[네트워크] HTTP와 HTTPS의 동작과정(Feat. SSL, DNS)

📆 날짜: **2023-10-18**

❓ 이 주제를 택한 이유

- 문득, HTTP와 HTTPS는 클라이언트와 서버 간 데이터 전송이 발생할 때, 데이터를 암호화 해주는지에 대한 차이라고 들었다. HTTPS는 통신 규약인데, 똑같은 방식으로 암호화, 복호화를 하게 된다면 어차피 중간에 가로채사람이 복호화를 하게 되면 똑같은거 아닌가? 라는 생각이 들었다. 그래서 HTTPS의 암,복호화가 어떻게 이루어지는지 중심으로 알아보고 싶어졌다.

HTTP와 HTTPS의 정의

`HTTP`

- 인터넷 상에서 정보를 주고 받기 위한 프로토콜(약속)
- 클라이언트와 서버 사이에 이루어지는 요청과 응답의 과정을 담는 프로토콜
- HTTPS와의 차이는 암호화 되지 않은 상태로 데이터를 주고 받는 다는 점이다.

`HTTPS`

- 보안이 강화된 HTTP
- 요청, 응답 데이터를 주고 받기 전에 해당 데이터들을 암호화하여 제공하여 보안을 높인다.

HTTP 동작 과정

1. 클라이언트 **URL** 주소를 입력 및 **DNS** 서버를 통해 **IP**를 획득

DNS 동작 과정

기본적으로 http 요청이 발생할 때 마다, DNS 서버에 IP를 요청하는 과정을 거치게 되면, 오버헤드가 과도하게 발생할 것이다. 그래서, 브라우저는 자주 방문하는 사이트는 DNS 캐시를 통해 IP를 저장하여 DNS 서버를 통하는 오버헤드를 최소화 한다.

브라우저가 **DNS** 캐시를 찾는 과정은 다음과 같다**.**

1. 브라우저 내부 캐시를 확인

   -> 이전에 브라우저에서 방문한 사이트인지 확인하는 절차

2. 운영체제 내부 캐시 확인

   -> 브라우저에 없는 도메인이면, 운영체제 내부에 있는 DNS 캐시에 해당 도메인이 등록되어 있는지 확인한다.

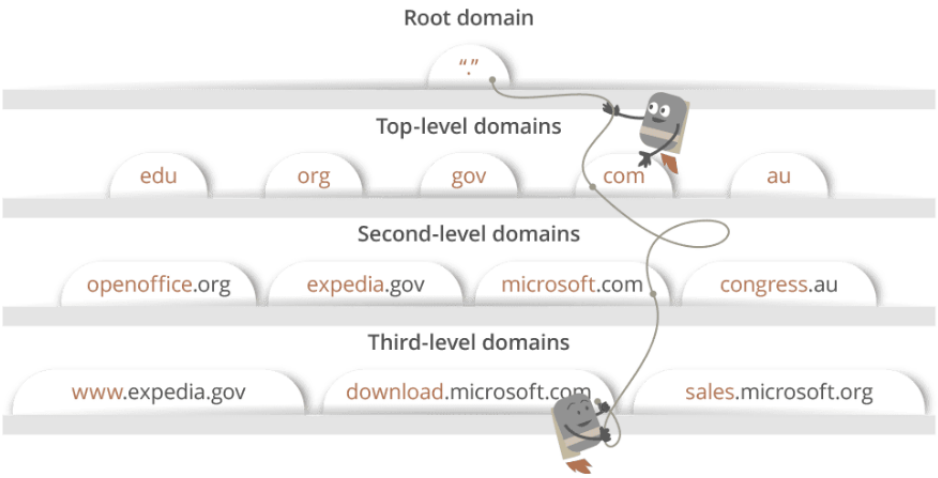
3. 라우터 캐시 확인

   -> OS 캐시에 일치하는 DNS 기록이 없다면 라우터도 DNS 기록을 일시적으로 보관함.

4. **ISP** 캐시 확인

*ISP(Internet Service Provider): 인터넷 서비스 제공 주체(ex. KT, LG, SKT 등)

요청한 URL이 캐시에 없으면 ISP의 DNS server가 재귀적으로 도메인을 찾는다.



📎 도메인 아키텍쳐 **(Domain Architecture)**

- 루트 도메인 (**Root** Domain) ex. "."
- 최상위 도메인 (TLD : **Top Level** Domain) ex. "edu", "org", "gov", "com", "au"
- 2차 도메인 (**Second Level** Domain) ex. "openoffice.org", "expedia.gov", "microsoft.com"
- 3차 도메인 (**Third Level** Domain) ex. "www.openoffice.org", "download.microsoft.com"

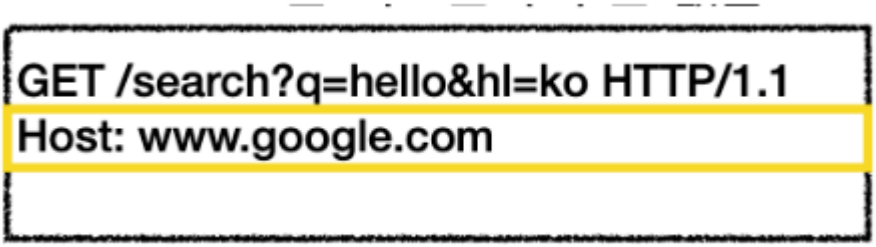
이렇게 주소창에 입력한 도메인을 IP주소로 변환하고, 이를 통해 해당 서버와 TCP 연결을 시작한다.

2. 웹 서버와 **TCP** 연결 시도 **(3-way handshaking)**을 통해 통신 연결 수립**,** 소켓 개방진행
    - 우리가 알고 있는 3-way-handshaking 과정을 통해 서버와 통신 가능 여부를 확인하고 연결 준비를 한다.

3. 클라이언트가 서버에게 요청 **(Request)**

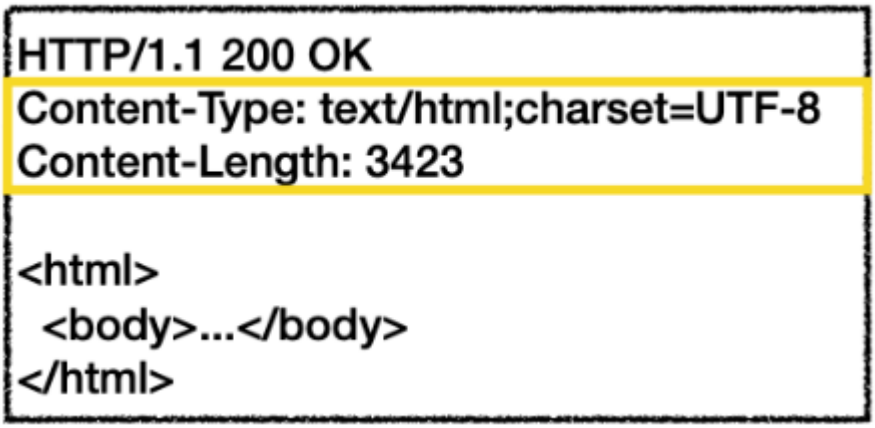
    그 이후에 우리가 알고 있는 HTTP의 응답객체를 주고 받는 과정이 나온다.

    url에 https://www.google.com을 쳤을 때 전송하는 Request 메시지



4. 서버는 클라이언트에게 응답 **(Response)**

브라우저가 해당 객체를 보내면 서버에서 주는 응답 객체를 받게 된다.



5. 연결 종료 **(4-way handshaking)**
    - 브라우저와 서버가 응답을 주고 받았기 때문에, 4-way-handshaking 과정을 통해 연결을 종료한다.

6. 클라이언트는 응답을 기반하여 웹 문서 출력
    - 서버로 부터 받은 응답 메시지를 통해 브라우저는 html과 css를 렌더링하고 자바스크립트를 통해 이벤트를 발생시킨다. 이 과정을 반복하면서 브라우저와 웹과의 TCP/IP 통신이 지속적으로 이루어진다.

그렇다면 **HTTPS**는**?**

HTTP와의 차이는 **three-way handshake** 과정에서 **SSL** 인증을 시도한다**.**

위에서 확인한 것처럼 TCP 연결은 연결 시작할 때, 3-way-handshaking을 통해 연결 상태가 정상인지 확인 한다. HTTPS는 위에서 설명한 HTTP연결 과정 중 3-way-handshaking에서 서버에 있는 ssl 인증서를 확인하고, 서버와 클라이언트 Session Key를 활용해 데이터 암복호화를 요청한다.

SSL 인증을 알기 위해 먼저 알아야할 대칭키, 공개키

`대칭키`

- 암호화와 복호화에 사용하는 키가 같을 경우 이를 대칭키 방식이라고 한다.
- 수신자와 송신자가 같은 키를 갖기 때문에, 둘 다 공유키를 가져야한다. 또한, 대칭키를 전달하는 과정에서 유출하게 되면, 데이터 보안에 위험이 발생하게 된다.

`공개키(비대칭키)`

- 공개키는 대칭키의 단점을 보완하는 방식이다.
- 대칭키와 반대로, 암호화와 복호화에 사용하는 키가 다르다.
- 공개키 방식은 두 개의 키를 갖고 있다. 두 개의 키를 각각 A키와 B키라고 가정을 한다면, A키로 암호화된 데이터를 B키로 복호화 할 수 있고, B키로 암호화된 데이터는 A키로 복호화할 수 있다.

  브라우저는 A키를 갖고 데이터를 복호화하게 되면, B키는 서버만이 갖고 있기 때문에, 해커에 의해 브라우저가 제공한 데이터를 탈취당하더라도, 해커는 데이터를 복호화를 할 수 없다.
- 의문점
  - 이 방식에서는 A키를 알고 있는 해커가 응답 데이터를 탈취하면, 그대로 데이터를 탈취당할 수 있지 않을까 라는 의문점이 생긴다.

Handshaking 과정에서 **SSL**인증을 확인하는 과정

1. 클라이언트는 랜덤데이터 생성하여 서버 전달(Client Hello 단계)
2. 서버측에서 생성한 무작위 데이터와 해당 서버 인증서를 실어서 클라이언트에게 보냄(Server Hello 단계)
3. 브라우저에 내장된 CA정보로 서버 인증서를 비대칭키 시스템을 통해 확인

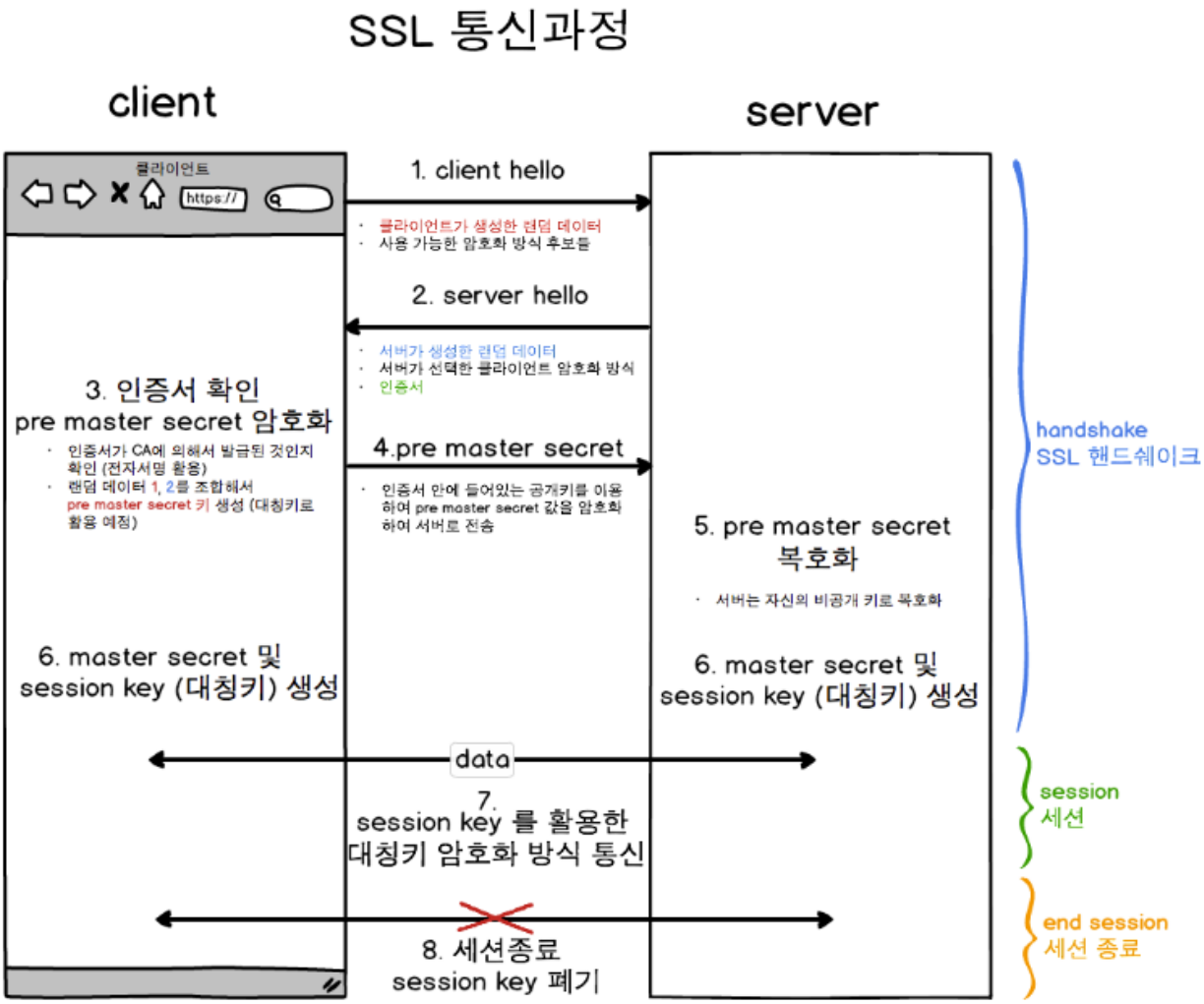
   3-1 브라우저에 저장된 CA의 공개키로 인증서 복호화

   3-2 만약 복호화가 되지 않는다면 정상적인 서버 인증서가 아님으로 브라우저 주소창에 Not secure뜸

   3-3 복호화가 된다면, Server Hello 단계에서 함께 받아온 ssl 인증서를 통해 pre master secret 키 생성한다.
4. 인증서에 있는 공개키로 pre master secret key를 암호화해서 서버로 보냄
5. 서버는 자신의 비공개 키로 복호화
6. 클라이언트와 마찬가지로 session key 생성
7. 이후 서로 주고받는 메시지는 대칭키로 보안유지

즉**, HTTPS**의 보안 통신 과정은 위에서 언급한 공개키를 통해 대칭키를 만들어 사용한다**.**

CA: Certification Authority의 약어로, 인증 기관을 의미한다.



의문점

❓ 브라우저는 받은 인증서가 **CA**에 의해 발급된 것인지 확인을 어떻게 하는가**?**

- 브라우저는 인증된 CA 리스트와 공개키를 브라우저에 저장한다. 그렇기 때문에, 서버가 브라우저에 등록된 CA의 인증서를 활용해 SSL 인증서를 발급했다면, 브라우저는 SSL 인증서를 받을 때, 이를 내부에서 탐색한 후에 해당하는 인증서가 있다면, 이를 공개키로 활용 한다.

그 외로 추가 되었으면 하는 내용이 있으면 말씀해주세요.

참고 자료

https://velog.io/@narcoker/네트워크-HTTP와-HTTPS-동작-과정 - HTTP와 HTTPS 동작 과정 요약 설명

https://velog.io/@kti0940/HTTPS-SSL-복호화는-어디서-진행될까 - HTTPS SSL 복호화 설명

https://wayhome25.github.io/cs/2018/03/11/ssl-https/ - SSL 동작 과정 설명

HTTPS 통신과정 쉽게 이해하기 #5(CA, 인증기관) - CA(인증기관) 설명

HTTPS와 SSL 인증서 - 생활코딩 -생활코딩 유튜브 SSL 인증서 설명

https://www.cloudflare.com/ko-kr/learning/ssl/what-is-a-session-key/ - cloudflare 홈페이지 세션키 설명

https://www.cloudflare.com/ko-kr/learning/ssl/what-happens-in-a-tls-handshake/ - cloudflare - TLS(SSL) handshake 설명

HTTP 메시지 - dodeon - HTTP 메시지 구조