

모바일 컴퓨팅 HW8

2018037356 안동현

수업시간에 배운 SIP, STUN의 작동원리를 기반으로 다음 trace file을 분석하세요. SIP, STUN과 연계된 각 packet을 살펴보고 어떤 과정을 거쳤는지 문서로 작성해서 업로드 하면 됩니다. 구체적 일수록 좋습니다.

먼저 간략하게 STUN과 SIP를 통해서 연결을 맺는 과정을 설명하자면,

STUN 서버는 요청시 현재 내 머신의 public 주소를 알려줍니다. 이후 이를 이용해서 SIP 서버를 통해 반대편 클라이언트와 연결을 맺고 VOIP 통신을 하다가, 통신을 끝낼 때 다시 SIP 서버를 통해 끝을 맺는 과정입니다. 그 과정에 좀더 디테일한 과정이 존재하고, 패킷의 흐름을 통해서 더욱 자세하게 알아보겠습니다.

A와 B가 서로 통신을 하고싶다고 가정하겠습니다.

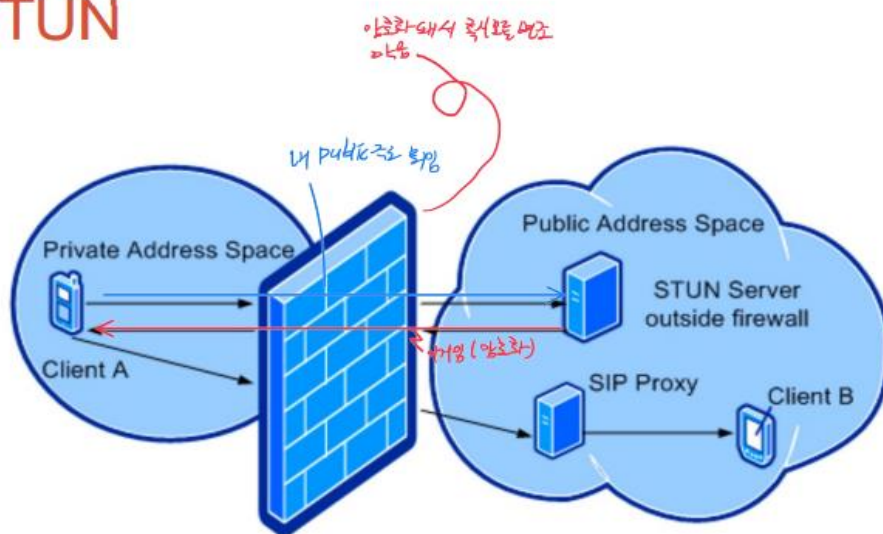
A = 사용자

B = 상대방

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.17.204.164	217.10.68.152	STUN	62	Binding Request
2	0.015431	217.10.68.152	10.17.204.164	STUN	130	Binding Success Response MAPPED-ADDRESS: 194.95.84.197:5062 XOR-MAPPED-ADDRESS: 194.95.84.197:5062
3	0.030154	10.17.204.164	217.10.68.152	STUN	62	Binding Request
4	0.041734	217.10.68.152	10.17.204.164	STUN	130	Binding Success Response MAPPED-ADDRESS: 194.95.84.197:5063 XOR-MAPPED-ADDRESS: 194.95.84.197:5063

가장 먼저 보이는 패킷들은 STUN 서버에 요청을 보내고 답을 받는 과정입니다. NAT을 통해 할당된 A의 private IP 주소 10.17.204.164 에서 STUN 서버인 217.10.68.152로 자신의 public IP 주소를 묻습니다.

STUN



이 과정에서 STUN서버로부터 STUN 메시지의 매직 쿠키값과 XOR 돼서 암호화된 형태로

public IP와 port 번호를 알아냅니다. A는 다시한번 XOR과정을 거쳐 주소들을 복호화하여 원래의 public 주소들을 얻어냅니다.

```
Message Length: 00
Message Cookie: 2112a442
.....
MAPPED-ADDRESS: 194.95.84.197:5062
  > Attribute Type: MAPPED-ADDRESS
  Attribute Length: 8
  Reserved: 00
  Protocol Family: IPv4 (0x01)
  Port: 5062
  IP: 194.95.84.197
  .....
XOR-MAPPED-ADDRESS: 194.95.84.197:5062
  > Attribute Type: XOR-MAPPED-ADDRESS
  Attribute Length: 8
  Reserved: 00
  Protocol Family: IPv4 (0x01)
  Port (XOR-d): 32d4
  [Port: 5062]
  IP (XOR-d): e34df087
  [IP: 194.95.84.197]
```

이렇게 해서 알아낸 값은 IP 194.95.84.197에 port번호 5062 입니다.

이후에 또 STUN서버를 통해 요청하고, IP 194.95.84.197에 port번호 5063을 알아내지만, 이후의 패킷들에게서 보이지 않는 것을 보아 쓰이지 않는 것 같고, src 주소가 10.17.204.164로 같았다는 점을 통해, 같은 머신에서 다른 어플리케이션에서 잠시 요청을 보냈지 않았나 하는 추측을 했습니다.

이제 SIP를 통해서 B와 연결을 맺기 위해 INVITE 메시지를 날려줍니다. 이때 목적지 주소가 217.10.79.9인 것과 밑의 연결된 후 UDP 통신에서

14	0.363775	10.17.204.164	217.10.79.30	UDP
15	0.366801	217.10.79.30	10.17.204.164	UDP

이렇게 217.10.79.30의 주소로 다이렉트로 보내는 것을 보아하니, IP 217.10.79.9 주소는 A와 B가 연결을 맺기 위한 SIP Proxy 서버라고 생각되고, B의 주소는 IP 217.10.79.30이라고 추측됩니다.

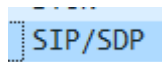
SIP는 연결과 끝맺음 만을 담당해주므로 이러한 일이 발생하는 것으로 생각됩니다.

5번 패킷 INVITE를 계속해서 보겠습니다.

```
Request-Line: INVITE sip:10000@sipgate.de SIP/2.0
Method: INVITE
  Request-URI: sip:10000@sipgate.de
    Request-URI User Part: 10000
    Request-URI Host Part: sipgate.de
  [Resent Packet: False]
Message Header
  > Via: SIP/2.0/UDP 194.95.84.197:5060;branch=z9hG4bK00eb593dc115e111b2ab0022fa24bf98;rport
  > From: "PhonerLite" <sip:8002424@sipgate.de>;tag=3818623237
  > To: <sip:10000@sipgate.de>
  Call-ID: 00EB593D-C115-E111-B2AA-0022FA24BF98@194.95.84.197
  [Generated Call-ID: 00EB593D-C115-E111-B2AA-0022FA24BF98@194.95.84.197]
  > CSeq: 6 INVITE
  > Contact: <sip:8002424@194.95.84.197:5060>
  Content-Type: application/sdp
  Allow: INVITE, OPTIONS, ACK, BYE, CANCEL, INFO, NOTIFY, MESSAGE, UPDATE
  Max-Forwards: 70
  Supported: 100rel, replaces, from-change
  User-Agent: SIPPER for PhonerLite
  > P-Preferred-Identity: <sip:8002424@sipgate.de>
  Content-Length: 416
```

해당 패킷은 SIP Proxy 서버를 향하지만, Request URI는 B의 것을 나타냅니다. 10000@sipgate.de가 그것을 의미합니다.

Via 부분을 보아, 194.95.84.197 즉 A의 public IP를 확인할 수 있고, From과 To는 SIP 연결의 첫 시도 A -> B를 나타내므로 8002424@sipgate.de는 A의 URI를 나타낸다고 할 수 있습니다.



```
Content-Length: 416
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      > Owner/Creator, Session Id (o): - 2602958595 0 IN IP4 194.95.84.197
      Session Name (s): SIPPER for PhonerLite
      > Connection Information (c): IN IP4 194.95.84.197
      > Time Description, active time (t): 0 0
      > Media Description, name and address (m): audio 5062 RTP/AVP 8 0 2 3 97 110
      > Media Attribute (a): rtpmap:8 PCMA/8000
      > Media Attribute (a): rtpmap:0 PCMU/8000
      > Media Attribute (a): rtpmap:2 G726-32/8000
      > Media Attribute (a): rtpmap:3 GSM/8000
      > Media Attribute (a): rtpmap:97 iLBC/8000
      > Media Attribute (a): rtpmap:110 speex/8000
      > Media Attribute (a): rtpmap:111 speex/16000
      > Media Attribute (a): rtpmap:9 G722/8000
      > Media Attribute (a): rtpmap:101 telephone-event/8000
      > Media Attribute (a): fmp:101 0-16
      > Media Attribute (a): ssrc:2510573407
      Media Attribute (a): sendrecv
      [Generated Call-ID: 00EB593D-C115-E111-B2AA-0022FA24BF98@194.95.84.197]
```

메시지에는 이렇게 SDP 요청까지 부탁드립니다.

이렇게 INVITE를 살펴 보았고, 이 이전에 REGISTER 요청이 없었던 것을 보면 A는 이전에 이미 SIP에 등록 과정을 거쳤었다는 것을 추측할 수 있습니다.

그런데 이 이후 6번 패킷은

217.10.79.9 -> 10.17.204.164 로 407 상태 코드를 답장으로 쏘았습니다.

```
▼ Status-Line: SIP/2.0 407 Proxy Authentication Required
  Status-Code: 407
  [Resent Packet: False]
  [Request Frame: 5]
  [Response Time (ms): 11]
▼ Message Header
```

1xx	Informational
2xx	Final
3xx	Redirection
4xx	Client Error
5xx	Server Error
6xx	Global Failure

Example: 404 Not Found

Responder

강의 자료에 근거해서 클라이언트 에러로, 연결에 실패했음을 알 수 있습니다. 상태 라인에 근거해서 Proxy 서버의 인증에 문제가 있었던 것 같습니다.

7번 패킷은 그것에 대한 ACK를 쳐줌으로 다시 연결 시도 전으로 돌아갔음을 추측할 수 있습니다. 이때 A -> Proxy 쪽으로 ACK를 날린 것 이므로, INVITE와 패킷의 SIP 부분이 크게 다르지 않습니다.

이제 다시 8번 패킷부터 A -> Proxy로 INVITE 리퀘스트를 날립니다.

```

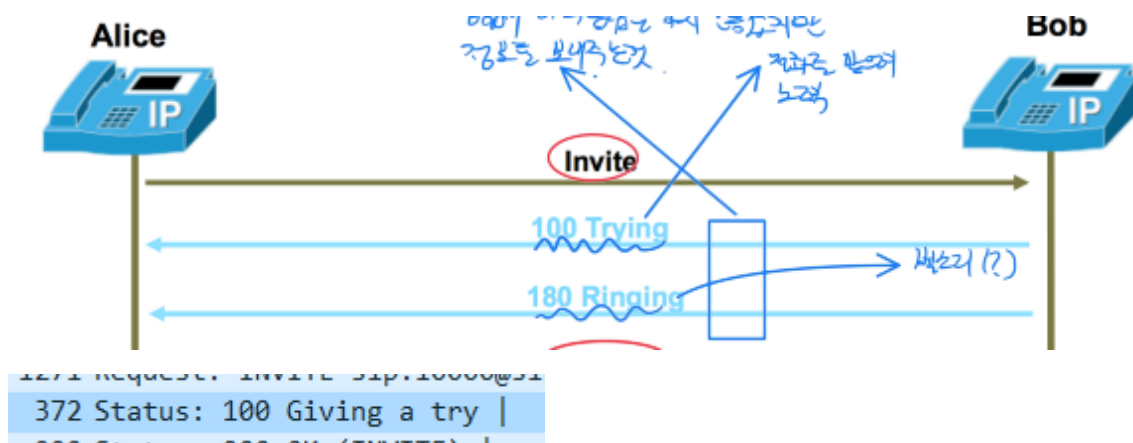
v Message Header
> Via: SIP/2.0/UDP 194.95.84.197:5060;branch=z9hG4bK00eb593dc115e111b2ac0022fa24bf98
> From: "PhonerLite" <sip:8002424@sipgate.de>;tag=3818623237
> To: <sip:10000@sipgate.de>
  Call-ID: 00EB593D-C115-E111-B2AA-0022FA24BF98@194.95.84.197
  [Generated Call-ID: 00EB593D-C115-E111-B2AA-0022FA24BF98@194.95.84.197]
> CSeq: 7 INVITE
> Contact: <sip:8002424@194.95.84.197:5060>
> Proxy-Authorization: Digest username="8002424", realm="sipgate.de", nonce="4ecf75b
  Content-Type: application/sdp
  Allow: INVITE, OPTIONS, ACK, BYE, CANCEL, INFO, NOTIFY, MESSAGE, UPDATE
  Max-Forwards: 70
  Supported: 100rel, replaces, from-change
  User-Agent: SIPPER for PhonerLite
> P-Preferred-Identity: <sip:8002424@sipgate.de>
  Content-Length: 416

```

여기서

Proxy-Authorization: Digest username="8002424", realm="sipgate.de",
 nonce="4ecf75b84949fb1f919542798bc1bbdd38e32b81", uri="sip:10000@sipgate.de",
 response="7eab72535014985b6115bbb9a1cc8f28", algorithm=MD5 이 부분을 보아하니 인증에
 문제가 없었던 것으로 보이고 그 결과,

9번 패킷은



이렇게 proxy -> A로 연결을 받기 위해서 노력중이라는 응답을 보내줍니다.

```
Status: 200 OK (INVITE) |
Request: ACK id=100000217
```

OK를 받아내고, 해당 패킷에는 RR 헤더 역시 발견할 수 있었습니다.

UAC P1 P2 P3 UAS

--REQ-->

--REQ--> Record-Route: P1

--REQ--> Record-Route: P2, P1

--REQ--> Record-Route: P3, P2, P1

--RSP--- Record-Route: P3, P2, P1
Contact: UAS

--RSP--- Record-Route: P3, P2, P1
Contact: UAS

--RSP--- Record-Route: P3, P2, P1
Contact: UAS

--REQ--> Route: P2, P3, UAS

--REQ--> Route: P3, UAS

--REQ--> Route: UAS

--REQ-->

Invite

RR

RR

ok

Trace

Ack

route

해당 자료를 보면 쉽게 이해를 할 수 있습니다.

또한, 이렇게 요청했던 SDP 정보들까지 채워서 같이 보내줬습니다.

```

Message Body
  Session Description Protocol
    Session Description Protocol Version (v): 0
    > Owner/Creator, Session Id (o): root 15126 15126 IN IP4 217.10.79.30
    Session Name (s): session
    > Connection Information (c): IN IP4 217.10.79.30
    > Time Description, active time (t): 0 0
    > Media Description, name and address (m): audio 17750 RTP/AVP 8 0 3 97 2 101
    > Media Attribute (a): rtpmap:8 PCMA/8000
    > Media Attribute (a): rtpmap:0 PCMU/8000
    > Media Attribute (a): rtpmap:3 GSM/8000
    > Media Attribute (a): rtpmap:97 iLBC/8000
    > Media Attribute (a): fmtp:97 mode=30
    > Media Attribute (a): rtpmap:2 G726-32/8000
    > Media Attribute (a): rtpmap:101 telephone-event/8000
    > Media Attribute (a): fmtp:101 0-16
    > Media Attribute (a): silenceSupp:off - - - -
    > Media Attribute (a):ptime:20
    Media Attribute (a): sendrecv
    [Generated Call-ID: 00EB593D-C115-E111-B2AA-0022FA24BF98@194.95.84.197]

```

이제 11번 패킷이

```

712 Request: ACK sip:10000@217.10.79.3

```

이렇게 ACK를 쳐주고 연결이 완성되었음을 알립니다. 여기서는 당연히 SDP가 필요 없으므로 메시지 안에서는 찾아볼 수 없었습니다.

```

Message Header
  > Via: SIP/2.0/UDP 194.95.84.197:5060;branch=z9hG4bK8081f23dc115e111b2ac0022fa24bf98;rport
  > Route: <sip:217.10.79.9;lr;ftag=3818623237>
    > Route URI: sip:217.10.79.9;lr;ftag=3818623237
      Route Host Part: 217.10.79.9
      Route URI parameter: lr
      Route URI parameter: ftag=3818623237
  > Route: <sip:172.20.40.1;lr=on>
    > Route URI: sip:172.20.40.1;lr=on
      Route Host Part: 172.20.40.1
      Route URI parameter: lr=on

```

하지만 Route 헤더는 찾아볼 수 있었습니다.

다음 12번 패킷은 RTCP로 실시간 전송 프로토콜을 나타냅니다. 찾아보니, RTP 세션의 품질 제어

를 위한 별도의 제어용 프로토콜로 세션 참여자들이 주기적으로 패킷 지연, 손실 지터등의 정보를 주고받는 용이라고 하고, 원래 주기적으로 서로 교환하지만 해당 통신 세션에서는 한번만 보였습니다.

RTCP	102	Sender Report	Source description
------	-----	---------------	--------------------

설명과 출발지, 목적지를 보아하니 연결이 완료된 후, 출발지의 현 상황을 알리기 위해서 보낸 패킷으로 추측됩니다.

알고있는 지식 상으로는 이제 연결이 완료됐으니 UDP를 통해서 A(10.17.204.164, public 194.95.84.197) 와 B(217.10.79.30)가 직접 통신을 시작해야 했지만, 또다시 이번에는 A -> B로 STUN 프로토콜 요청을 하는 것을 확인 할 수 있었습니다.

13 0.363652	10.17.204.164	217.10.79.30	STUN	62 Binding Request
-------------	---------------	--------------	------	--------------------

이게 대체 뭔지 고민하다가, 먼저 A와 기존의 STUN 서버간이 아닌 A와 B 사이인 점과 이후 답장인 16번 패킷에서

Binding Success Response MAPPED-ADDRESS: 194.95.84.197:5062

이전 답장과는 다르게 쿠키값을 통해서 XOR 암호화한 결과를 포함하지 않고 알려주는 것을 미루어 보면, 먼저 B는 A의 주소를 알고 있었다는 의미이고 암호화할 필요가 없음을 알려줍니다. 이는 A 역시 자신의 public IP를 이미 알고있음 역시 의미하고 결국 A가 B에게 "정말로 내 IP를 알고 있고 너와 연결된 거야?" 라는 것을 확인하고 싶었다. 라고 추측됩니다.

암호화가 되어있지 않았다는 것은 해당 주소가 변조가 되어도 상관이 없었다는 의미로 생각되고, A는 이미 자신의 public 주소를 알고있기에, 만약 B에서 답장이 온 주소값이 A가 알고 있는 것과 다르다면 연결을 빠르게 해제하는 것이 가능하기 때문이라고 생각합니다.

17 0.382194	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
18 0.386094	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
19 0.402200	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
20 0.403903	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
21 0.422212	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
22 0.442193	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
23 0.462225	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
24 0.482500	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
25 0.502820	10.17.204.164	217.10.79.30	UDP	214 5062 → 17750 Len=172
26 0.512073	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
27 0.514049	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
28 0.515152	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
29 0.515155	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172

이후 244번 패킷까지는 A와 B가 UDP로 서로 통신하는 패킷들로 구성되어 있고,

245번 패킷에서 A -> Proxy 서버로(정확히는 B를 향하며 Proxy를 거쳐갑니다.)

747 Request: BYE sip:10000@217.10.79.30 |

BYE 메시지를 알려줍니다. 연결을 종료하겠다는 것을 의미하고, 보내는 쪽 이므로

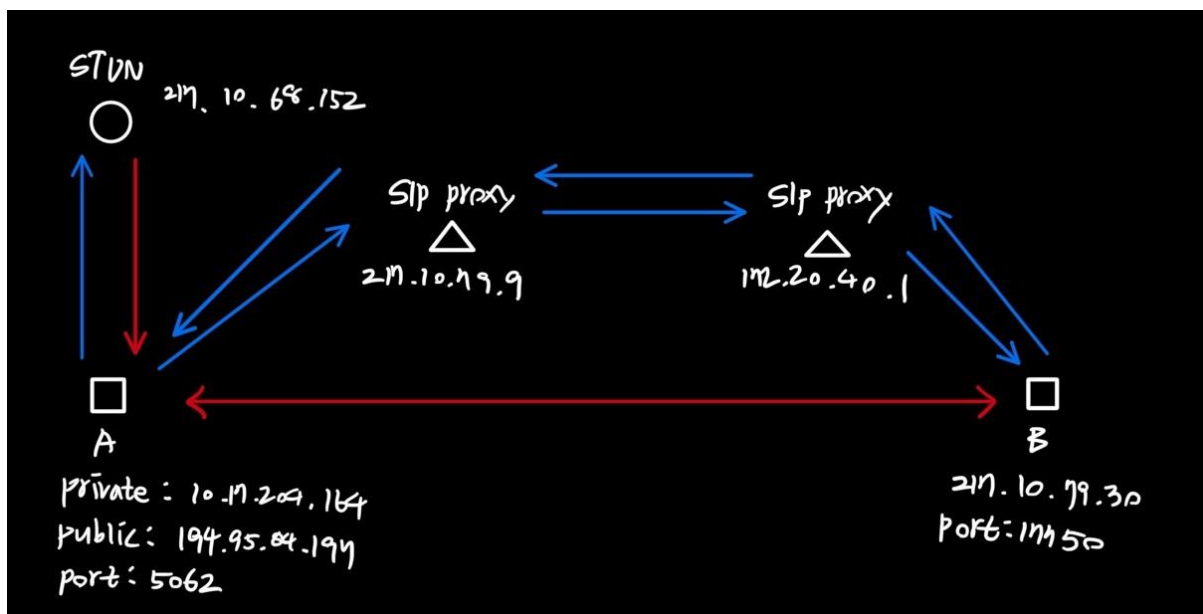
Route 헤더가 존재함을 확인할 수 있었습니다.

```
[Kesent Packet: False]
▼ Message Header
  > Via: SIP/2.0/UDP 194.95.84.197:5060;branch=z9hG4bK80ae233fc115e111b2ac0022fa24bf98;rport
  ▼ Route: <sip:217.10.79.9;lr;ftag=3818623237>
    ▼ Route URI: sip:217.10.79.9;lr;ftag=3818623237
      Route Host Part: 217.10.79.9
      Route URI parameter: lr
      Route URI parameter: ftag=3818623237
    ▼ Route: <sip:172.20.40.1;lr=on>
      ▼ Route URI: sip:172.20.40.1;lr=on
        Route Host Part: 172.20.40.1
        Route URI parameter: lr=on
```

246	2.763875	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
247	2.872514	217.10.79.30	10.17.204.164	UDP	214 17750 → 5062 Len=172
248	2.872722	217.10.79.9	10.17.204.164	SIP	469 Status: 200 OK (BYE)

이제 못 도착한 UDP 패킷들이 들어오고, 마지막으로 BYE에 대한 OK 메시지가 들어오면서 연결을 끝냅니다.

마지막으로 그림으로 정리를 해보자면 결국



이런 연결 관계라고 생각합니다.

느낀점

과제를 진행하면서 재미있던 점은 먼저 패킷들을 트레이스 해가며 머릿속에 그림을 그리는 것이었습니다. 그 과정에서 SIP와 NAT에 대해서 더 깊게 이해한 기분이고 어려웠던 점은 강의에서는 본적이 없었던 것 같은 패킷들이 나왔을 때입니다. 주변의 상황을 잘 보고 추측한 결과밖에 도출하지는 못했지만, 그 과정이 저를 더욱 더 이해하게 만들었다고 생각합니다.