01.30

📌Https

Https는 기존의 평문 데이터를 전송했던 Http의 암호화 버전으로, SSL이나 TLS 프로토콜을 사용하여 세션 데이터를 암호화하는 방식입니다. 클라이언트(브라우저)와 웹 서버 간의 통신에 사용됩니다.

인증서 발급

1. 먼저 Https를 사용하기 위해 서버는 CA(Certificate Authority, 인증 기관)로 서버를 사용하는 회사의 신원 정보 및 공개키를 제공하여 **SSL 인증서 발급을 요청**합니다. 이 때 서버는 자신의 공개키와 비밀 키를 소유한 상태일 수 있습니다.
2. CA는 받은 서버의 정보를 검토한 뒤, 인증서를 발급해줍니다. 인증서 내부에는 서버 정보와 CA 정보, 인증서에 해시 알고리즘을 사용하여 해시 값을 생성하고, 이를 CA 개인키로 암호화한 전자 서명이 담깁니다. 그리고 인증서는 CA 소유의 개인 키로 암호화됩니다. 한편, 인증서는 DNS 서버처럼 여러 인증 기관 간의 Chain으로 연결되어 신뢰성을 가집니다.
3. 서버는 이제 CA로부터 발급 받은 인증서와, 공개키, 비밀키를 가지게 됩니다.

SSL Handshake(TLS Handshake)

4. 클라이언트는 접속하고자 하는 서버에 인증서를 요청합니다. 이 때 Cipher suite, 세션 id, ssl protocol version 등을 함께 전달합니다.
5. 서버는 인증서와 선택한 Cipher Suite를 전달하고, 이를 받은 사용자는 CA에게 인증서 검증을 요청합니다. 인증서를 CA의 비밀키로 암호화했으므로, CA의 공개키를 통해 복호화할 수 있습니다.
6. CA는 CA 공개키를 클라이언트에게 전달하고, 클라이언트는 이를 통해 인증서를 복호화하여 서버의 공개키와 서버의 신원에 대한 정보를 얻습니다. 전자 서명을 통해 데이터 위변조 여부도 판단할 수 있습니다. 이 과정이 끝나면, 클라이언트는 서버를 신뢰할 수 있게 됩니다.

   만일 공개키가 인증서에 포함되지 않았다면, 서버는 DH 알고리즘과 ECDHE를 사용한 것입니다. 이 때는 파라미터 값을 보내는 server key exchange가 발생합니다.
7. 클라이언트는 데이터 암호화에 사용할 대칭키를 만들고, 이를 서버의 공개키로 암호화하여 서버에 전송합니다. 이를 Client key Exchange라고 합니다. 서버는 클라이언트로

부터 암호화된 대칭 키를 받고, 이를 서버의 개인 키로 복호화합니다. 이제 같은 대칭키를 통해 세션 내에서 통신이 가능합니다. 만일 RSA가 아닌 DH 알고리즘, ECDHE를 사용한 것이었다면 서버로 파라미터 값을 보내어 서로 간 주고 받은 파라미터 값을 사용하여 대칭키를 생성합니다.

8. 서버와 클라이언트 모두 교환할 정보를 모두 교환한 뒤 통신 준비가 되었으면, Finished 패킷을 보내어 ssl handshake를 종료하게 됩니다.

***키 교환 방식**

키 교환 시 주로 **RSA, ECDHE(**DH계열**)**가 사용되는데, RSA의 경우 공개키와 비밀키를 사용하는 방식이고, ECDHE는 양측 간 키를 생성할 재료인 Parameter를 교환하는 방식입니다. 이 때 Parameter는 노출되어도 상관 없습니다. RSA의 경우 client key exchange 때 클라이언트가 생성한 대칭키를 공개키로 암호하한 뒤 서버로 전송하고, DH의 경우 client key exchange와 server key exchange 때 키 매개변수를 교환하여 대칭키를 얻습니다. 따라서 ssl 인증서 내부에 서버의 공개키가 존재하지 않습니다.

한편, RSA는 서버의 비밀키 탈취 시 해당 비대킹키를 사용한 모든 과거의 네트워크 패킷 정보가 복호화될 수 있으므로 안전하지 않다고 볼 수 있습니다. 반면 DHE 방식은 탈취 당하더라도 일정 기간의 데이터만 탈취된 것이므로 비교적 안전하다고 볼 수 있습니다.

📌AOP

- Aspect Oriented Programming. 관점 지향 프로그래밍.

  코드를 핵심 관심사와 횡단 관심사로 구분하여 모듈 별로 핵심 관심사에 집중하고, 프록시 패턴을 통해 컴파일 시, 혹은 런타임 시 횡단 관심사를 주입하는 것을 의미합니다.

- 횡단 관심사(cross-cutting corcern)는 다수의 모듈에 공통적으로 나타나는 코드 부분을 말하며, 핵심 관심사는 모듈 별로 다르게 나타나는 코드, 즉 비즈니스 로직을 의미합니다.

  횡단 관심사를 분리함으로써, 모듈은 SRP를 준수할 수 있게 됩니다.

- 스프링에서 사용되는 AOP 라이브러리는 AspectJ, JDK Dynamic Proxy, CGLib 등이 있습니다.

용어

- Pointcut

Aspect 적용 위치 지정자로, 실제 횡단관심사를 적용할 위치를 선정하는 역할을 합니다. Spring에서는 Aspect를 Target 메서드에만 적용 가능하므로, Pointcut을 메서드 선정 알고리즘이라고 부르기도 합니다.

- JoinPoint

  넓은 의미와 좁은 의미로 해석이 가능합니다. 넓은 의미로는 Aspect 적용이 가능한 모든 지점을 말합니다. 스프링 AOP에서는 스프링 프레임워크가 관리하는 빈의 모든 메서드에 해당합니다.

  좁은 의미의 JoinPoint는 Advice의 인자로 사용되는 Joinpoint 인터페이스를 의미하며, 이는 Target 객체의 메서드를 말합니다.

- Advice

  Pointcut에 적용할 로직, 즉 메서드와 시점(before, after, after-throwing, after-returning, Around)을 포함한 개념으로, Pointcut에 언제, 무엇을 적용할지를 정의한 메서드를 말합니다.

- Advisor

  1개의 Advice와 1개의 Pointcut이 결합된 형태를 말합니다.

- weaving

  Aspect가 지정된 객체를 새로운 프록시 객체를 생성하는 과정을 말합니다.