

파이썬을 이용한 UDP, TCP 통신 연산서버 개발 및 와이어샤크를 이용한 패킷분석

해당 프로젝트를 구현하기 앞서 곱, 나눗셈 연산서버와 덧셈, 뺄셈 연산서버를 분리하라 지시하였으나, 파이썬의 eval 함수를 이용해 연산을 구현하였으므로 각 연산서버와 클라이언트의 코드 내용은 같음 또한 해당 서버는 단순 연산과제를 제출하기 위함이기때문에 eval함수의 보안적 취약점은 따로 시큐어코딩하지않았음

PDServer.py (UDP 프로토콜 연산 서버 소스코드)

```
# 통신을 위한 socket 모듈 import
import socket
# udp 소켓을 생성
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# host 의 port 번호로 서버 실행
sock.bind(("127.0.0.1", 4000))
while True:
    # data 를 data 변수에, 클라이언트의 정보를 addr 변수에 저장
    data, addr = sock.recvfrom(1024)
    # data 변수에 저장된 값을 디코딩하여 계산한 뒤 str 로 형변환하여 data 에 재저장
    data = str(eval(data.decode()))
    # data 변수에 저장된 값을 클라이언트로 전송
    sock.sendto(data.encode(), addr)
# 소켓 종료
sock.close()
```

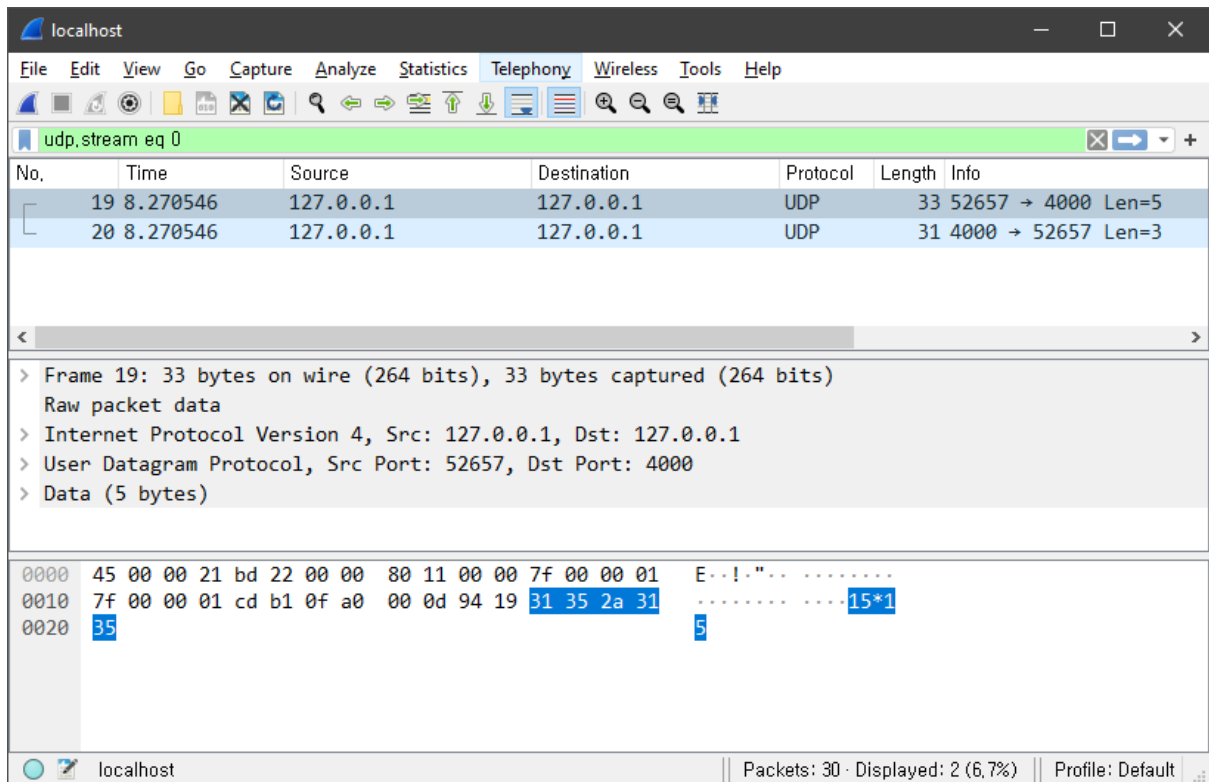
PDClient.py (UDP 프로토콜 연산 클라이언트 소스코드)

```
# 통신을 위한 socket 모듈 import
import socket
# 사용할 udp 소켓 생성
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# HOST 의 PORT 로 소켓 접속 시도
sock.connect(('127.0.0.1', 4000))
while True:
    # 계산식을 입력받고 msg 변수에 저장
    msg = input("->")
    # 연결된 소켓으로 msg 를 인코딩하여 서버로 전송
    sock.sendto(msg.encode(), ("127.0.0.1", 4000))
    # 서버로부터 받은 데이터를 data 변수에 저장
    recvMsg, addr = sock.recvfrom(2048)
    # data 변수의 값을 디코딩하여 출력
    print(recvMsg.decode())
# 소켓 접속 할당 해제
sock.close()
```

UDP 프로토콜을 이용한 서버 & 클라이언트의 패킷 송수신 내용

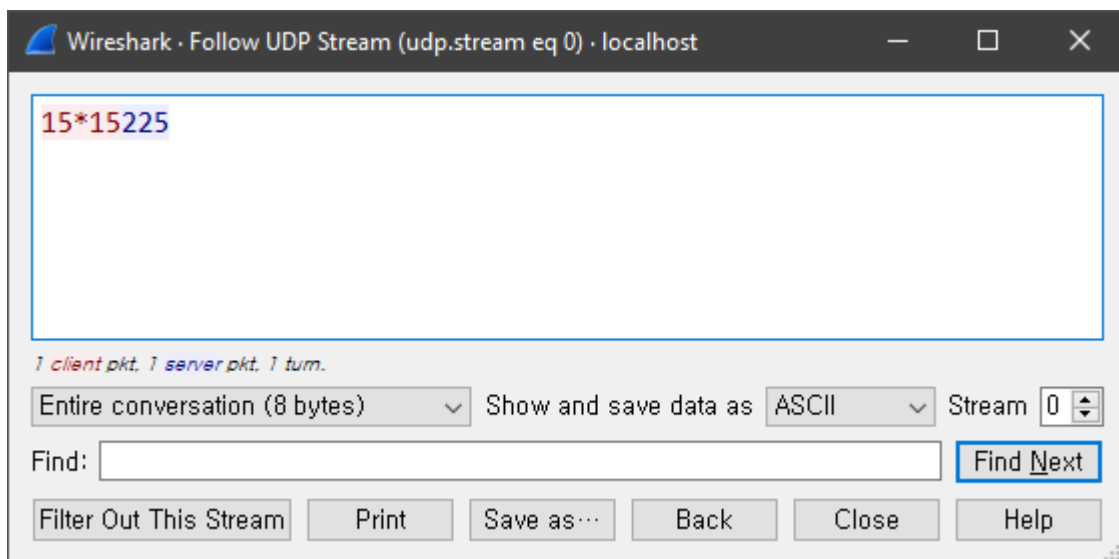
Server HOST : 127.0.0.1(Loofback) Server PORT : 4000, Client PORT : 52657

아래 이미지는 전체적인 패킷의 송수신 내용임



아래 이미지는 클라이언트와 서버의 통신내용임

15*15라는 내용을 클라이언트에서 서버로 전송하였으며, 서버는 225라는 연산결과를 클라이언트로 송신함



PDServer.py (TCP 프로토콜 연산 서버 소스코드)

```
# 통신을 위한 socket 모듈 import
import socket
# 사용할 tcp 소켓 생성
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 소켓에 Host 의 포트를 바인딩하고 TCP 연결 요청 기다림
sock.bind(('127.0.0.1', 4000))
# 클라이언트 접속 대기
sock.listen()
# 클라이언트 접속시 con, addr 에 클라이언트의 접속 정보를 저장
con, addr = sock.accept()
while True:
    # data 변수에 클라이언트로부터 전송받은 데이터를 디코딩하여 저장
    data = con.recv(1024).decode()
    # 전달받은 데이터가 존재하지않으면
    if not data:
        # 루프를 종료하고 23 번째 라인으로 이동 (sock.close()) << 맨 마지막줄
        break
    # data 변수에 이전의 data 값을 eval 함수로 계산하여 data 변수에 저장
    data = eval(data)
    # data 변수에 저장된 값을 str(문자형 데이터)로 형변환하고 인코딩하여 전송
    con.send(str(data).encode())
# 소켓 서버 할당 종료
sock.close()
```

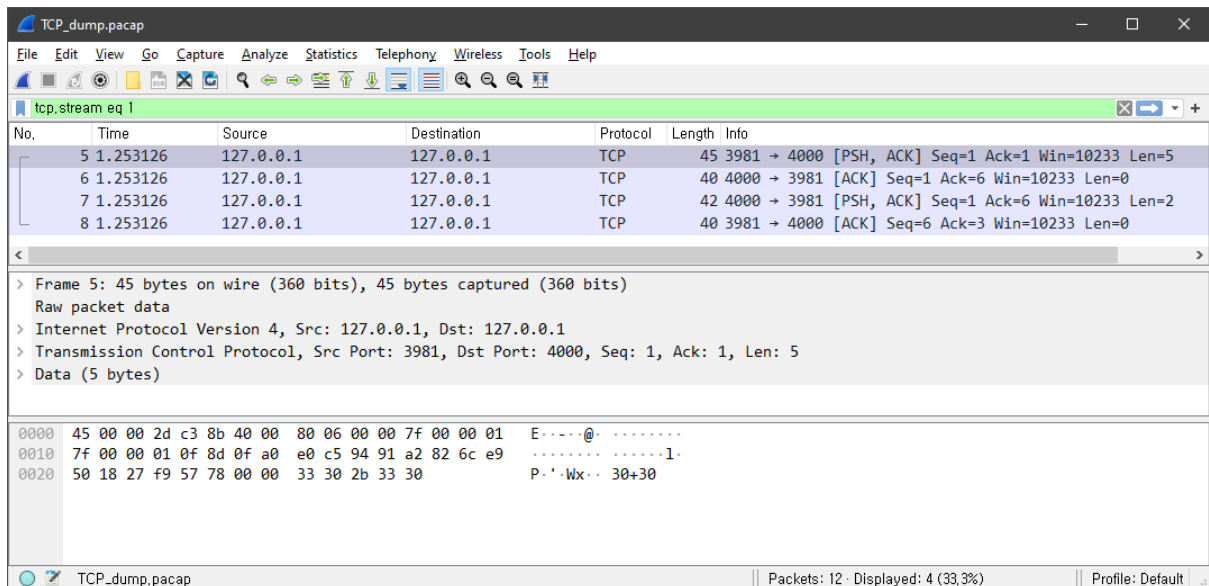
PDClient.py (TCP 프로토콜 연산 클라이언트 소스코드)

```
# 통신을 위한 socket 모듈 import
import socket
# 사용할 tcp 소켓 생성
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# HOST 의 PORT 로 소켓 접속 시도
sock.connect(('127.0.0.1', 4000))
while True:
    # 계산식을 입력받고 msg 변수에 저장
    msg = input(">")
    # 연결된 소켓으로 msg 를 인코딩하여 서버로 전송
    sock.send(msg.encode())
    # 서버로부터 받은 데이터를 data 변수에 저장
    data = sock.recv(1024)
    # data 변수의 값을 디코딩하여 출력
    print(data.decode())
# 소켓 접속 할당 해제
sock.close()
```

TCP 프로토콜을 이용한 서버 & 클라이언트의 패킷 송수신 내용

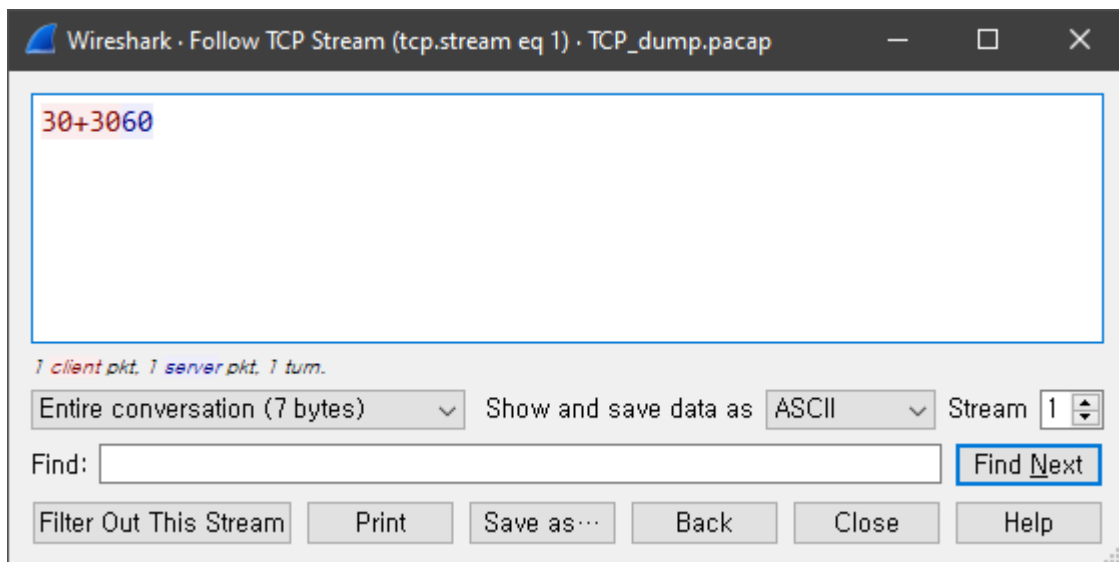
Server HOST : 127.0.0.1(Loofback) Server PORT : 4000, Client PORT : 3981

아래 이미지는 전체적인 패킷의 송수신 내용임



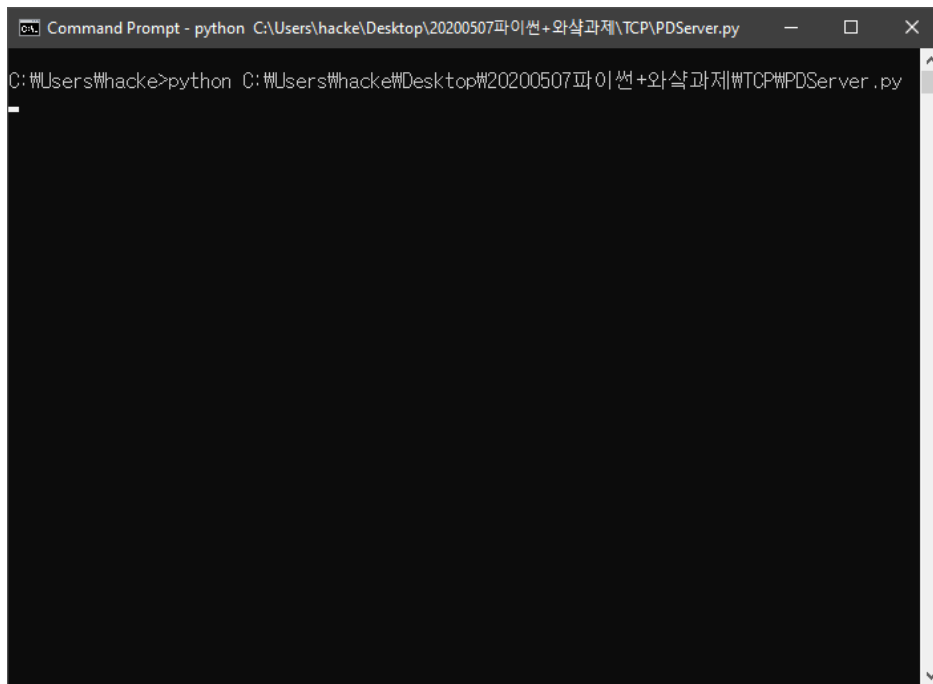
아래 이미지는 클라이언트와 서버의 통신내용임

30+30이라는 내용을 클라이언트에서 서버로 전송하였으며, 서버는 60이라는 연산결과를 클라이언트로 송신함



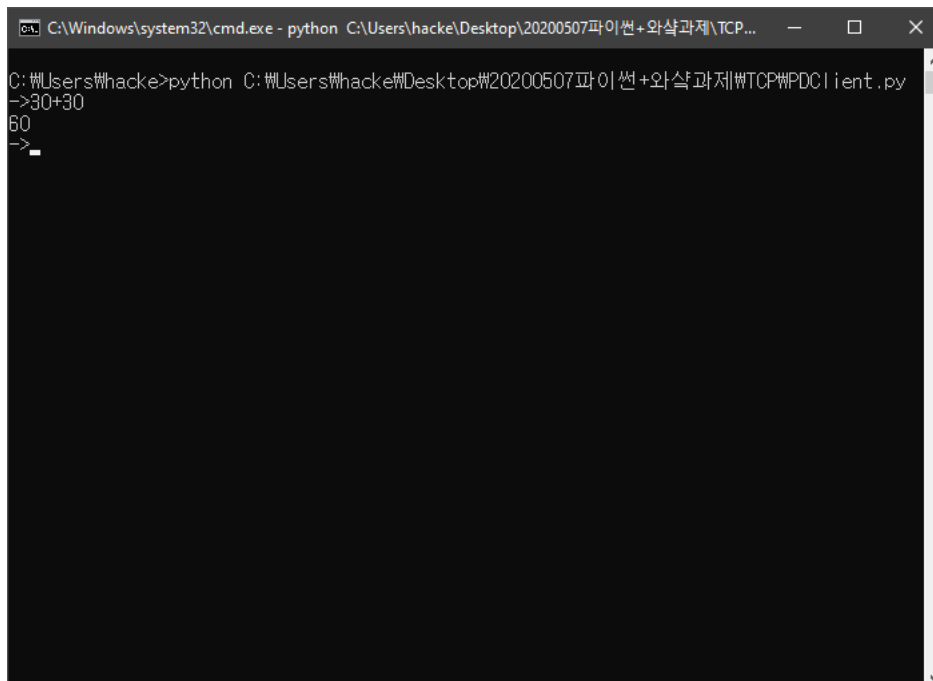
TCP는 핸드셰이킹 과정이 일어나지만 UDP는 핸드셰이킹 과정이 존재하지 않으므로 TCP에 비해 비교적 간단한 패킷교환 구조를 가지고있음

Server 프로그램의 Interface



```
Command Prompt - python C:\Users\hacke\Desktop\20200507파이썬+와샹과제\TCP\PDServer.py
C:\Users\hacke>python C:\Users\hacke\Desktop\20200507파이썬+와샹과제\TCP\PDServer.py
```

Client 프로그램의 Interface



```
C:\Windows\system32\cmd.exe - python C:\Users\hacke\Desktop\20200507파이썬+와샹과제\TCP...
C:\Users\hacke>python C:\Users\hacke\Desktop\20200507파이썬+와샹과제\TCP\PDCClient.py
->30+30
60
->
```

인터페이스는 간결하게 작성하였음.

서버에는 따로 프린팅 해주는 문구가 없으며 클라이언트에서는 서버와 정상적으로 연결이 수렴되면 "->"를 출력해 사용자로부터 연산할 내용을 입력받고, 서버로부터 받은 결과값을 출력함.