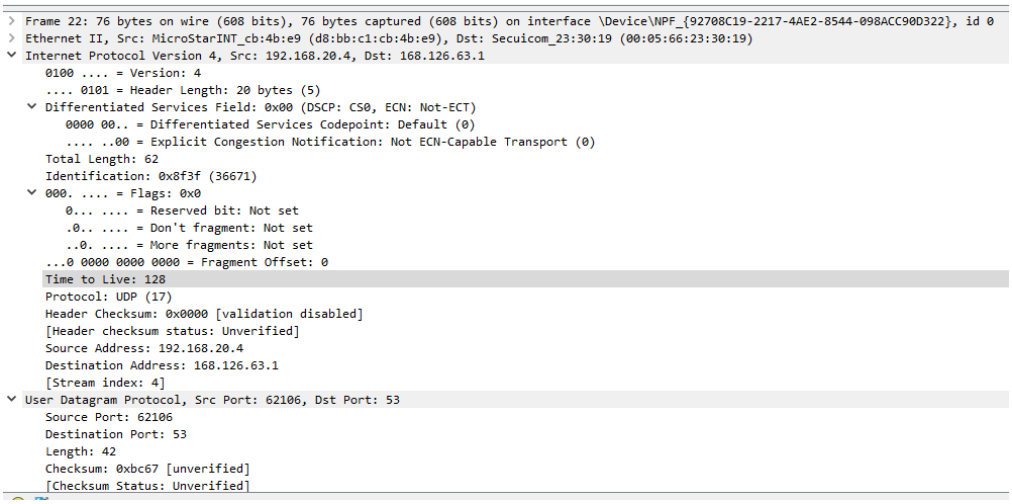
- **edge 브라우저에서 네이버에서 “라우팅” 검색 시 캡처한 UDP 패킷**

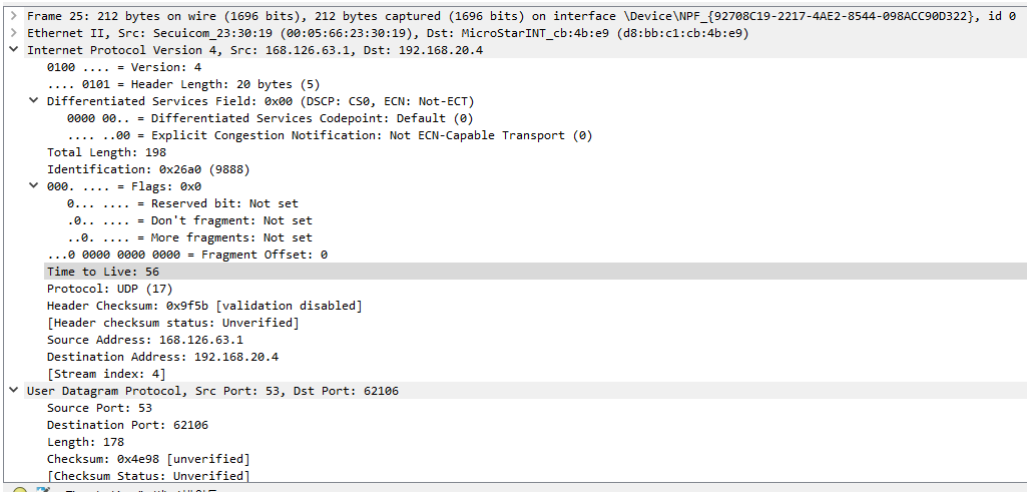




- DNS 서비스 이용 시 UDP 통신이 사용됨
- 초기에 클라이언트(192.168.20.4)가 DNS 서버(168.126.63.1)에게 DNS 요청을 보내고 서버로부터 수신받음
- 서버로 전송하는 4 개의 패킷에서 목적지 포트번호는 53 번이고, 출발지 포트번호는 각각 다른 번호임(63138, 62106)
- search.naver.com 에 대해 질의하였음
- UDP 헤더의 필드 크기는 출발지 포트번호와 목적지 포트번호가 각각 2 바이트, 데이터그램 크기가 2 바이트, checksum 이 2 바이트임
- UDP 데이터그램의 총 크기는 42 바이트이며 payload(헤더를 제외한 데이터)의 크기는 34 바이트임







- 클라이언트가 서버에게 요청을 보내는 경우 TTL 값은 128(윈도우 운영체제의 기본값)임
- 서버에서 클라이언트에게 보낸 패킷의 TTL 값은 56 임
- 서버에서 클라이언트에게 보낸 패킷 중 63138 번 포트로 보낸 패킷은 don't fragment 가 설정되어 있으며 62106 번 포트로 보낸 패킷은 설정되어 있지 않음