

# -컴퓨터 네트워크(2주차)-

학번 : 201404377

이름 : 진승언

환경

윈도우 노트북에 vmware ubuntu

패킷분석 : Wireshark 사용

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;... ) Firefox/51.0
Accept: text/html,application/xhtml+xml,..., */*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Request headers

General headers

Entity headers

wireshark에서 이 그림과 같이 요청 헤더를 보고 대략적인 정보들을 파악할 수 있다.

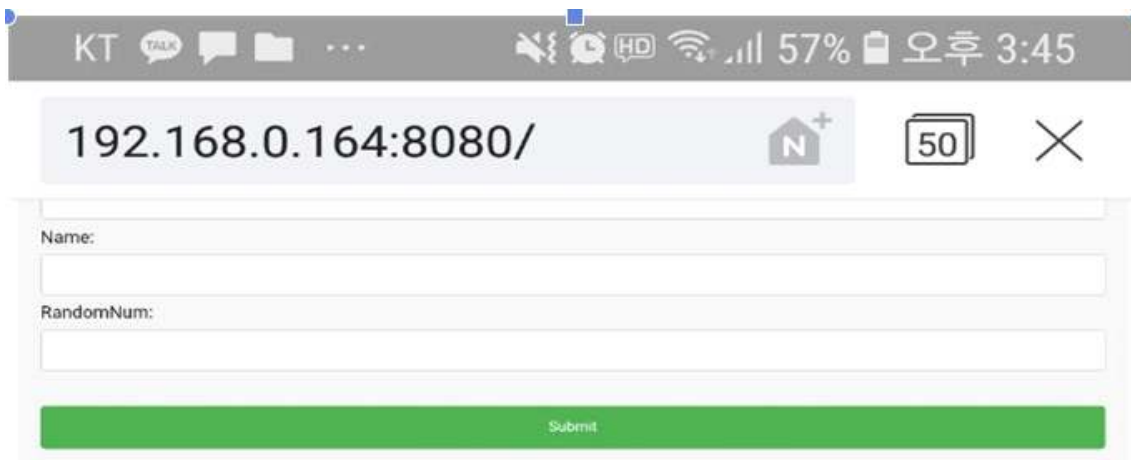
## 1. Telnet으로 만들기

내 스마트폰을 클라이언트로하고 노트북의 vmware의(가상머신) 도커의 nginx를 서버로 했다. 즉 클라이언트는 스마트폰이 하고 도커호스트, 컨테이너 역할을 노트북이 하였다. (접속이 가능하게 노트북과 스마트폰은 같은 와이파이를 사용하였다.)

### -노트북 vmware ifconfig 결과-

[illegible]

도커 컨테이너(nginx서버) ip는 172.17.0.1이고 도커호스트 ip는 192.168.0.164로 와이파이에게 부여받은 아이피이다.



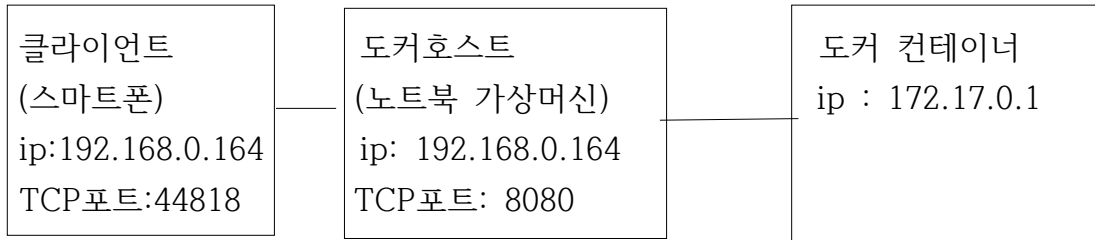
스마트폰에서 접속한 스크린샷이다.

No.	Time	Source	Destination	Protocol	Length	Info
75	1.095905	125.209.230.135	192.168.0.100	TCP	54	443 → 55150 [ACK] Seq=7356 Ack=1749 Win=32512 Len=0
76	1.111323	125.209.230.135	192.168.0.100	TCP	54	443 → 55149 [ACK] Seq=6939 Ack=638 Win=30336 Len=0
77	1.642842	192.168.0.14	192.168.0.164	TCP	74	44818 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=68296801 TSecr=0 WS=256
78	1.643655	192.168.0.164	192.168.0.14	TCP	74	8080 → 44818 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1378827977 TSecr=68296801 WS=128
79	1.643668	192.168.0.164	192.168.0.14	TCP	74	[TCP Out-Of-Order] 8080 → 44818 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1378827977 TSecr=68296801 WS=128
80	1.647733	192.168.0.14	192.168.0.164	TCP	66	44818 → 8080 [ACK] Seq=1 Ack=1 Win=87800 Len=0 TSval=68296814 TSecr=1378827977
81	1.648572	192.168.0.14	192.168.0.164	HTTP	591	GET / HTTP/1.1
82	1.649185	192.168.0.164	192.168.0.14	TCP	66	8080 → 44818 [ACK] Seq=1 Ack=526 Win=30080 Len=0 TSval=1378827982 TSecr=68296814
83	1.649196	192.168.0.164	192.168.0.14	TCP	66	[TCP Dup ACK 82#1] 8080 → 44818 [ACK] Seq=1 Ack=526 Win=30080 Len=0 TSval=1378827982 TSecr=68296814
84	1.650141	192.168.0.164	192.168.0.14	TCP	304	8080 → 44818 [PSH, ACK] Seq=1 Ack=526 Win=30080 Len=238 TSval=1378827983 TSecr=68296814 [TCP segment of a reassembled PDU]
85	1.650158	192.168.0.164	192.168.0.14	TCP	304	[TCP Retransmission] 8080 → 44818 [PSH, ACK] Seq=1 Ack=526 Win=30080 Len=238 TSval=1378827983 TSecr=68296814
86	1.650590	192.168.0.164	192.168.0.14	HTTP	1019	HTTP/1.1 200 OK (text/html)
87	1.650604	192.168.0.164	192.168.0.14	TCP	1019	[TCP Retransmission] 8080 → 44818 [PSH, ACK] Seq=239 Ack=526 Win=30080 Len=993 TSval=1378827983 TSecr=68296814
88	1.652096	192.168.0.14	192.168.0.164	TCP	66	44818 → 8080 [ACK] Seq=526 Ack=239 Win=88032 Len=0 TSval=68296815 TSecr=1378827983
89	1.653902	192.168.0.14	192.168.0.164	TCP	66	44818 → 8080 [ACK] Seq=526 Ack=1192 Win=90624 Len=0 TSval=68296815 TSecr=1378827983

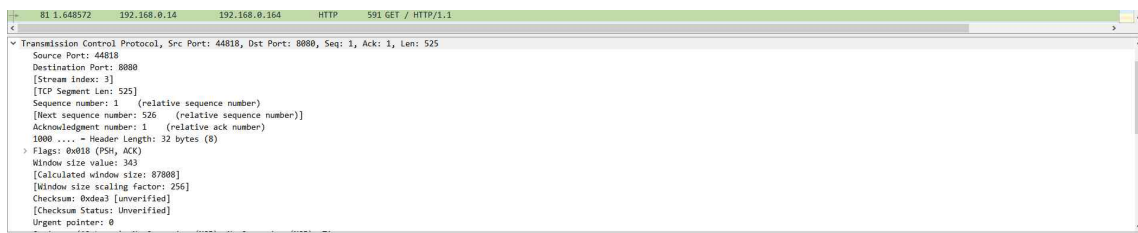
wireshark 결과이다. ip주소로 필터링을 하였다. 초록색 줄로 HTTP GET 요청과 200 OK 응답을 받았음을 알 수 있다. 그리고 source와 destination을 통해 192.168.0.14(스마트폰)에서 192.168.0.14(도커호스트)로 응답과 요청이 서로 잘 갔음을 볼 수 있다. 같은 와이파이내에서 한거라 ip가 동일하다. 그리고 HTTP요청의 검은 줄이나 HTTP 상세정보를 통해서 클라이언트인 스마트폰에서는 44818 포트에서 도커호스트의 8080 포트에 접속했음을 볼 수 있었다.

도커에서 nginx를 pull했을 때 기본 포트가 8080으로 해놨기 때문이다. (스마트폰에서 url을 보면 :8080 으로 접속도 하고 말이다.)

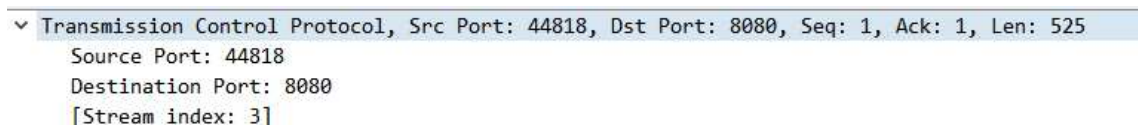
그림은 다음과 같이 정리했다.



### -요청-



스마트폰에서 서버로 HTTP Body에 따로 Data를 넣어서 전송을 한 것이 아니므로 get요청을 보내게 된다. 사진은 HTTP/1.1 get요청 패킷의 TCP 상세 정보이다.

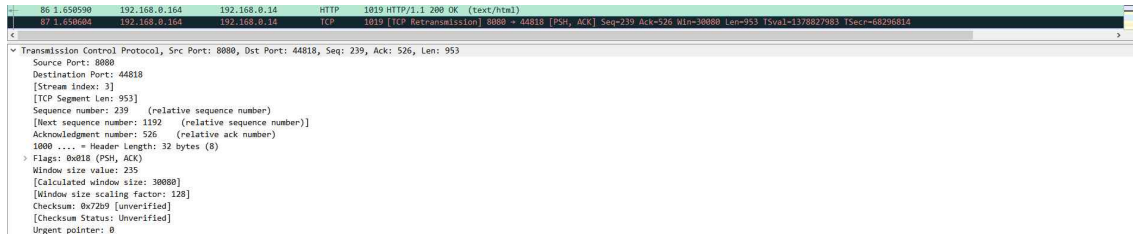


스마트폰에서 도커호스트로 가는데 44818포트로 나와 8080포트로 접속함을 볼 수 있다.



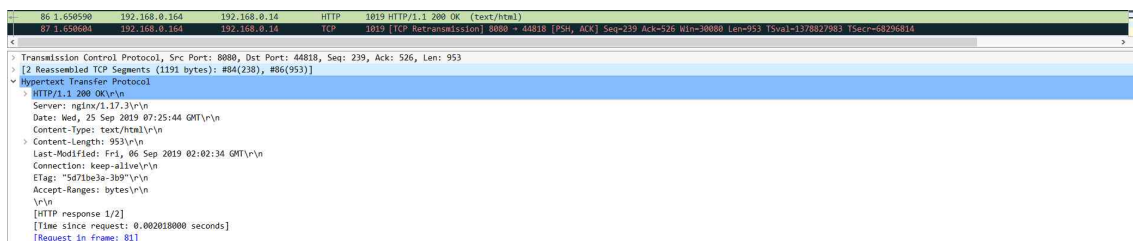
HTTP 상세정보이다. GET/ HTTP/ 1.1 로 요청을 했음을 볼 수 있고 Host의 ip주소가 192.168.0.164임을 볼 수 있다.

-응답-

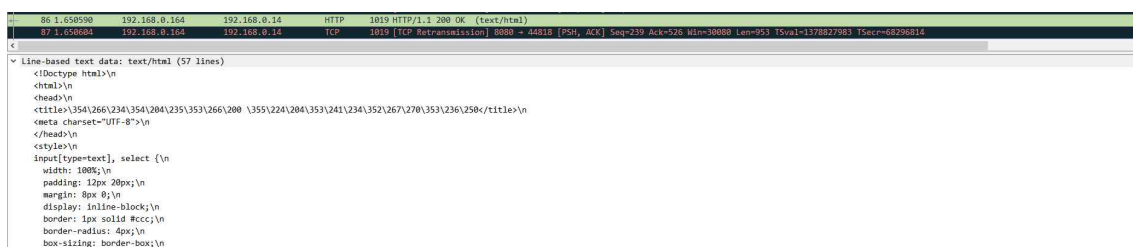


응답의 TCP 상세정보이다. 요청과는 반대로 8080포트에서나와 44818포트로 접속함을 볼 수 있다.

마찬가지로 파란줄쪽을 보면 HTTP/1.1 200 OK를 보면 응답코드 200 OK로 응답이 잘 왔음을 알 수 있다. 또한 TCP쪽을 보면 포트가 8080임을 알 수 있다.



응답의 HTTP 상세정보이다. HTTP/1.1 200 OK로 응답이 성공적으로 잘 왔다는 것을 알 수 있었다. 200코드는 응답이 이상없이 성공적으로 왔음을 뜻한다. 그리고 서버는 nginx임을 알 수 있었다. content/type이 text/html임도 볼 수 있다.

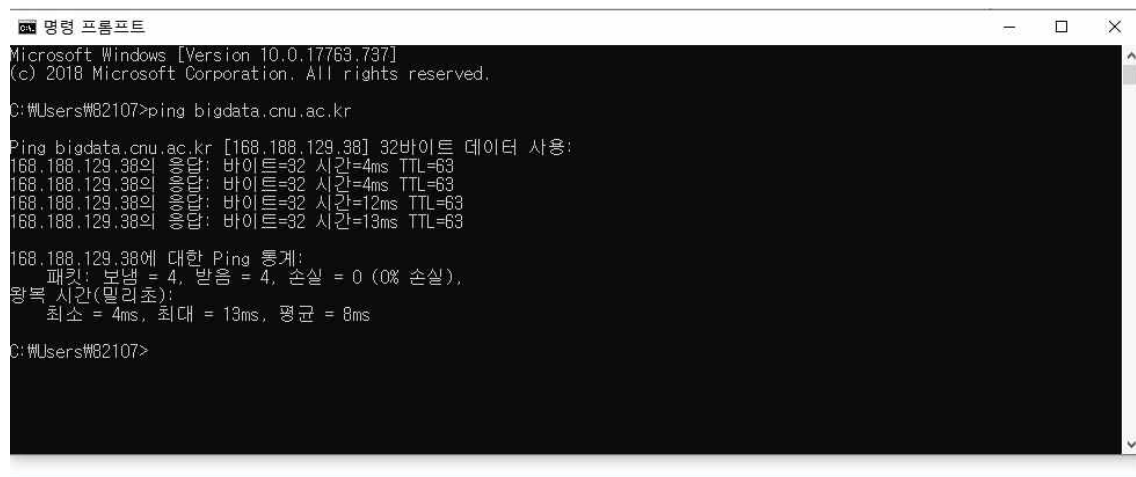


전달받은 text data인 text/html은 위와 같다.

## 2. cURL 로 만들기

vmware ubuntu 커맨드창에서 curl <http://bigdata.cnu.ac.kr:8080> 을 한 결과를 패킷분석해봤다. cURL은 다양한 통신 프로토콜을 이용하여 데이터를 전송하기 위한 라이브러리와 명령 줄 도구를 제공하는 컴퓨터 소프트웨어 프로젝트이다.

<http://bigdata.cnu.ac.kr:8080> 으로 ping을 날려 ip주소가 168.188.129.38임을 알 수 있었다.



```
명령 프롬프트
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

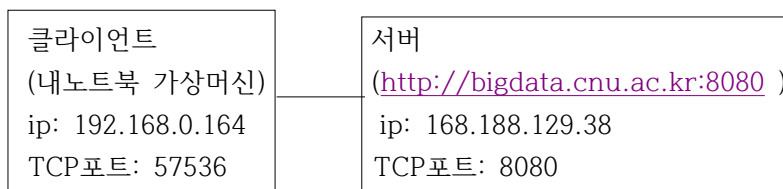
C:\Users\#82107>ping bigdata.cnu.ac.kr

Ping bigdata.cnu.ac.kr [168.188.129.38] 32바이트 데이터 사용:
168.188.129.38의 응답: 바이트=32 시간=4ms TTL=63
168.188.129.38의 응답: 바이트=32 시간=4ms TTL=63
168.188.129.38의 응답: 바이트=32 시간=12ms TTL=63
168.188.129.38의 응답: 바이트=32 시간=13ms TTL=63

168.188.129.38에 대한 Ping 통계:
    패킷: 보낸 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 4ms, 최대 = 13ms, 평균 = 8ms

C:\Users\#82107>
```

그림은 다음과 같이 정리했다.









37	1.948140	192.168.0.164	168.188.129.38	HTTP	152 GET / HTTP/1.1
38	1.948149	192.168.0.164	168.188.129.38	TCP	152 [TCP Retransmission] 57536 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=86 TSval=2393075800 TSecr=1678736831
<pre> &gt; Frame 37: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface 0 &gt; Ethernet II, Src: IntelCor_6f:59:91 (50:76:af:6f:59:91), Dst: EfmNetwo_42:6b:38 (88:36:6c:42:6b:38) &gt; Internet Protocol Version 4, Src: 192.168.0.164, Dst: 168.188.129.38 &gt; Transmission Control Protocol, Src Port: 57536, Dst Port: 8080, Seq: 1, Ack: 1, Len: 86   Hypertext Transfer Protocol     GET / HTTP/1.1\r\n     Host: bigdata.cnu.ac.kr:8080\r\n     User-Agent: curl/7.58.0\r\n     Accept: */*\r\n     \r\n     [Full request URI: http://bigdata.cnu.ac.kr:8080/]     [HTTP request 1/1]     [Response in frame: 41] </pre>					

요청의 HTTP 상세정보이다. GET/ HTTP/ 1.1을 통해 GET 요청을 보냈음을 알 수 있다. 또한 Host는 충남대 빅데이터 연구실 사이트이며 user-agent도 확인할 수 있다. user-agent는 사용자를 대신하여 일을 수행하는 소프트웨어 에이전트라고 볼 수 있다. curl로 요청했으므로 curl이 뜨고 있다.

## -응답-

41	1.953160	168.188.129.38	192.168.0.164	HTTP	686 HTTP/1.1 200 OK (text/html)
<pre> &gt; Transmission Control Protocol, Src Port: 8080, Dst Port: 57536, Seq: 239, Ack: 87, Len: 620   Source Port: 8080   Destination Port: 57536   [Stream index: 1]   [TCP Segment Len: 620]   Sequence number: 239 (relative sequence number)   [Next sequence number: 859 (relative sequence number)]   Acknowledgment number: 87 (relative ack number)   1000 .... = Header Length: 32 bytes (8)   &gt; Flags: 0x018 (PSH, ACK)   Window size value: 227   [Calculated window size: 29056]   [Window size scaling factor: 128]   Checksum: 0x549c [unverified]   [Checksum Status: Unverified]   Urgent pointer: 0 </pre>					

응답의 TCP 상세정보이다. 클라이언트가 요청한 것을 서버에서 응답을 해주는 것이므로 요청과 TCP포트가 8080에서 57536으로 반대로 되었음을 볼 수 있다.

41	1.953160	168.188.129.38	192.168.0.164	HTTP	686 HTTP/1.1 200 OK (text/html)
<pre> &gt; Transmission Control Protocol, Src Port: 8080, Dst Port: 57536, Seq: 239, Ack: 87, Len: 620   [2 Reassembled TCP Segments (858 bytes): #40(238), #41(620)]   Hypertext Transfer Protocol     HTTP/1.1 200 OK\r\n     Server: nginx/1.17.3\r\n     Date: Wed, 25 Sep 2019 07:31:22 GMT\r\n     Content-Type: text/html\r\n     Content-Length: 620\r\n     Last-Modified: Tue, 10 Sep 2019 10:32:32 GMT\r\n     Connection: keep-alive\r\n     ETag: "5d777bc0-26c"\r\n     Accept-Ranges: bytes\r\n     \r\n     [HTTP response 1/1]     [Time since request: 0.005020000 seconds]     [Request in frame: 37] </pre>					

응답의 HTTP 정보이다. 200 OK 코드를 통해 성공적으로 응답을 받았음을 알 수 있다. 서버는 nginx를 사용하고 기타 날짜나 content-type등의 정보도 알 수 있었다. 1번과 겹치는 내용이므로 생략하도록 하겠습니다

### 3. Python3 Request 모듈이용

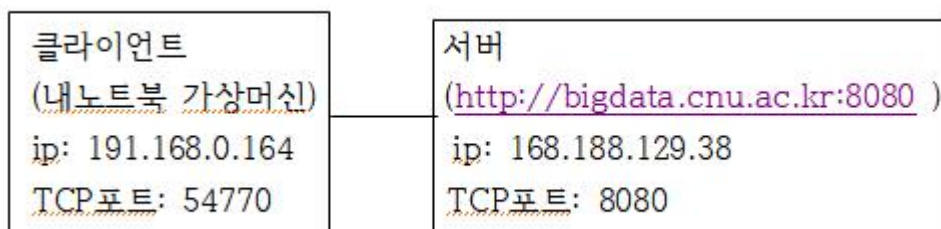
python을 사용해서 1,2에서 했던거와 마찬가지로 get요청을 한다.

```
u201404377@ubuntu: ~/u201404377
File Edit View Search Terminal Help
u201404377@ubuntu:~/u201404377$ python3
Python 3.6.8 (default, Aug 20 2019, 17:12:48)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> r = requests.get('http://bigdata.cnu.ac.kr:8080')
>>> r.status_code
200
>>> r.headers['content-type']
'text/html'
>>> r.encoding
'ISO-8859-1'
>>> r.text
'<!DOCTYPE html>\n<html>\n<head>\n<title>Welcome to nginx!</title>\n<style>\n
  body {\n          width: 35em;\n          margin: 0 auto;\n          font-family: Tah
oma, Verdana, Arial, sans-serif;\n      }\n</style>\n</head>\n<body>\n<h1>Welcome
to Yslee\'s nginx!</h1>\n<p>If you see this page, the nginx web server is succes
sfully installed and\nworking. Further configuration is required.</p>\n\n<p>For
online documentation and support please refer to\n<a href="http://nginx.org/">ng
inx.org</a>.<br/>\nCommercial support is available at\n<a href="http://nginx.com
/">nginx.com</a>.</p>\n\n<p><em>Thank you for using nginx.</em></p>\n</body>\n</
html>\n'
>>>
```

요청한 결과를 다음과 같이 볼 수 있다.

먼저 Request모듈을 사용해서 requests.get()으로 get요청을 하였고 그 요청의 응답값 r에 저장했다. 응답값인 r의 상태코드가 200 OK로 잘 뜬을 볼 수 있고 인코딩 방식이나 html형식의 text를 받아볼 수 도 있었다.

그림으로 나타내면 다음과 같다. 2번과 동일할텐데 다른 장소에서 한 것이라 ip와 TCP포트번호가 달라졌음을 볼 수 있다.



## -요청-

Capturing on Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Filter: ip.addr == 168.188.129.38

No.	Time	Source	Destination	Protocol	Length	Info
32	3.685484	192.168.0.164	168.188.129.38	TCP	74	54770 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3308833858 TSecr=0 WS=128
33	3.695493	192.168.0.164	168.188.129.38	TCP	74	[TCP Out-Of-Order] 54770 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3308833858 TSecr=0 WS=128
34	3.613465	168.188.129.38	192.168.0.164	TCP	74	8080 → 54770 [SYN, ACK] Seq=0 Ack=1 Win=20900 Len=0 MSS=1460 SACK_PERM=1 TSval=1854838612 TSecr=3308833858 WS=128
35	3.613840	192.168.0.164	168.188.129.38	TCP	66	54770 → 8080 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3308833858 TSecr=1854838612
36	3.613848	192.168.0.164	168.188.129.38	TCP	66	[TCP Dup ACK 3591] 54770 → 8080 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3308833858 TSecr=1854838612
37	3.614153	192.168.0.164	168.188.129.38	HTTP	219	GET / HTTP/1.1
38	3.614155	192.168.0.164	168.188.129.38	TCP	219	[TCP Retransmission] 54770 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=153 TSval=3308833858 TSecr=1854838612
39	3.615428	168.188.129.38	192.168.0.164	TCP	66	8080 → 54770 [ACK] Seq=1 Ack=154 Win=30080 Len=0 TSval=1854838619 TSecr=3308833858
40	3.615539	168.188.129.38	192.168.0.164	TCP	384	8080 → 54770 [PSH, ACK] Seq=1 Ack=154 Win=30080 Len=238 TSval=1854838620 TSecr=3308833858 [TCP segment of a reassembled PDU]
41	3.615655	168.188.129.38	192.168.0.164	HTTP	686	HTTP/1.1 200 OK (text/html)
42	3.615734	192.168.0.164	168.188.129.38	TCP	66	54770 → 8080 [ACK] Seq=154 Ack=239 Win=30336 Len=0 TSval=3308833860 TSecr=1854838620
43	3.615740	192.168.0.164	168.188.129.38	TCP	66	[TCP Dup ACK 4281] 54770 → 8080 [ACK] Seq=154 Ack=239 Win=30336 Len=0 TSval=3308833860 TSecr=1854838620
44	3.615909	192.168.0.164	168.188.129.38	TCP	66	54770 → 8080 [ACK] Seq=154 Ack=859 Win=31616 Len=0 TSval=3308833860 TSecr=1854838620
45	3.615924	192.168.0.164	168.188.129.38	TCP	66	[TCP Dup ACK 4441] 54770 → 8080 [ACK] Seq=154 Ack=859 Win=31616 Len=0 TSval=3308833860 TSecr=1854838620
46	3.620105	192.168.0.164	168.188.129.38	TCP	66	54770 → 8080 [FIN, ACK] Seq=154 Ack=239 Win=31616 Len=0 TSval=3308833864 TSecr=1854838620
47	3.620113	192.168.0.164	168.188.129.38	TCP	66	[TCP Out-Of-Order] 54770 → 8080 [FIN, ACK] Seq=154 Ack=859 Win=31616 Len=0 TSval=3308833864 TSecr=1854838620
48	3.622094	168.188.129.38	192.168.0.164	TCP	66	8080 → 54770 [FIN, ACK] Seq=859 Ack=155 Win=30080 Len=0 TSval=1854838625 TSecr=3308833864
49	3.622218	192.168.0.164	168.188.129.38	TCP	66	54770 → 8080 [ACK] Seq=155 Ack=860 Win=31616 Len=0 TSval=3308833867 TSecr=1854838625

[Calculated window size: 29312]  
 [Window size scaling factor: 128]  
 Checksum: 0d6e7e [unverified]  
 [Checksum Status: Unverified]  
 Urgent pointer: 0  
 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps  
 [SEQ/ACK analysis]  
 [Timestamps]  
 TCP payload (153 bytes)  
 Hypertext Transfer Protocol  
 GET / HTTP/1.1\r\n  
 Host: bigdata.cnu.ac.kr:8080\r\n  
 User-Agent: python-requests/2.18.4\r\n  
 Accept-Encoding: gzip, deflate\r\n  
 Accept: \*/\*\r\n  
 Connection: keep-alive\r\n  
 \r\n  
 [Full request URI: http://bigdata.cnu.ac.kr:8080/]  
 [HTTP request 1/1]  
 [Response in frame: 41]

0000 97 54 47 45 54 30 2f 20 48 54 54 50 2f 31 2e 31 --TGET / HTTP/1.1  
 0050 0d 0a 48 6f 73 74 3a 20 62 69 67 64 61 76 31 2e --Host: bigdata.  
 0060 63 6e 75 2e 63 63 2e 60 72 3a 38 38 30 0d 0a cnu.ac.kr:8080-  
 0070 55 73 65 72 2d 61 67 65 6e 74 3a 20 70 79 74 68 User-Age nt: pyth  
 0080 6f 6e 2d 72 65 71 75 65 73 74 73 2f 32 2e 31 38 on-reqe sts/2.18

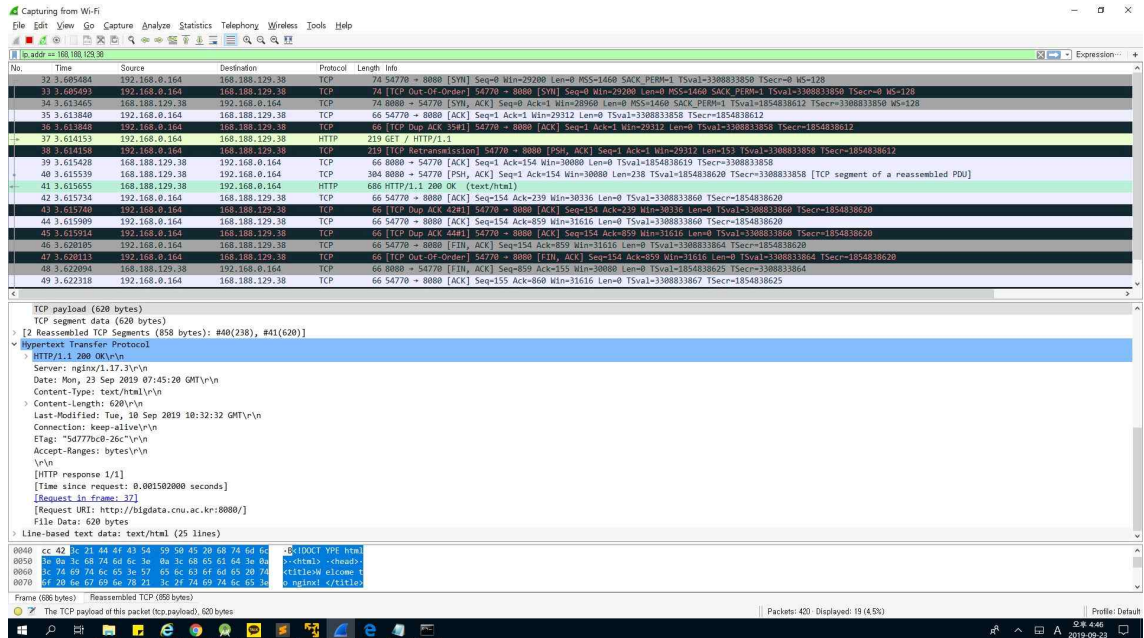
The TCP payload of this packet (tcp.payload), 153 bytes

Packets: 205 | Displayed: 19 (7.2%) | Profile: Default

오류 4.45  
 2019-09-23

요청부분도 2번과 동일하다고 보는데 다른점이 두가지가 정도가 있다. 하나는 바뀐 포트번호와 User-agent가 2번에서는 curl이 었지만 이번엔 파이썬 모듈을 사용해 통신을 했으므로 python-requests 가 적혀있음을 볼 수 있다. 다른 하나는 다른 장소에서 했으므로 클라이언트의 ip와 TCP포트가 다르다는 점이다.

-응답-



응답이 200 OK로 잘 왔음을 확인할 수 있었다. 마찬가지로 이 결과도 포트번호와 클라이언트 ip 제외 2번과 동일하다고 볼 수 있다.

## [과제 후기]

보고서를 쓰긴 했는데 이게 정확히 맞는건지는 모르겠다. 분발해야겠다.