7. 웹을 안전하게 지켜주는 HTTPS

↗ Books	그림으로 배우는 HTTP & Network Basic

1️⃣ HTTP의 약점

HTTP는 도청, 위장, 변조가 가능하다는 약점을 가지고 있다.

1. 평문이기 때문에 도청 가능

HTTP를 사용한 리퀘스트나 리스폰스 통신 내용은 암호화되지 않기 때문에 TCP/IP 구조의 통신 내용은 전부 통신 경로 도중에 엿볼 수 있다.

네트워크 상을 흐르고 있는 패킷을 수집하는 것만으로 도청이 가능하다.

이와 같은 문제를 해결하기 위해 몇 가지 암호화 방식을 사용한다.

1. 통신 암호화
   SSL 등을 이용해 안전한 통신로를 확립하고 나서 그 통신로를 사용해 HTTP 통신을 한다.
   이렇게 SSL을 조합한 HTTP를 HTTPS나 HTTP over SSL이라고 부른다.
2. 콘텐츠 암호화
   콘텐츠 내용 자체를 암호화하는 방식이다.
   이 경우 클라이언트에서 HTTP 메시지를 암호화해서 출력하는 처리가 필요하기 때문에 유저가 사용하는 브라우저나 웹 서버에서 이용하는 것은 어렵고 주로 웹 서비스 등에서 이용되는 방법이다.

2. 통신 상대를 확인하지 않기 때문에 위장 가능

HTTP 통신은 리퀘스트를 보낸 서버가 정말 URI에서 지정된 호스트인지, 리스폰스를 반환한 클라이언트가 정말 리퀘스트를 출력한 클라이언트인지 모른다.

하지만 SSL로 상대를 확인하는 것이 가능하다. SSL은 암호화뿐만 아니라 상대를 확인하는 수단으로 증명서를 제공한다.

3. 완전성을 증명할 수 없기 때문에 변조 가능

완전성이란 정보의 정확성을 가리킨다.

HTTP가 완전성을 증명할 수 없다는 뜻은 통신 과정에서 변조되었다고 하더라도 이 사실을 알 수 없다는 뜻이다.

예를 들어, 어떤 웹 사이트에서 콘텐츠를 다운로드 했다고 하면 서버 상에 있는 파일과 정말 동일한지 모른다는 것이다.
이처럼 공격자가 중간에 리퀘스트나 리스폰스를 빼앗아 변조하는 공격을 중간자 공격(Man-in-the-Middle 공격)이라고 부른다.

HTTP를 사용해서 완전성을 확인하기 위한 방법은 있지만, 확실하면서 편리한 방법은 현재로서는 존재하지 않는다.
그 중 자주 사용하는 방법은 MD5나 SHA-1 등의 해시 값을 확인하는 방법과 파일의 디지털 서명을 확인하는 방법이다.

2️⃣ HTTP + 암호화 + 인증 + 완전성 보호 = HTTPS

HTTP에 암호화나 인증 등의 구조를 더한 것을 HTTPS(HTTP Secure)라고 부른다.

보통 HTTP는 직접 TCP와 통신하지만 SSL을 사용한 경우에는 HTTP는 SSL과 통신하고 SSL이 TCP와 통신하게 된다.

3️⃣ 상호간에 키를 교환하는 공개키 암호화 방식

SSL에서는 공개키 암호화 방식이라고 불리는 암호화 방식을 채용하고 있다.
공개키 암호화 방식은 비밀키(private key)-공개키(public key)로 이루어진 키 페어를 사용한다.
암호를 보내는 측은 암호화에 상대의 공개키를 사용하고 암호를 받는 측은 복호화에 자신의

비밀키를 사용한다.
키를 통신으로 보낼 필요가 없기 때문에 도청에 의해 빼앗길 걱정이 없는 것이다.

HTTP는 공통키 암호화 공개키 암호의 양쪽 성질을 가진 하이브리드 암호 시스템이다.
키를 교환하는 곳에서는 공개키 암호를 사용하고 그 후의 통신에서 메시지를 교환하는 곳에서는 공통키 암호를 사용한다.

4️⃣ 공개키가 정확한지 아닌지를 증명하는 증명서

하지만 공개키 암호에도 문제점이 있다. 공개키가 진짜인지 증명할 수 없다는 것이다.
이 문제를 해결하는 데는 인증 기관과 그 기관이 발행하는 공개키 증명서가 이용되고 있다.

인증 기관은 다음과 같이 이용된다.

서버 운영자가 인증 기관에 공개키를 제출한다.
→ 인증 기관은 제출된 공개키에 디지털 서명을 하고 서명이 끝난 공개키를 만든다.
→ 공개키 인증서에 서명이 끝난 공개키를 담는다.

서버는 이 인증 기관에 의해 작성된 공개키 인증서를 클라이언트에 보내고 공개키 암호로 통신한다.

- **조직의 실제성을 증명하는 EV SSL 증명서**

상대방이 실제로 있는 기업인지 확인하는 역할을 가진 증명서를 EV SSL 증명서라고 한다.

- **클라이언트를 확인하는 클라이언트 증명서**

서버가 통신하고 있는 상대가 의도한 클라이언트인 것을 증명하는 역할을 한다.

클라이언트 증명서는 몇 가지 문제점이 있다.

1. 클라이언트 증명서는 유료로 구입을 해야 하기 때문에 유저 수만큼 비용이 들게 된다. 때문에 그 만큼 비용을 들일 필요가 있는 인터넷 뱅킹 등에서만 사용되고 있다.
2. 사용자의 존재 유무를 증명하는 증명서가 아니기 때문에 클라이언트 증명서가 들어간 컴퓨터를 사용할 권한이 있다면 누구든지 클라이언트 증명서를 이용할 수 있다.

- **인증 기관은 신용이 제일**

SSL은 인증 기관을 신용할 수 있다는 전제로 이루어져 있다.

잘못 액세스된 인증 기관이 발행한 증명서도 올바른 증명서로 인식하기 때문에 서버로 위장할 때 사용하더라도 유저가 알아채기 어렵다.

- **자기 인증 기관 발행 증명서는 '나야 나' 증명서**

OpenSSL 등의 소프트웨어를 사용하면 누구든지 인증 기관을 구축할 수 있어서 서버 증명서를 발행할 수 있다.
하지만 이 서버 증명서는 인터넷 상에서는 증명서로 구실을 하지 못한다.

독자적으로 구축한 인증 기관을 자기 인증 기관이라고 부르고 거기서 발행한 증명서를 비하해서 '나야 나 증명서'라고 부르는 일이 있었다.

자기 인증 기관이 발행한 서버 증명서는 위장의 가능성을 불식할 수 없기 때문에 안전하지 않다.

자기 인증 기관을 사용하지 않더라도 마이너 인증 기관을 사용하면 '나야 나' 증명서가 될 수 있다.

5️⃣ 안전한 통신을 하는 HTTPS의 구조

통신의 수순을 살펴보자.

1. 클라이언트가 Client Hello 메시지를 송신하면서 SSL 통신을 시작한다.
   메시지에는 클라이언트가 제공하는 SSL 버전을 지정하고, 암호 스위트로 불리는 리스트 등이 포함된다.
2. 서버가 SSL 통신이 가능한 경우 Server Hello 메시지로 응답한다.
   클라이언트와 같이 SSL 버전과 클라이언트에서 받은 암호 스위트에서 선택한 일부 내용을 포함한다.
3. 서버가 Certificate 메시지를 송신한다.
   메시지에는 공개키 증명서가 포함된다.
4. 서버가 Server Hello Done 메시지를 송신하여 최초의 SSL 네고시에이션 부분이 끝났음을 통지한다.
5. SSL의 최초 네고시에이션이 종료되면 클라이언트가 Client Key Exchange 메시지로 응답한다.
   메시지에는 통신을 암호화하는데 사용하는 Pre-Master secret이 포함되며 3번의 공개키 증명서에서 꺼낸 공개키로 암호화되어 있다.

6. 클라이언트는 Change Cipher Spec 메시지를 송신한다.
   이 메시지는 이 메시지 이후의 통신은 암호키를 사용해서 진행한다는 것을 나타낸다.
7. 클라이언트는 Finished 메시지를 송신한다.
   접속 전체의 체크 값을 포함한다. 네고시에이션이 성공했는지는 서버가 이 메시지를 올바르게 복호화할 수 있는지 아닌지가 결정된다.
8. 서버에서도 Change Cipher Spec 메시지를 송신한다.
9. 서버에서 Finished 메시지를 송신한다.
10. 서버와 클라이언트의 Finished 메시지 교환이 완료되면 SSL에 의해 접속은 확립된다.
    HTTP 리퀘스트를 송신한다. 물론 통신은 SSL에 의해 보호되고 있다.
11. HTTP 리스폰스를 송신한다.
12. 클라이언트가 close_notify 메시지를 송신하며 접속을 끊는다.
13. 클라이언트가 TCP FIN 메시지를 보내 TCP 통신을 종료한다.

서버 측의 공개키 증명서만을 이용한 HTTPS에 의한 통신을 시작하여 HTTP에 의한 통신을 개시하는 곳까지의 설명이다.