# Expressway SSL 암호화 컨피그레이션 사용자 지정

## 목차



## 소개

이 문서에서는 Expressway에서 사전 구성된 암호 문자열을 사용자 지정하는 단계에 대해 설명합니다.

## 사전 요구 사항

### 요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Cisco Expressway 또는 Cisco VCS
- TLS 프로토콜.

### 사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- Cisco Expressway 버전 X15.0.2

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바

이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

# 배경 정보

기본 Expressway 컨피그레이션에는 사전 구성된 암호 문자열이 포함되어 있습니다. 이 문자열은 호환성의 이유로 일부 기업 보안 정책에서 약한 것으로 간주될 수 있는 일부 암호에 대한 지원을 활성화합니다. 각 환경의 특정 정책에 맞게 세부적으로 조정하기 위해 암호 문자열을 사용자 지정할 수 있습니다.

Expressway에서는 다음 각 프로토콜에 대해 독립적인 암호 문자열을 구성할 수 있습니다.

- HTTPS
- LDAP
- 역방향 프록시
- SIP
- SMTP
- TMS 프로비저닝
- UC 서버 검색
- XMPP

암호 문자열은 OpenSSL Ciphers Manpage에 설명된 OpenSSL 형식을 따릅니다. 현재 Expressway 버전 X15.0.2에는 모든 프로토콜에 대해 동일하게 사전 구성된 기본 문자열 EDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH가 포함되어 있습니다. 웹 관리 페이지의 Maintenance > Security > Ciphers에서는 공통 알고리즘을 사용하여 특정 암호 또는 암호 그룹을 추가 또는 제거하기 위해 각 프로토콜에 할당된 암호 문자열을 수정할 수 있습니다.

## 암호 문자열 검사

openssl ciphers -V "<cipher string>" 명령을 사용하면 특정 문자열이 허용하는 모든 암호가 포함된 목록을 출력할 수 있습니다. 이는 암호를 시각적으로 검사하는 데 유용합니다. 다음 예에서는 기본 Expressway 암호 문자열을 검사할 때의 출력을 보여 줍니다.

```
<#root>

~ #

openssl ciphers -V "EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH"


0x13,0x02 - TLS_AES_256_GCM_SHA384 TLSv1.3 Kx=any Au=any Enc=AESGCM(256) Mac=AEAD
0x13,0x03 - TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any Au=any Enc=CHACHA20/POLY1305(256) Mac=AEAD
0x13,0x01 - TLS_AES_128_GCM_SHA256 TLSv1.3 Kx=any Au=any Enc=AESGCM(128) Mac=AEAD
0xC0,0x2C - ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(256) Mac=AEAD
0xC0,0x30 - ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(256) Mac=AEAD
0xCC,0xA9 - ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2 Kx=ECDH Au=ECDSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
0xCC,0xA8 - ECDHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=ECDH Au=RSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
0xC0,0xAD - ECDHE-ECDSA-AES256-CCM TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM(256) Mac=AEAD
0xC0,0x2B - ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(128) Mac=AEAD
0xC0,0x2F - ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(128) Mac=AEAD
```

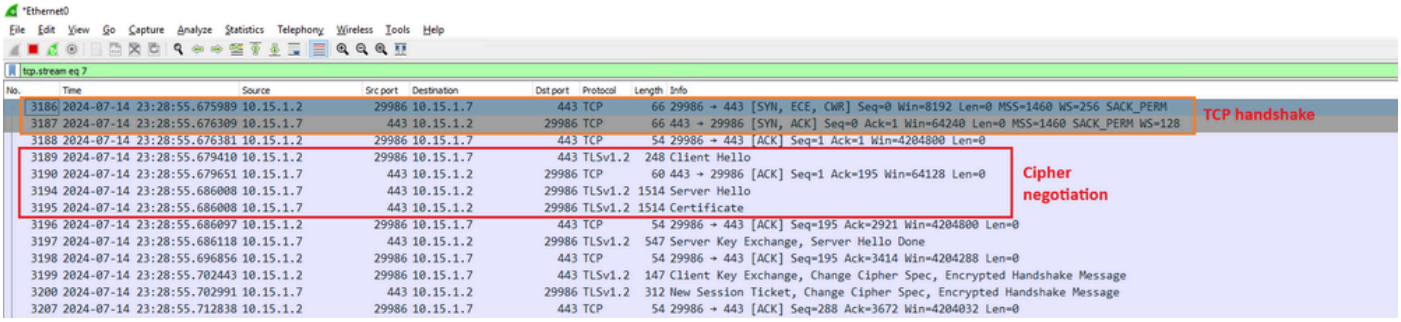
```
0xC0,0xAC - ECDHE-ECDSA-AES128-CCM TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM(128) Mac=AEAD
0xC0,0x24 - ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(256) Mac=SHA384
0xC0,0x28 - ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(256) Mac=SHA384
0xC0,0x23 - ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA256
0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA256
0xC0,0x09 - ECDHE-ECDSA-AES128-SHA TLSv1 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA1
0xC0,0x13 - ECDHE-RSA-AES128-SHA TLSv1 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA1
0x00,0xA3 - DHE-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH Au=DSS Enc=AESGCM(256) Mac=AEAD
0x00,0x9F - DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(256) Mac=AEAD
0xCC,0xAA - DHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=DH Au=RSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
0xC0,0x9F - DHE-RSA-AES256-CCM TLSv1.2 Kx=DH Au=RSA Enc=AESCCM(256) Mac=AEAD
0x00,0xA2 - DHE-DSS-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AESGCM(128) Mac=AEAD
0x00,0x9E - DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(128) Mac=AEAD
0xC0,0x9E - DHE-RSA-AES128-CCM TLSv1.2 Kx=DH Au=RSA Enc=AESCCM(128) Mac=AEAD
0x00,0x6B - DHE-RSA-AES256-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AES(256) Mac=SHA256
0x00,0x6A - DHE-DSS-AES256-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AES(256) Mac=SHA256
0x00,0x67 - DHE-RSA-AES128-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AES(128) Mac=SHA256
0x00,0x40 - DHE-DSS-AES128-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AES(128) Mac=SHA256
0x00,0x33 - DHE-RSA-AES128-SHA SSLv3 Kx=DH Au=RSA Enc=AES(128) Mac=SHA1
0x00,0x32 - DHE-DSS-AES128-SHA SSLv3 Kx=DH Au=DSS Enc=AES(128) Mac=SHA1
0x00,0x9D - AES256-GCM-SHA384 TLSv1.2 Kx=RSA Au=RSA Enc=AESGCM(256) Mac=AEAD
0xC0,0x9D - AES256-CCM TLSv1.2 Kx=RSA Au=RSA Enc=AESCCM(256) Mac=AEAD
0x00,0x9C - AES128-GCM-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AESGCM(128) Mac=AEAD
0xC0,0x9C - AES128-CCM TLSv1.2 Kx=RSA Au=RSA Enc=AESCCM(128) Mac=AEAD
0x00,0x3D - AES256-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AES(256) Mac=SHA256
0x00,0x3C - AES128-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA256
0x00,0x2F - AES128-SHA SSLv3 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA1
~ #
```

## 패킷 캡처로 TLS 핸드셰이크의 암호화 협상 검사

패킷 캡처에서 TLS 협상을 캡처하여 Wireshark를 사용하여 암호화 협상의 세부사항을 검사할 수 있습니다.

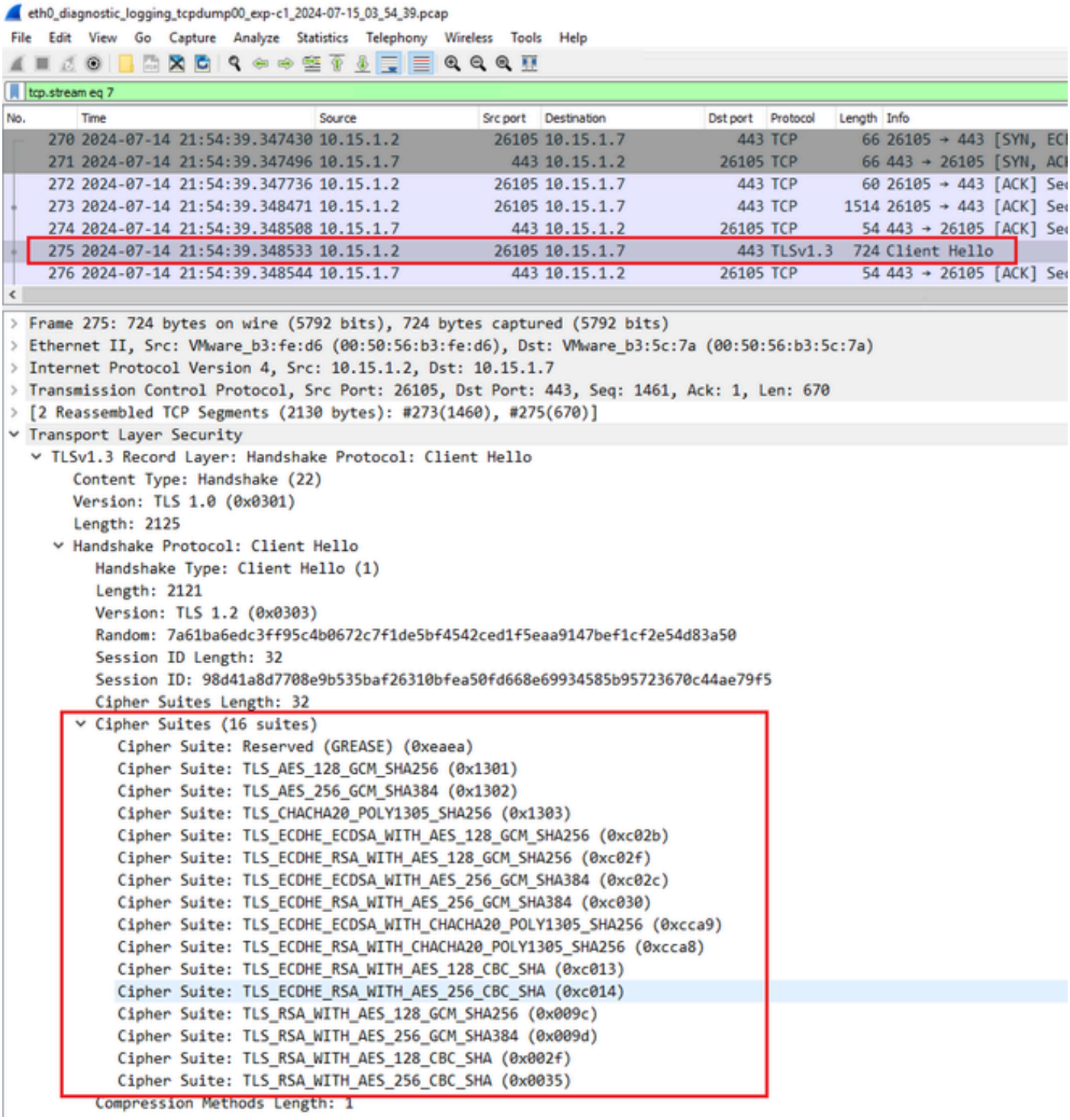
TLS 핸드셰이크 프로세스에는 클라이언트 디바이스에서 전송한 ClientHello 패킷이 포함되어 있으며, 연결 프로토콜에 대해 구성된 암호 문자열에 따라 지원하는 암호 목록을 제공합니다. 서버는 목록을 검토하고, 목록을 허용되는 자체 암호 목록(자체 암호 문자열에 의해 결정됨)과 비교하고, 두 시스템이 모두 지원하는 암호를 선택하여 암호화된 세션에 사용합니다. 그런 다음 선택한 암호를 나타내는 ServerHello 패킷으로 응답합니다. TLS 1.2와 1.3 핸드셰이크 대화 상자 사이에는 중요한 차이점이 있지만, 암호 협상 메커니즘은 두 버전에서 동일한 원리를 사용합니다.

다음은 Wireshark에서 볼 수 있는 포트 443의 웹 브라우저와 Expressway 간의 TLS 1.3 암호 협상의 예입니다.



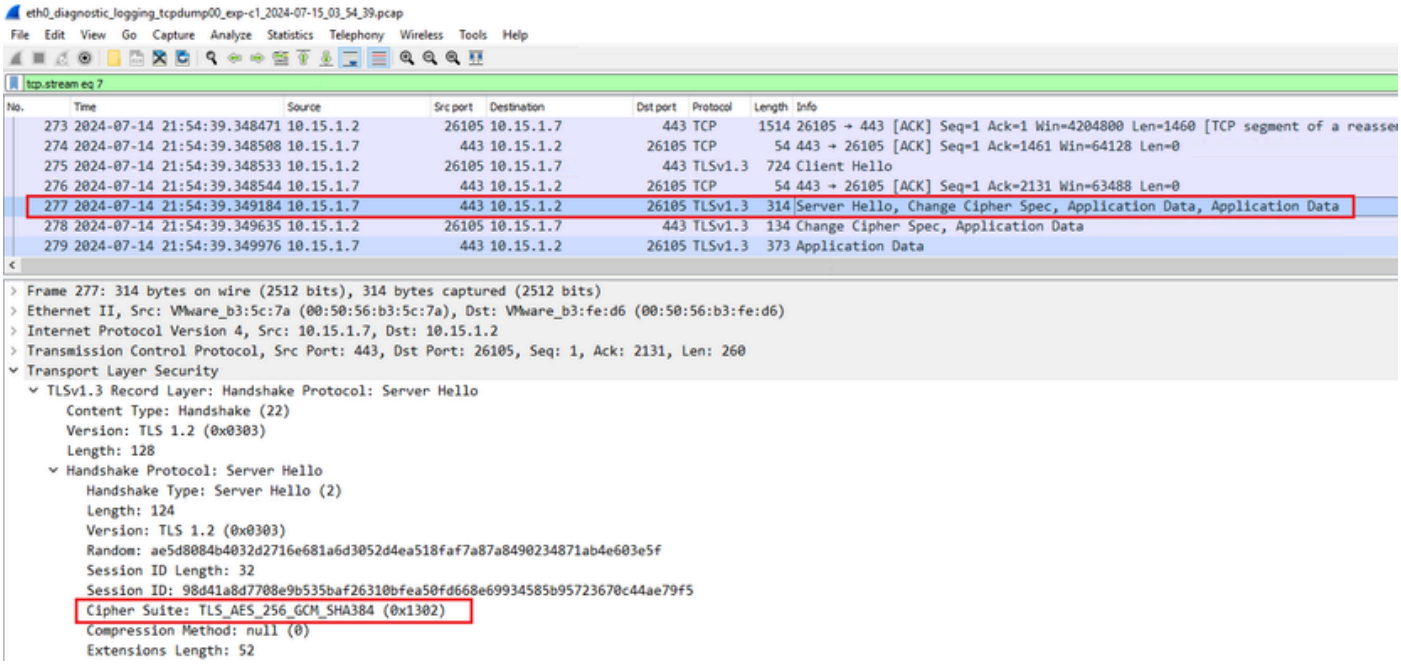

Wireshark에서 TLS 핸드셰이크의 예

먼저 브라우저에서 지원하는 암호 목록과 함께 ClientHello 패킷을 보냅니다.




Wireshark의 ClientHello 패킷의 예

Expressway는 HTTPS 프로토콜에 대해 구성된 암호 문자열을 확인하고, 자신과 클라이언트가 모두 지원하는 암호를 찾습니다. 이 예에서는 ECDHE-RSA-AES256-GCM-SHA384 암호가 선택됩니다. Expressway는 선택한 암호를 나타내는 ServerHello 패킷으로 응답합니다.



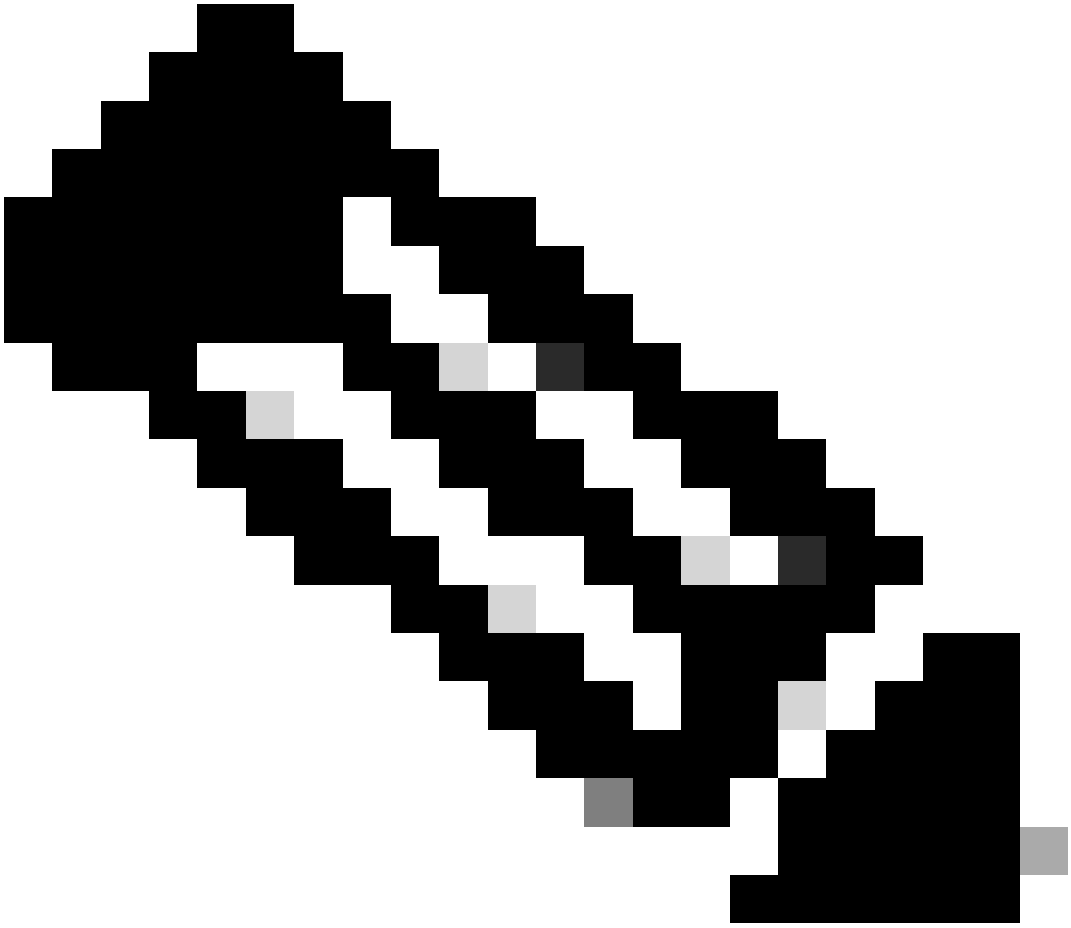

Wireshark의 ServerHello 패킷의 예

# 구성

OpenSSL 암호 문자열 형식은 특정 암호 또는 공통 구성 요소를 공유하는 암호 그룹을 제거하는 것과 같은 문자열 작업을 수행하기 위해 여러 특수 문자를 포함합니다. 이러한 사용자 지정의 목적은 일반적으로 암호를 제거하는 것이므로 다음 예에서 사용되는 문자는 다음과 같습니다.

- 목록에서 암호를 제거하는 데 사용되는 문자. 제거된 암호의 일부 또는 전부는 문자열의 뒤에 나타나는 옵션을 통해 다시 허용할 수 있습니다.
- 목록에서 ! 문자를 제거하는 데 사용되는 문자입니다. 이 암호를 사용할 때 문자열의 뒤에 나타나는 다른 옵션을 사용하여 제거된 암호를 다시 허용할 수 없습니다.
- 문자: 목록의 항목 간 구분 기호 역할을 합니다.

둘 다 문자열에서 암호를 제거하는 데 사용할 수 있지만 !이(가) 우선입니다. 특수 문자의 전체 목록을 보려면 OpenSSL Ciphers Manpage를 검토하십시오.



참고: OpenSSL 사이트에서는 ! 문자를 사용할 때 "삭제된 암호는 명시적으로 지정되었더라도 목록에 다시 나타나지 않습니다."라고 말합니다. 이는 시스템에서 암호가 영구적으로 삭제된다는 것을 의미하는 것은 아니며, 암호 문자열의 해석 범위를 의미한다.

## 특정 암호 비활성화

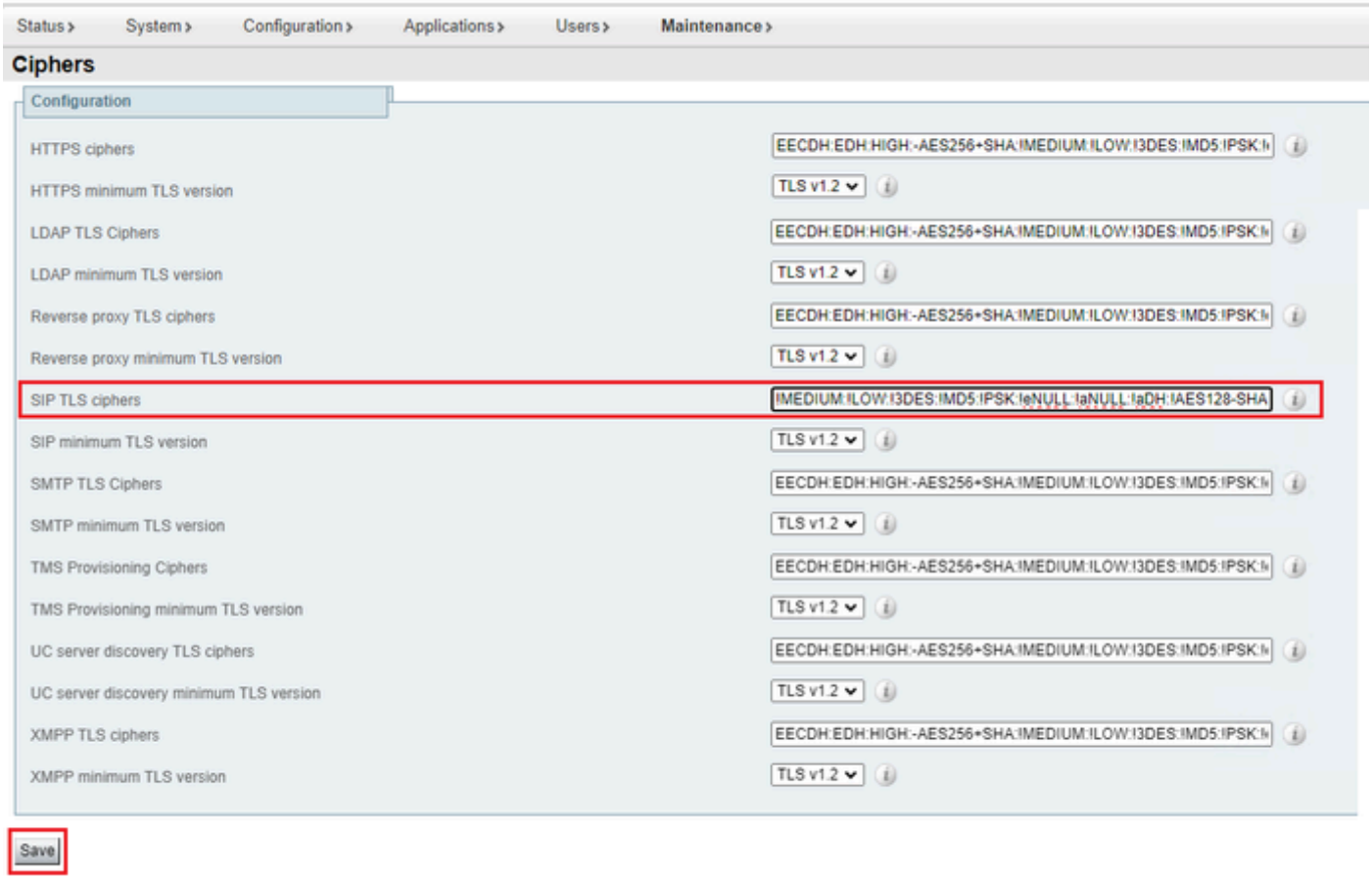
특정 암호를 비활성화하려면 기본 문자열에 separator, the ! or-sign 및 비활성화할 암호 이름을 추가합니다. 암호 이름은 OpenSSL Ciphers Manpage에서 사용할 수 있는 OpenSSL 명명 형식을 따라야 합니다. 예를 들어 SIP 연결에 대해 AES128-SHA 암호를 비활성화해야 하는 경우 다음과 같이 암호 문자열을 구성합니다.

<#root>

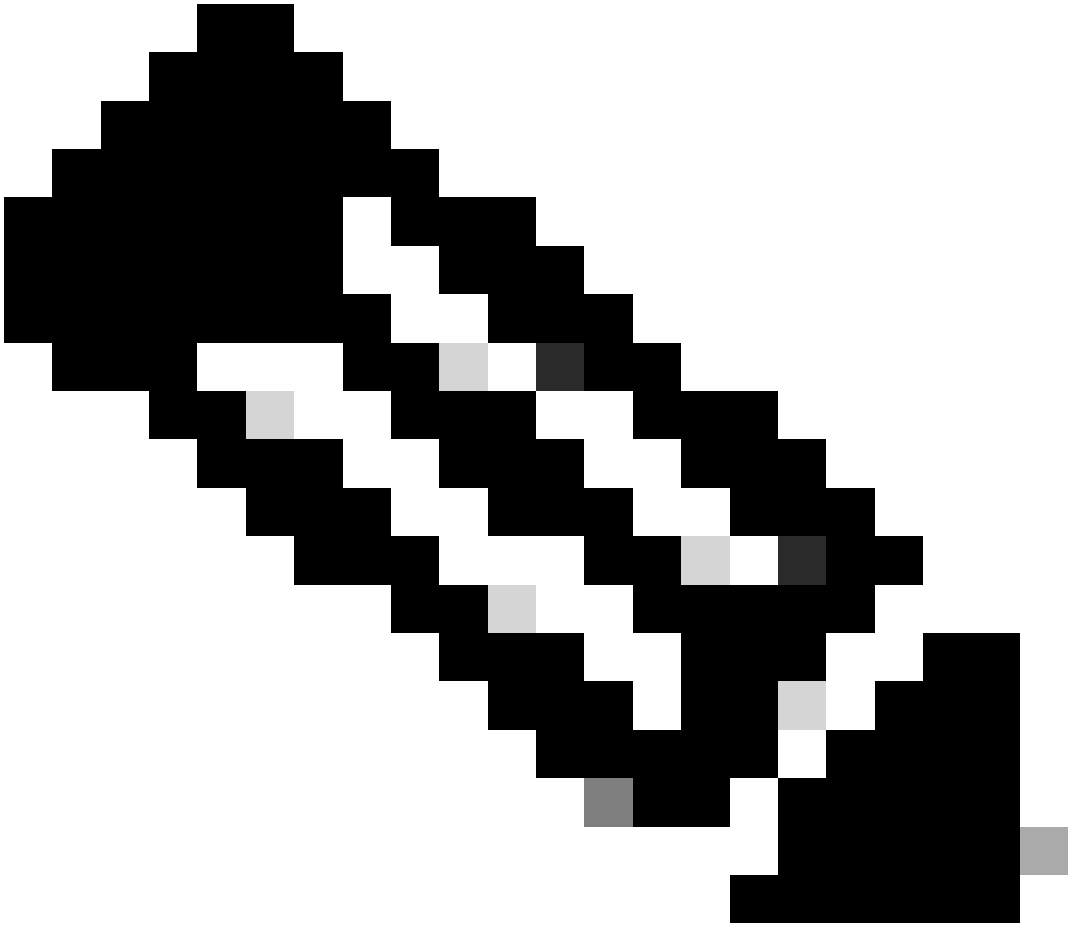
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH

**:!AES128-SHA**

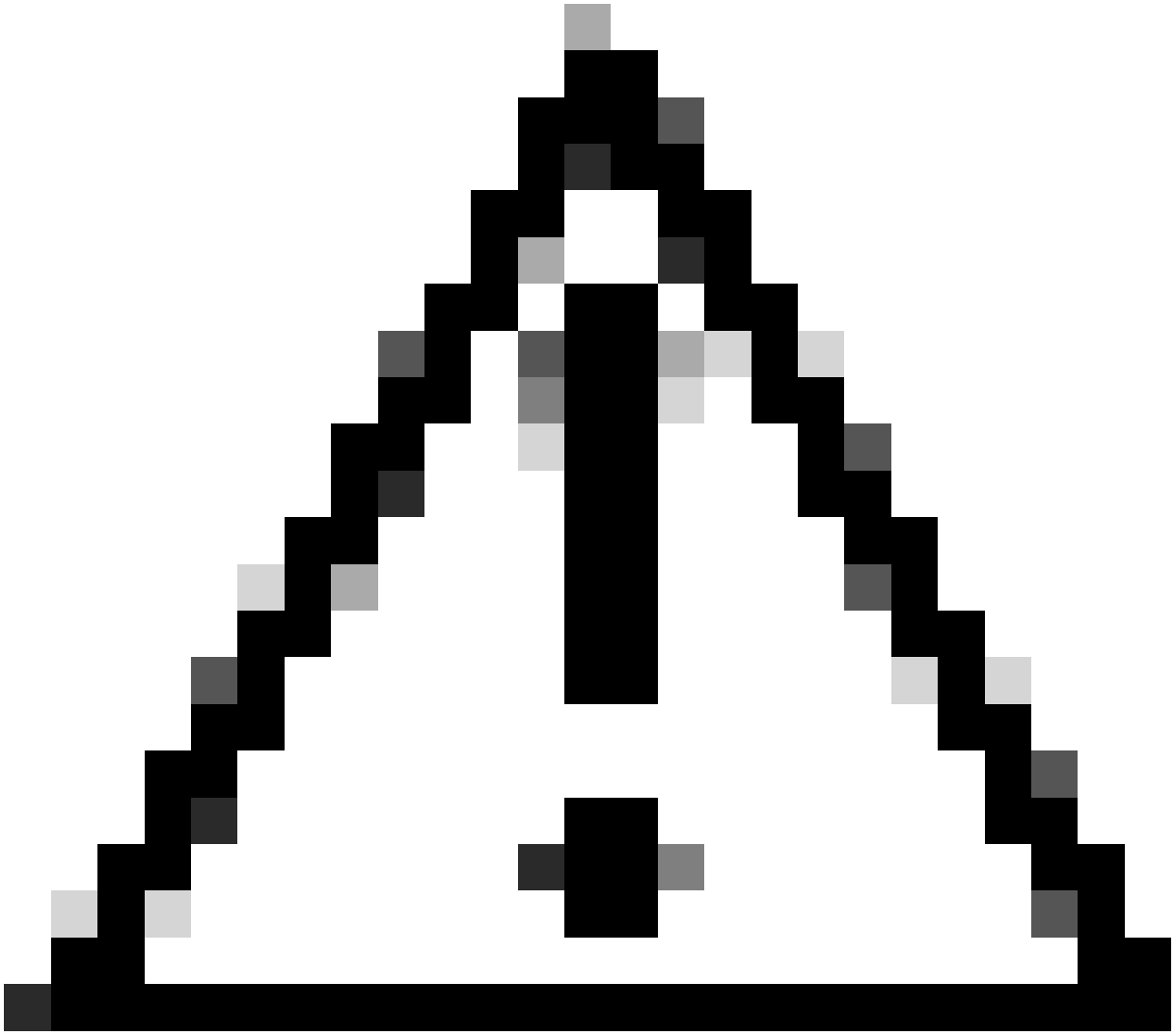
그런 다음 Expressway 웹 관리 페이지로 이동하고 Maintenance(유지 관리) > Security(보안) > Ciphers(암호)로 이동하여 필요한 프로토콜에 사용자 지정 문자열을 지정하고 Save(저장)를 클릭합니다. 새 컨피그레이션을 적용하려면 시스템을 다시 시작해야 합니다. 이 예에서는 사용자 지정 문자열이 SIP TLS 암호에서 SIP 프로토콜에 할당됩니다.




Expressway 웹 관리 포털의 암호화 설정 페이지



참고: Expressway 클러스터의 경우 기본 서버에서만 변경합니다. 새 컨피그레이션이 나머지 클러스터 멤버에 복제됩니다.



주의: Cisco Expressway Cluster Creation and Maintenance Deployment Guide에 나와 있는 권장 클러스터 재부팅 시퀀스를 사용합니다. 먼저 기본 서버를 다시 시작하고 웹 인터페이스를 통해 액세스할 수 있을 때까지 기다린 다음 System > Clustering에 구성된 목록에 따라 각 피어에 대해 같은 작업을 수행합니다.

## 공통 알고리즘을 사용하여 암호 그룹 비활성화

공통 알고리즘을 사용하여 암호 그룹을 비활성화하려면 기본 문자열에 separator, ! 또는-sign 및 비활성화할 알고리즘 이름을 추가합니다. 지원되는 알고리즘 이름은 OpenSSL Ciphers Manpage에서 사용할 수 있습니다. 예를 들어, DHE 알고리즘을 사용하는 모든 암호를 비활성화해야 하는 경우 다음과 같이 암호 문자열을 구성합니다.

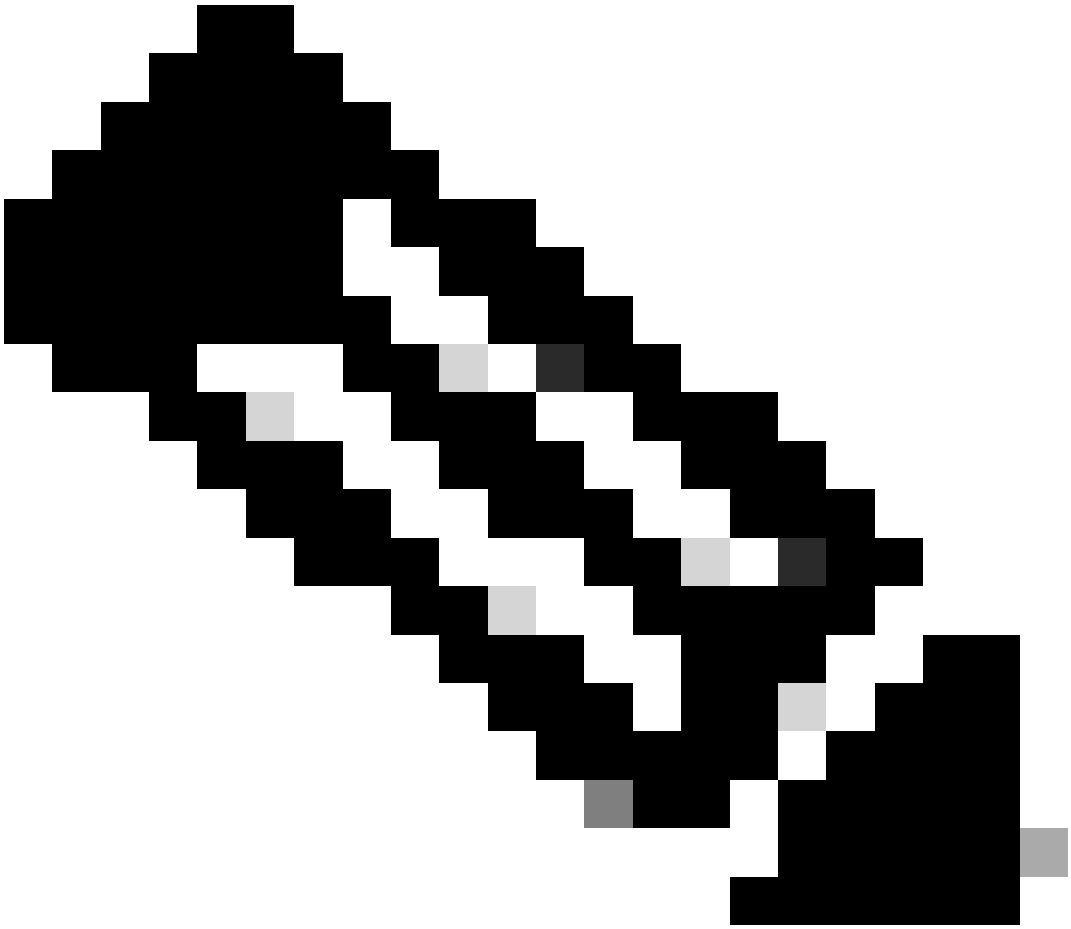
<#root>

EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH

**:!DHE**

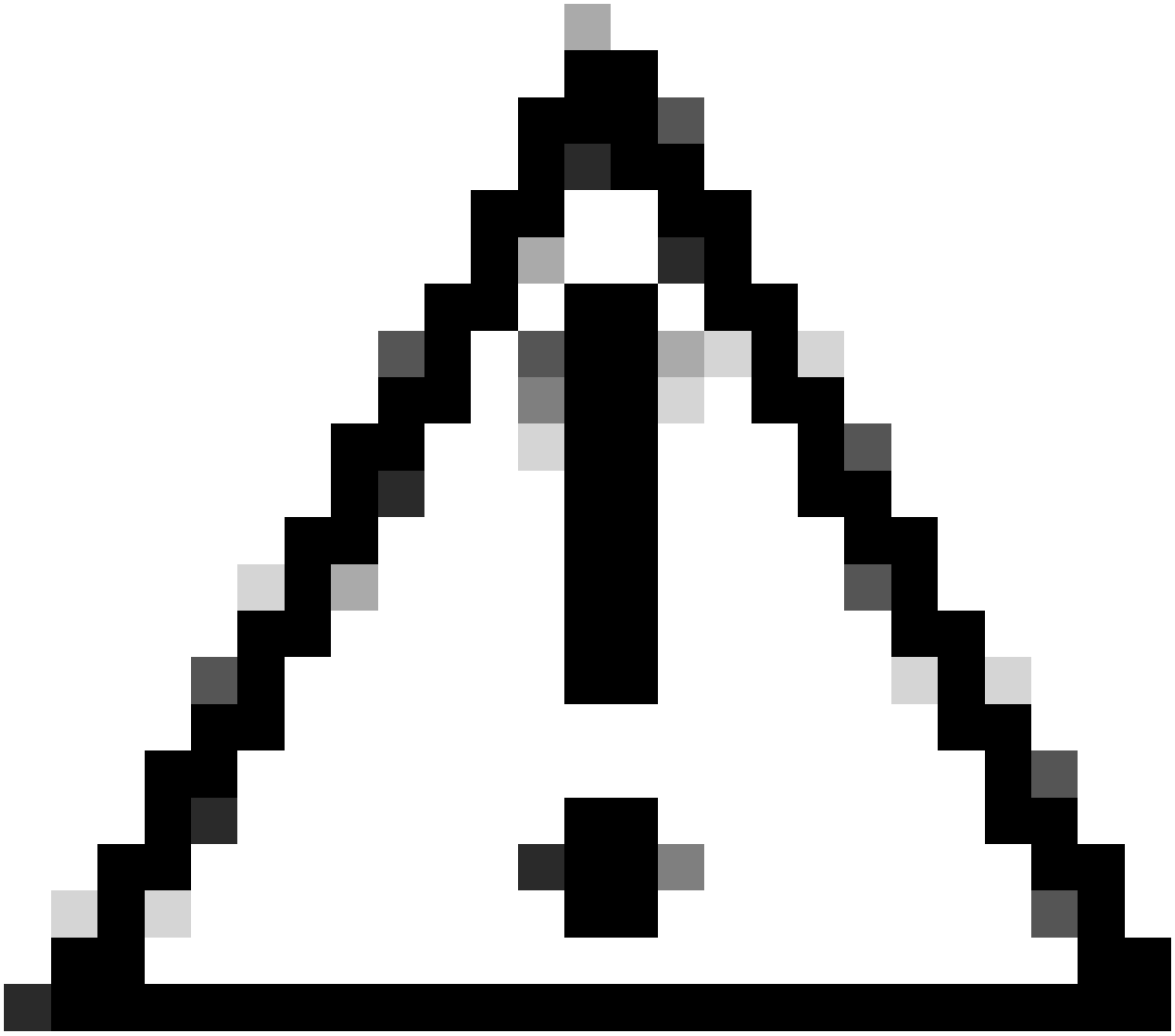
Expressway 웹 관리자 페이지로 이동하고 Maintenance(유지 관리) > Security(보안) > Ciphers(암호)로 이동하여 필요한 프로토콜에 사용자 지정 문자열을 지정하고 Save(저장)를 클릭합니다. 새 컨피그레이션을 적용하려면 시스템을 다시 시작해야 합니다.

---



참고: Expressway 클러스터의 경우 기본 서버에서만 변경합니다. 새 컨피그레이션이 나머지 클러스터 멤버에 복제됩니다.

---



주의: Cisco Expressway Cluster Creation and Maintenance Deployment Guide에 나와 있는 권장 클러스터 재부팅 시퀀스를 사용합니다. 먼저 기본 서버를 다시 시작하고 웹 인터페이스를 통해 액세스할 수 있을 때까지 기다린 다음 System > Clustering에 구성된 목록에 따라 각 피어에 대해 같은 작업을 수행합니다.

## 다음을 확인합니다.

### 암호 문자열에서 허용하는 암호 목록 검사

openssl ciphers -V "<cipher string>" 명령을 사용하여 사용자 지정된 암호 문자열을 검사할 수 있습니다. 변경 후 원하지 않는 암호가 더 이상 나열되지 않는지 확인하려면 출력을 검토합니다. 이 예에서는 ECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHE 암호 문자열을 검사합니다. 명령 출력에서는 문자열이 DHE 알고리즘을 사용하는 암호를 허용하지 않음을 확인합니다.

<#root>

```
    ~ # openssl ciphers -V "EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
:!DHE
"
0x13,0x02 - TLS_AES_256_GCM_SHA384 TLSv1.3 Kx=any Au=any Enc=AESGCM(256) Mac=AEAD
0x13,0x03 - TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any Au=any Enc=CHACHA20/POLY1305(256) Mac=AEAD
0x13,0x01 - TLS_AES_128_GCM_SHA256 TLSv1.3 Kx=any Au=any Enc=AESGCM(128) Mac=AEAD
0xC0,0x2C - ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(256) Mac=AEAD
0xC0,0x30 - ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(256) Mac=AEAD
0xCC,0xA9 - ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2 Kx=ECDH Au=ECDSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
0xCC,0xA8 - ECDHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=ECDH Au=RSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
0xC0,0xAD - ECDHE-ECDSA-AES256-CCM TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM(256) Mac=AEAD
0xC0,0x2B - ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(128) Mac=AEAD
0xC0,0x2F - ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(128) Mac=AEAD
0xC0,0xAC - ECDHE-ECDSA-AES128-CCM TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM(128) Mac=AEAD
0xC0,0x24 - ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(256) Mac=SHA384
0xC0,0x28 - ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(256) Mac=SHA384
0xC0,0x23 - ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA256
0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA256
0xC0,0x09 - ECDHE-ECDSA-AES128-SHA TLSv1 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA1
0xC0,0x13 - ECDHE-RSA-AES128-SHA TLSv1 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA1
0x00,0x9D - AES256-GCM-SHA384 TLSv1.2 Kx=RSA Au=RSA Enc=AESGCM(256) Mac=AEAD
0xC0,0x9D - AES256-CCM TLSv1.2 Kx=RSA Au=RSA Enc=AESCCM(256) Mac=AEAD
0x00,0x9C - AES128-GCM-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AESGCM(128) Mac=AEAD
0xC0,0x9C - AES128-CCM TLSv1.2 Kx=RSA Au=RSA Enc=AESCCM(128) Mac=AEAD
0x00,0x3D - AES256-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AES(256) Mac=SHA256
0x00,0x3C - AES128-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA256
0x00,0x2F - AES128-SHA SSLv3 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA1
~ #
```

## 비활성화된 암호를 협상하여 TLS 연결 테스트

비활성화된 암호를 사용한 연결 시도가 거부되었는지 확인하려면 openssl s_client 명령을 사용할 수 있습니다. Expressway 주소 및 포트를 지정하려면 -connect 옵션을 사용하고, TLS 핸드셰이크 중에 클라이언트가 협상할 단일 암호를 지정하려면 -cipher 옵션을 사용합니다.

openssl s_client -connect <주소>:<포트> -cipher <암호> -no_tls1_3

이 예에서는 openssl이 설치된 Windows PC에서 Expressway로 TLS 연결을 시도합니다. PC는 클라이언트로서 DHE 알고리즘을 사용하는 원하지 않는 DHE-RSA-AES256-CCM 암호만 협상합니다
.

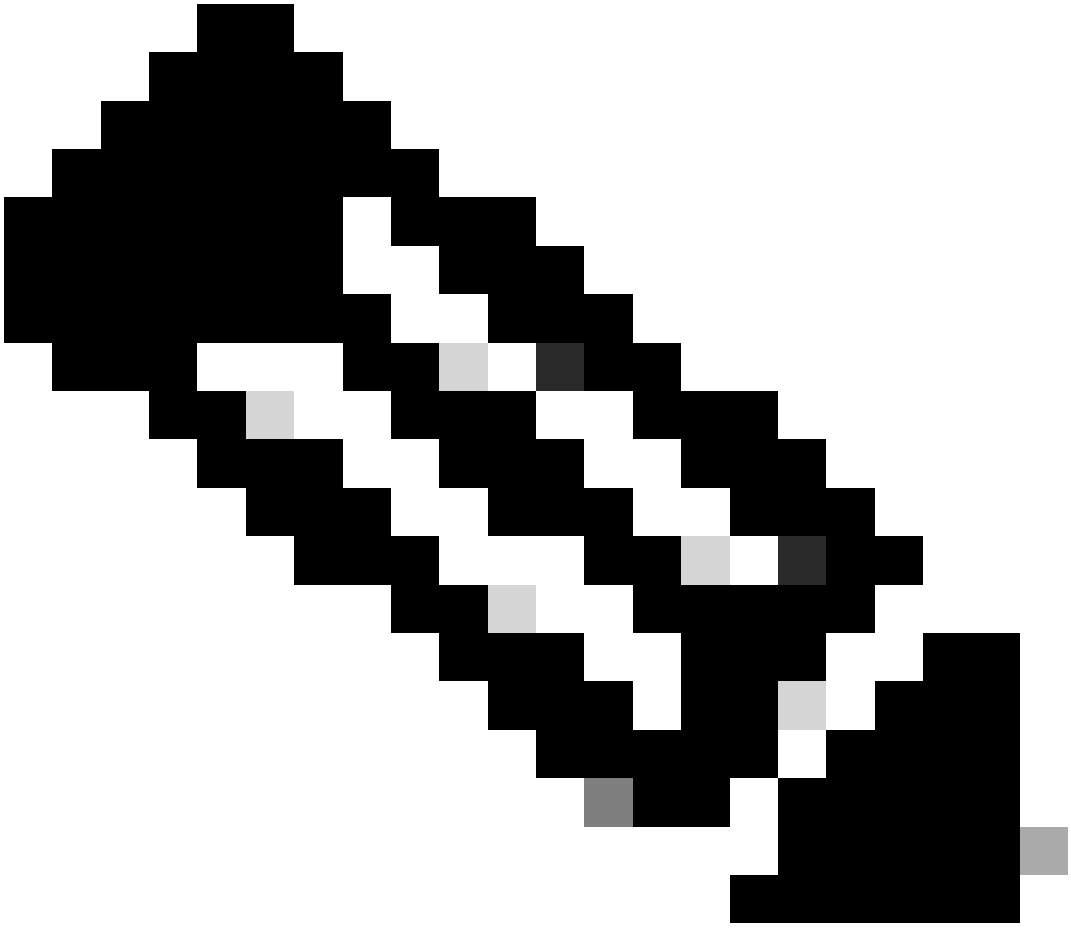
<#root>

```
C:\Users\Administrator>

openssl s_client -connect exp.example.com:443 -cipher DHE-RSA-AES256-CCM -no_tls1_3


Connecting to 10.15.1.7
CONNECTED(00000154)
D0130000:error:0A000410:SSL routines:ssl3_read_bytes:

ssl/tls alert handshake failure

:..\ssl\record\rec_layer_s3.c:865:
```

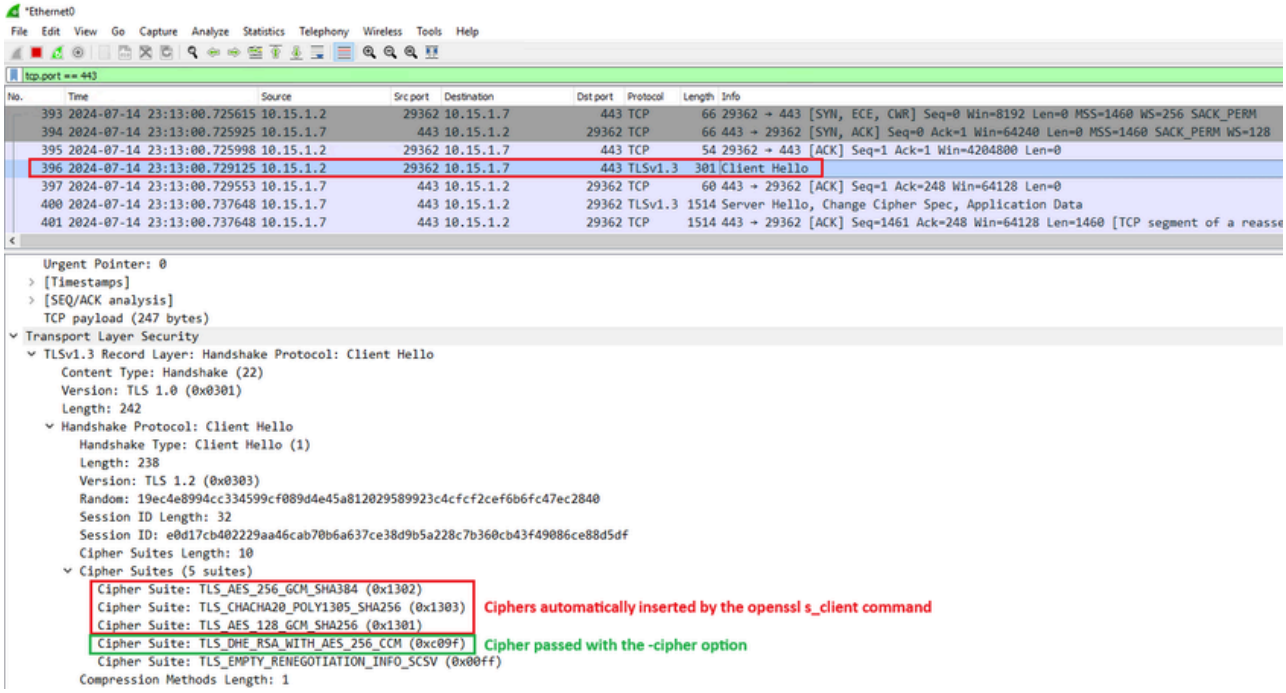
**SSL alert number 40**

```
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 7 bytes and written 118 bytes
Verification: OK
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
No ALPN negotiated
SSL-Session:
Protocol : TLSv1.2
Cipher : 0000
Session-ID:
Session-ID-ctx:
Master-Key:
PSK identity: None
PSK identity hint: None
SRP username: None
Start Time: 1721019437
Timeout : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
---

C:\Users\Administrator>
```

DHE 알고리즘을 사용하는 암호를 비활성화하는 HTTPS 연결에 EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHE 암호 문자열을 사용하도록 Expressway가 구성되었으므로 명령 출력에 "ssl/tls alert handshake failure:..\ssl\record\rec_layer_s3.c:865:SSL alert number 40" 오류 메시지와 함께 연결 시도가 실패한 것으로 표시됩니다.



참고: openssl s_client 명령을 사용하는 테스트가 설명한 대로 작동하려면 -no_tls1_3 옵션을 명령에 전달해야 합니다. 포함되지 않은 경우 클라이언트는 ClientHello 패킷에 TLS 1.3 암호를 자동으로 삽입합니다.



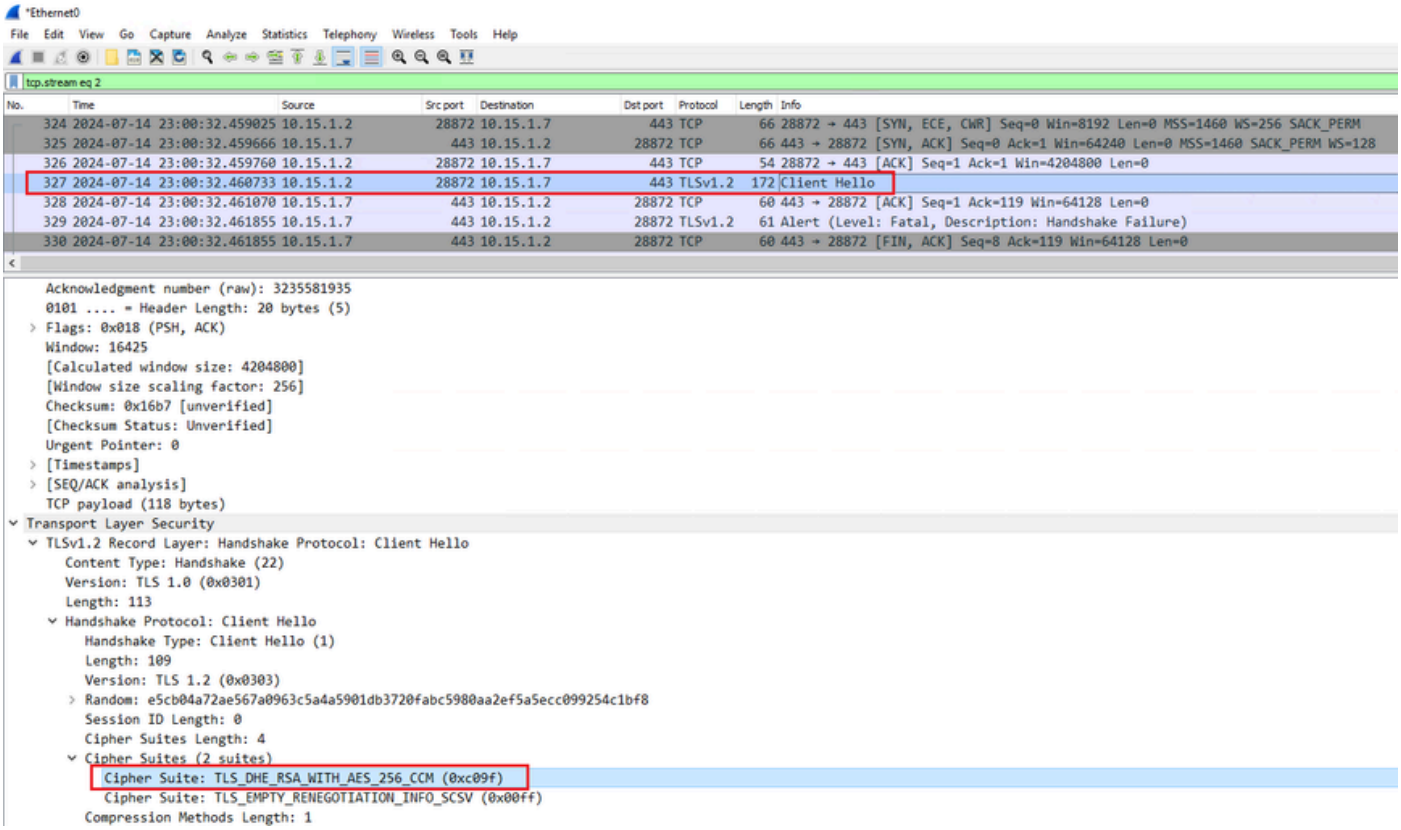

자동으로 암호가 추가된 ClientHello 패킷

대상 Expressway가 이러한 암호를 지원하는 경우 테스트해야 하는 특정 암호 대신 둘 중 하나를 선택할 수 있습니다. 연결에 성공했습니다. 그러면 -cipher 옵션을 사용하여 명령에 전달된 비활성화된 암호를 사용하여 연결이 가능했다고 생각할 수 있습니다.

## 비활성화된 암호를 사용하여 TLS 핸드셰이크의 패킷 캡처 검사

비활성화된 암호 중 하나를 사용하여 연결 테스트를 수행하는 동안 테스트 디바이스 또는 Expressway에서 패킷 캡처를 수집할 수 있습니다. 그런 다음 Wireshark로 검사하여 핸드셰이크 이벤트를 더 분석할 수 있습니다.

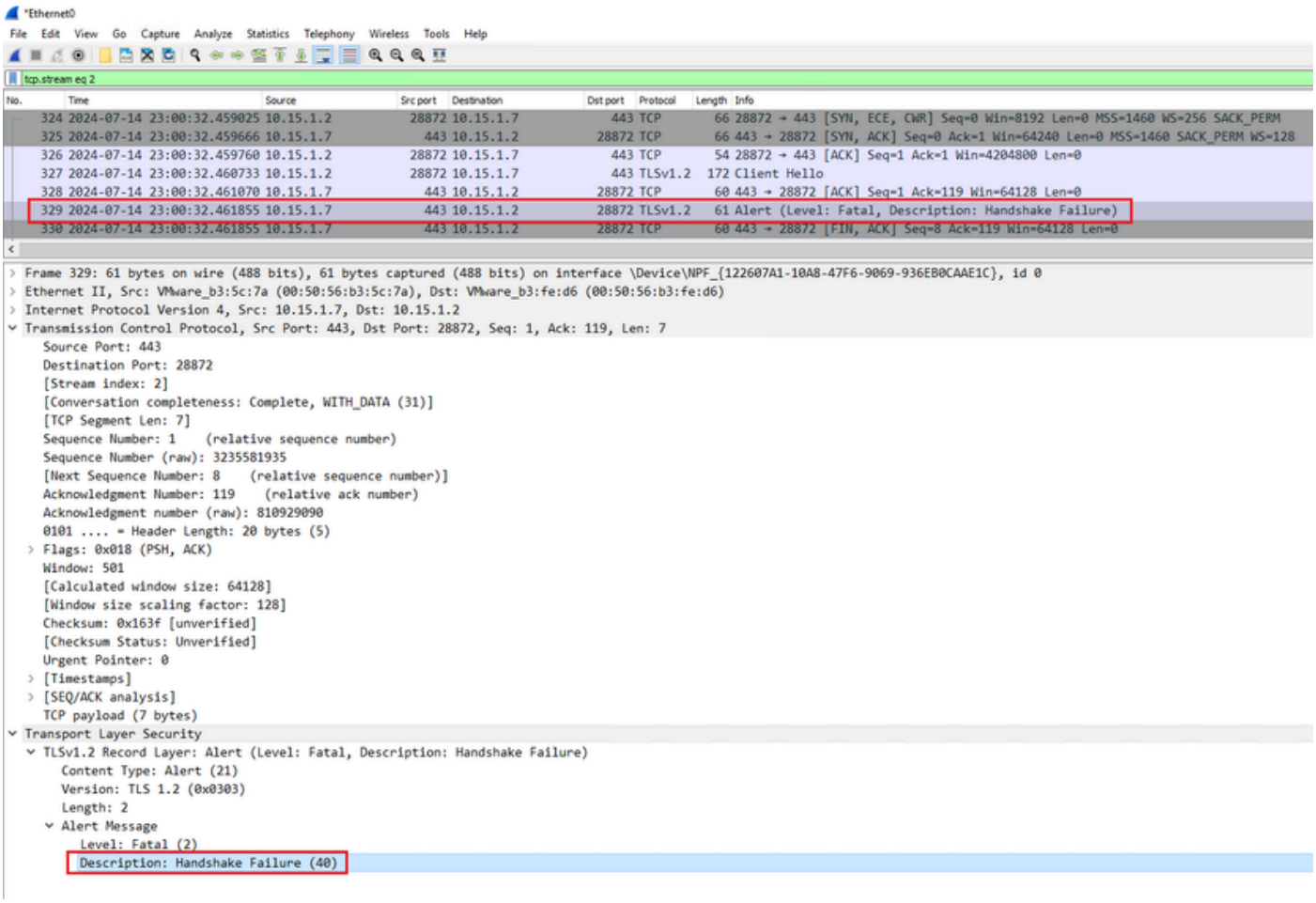
테스트 디바이스에서 전송한 ClientHello를 찾습니다. 원하지 않는 테스트 암호만 협상하는지 확인합니다. 이 예에서는 DHE 알고리즘을 사용하는 암호입니다.




Wireshark의 ClientHello 패킷의 예

:

Expressway가 연결을 거부하면서 치명적인 TLS 알림 패킷으로 응답하는지 확인합니다. 이 예에서 Expressway는 HTTPS 프로토콜에 대해 구성된 암호 문자열당 DHE 암호를 지원하지 않으므로 오류 코드 40을 포함하는 치명적인 TLS 알림 패킷으로 응답합니다.




Wireshark의 TLS 치명적 알림 패킷

# 관련 정보

- [OpenSSL 암호 맨페이지]
- [Cisco Expressway 관리자 설명서(X15.0) - 장: 보안 관리 - 최소 TLS 버전 및 암호 그룹 구성]

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.