<HTTP & HTTPS>

HTTP(HyperText Transfer Protocol)

인터넷 상에서 클라이언트와 서버가 자원을 주고 받을 때 쓰는 통신 규약

HTTP는 텍스트 교환이므로, 누군가 네트워크에서 신호를 가로채면 내용이 노출되는 보안 이슈가 존재한다.

이런 보안 문제를 해결해주는 프로토콜이 **'HTTPS'**

HTTPS(HyperText Transfer Protocol Secure)

인터넷 상에서 정보를 암호화하는 SSL 프로토콜을 사용해 클라이언트와 서버가 자원을 주고 받을 때 쓰는 통신 규약

HTTPS는 텍스트를 암호화한다. (공개키 암호화 방식으로!) : 공개키 설명

HTTPS 통신 흐름

1) 애플리케이션 서버(A)를 만드는 기업은 HTTPS를 적용하기 위해 공개키와 개인키를 만든다.
2) 신뢰할 수 있는 CA 기업을 선택하고, 그 기업에게 내 공개키 관리를 부탁하며 계약을 한다.

CA란? : Certificate Authority로, 공개키를 저장해주는 신뢰성이 검증된 민간기업

3) 계약 완료된 CA 기업은 해당 기업의 이름, A서버 공개키, 공개키 암호화 방법을 담은 인증서를 만들고, 해당 인증서를 CA 기업의 개인키로 암호화해서 A서버에게 제공한다.
4) A서버는 암호화된 인증서를 갖게 되었다. 이제 A서버는 A서버의 공개키로 암호화된 HTTPS 요청이 아닌 요청이 오면, 이 암호화된 인증서를 클라이언트에게 건내준다.
5) 클라이언트가 main.html 파일을 달라고 A서버에 요청했다고 가정하자. HTTPS 요청이 아니기 때문에 CA기업이 A서버의 정보를 CA 기업의 개인키로 암호화한 인

증서를 받게 된다.

CA 기업의 공개키는 브라우저가 이미 알고있다. (세계적으로 신뢰할 수 있는 기업으로 등록되어 있기 때문에, 브라우저가 인증서를 탐색하여 해독이 가능한 것)

6) 브라우저는 해독한 뒤 A서버의 공개키를 얻게 되었다. 이제 A서버와 통신할 대는 얻은 A서버의 공개키로 암호화해서 요청을 날리게 된다.

HTTPS도 무조건 안전한 것은 아니다. (신뢰받는 CA 기업이 아닌 자체 인증서 발급한 경우 등) 이때는 HTTPS지만 브라우저에서 주의 요함, 안전하지 않은 사이트와 같은 알림으로 주의 받게 된다.

Reference

https://jeong-pro.tistory.com/89