HTTP 분석

수학과 201621136 이재협

1. 관찰 환경 및 방법

   1) 언제: 4월 3일 Wireshark 프로그램을 처음 사용해 보았다. 이후 2~3일에 한 번씩 it버팀목 사이트, '청오회' 사이트를 접속하여 HTTP 헤더 분석, Web-Caching 등의 분석을 하였고, 사진에 있는 자료는 4월 9일부터 20일 사이에 해당 사이트들을 접속한 기록이다.

   2) 어디서: 내 방에 있는 iptime 공유기의 WIFI 혹은 내가 다니는 스터디 카페의 WIFI를 이용했다. 다른 곳에서 Wireshark를 사용하게 될 경우, IP주소를 혼동하는 등의 내가 인지하지 못한 오차를 방지하기 위해서 한 번을 제외하고는 내 방에서 관찰했다.

2. Protocol 분석

   1) HTTP request와 response message의 header 분석

   Facebook이나 Google 등의 유명 해외 사이트는 메시지가 암호화되어 전송되기 때문에 Wireshark로 해석하기 힘들다. 따라서, 국내 웹 서버 중 내가 컴퓨터 활용능력 1급 강의를 수강하고 있는 it버팀목(https://www.itbtm.com) 사이트를 이용하였다.

   <처음으로 보낸 Request Message>

```
✓ Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Accept: text/html, application/xhtml+xml, image/jxr, */*\r\n
    Accept-Language: ko-KR\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: www.itbtm.com\r\n
    If-Modified-Since: Wed, 08 Apr 2020 16:30:51 GMT\r\n
    Connection: Keep-Alive\r\n
  > Cookie: 96b28b766b7e0699aa91c9ff3d890663=aHR0cDovL3d3dy5pdGJ0bS5jb20v; f33d2ed86bd82d4(
    \r\n
    [Full request URI: http://www.itbtm.com/]
    [HTTP request 1/8]
    [Response in frame: 49]
    [Next request in frame: 54]
```

먼저, 매 행마다 적힌 ₩r₩n 부분은 행 바꿈을 의미한다.

① HTTP method 종류를 뜻한다.

- GET: 지정된 URL이 가진 정보를 서버 측에 요청할 때 사용된다. 사실, 우리가 인터넷 주소 창에 www.naver.com 이라고 입력하는 것은 www.naver.com이라는 URL을 GET이라는 HTTP method를 이용하여 서버에 정보를 요청하는 것과 같다.
- DELETE: 서버에 파일을 삭제를 요청할 때 사용된다,
- HEAD: 서버에서 헤더 정보만 받아올 때 사용된다.
- TRACE: 클라이언트가 요청한 자원에 도달하기까지 경로를 기록할 때(loop back) 사용된다.

  이외에도 PUT, PATCH, OPTION 등 다양한 형태의 HTTP method가 있다. 추가적으로, HTTP/1.1 부분은 HTTP가 1.1버전인 것을 뜻한다.

② Accept: 클라이언트가 원하는 미디어 타입을 말한다. 위의 사진의 경우, 클라이언트가 html 형식의 text, xtml 혹은 xml 형태의 application, jxr형식의 image을 원한다는 뜻이다.

은 와일드 카드를 뜻하며, 모든 미디어 타입이 가능하다는 뜻이다. 예를 들어, text/ 라고 표시되어 있으면 클라이언트는 모든 형식의 text를 받을 수 있다는 뜻이다. */*은 어떤 미디어 타입도 가능하다는 뜻이다.

간혹 image/*;q=0.8, */*;q=0.5 와 같이 q라는 숫자가 적혀 있는 경우를 볼 수 있는데, q는 q value라고 하며 0~1 사이의 일종의 가중치이다. q value가 클수록 클라이언트의 선호도가 높다는 것을 의미한다.

Accept, Accept-Language, Accept-Encoding 등의 헤더를 통해 클라이언트가 다룰 수 있는 형식의 데이터를 가져올 수 있다. 어떤 자원을 영어보단 한글, gif파일 보다는 jxr 등 여러가지의 방법으로 표현할 수 있다. 이 경우, 해당 자원을 negotiable하다고 말하며, 이 때의 각 표현 방식을 variant 라고 한다. 클라이언트의 선호도에 따라 자원의 적합한 표현을 찾는 것을 Content Negotiation이라고 한다.

③ Accept-Language: 클라이언트가 받아들일 문자열을 말한다. ko-KR 은 한국어를 뜻하며 영어를 원한다면 en-US, 불어를 원한다면 fr이라고 표현하면 된다. 여기서도 fr;q=1.0, ko-KR;q=0.8 과 같이 q value를 사용할 수 있다.

④ User-Agent: IE11, Chrome, Firefox 등의 브라우저 종류, 심지어 Windows, Mac OS 등의 운영체제 종류나 PC와 모바일의 여부도 알 수 있다. IE를 지원하지 않는 서버의 경우, User-Agent에 표시된 정보를 토대로 "Chrome으로 접속하세요" 와 같은 안내를 할 수 있다.

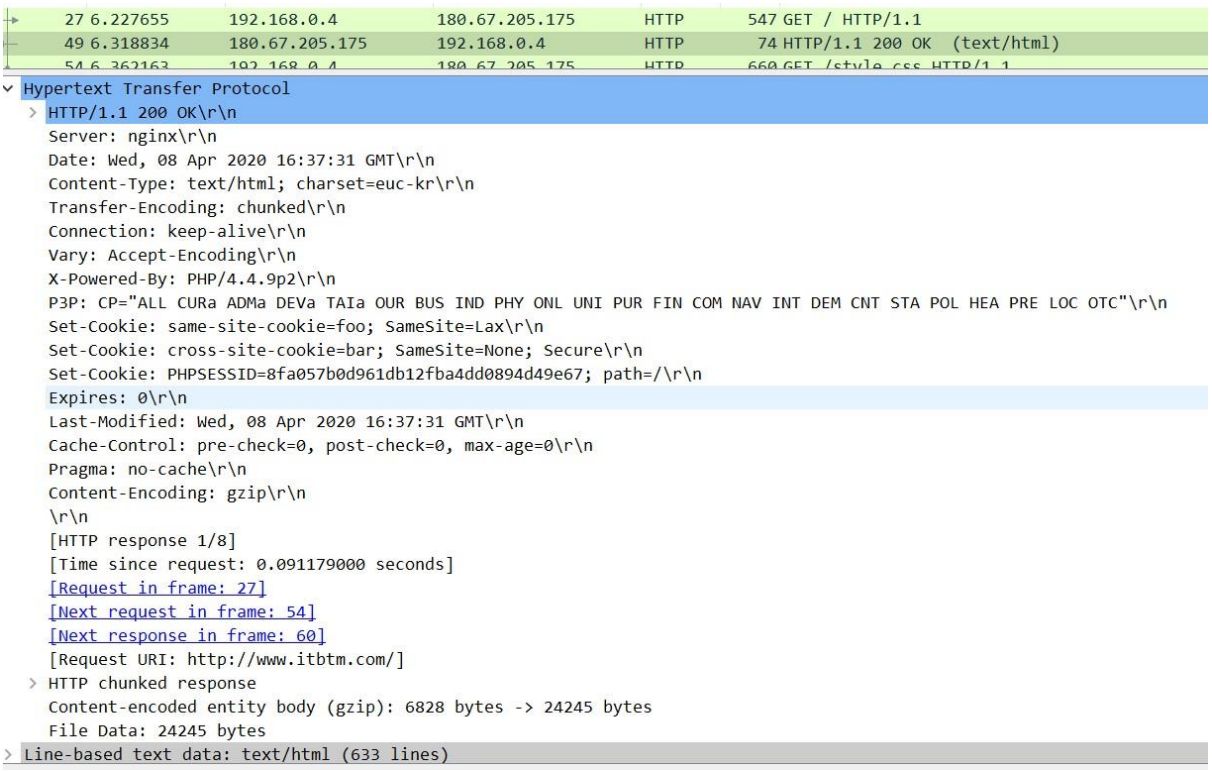
⑤ Host: 서버의 도메인 네임이 나타나는 곳이며 HTTP/1.1 이후부터는 반드시 하나가 존재해야 한다.

⑥ If-Modified-Since: Web caching과 관련된 헤더이며, 클라이언트가 이전에도 요청한 적 있었던 특정 컨텐츠를 요청했을 때, 프록시 서버가 caching 해 두었던 해당 컨텐츠가 최신 것인지 서버에게 확인하는 헤더이다. 위의 사진의 경우, 프록시 서버가 가지고 있던 컨텐츠는 2020년 4월 8일 수요일 16:30:51에 최종 수정된 자료임을 말한다. GMT는 표준 시간을 의미한다. 만약, 클라이언트가 처음으로 특정 컨텐츠를 요청하는 경우, 서버는 Last-Modified 라는 헤더를 통해서 그 컨텐츠의 최종 수정 시간을 알려준다.
Caching에 대한 상세한 설명은 4)Web caching 기능에서 서술할 예정이다.
If-Modified-Since 와 같이 조건에 따라 경우가 바뀌는 헤더를 조건부 헤더라고 한다. 조건부 헤더에는 If-Match, If-Range, If-Unmodified-Since 등이 있다고 한다.

⑦ Connection: Keep-alive 혹은 close 둘 중 하나의 값을 가지며, 서버와 계속해서 연결을 유지할 지에 대한 여부를 판단한다.
HTTP/1.1의 경우 대부분 keep-alive를 지원하고, 이는 처음 성사된 클라이언트와 서버의 연결을 끊지 않고 유지시킬 수 있다. 이는 persistent connection, 더 나아가 pipelining을 위해 사용된다. 한 번 열린 TCP connection을 계속 이용하기 때문에, 새로운 연결(TCP handshake라고도 함)에 필요한 비용을 줄여 TCP 성능을 향상시킬 수 있다.
Connection이 close인 경우, 컨텐츠를 보내주는 용무가 끝나면 클라이언트와 서버 사이의 연결을 끊어버린다. 이는 non-persistent connection에 해당한다.

⑧ Cookie: 클라이언트가 가지고 있던 쿠키를 말한다. 쿠키를 이용하여 서버는 해당 쿠키를 가진 클라이언트가 누구인지, 이전에 어떤 활동을 했는지 등을 알 수 있다. 이후 클라이언트가 다시 접속할 때 쿠키를 통해 어떤 클라이언트인지 식별할 수 있는 기준이 된다.
만약 클라이언트가 해당 사이트를 처음 방문하는 것이라면, 서버는 Set-Cookie라는 헤더를 이용해서 클라이언트에게 쿠키를 배정해준다

<처음으로 받은 Response Message>



①Server: 요청을 처리하는 소프트웨어 혹은 하위 제품의 이름이다. Apache 혹은 ngnix가 가장 대표적인 예시이다.

② Date: 날짜를 의미하며, If-modified-Since와 동일한 형식이다. 2020년 4월 8일 수요일 16:37:31분에 response가 전송되었다는 것을 알 수 있다.

③ Content-Type: 클라이언트에게 반환된 컨텐츠의 유형이 실제로 무엇인지 알려준다. 위의 사진의 경우, 메시지 내용은 text/html 타입이고 문자열은 euc-kr은 ASCII코드와 비슷한 문자이며, 2바이트로 한글을 표현할 수 있게 만든 방식이다.

④ Transfer-Encoding: 안전한 전송을 위해 사용하는 인코딩 형식을 말한다. Chunked는 file이 여러 개의 덩어리로 나누어진 상태로 전송된다는 뜻이다. 이외에도 gzip, compress 등 여러 형식이 있다.

⑤ Connection: Request에서의 connection과 동일하다. Keep-alive 인 경우 성사된 연결을 끊지 않고 persistent-connection을 수행한다.

⑥ Vary: 같은 URL이라도 클라이언트에 따라 반환 결과가 다를 수 있다. 이 때, 반환 결과에 영향을 줄 수 있는 헤더들의 목록을 클라이언트에 알려주는 것이 Vary 헤더이다.

만약, Accept-Encoding이라고 적혀 있다면 클라이언트가 다룰 수 있는 인코딩(gzip, compress 등)에 따라 반환 결과가 바뀔 수 있다는 뜻이다.
우리가 특정 사이트를 PC로 접속할 때와 모바일로 접속할 때의 표시 형식이 달라야 하는 경우를 생각해보자. Request Message의 ④번에서도 언급한 것처럼 클라이언트가 PC인지 모바일인지의 여부는 User-Agent를 통해 알 수 있다. 따라서 Vary: User-Agent 라고 한다면 유용하다.

⑦ X-Powered-By: 해당 사이트 어떤 기술로 개발되었는지 알 수 있다. PHP, JSP 등이 있으며 이러한 정보는 보안을 위협하려는 사람들에게 악용될 가능성이 있어 X-Powered-By 정보를 숨긴 채 전송할 수 있다.

⑧ P3P: Platform for Privacy Preference 라고 하며 프라이버시 보호 관련 표준 기술이다. 사용자들이 허용하는 정보만 웹 사이트가 수집할 수 있게 통제하는 도구이다. 사용자의 웹 브라우저 소프트웨어는 사용자의 프라이버시 조항과 웹 사이트에서 받은 조항들을 대조한다. 웹 사이트 P3P정책과 사용자의 설정과 비교해 본 후 허용, 제한, 금지 등의 적절한 조치를 취한다.

⑨ Set-Cookie: 클라이언트가 처음으로 서버에 컨텐츠를 요청한 경우, 클라이언트가 사용할 쿠키를 알려준다. 3번째로 나오는 Set-Cookie가 클라이언트에 저장된 쿠키이다. Cookie 헤더 뒷부분에 나오는 SameSite는 해당 쿠키를 전송하지 않게 할 지에 대한 여부를 알려준다. 기본적으로 CSRF(사용자의 의지와는 관련없이 공격자가 사용자의 쿠키를 악용하여 웹사이트에 요청하는 행위)를 방지하기 위해 추가되었다. Same-site-cookie 부분에는 Strict, Lax, None 등이 올 수 있다.

⑩ Expires: 날짜와 시간이 들어가며 응답된 컨텐츠가 언제 만료되는지를 나타낸다. 지정한 날짜가 지나지 않았다면 캐시를 신선하다고 판단한다. 0과 같이 유효하지 않은 날짜는 과거의 시간을 나타내어 이미 만료되었음을 의미한다. Cache-Control 헤더에 초 단위의 시간만큼 캐시가 유효하도록 설정해주는 max-age가 사용된 경우, Expires 헤더보다 우선적으로 사용된다.

⑪Last-Modified: 클라이언트가 특정 컨텐츠를 요청했을 때, 해당 컨텐츠가 서버에서 언제 마지막으로 수정된 것인지 알려준다. 컨텐츠를 처음 요청하는 경우, 프록시 서버에서 해당 컨텐츠를 저장한다. 해당 컨텐츠를 이전에 요청한 적이 있으면 프록시 서버에 해당 컨텐츠가 저장이 되어있다. 따라서, 프록시 서버에 저장된 컨텐츠의 날짜와 Last-Modified의 날짜를 비교하여 서버에서 컨텐츠를 새로 다운받을지 결정한다.

⑫Cache-Control: 서버는 Cache-Control 헤더로 더 유연하게 Cache와 관련한 일을 처리할 수 있으며 Expires 보다 먼저 처리된다. 몇 가지 예시를 살펴보자.

- Private: 같은 컴퓨터를 사용하는 다른 사용자에게는 캐시를 재사용 하지 않게 한다.
- Public: 같은 컴퓨터를 사용하는 복수의 사용자에게 캐시 재사용을 허가한다.
- Max-age=n: 캐시가 유효한 시간을 초 단위로 설정한다. 60x60x24 = 86400을 지정하면 하루동안 캐시가 유효하다. 설정한 시간이 지나면 서버에 문의한 뒤 304 Not Modified가 반환된 경우에만 캐시를 이용한다.
- No-cache: 캐시가 유효한지 항상 문의한다.
- No-store: Cache하지 않는다.

⑬Pragma: request 와 response 사이에 영향을 줄 수 있는 구현 관련 헤더이다. no-cache는 요청한 컨텐츠가 이미 저장되어 있어도, 원래 서버로 request 보내도록 강요하는 명령이다.

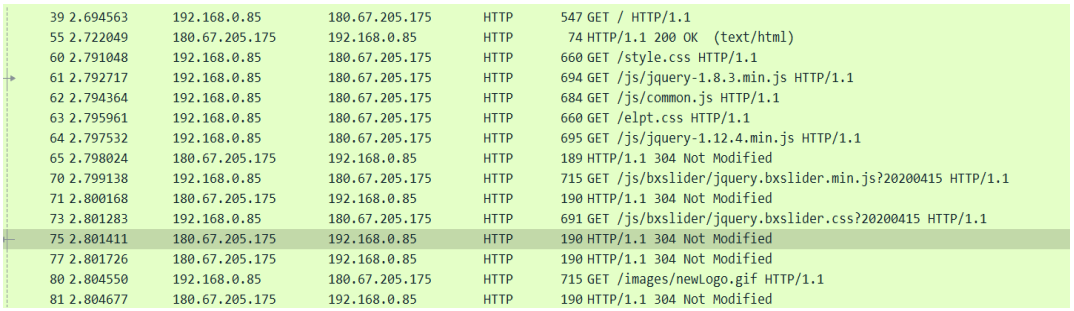
⑭Content-Encoding: 미디어 타입을 압축하기 위해 사용된다. 이 헤더에 적힌 값은 본문에 어떤 인코딩이 적용될지를 나타낸다.

2) Persistent – connection

먼저, Non-persistent connection 과 Persistent connection 그리고 Pipelining에 대한 설명은 아래의 표를 보면 알 수 있다.

HTTP connection 방식	2개의 컨텐츠 전달 순서
Non-persistent connection	연결 -> 컨텐츠1 요청 -> 컨텐츠1 전달 -> 연결 끊기 연결 -> 컨텐츠2 요청 -> 컨텐츠2 전달 -> 연결 끊기
Persistent connection	연결 -> 컨텐츠1 요청 -> 컨텐츠1 전달 ->컨텐츠 2 요청 -> 컨텐츠2 전달 -> 연결 끊기
Pipelining	연결 -> 컨텐츠1 요청, 컨텐츠2 요청 -> 컨텐츠 1 전달 -> 컨텐츠2 전달 -> 연결 끊기

https://www.ltbtm.com 에 접속한 후 클라이언트와 서버사이에 교환된 메시지이다.



39	2.694563	192.168.0.85	180.67.205.175	HTTP	547	GET / HTTP/1.1
55	2.722049	180.67.205.175	192.168.0.85	HTTP	74	HTTP/1.1 200 OK (text/html)
60	2.791048	192.168.0.85	180.67.205.175	HTTP	660	GET /style.css HTTP/1.1
61	2.792717	192.168.0.85	180.67.205.175	HTTP	694	GET /js/jquery-1.8.3.min.js HTTP/1.1
62	2.794364	192.168.0.85	180.67.205.175	HTTP	684	GET /js/common.js HTTP/1.1
63	2.795961	192.168.0.85	180.67.205.175	HTTP	660	GET /elpt.css HTTP/1.1
64	2.797532	192.168.0.85	180.67.205.175	HTTP	695	GET /js/jquery-1.12.4.min.js HTTP/1.1
65	2.798024	180.67.205.175	192.168.0.85	HTTP	189	HTTP/1.1 304 Not Modified
70	2.799138	192.168.0.85	180.67.205.175	HTTP	715	GET /js/bxslider/jquery.bxslider.min.js?20200415 HTTP/1.1
71	2.800168	180.67.205.175	192.168.0.85	HTTP	190	HTTP/1.1 304 Not Modified
73	2.801283	192.168.0.85	180.67.205.175	HTTP	691	GET /js/bxslider/jquery.bxslider.css?20200415 HTTP/1.1
75	2.801411	180.67.205.175	192.168.0.85	HTTP	190	HTTP/1.1 304 Not Modified
77	2.801726	180.67.205.175	192.168.0.85	HTTP	190	HTTP/1.1 304 Not Modified
80	2.804550	192.168.0.85	180.67.205.175	HTTP	715	GET /images/newLogo.gif HTTP/1.1
81	2.804677	180.67.205.175	192.168.0.85	HTTP	190	HTTP/1.1 304 Not Modified

55번 메시지인 200 OK 라는 서버의 응답 이후로, 클라이언트는 자신이 필요한 컨텐츠들을 한꺼번에 요청(GET)한다. 이 요청은 컨텐츠의 response가 오기 전에 request를 연이어 하였으므로 이 경우 persistent connection의 pipelining이 발생했다고 할 수 있다.

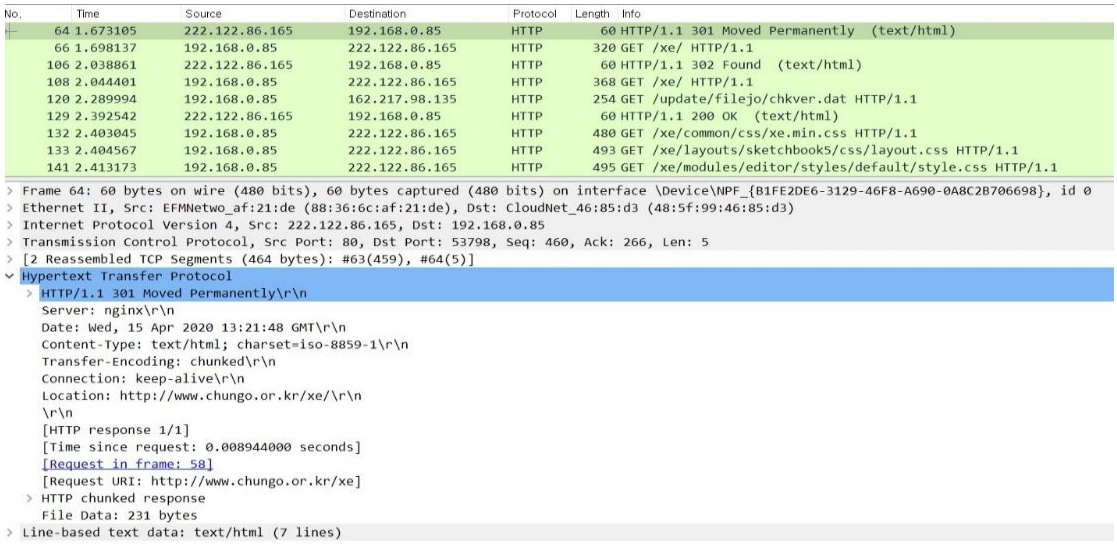
만약 Get을 이용한 클라이언트 request와 서버의 response(200 OK 혹은 304 Not found)가 번갈아 가면서 나타난다고 생각해보자. 이 경우 클라이언트와 서버의 연결은 유지한 채 컨텐츠1 요청-> 컨텐츠1 전달 -> 컨텐츠2 요청-> 컨텐츠2 전달을 받는 방식에 해당한다. 따라서 persistent connection이다.

Non-persistent connection이면, 하나의 컨텐츠를 받을 때 마다 서버와 클라이언트의 통신이 끊어졌다가 새로 연결된다. 따라서 GET/HTTP/1.1 메시지와 HTTP/1.1 200 OK 메시지가 반복해서 나타나게 될 것이다. 대부분의 경우 빠른 속도를 위해 persistent connection을 지원하므로 non-persistent connection은 흔치 않다.

Request message의 ⑦을 보면 persistent connection이 TCP 성능을 향상시켜 준다고 하였다. 그렇다면 persistent connection이 항상 좋은 것일까? 그렇지 않다. Persistent connection으로 인해 서버와 연결된 클라이언트의 접속이 지나치게 늘어나면, 서버가 이를 감당할 수 없는 상황이 발생할 수도 있기 때문이다. 따라서, 클라이언트의 접속이 잦은 메인 페이지와 같은 경우, persistent connection을 사용할 지에 대한 여부를 신중히 고민해야 한다.

3) Redirection 사례

앞서 방문했던 it버팀목 사이트에서는 다른 URL을 알려주는 경우를 찾기 어려워 다른 사이트들을 찾아다녔다. 다양한 사이트들을 돌아다닌 결과 '청오회'라는 사이트(www.chungo.or.kr)에서 아래와 같은 결과를 얻을 수 있었다.

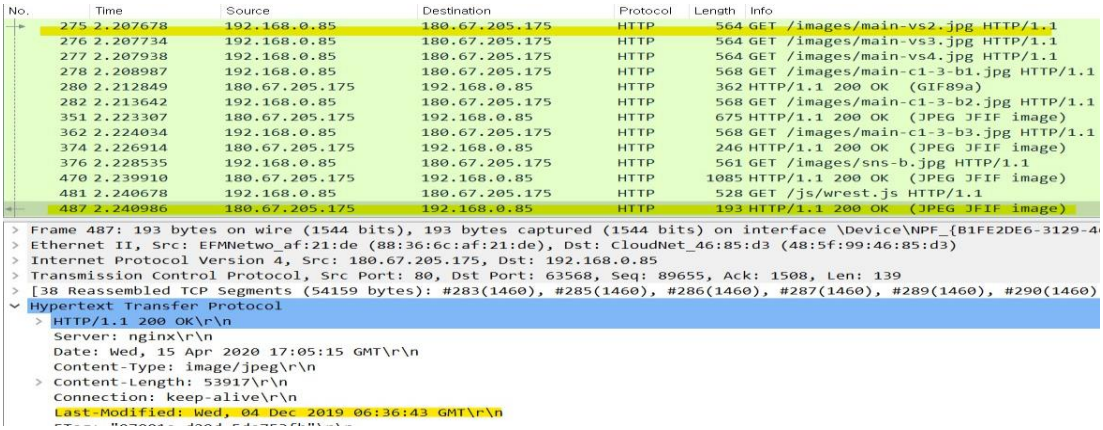


첫 번째 줄을 보면 301 Moved Permanently 라는 response를 볼 수 있다. 내가 요청했던 정보가 다른 URL로 이동했으며 Location 헤더를 통해 그 URL은 http://www.chungo.or.kr/xe 라고 알려주고 있다. 그 이후로, 클라이언트는 계속해서 서버가 알려준 URL에 GET request message를 보내는 것을 확인할 수 있다.

4) Web caching 기능

Web- Caching이란, 클라이언트가 특정 컨텐츠를 요청했을 때, 프록시 서버라는 곳에서 해당 컨텐츠를 저장해 두는 것이다. 같은 컨텐츠를 다시 요청할 때 프록시 서버는 서버에게 컨텐츠의 최종 변경 시간을 확인한다. 만약, 프록시 서버의 컨텐츠가 최신의 것이라는 것이 확인되면 프록시 서버가 자신의 컨텐츠를 클라이언트에게 제공한다. Web caching의 정확한 비교를 위해 인터넷 옵션에서 캐시메모리를 제거한 후 처음 it 버팀목 사이트에 접속했을 때와 그 이후에 접속한 경우를 비교하였다.

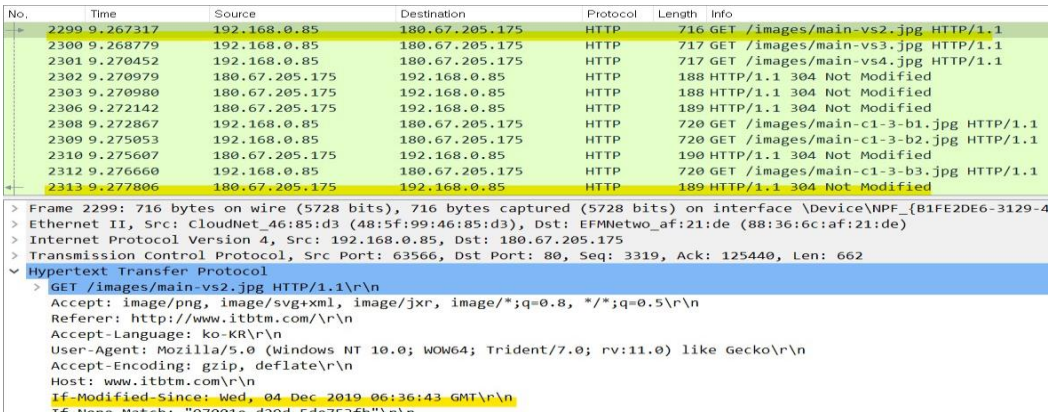
<해당 사이트를 처음 접속했을 때>



캐시 메모리에 저장된 컨텐츠가 없기 때문에 서버에서 해당 컨텐츠를 받아와야 하므로 200 OK의 response message가 나타났다. 이 컨텐츠가 마지막으로 수정된 시기

가 Last-Modified헤더에 적힌 날짜이고, 이 컨텐츠는 프록시 서버에 저장된다.

<해당 사이트를 반복해서 재접속한 경우>



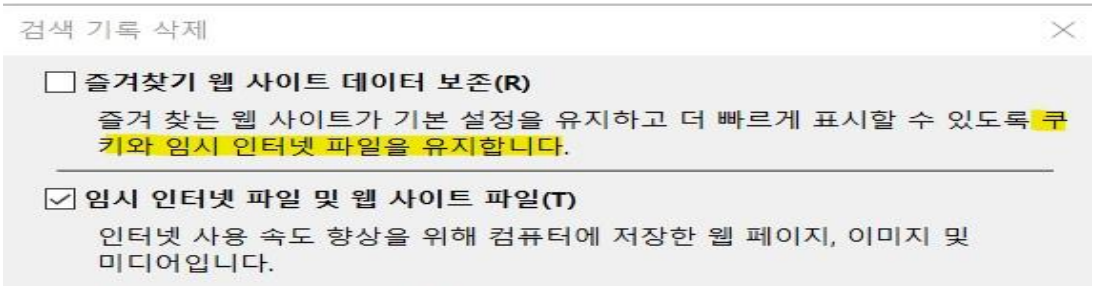
If-Modified-Since 헤더를 이용해서 프록시 서버가 가지고 있는 컨텐츠가 최신 것인지 서버에게 확인한다. 이 때, 서버는 프록시 서버가 컨텐츠를 대신 전달하라는 응답인 304 Not modified 남겼으므로 직전에 Cache된 컨텐츠가 최신 것이라는 것을 알 수 있다.

그런데, If-Modified-Since는 프록시 서버에서 원래 서버로 보낼 때 사용하는 것인데, 프록시 서버의 위치는 어디인지 궁금했다. 담당 조교에게 질문을 하여 프록시 서버는 인터넷 브라우저 내부에 있는 캐시임을 알게 되었다.

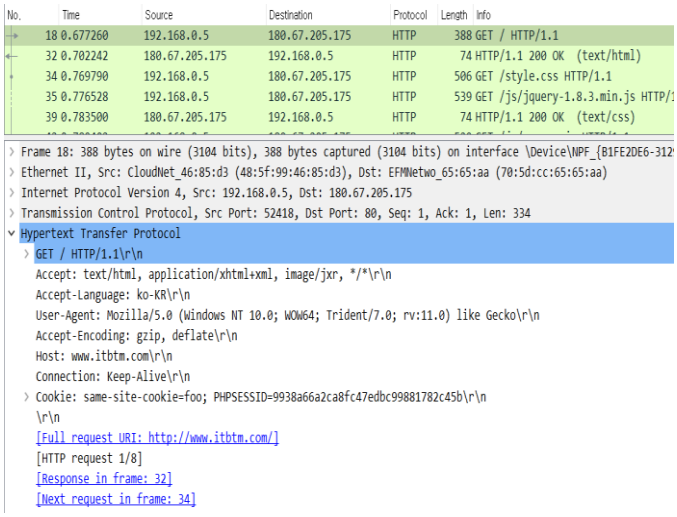
그리고 프록시는 Forward 프록시와 Reverse 프록시가 있다는 것도 알게 되었다. Forward 프록시는 클라이언트의 요청이 원래 서버에 직접 전송되지 않고 프록시를 거쳐 전달되는 방식이다. Reverse 프록시는 클라이언트가 원래 서버 대신 프록시로 설정된 주소로 요청을 한다. 그 후, 프록시 서버가 공개되지 않은 원래 서버에게 요청하는 방식으로 클라이언트는 원래 서버의 정보를 알 수 없다.

5) 자유 주제

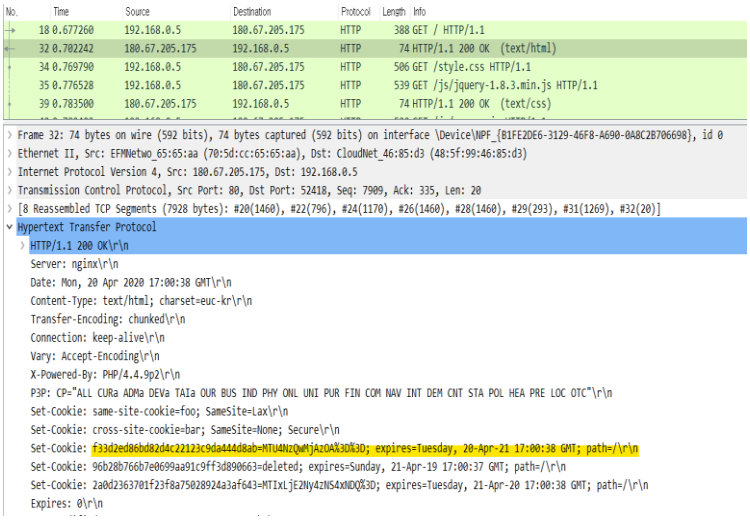
수업 시간에 Cookie에 관해서 자세히 배웠는데 Request와 Response message에 쿠키 관련 헤더 내용만으로는 부족한 것 같아서 Cookie 관련 실험을 해 보았다.
쿠키를 삭제한 후 사이트를 들어가면 어떤 메시지를 주고받는지 궁금증이 생겼다. 인터넷 옵션 - 검색 기록을 제거할 때 쿠키를 삭제한다는 것을 본 기억이 있었다. 따라서 이 방법을 통해 쿠키 파일을 모두 삭제한 후 접속해 보았다. 체크를 하지 않는 것이 쿠키를 보존하는 것인지 확실치 않아 체크를 한 상태와 체크를 하지 않은 상태로 두 번에 걸쳐 삭제를 하였다. 그 후 it버팀목 사이트를 접속했고 이 때, 시간은 2020년 4월 21일 오전 2시 정각이었다.



<Request>

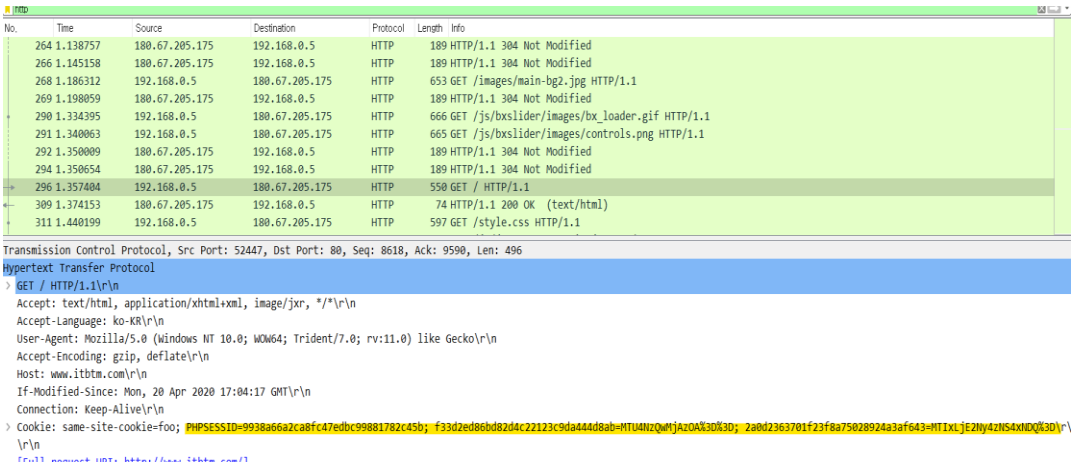


<Response>



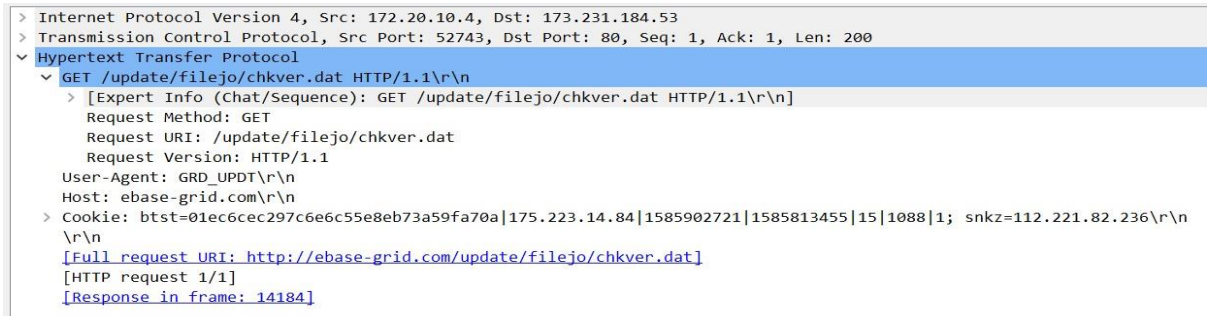
Response의 Set-Cookie헤더에서 새로 발급받은 쿠키를 볼 수 있다.

request문의 Cookie헤더에 PHPSESSID 라는 것은 세션 관련 정보이다. 이는 쿠키에서 저장하지 않는 아이디, 개인 정보 등의 중요한 데이터를 저장할 때 주로 쓰인다. 다시 말해, 서버에게 주는 내 정보이며 서버에 직접 저장되는 쿠키라고 생각하면 된다. 특정 사이트에서의 로그인도 쿠키와 세션의 연결에 의해서 가능한 것이다. 만약 해커가 A의 PHPSESSID를 알아내어 해커의 PHPSESSID를 A의 PHPSESSID로 바꾸면 A의 아이디로 로그인할 수 있다. 이러한 해킹 방법을 세션 탈취(Session Hijacking)이라고 한다. 이를 막는 방법 중 하나는 세션에 로그인 했을 때의 IP를 저장해두고, 이후에 세션에 해당하는 IP가 맞는지 확인하는 방법이 있다.



다시 접속을 했을 때에는, 처음에 발급받은 쿠키를 사용하고 있는 것을 볼 수 있다. 보통 세션 쿠키는 expire를 명시하지 않는다고 알고 있었다. 처음 접속할 때 response message를 보면, expire 시간이 과거의 시간으로 설정되어 있는 쿠키를 볼 수 있다. 이는 일종의 세션 쿠키(브라우저가 종료되면 지워지는 쿠키)의 역할 혹은 기존의 쿠키를 지우도록 하는 쿠키인 것 같다.

3. 새로 알게 된 지식



Wireshark를 처음 사용했을 때, 위의 사진과 같은 request message를 반복해서 볼 수 있었다. 처음에는 내가 접속한 사이트에서 문제가 생긴 것 같아 인터넷 브라우저를 종료한 여도 계속해서 나타났다.

검색을 해 보니 Grid라는 것을 알게 되었다. Grid는 네트워크상 연결된 컴퓨터끼리 데이터를 주고받아 회사(서버) 측의 자료를 분산 저장함으로 보다 빠른 데이터 전송을 꾀하는 것이라고 한다. 처음에는 개인의 유휴 컴퓨터 자원을 이용할 수 있다는 장점이 있었지만 이를 악용하면 보안 문제가 발생할 수도 있다고 한다. Application layer protocol P2P를 사용하는 대표적인 예시인 웹하드에서 Grid를 많이 사용한다. 작년 여름에 filejo 라는 웹하드를 사용했는데 그 이후로 계속 유지된 것 같다. 안전을 위해 검색을 통해 알게 된 방법으로 Grid를 삭제하였다.

4. 느낀 점

Wireshark라는 프로그램을 처음 사용하다 보니 사용 방법도 익숙하지 않았고 각 헤더들의 의미도 알기 힘들었다. 헤더에 매칭되는 message들을 보며 유추를 해보고 직접 검색하면서 HTTP가 돌아가는 대략적인 흐름들을 이해하게 되었다. 이를 통해, 수업 때 내용으로 배우는 것들이 실제로 어떻게 사용되는지 직접 체득할 수 있었다.

특히, Web Cache나 Cookie처럼 수업 시간에만 들었던 내용들을 Wireshark를 통해 어떻게 이용되는지 이해할 수 있었다. Grid에 대한 내용 또한 Wireshark를 이용하지 않았다면 전혀 모른 채로 살고 있었을 것 같다.