인터넷에서 사용되는 여러 기술 : HTTPS 9

조인준
KBS 미디어기술연구소 차장

TLS 1.2에서 TLS 1.3으로 발전하는 과정에서 가장 큰 변화를 보인 부분은 핸드셰이크와 암호화 스위트(Cipher Suite) 관련입니다. 지난 편들을 통해 핸드셰이크의 차이점과 TLS 1.2 암호화 스위트의 특징을 간략히 살펴보았습니다.
이번 편에서는 TLS 1.3의 암호화 스위트가 TLS 1.2와 어떻게 다른지에 관해 다루어 보겠습니다. SSL/TLS 기술 발전에 대한 이야기를 시작할 때 TLS 1.3은 이전 버전인 TLS 1.2의 보안 결함과 성능 이슈 개선을 위해 설계되었고 이런 변화와 개선에서 가장 중요하게 작용한 원칙은 보안성과 단순화이며 이를 위해 하위호환(Backward Compatibility)을 포기한 부분도 있다고 소개해 드렸습니다. 예를 들어 [표 1]에 나열된 TLS 1.2 암호화 스위트의 요소 중 암호화 계열의 DES 방식은 1975년에 발표되어 2006년에 TLS 1.1에 포함된 것이고, DES 방식을 개선한 3DES 방식은 1981년에 발표되어 2008년 TLS 1.2에 포함되었습니다. 이렇듯 TLS 1.2까지는 기존 버전들과의 호환성을 위해 오래된 기술도 지원하다 보니 이것이 오히려 보안에 불안한 요소가 되는 경우도 생겼습니다. TLS 1.3에서는 보안성 강화를 위해 이런 기술들의 하위호환을 포기하고 암호화 스위트를 단순화합니다.

키 교환 (Key Exchange)	인증 (Authentification)	암호화 (Encryption)	해시 (Hash)
RSA DH DHE ECDH PSK 등	RSA ECDSA DSS PSK 등	AES-128-CBC AES-256-CBC AES-128-GCM AES-256-GCM 3DES-CBC DES-CBC RC4-128 등	SHA-1 SHA256 SHA384 MD5 등

표 1. TLS 1.2 암호화 스위트 알고리즘 예 (TLS 1.2)

어떤 방식으로 TLS 1.3에서 암호화 스위트가 단순화되었는지 자세히 들어가 보겠습니다. TLS 1.2에서 암호화 스위트는 [표 1]의 요소별 알고리즘을 조합하여 [그림 1]과 같은 형식으로 표시되고 핸드셰이크 단계에서 Client Hello에 [표 2]의 예와 같이 클라이언트가 지원 가능한 암호화 스위트의 리스트를 서버에 보냅니다. [표 2]에서 인증 알고리즘이 빠진 경우는 키 교환 알고리즘과 동일 알고리즘을 인증에 사용하는 경우입니다.

TLS_[키 교환]_[인증]_WITH_[암호화]_[해시]

그림 1. 암호화 스위트 형식

TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
TLS_PSK_WITH_AES_128_CBC_SHA256
TLS_DH_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

표 2. 클라이언트 지원 암호화 스위트 리스트 예

그림 2. TLS 1.3에서의 암호화 스위트 단순화

[그림 1]과 같은 방식으로 [표 1]의 요소별 알고리즘을 조합하는 경우 TLS 1.2 암호화 스위트의 개수는 300개를 넘는다고 알려져 있습니다. 이런 가운데 TLS 1.2의 수많은 암호화 스위트 중 상당량이 현대적 관점에서 보안 우려가 있다고 합니다. 그래서 TLS 1.3에서는 TLS 1.2에서 하나로 묶여 있던 키 교환, 인증, 암호화, 해시의 묶음을 독립된 세 단위로 [그림 2]와 같이 분리하여 암호화 스위트를 단순화하였습니다.

TLS 1.3에서 [그림 2]와 같이 독립적으로 분리된 세 요소를 각각 설명하면 다음과 같습니다.

키 교환

- **Forward Secrecy (순방향 비밀성)**
  - 제3자가 암호화된 통신 데이터를 취득하여 보유할 경우 암호화 키가 유출되면 과거의 통신 내용이 모두 해독되어 민감 정보가 제3자의 손에 들어갈 수 있음
  - 클라이언트와 서버가 각각의 세션(Session, 서버와 클라이언트의 논리적 연결)마다 임시로 키를 생성하여 암호화된 통신을 하면 현재 세션에서 암호화 키가 유출되어도 그 키로 과거 세션의 데이터를 해독할 수 없고, 마찬가지로 미래의 세션에서 암호화 키가 유출되어도 현재의 세션을 해독할 수 없음
  - 키 유출로 인해 과거에 주고받은 암호화된 데이터가 해독되어 정보가 유출되는 사고를 방지하는 것을 Forward Secrecy라고 함

- **TLS 1.3에 사용되는 키 교환 알고리즘**
  - TLS 1.2까지 사용되던 RSA의 경우 Forward Secrecy에 약점이 있음
  - RSA 이외에도 Forward Secrecy에 약점이 있는 알고리즘을 TLS 1.3에서는 제거
  - TLS 1.3은 [표 3]의 네 가지 알고리즘만을 키 교환에 사용함
  - DHE와 ECDHE 같이 마지막에 E(Ephemeral의 약자)가 붙은 알고리즘은 세션마다 임의로 키를 만들어 암호화를 수행하므로 Forward Secrecy를 보장
  - PSK는 클라이언트와 서버가 미리 공유한 비밀 키를 사용하여 통신을 설정하는 방식이며 Forward Secrecy를 보장하는 방식은 아님
  - PSK가 TLS 1.3에 남아 있는 이유는 IoT 디바이스와 같이 저전력에서 동작하는 장치들의 경우 성능 문제로 PSK와 같은 가벼운 프로토콜을 필요로 함
  - 내부 네트워크에서의 통신이나 일회성 세션 등에서는 PSK를 사용하는 것이 적절한 경우도 있음

DHE (Diffie-Hellman Ephemeral)
ECDHE (Elliptic Curve Diffie-Hellman Ephemeral)
Pre-Shared Key (PSK)
PSK with (EC)DHE

표 3. TLS 1.3 키 교환 알고리즘

인증

• **디지털 서명과 인증**

- 공용키와 사설키 쌍을 이용하는 비대칭키 암호화 방식을 이용
- 상대에게 자신의 신원을 증명하는 인증서의 해시값을 사설키로 암호화한 후 인증서와 같이 보내는 것이 디지털 서명
- 디지털 서명된 인증서(암호화된 해시값 포함)를 수신 후 자체적으로 계산한 인증서 해시값과 상대의 공용키로 복원한 해시값(수신된 해시값)이 같다면 인증서를 신뢰할 수 있으며 누군가 중간에서 변조하지 않았음을 확신할 수 있음

• **TLS 1.3에서는 아래 세 가지 비교적 강력한 인증 알고리즘만 사용**

- RSA
- ECDSA (Elliptic Curve Digital Signature Algorithm)
- EdDSA (Edwards-curve Digital Signature Algorithm)

암호화 스위트

• **TLS 1.3 암호화 스위트**

- TLS 1.3에서는 [표 4]의 다섯 가지 암호화 스위트만 사용
- AEAD(Authenticated Encryption with Associated Data)가 적용된 암호화 알고리즘([표 4] 파란색 표시)과 키 교환 등에 사용하는 해시 알고리즘([표 4] 주황색 표시)의 조합으로 구성

TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_CCM_SHA256
TLS_AES_128_CCM_8_SHA256

표 4. TLS 1.3 암호화 스위트

• **AEAD(Authenticated Encryption with Associated Data)**

- TLS 1.3은 암호화에 AEAD 방식을 사용([표 4] 파란색 표시)
- 데이터의 암호화를 통한 기밀성과 인증을 통한 무결성(데이터가 전송 중에 변경되지 않았음)을 동시에 보장하는 암호화 방식
- 암호화와 수신 데이터 인증을 결합하여 암호화된 데이터의 변조 여부를 확인할 수 있게 한 방식으로 보안 프로토콜에서 중요한 역할을 함

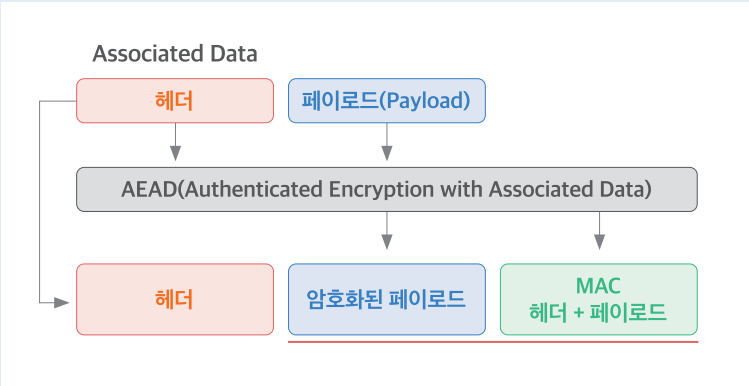

그림 3. AEAD 적용 결과

- 헤더 등에 해당할 수 있는 Associated Data(연관 데이터)는 암호화되지 않지만 인증 관련 데이터인 MAC(Message Authentication Code)에 포함
- 분리된 암호화와 인증 절차를 단일 과정으로 단순화
- 암호화된 데이터의 무결성을 보장함으로써 데이터 변조 및 제3자 공격에 대응
- [그림 3]은 헤더를 Associated Data로 설정한 AEAD 알고리즘 적용 결과에 대한 예시

- 예로써 Associated Data(연관 데이터)로 IP 주소와 포트 번호 등이 포함된 TCP 헤더를 사용할 경우 제3자가 추후 암호화된 페이로드 및 MAC 데이터를 재활용하여 악용하려 할 경우 헤더 데이터(IP 주소와 포트 번호 등)와 MAC 데이터의 불일치로 무결성에 문제가 있음을 발견할 수 있음

• **해시**

- TLS 1.3 핸드셰이크 과정에서 해시 함수인 SHA-256 등을 사용하여 보안을 강화
- 이를 통해 암호화된 통신 세션에서 사용할 대칭 키를 안전하게 생성
- 해시값을 이용하여 핸드셰이크 무결성 보장 등에도 이용

지금까지 TLS 1.2와 TLS 1.3의 차이에 대해 이야기해보았습니다. 그렇다면 지금 내가 접속하는 웹사이트가 보안이 적용되어 있는지, 적용되어 있다면 어떤 TLS 버전을 사용하는지 알 수 있을까요? 네, 알 수 있는 방법이 있습니다. 만약 여러분이 '구*'의 '크*' 브라우저를 사용하고 계신다면 아래와 같이 쉽게 보안 관련 사항을 확인할 수 있습니다. 다른 브라우저에서도 '크*'과 같이 보안 관련 정보를 확인할 수 있을 겁니다.

보안 적용 여부

'크*' 브라우저로 웹사이트에 접속 후 주소 입력창 왼쪽에 [그림 4]의 빨간 원으로 표시한 부분을 누르면 메뉴창이 펼쳐집니다. 이 메뉴창에 [그림 4]의 녹색 사각형으로 표시한 것과 같이 '이 연결은 안전합니다'라는 표시가 있으면 현재 접속한 웹사이트에 TLS 보안이 적용되어 있음을 나타냅니다.

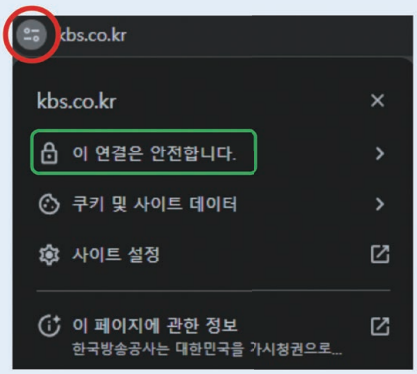

그림 4. 브라우저에서 웹사이트 보안 확인

TLS 버전 확인

'크*' 브라우저로 웹사이트에 접속 후 [그림 5] 우상단의 빨간 원으로 표시된 부분을 누르면 메뉴창이 열리고 이 메뉴에서 '도구 더보기' → '개발자 도구'를 선택하면 [그림 5]와 같이 여러 가지 데이터를 조회할 수 있는 창이 열립니다. 이 창의 상단 탭에서 [그림 5] 노란색으로 표시된 Security 메뉴를 선택하면 녹색 사각형으로 표시한 것과 같이 현재 웹사이트의 TLS 버전과 AEAD 암호화 알고리즘을 확인할 수 있습니다. [그림 5]에 보이는 TLS 버전은 1.3이며 AEAD 암호화 알고리즘은 AES_128_GCM입니다.

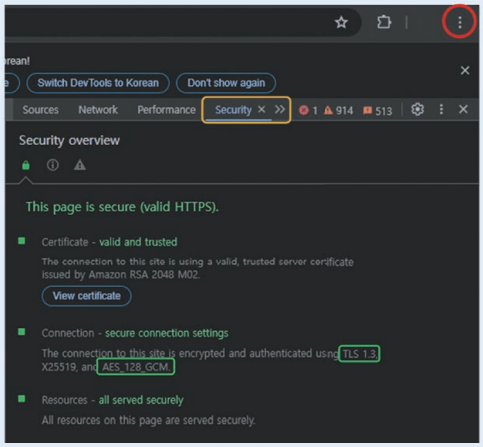

그림 5. 브라우저를 통한 TLS 버전 확인

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다.
본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.