

Anpassen der SSL- Verschlüsselungskonfiguration für Expressway

Inhalt

[Einleitung](#)

[Voraussetzungen](#)

[Anforderungen](#)

[Verwendete Komponenten](#)

[Hintergrundinformationen](#)

[Überprüfen der Chiffrierzeichenfolge](#)

[Überprüfen Sie die Cipher Negotiation beim TLS-Handshake mit einer Paketerfassung.](#)

[Konfigurieren](#)

[Deaktivieren einer bestimmten Verschlüsselung](#)

[Deaktivieren einer Gruppe von Chiffren mithilfe eines allgemeinen Algorithmus](#)

[Überprüfung](#)

[Überprüfen Sie die Liste der von der Cipher-Zeichenfolge zugelassenen Chiffren.](#)

[Testen einer TLS-Verbindung durch Aushandeln eines deaktivierten Verschlüsselungsverfahrens](#)

[Überprüfen der Paketerfassung eines TLSHandshake mithilfe eines deaktivierten Verschlüsselungsverfahrens](#)

[Zugehörige Informationen](#)

Einleitung

In diesem Dokument werden die Schritte zum Anpassen der vorkonfigurierten Verschlüsselungszeichenfolgen in Expressway beschrieben.

Voraussetzungen

Anforderungen

Cisco empfiehlt, dass Sie über Kenntnisse in folgenden Bereichen verfügen:

- Cisco Expressway oder Cisco VCS,
- TLS-Protokoll.

Verwendete Komponenten

Die Informationen in diesem Dokument basierend auf folgenden Software- und Hardware-Versionen:

- Cisco Expressway-Version X15.0.2

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die möglichen Auswirkungen aller Befehle kennen.

Hintergrundinformationen

Die standardmäßige Expressway-Konfiguration umfasst vorkonfigurierte Verschlüsselungszeichenfolgen, die aus Kompatibilitätsgründen die Unterstützung für einige Verschlüsselungsarten ermöglichen, die in einigen Sicherheitsrichtlinien des Unternehmens als schwach angesehen werden können. Sie können die Verschlüsselungszeichenfolgen anpassen, um sie an die spezifischen Richtlinien der jeweiligen Umgebung anzupassen.

In Expressway ist es möglich, eine unabhängige Verschlüsselungszeichenfolge für jedes dieser Protokolle zu konfigurieren:

- HTTPS
- LDAP
- Reverse-Proxy
- SIP
- SMTP
- TMS-Bereitstellung
- UC-Servererkennung
- XMPP

Die Verschlüsselungszeichenfolgen entsprechen dem OpenSSL-Format, das in der [OpenSSL Ciphers Manpage](#) beschrieben wird. Die aktuelle Expressway-Version X15.0.2 enthält die Standardzeichenfolge `EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH`, die für alle Protokolle gleichermaßen vorkonfiguriert ist. Auf der Web-Admin-Seite können Sie unter `Wartung > Sicherheit > Chiffren` die jedem Protokoll zugewiesene Chiffrierzeichenfolge ändern, um bestimmte Chiffren oder Verschlüsselungsgruppen mithilfe eines gemeinsamen Algorithmus hinzuzufügen oder zu entfernen.

Überprüfen der Chiffrierzeichenfolge

Mit dem Befehl `openssl ciphers -V "<cipher string>"` können Sie eine Liste mit allen Verschlüsselungen ausgeben, die eine bestimmte Zeichenfolge zulässt. Dies ist für die visuelle Überprüfung der Verschlüsselungen nützlich. Dieses Beispiel zeigt die Ausgabe bei der Überprüfung der standardmäßigen Expressway-Verschlüsselungszeichenfolge:

```
<#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
~ #

```

Überprüfen Sie die Cipher Negotiation beim TLS-Handshake mit einer Paketerfassung.

Wenn Sie eine TLS-Verhandlung in einer Paketerfassung erfassen, können Sie die Details der Verschlüsselungsverhandlung mithilfe von Wireshark überprüfen.

Der TLS-Handshake-Prozess umfasst ein vom Client-Gerät gesendetes ClientHello-Paket, das die Liste der unterstützten Verschlüsselungen gemäß der für das Verbindungsprotokoll konfigurierten Verschlüsselungszeichenfolge bereitstellt. Der Server überprüft die Liste, vergleicht sie mit seiner eigenen Liste zulässiger Chiffren (bestimmt durch seine eigene Chiffrierzeichenfolge) und wählt eine von beiden Systemen unterstützte Chiffre für die verschlüsselte Sitzung aus. Anschließend antwortet er mit einem ServerHello-Paket, das den ausgewählten Verschlüsselungscode angibt. Es gibt wichtige Unterschiede zwischen den TLS 1.2 und 1.3 Handshake Dialogen, aber der Cipher Negotiation Mechanismus verwendet dieses Prinzip in beiden Versionen.

Dies ist ein Beispiel für eine TLS 1.3-Verschlüsselungsverhandlung zwischen einem Webbrowser

und Expressway an Port 443, wie in Wireshark gezeigt:

No.	Time	Source	Src port	Destination	Dst port	Protocol	Length	Info
3186	2024-07-14 23:28:55.675989	10.15.1.2	29986	10.15.1.7	443	TCP	66	29986 → 443 [SYN, ECE, CW] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
3187	2024-07-14 23:28:55.676309	10.15.1.7	443	10.15.1.2	29986	TCP	66	443 → 29986 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3188	2024-07-14 23:28:55.676381	10.15.1.2	29986	10.15.1.7	443	TCP	54	29986 → 443 [ACK] Seq=1 Ack=1 Win=4204800 Len=0
3189	2024-07-14 23:28:55.679410	10.15.1.2	29986	10.15.1.7	443	TLsv1.2	248	Client Hello
3190	2024-07-14 23:28:55.679651	10.15.1.7	443	10.15.1.2	29986	TCP	60	443 → 29986 [ACK] Seq=1 Ack=195 Win=64128 Len=0
3194	2024-07-14 23:28:55.686008	10.15.1.7	443	10.15.1.2	29986	TLsv1.2	1514	Server Hello
3195	2024-07-14 23:28:55.686008	10.15.1.7	443	10.15.1.2	29986	TLsv1.2	1514	Certificate
3196	2024-07-14 23:28:55.686097	10.15.1.2	29986	10.15.1.7	443	TCP	54	29986 → 443 [ACK] Seq=195 Ack=2921 Win=4204800 Len=0
3197	2024-07-14 23:28:55.686118	10.15.1.7	443	10.15.1.2	29986	TLsv1.2	547	Server Key Exchange, Server Hello Done
3198	2024-07-14 23:28:55.696856	10.15.1.2	29986	10.15.1.7	443	TCP	54	29986 → 443 [ACK] Seq=195 Ack=3414 Win=4204288 Len=0
3199	2024-07-14 23:28:55.702443	10.15.1.2	29986	10.15.1.7	443	TLsv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
3200	2024-07-14 23:28:55.702991	10.15.1.7	443	10.15.1.2	29986	TLsv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
3207	2024-07-14 23:28:55.712838	10.15.1.2	29986	10.15.1.7	443	TCP	54	29986 → 443 [ACK] Seq=288 Ack=3672 Win=4204032 Len=0

Beispiel für einen TLS-Handshake in Wireshark

Zunächst sendet der Browser ein ClientHello-Paket mit der Liste der unterstützten Chiffren:

eth0_diagnostic_logging_tcpdump00_exp-c1_2024-07-15_03_54_39.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

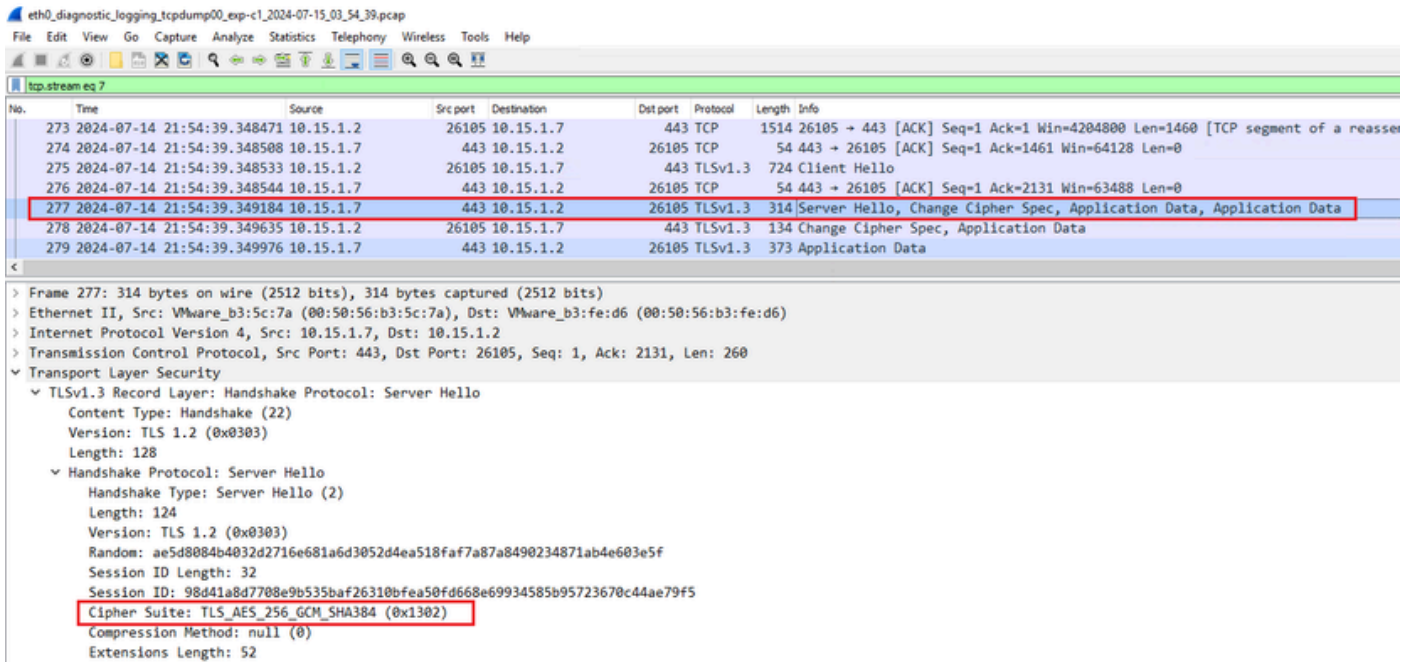
tcp.stream eq 7

No.	Time	Source	Src port	Destination	Dst port	Protocol	Length	Info
270	2024-07-14 21:54:39.347430	10.15.1.2	26105	10.15.1.7	443	TCP	66	26105 → 443 [SYN, EC
271	2024-07-14 21:54:39.347496	10.15.1.7	443	10.15.1.2	26105	TCP	66	443 → 26105 [SYN, AC
272	2024-07-14 21:54:39.347736	10.15.1.2	26105	10.15.1.7	443	TCP	60	26105 → 443 [ACK] Ser
273	2024-07-14 21:54:39.348471	10.15.1.2	26105	10.15.1.7	443	TCP	1514	26105 → 443 [ACK] Ser
274	2024-07-14 21:54:39.348508	10.15.1.7	443	10.15.1.2	26105	TCP	54	443 → 26105 [ACK] Ser
275	2024-07-14 21:54:39.348533	10.15.1.2	26105	10.15.1.7	443	TLSv1.3	724	Client Hello
276	2024-07-14 21:54:39.348544	10.15.1.7	443	10.15.1.2	26105	TCP	54	443 → 26105 [ACK] Ser

```
> Frame 275: 724 bytes on wire (5792 bits), 724 bytes captured (5792 bits)
> Ethernet II, Src: VMware_b3:fe:d6 (00:50:56:b3:fe:d6), Dst: VMware_b3:5c:7a (00:50:56:b3:5c:7a)
> Internet Protocol Version 4, Src: 10.15.1.2, Dst: 10.15.1.7
> Transmission Control Protocol, Src Port: 26105, Dst Port: 443, Seq: 1461, Ack: 1, Len: 670
> [2 Reassembled TCP Segments (2130 bytes): #273(1460), #275(670)]
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 2125
  ▼ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 2121
    Version: TLS 1.2 (0x0303)
    Random: 7a61ba6edc3ff95c4b0672c7f1de5bf4542ced1f5eaa9147bef1cf2e54d83a50
    Session ID Length: 32
    Session ID: 98d41a8d7708e9b535baf26310bfea50fd668e69934585b95723670c44ae79f5
    Cipher Suites Length: 32
  ▼ Cipher Suites (16 suites)
    Cipher Suite: Reserved (GREASE) (0xaeaa)
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xc0ca9)
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc0ca8)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
    Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
    Compression Methods Length: 1
```

Beispiel für ein ClientHello-Paket in Wireshark

Expressway überprüft die für das HTTPS-Protokoll konfigurierte Verschlüsselungszeichenfolge und findet eine Verschlüsselung, die sowohl von Expressway selbst als auch vom Client unterstützt wird. In diesem Beispiel wird die ECDHE-RSA-AES256-GCM-SHA384-Chiffre ausgewählt. Expressway antwortet mit dem ServerHello-Paket, das den ausgewählten Schlüssel angibt:



eth0_diagnostic_logging_tcpdump00_exp-cl_2024-07-15_03_54_39.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream eq 7

No.	Time	Source	Src port	Destination	Dst port	Protocol	Length	Info
273	2024-07-14 21:54:39.348471	10.15.1.2	26105	10.15.1.7	443	TCP	1514	26105 → 443 [ACK] Seq=1 Ack=1 Win=4204800 Len=1460 [TCP segment of a reasse...
274	2024-07-14 21:54:39.348508	10.15.1.7	443	10.15.1.2	26105	TCP	54	443 → 26105 [ACK] Seq=1 Ack=1461 Win=64128 Len=0
275	2024-07-14 21:54:39.348533	10.15.1.2	26105	10.15.1.7	443	TLSv1.3	724	Client Hello
276	2024-07-14 21:54:39.348544	10.15.1.7	443	10.15.1.2	26105	TCP	54	443 → 26105 [ACK] Seq=1 Ack=2131 Win=63488 Len=0
277	2024-07-14 21:54:39.349184	10.15.1.7	443	10.15.1.2	26105	TLSv1.3	314	Server Hello, Change Cipher Spec, Application Data, Application Data
278	2024-07-14 21:54:39.349635	10.15.1.2	26105	10.15.1.7	443	TLSv1.3	134	Change Cipher Spec, Application Data
279	2024-07-14 21:54:39.349976	10.15.1.7	443	10.15.1.2	26105	TLSv1.3	373	Application Data

> Frame 277: 314 bytes on wire (2512 bits), 314 bytes captured (2512 bits)

> Ethernet II, Src: VMware_b3:5c:7a (00:50:56:b3:5c:7a), Dst: VMware_b3:fe:d6 (00:50:56:b3:fe:d6)

> Internet Protocol Version 4, Src: 10.15.1.7, Dst: 10.15.1.2

> Transmission Control Protocol, Src Port: 443, Dst Port: 26105, Seq: 1, Ack: 2131, Len: 260

Transport Layer Security

- TLV1.3 Record Layer: Handshake Protocol: Server Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 128
 - Handshake Protocol: Server Hello
 - Handshake Type: Server Hello (2)
 - Length: 124
 - Version: TLS 1.2 (0x0303)
 - Random: ae5d8084b4032d2716e681a6d3052d4ea518faf7a87a8490234871ab4e603e5f
 - Session ID Length: 32
 - Session ID: 98d41a8d7708e9b535baf26310bfea50fd668e69934585b95723670c44ae79f5
 - Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
 - Compression Method: null (0)
 - Extensions Length: 52

Beispiel für ein ServerHello-Paket in Wireshark

Konfigurieren

Das OpenSSL-Verschlüsselungszeichenfolgenformat enthält mehrere Sonderzeichen, um Vorgänge für die Zeichenfolge auszuführen, z. B. das Entfernen einer bestimmten Verschlüsselung oder einer Gruppe von Verschlüsselungen, die eine gemeinsame Komponente verwenden. Da das Ziel dieser Anpassungen normalerweise darin besteht, Chiffren zu entfernen, werden in diesen Beispielen folgende Zeichen verwendet:

- Das - Zeichen, mit dem Chiffren aus der Liste entfernt werden. Einige oder alle der entfernten Chiffren können durch Optionen, die später in der Zeichenfolge erscheinen, wieder zugelassen werden.
- Das !-Zeichen, das auch zum Entfernen von Chiffren aus der Liste verwendet wird. Wenn Sie es verwenden, können die entfernten Chiffren nicht durch andere Optionen, die später in der Zeichenfolge erscheinen, wieder zugelassen werden.
- Das Zeichen ;, das als Trennzeichen zwischen Elementen in der Liste fungiert.

Beide können verwendet werden, um eine Chiffre aus der Zeichenfolge zu entfernen! ist jedoch bevorzugt. Eine vollständige Liste der Sonderzeichen finden Sie auf der [OpenSSL Ciphers Manpage](#).



Hinweis: Die OpenSSL-Site stellt fest, dass bei Verwendung des Zeichens ! "die gelöschten Chiffren in der Liste nie wieder auftauchen können, auch wenn sie explizit angegeben sind". Dies bedeutet nicht, dass die Chiffren dauerhaft aus dem System gelöscht werden, es bezieht sich auf den Umfang der Interpretation der Chiffre-Zeichenfolge.

Deaktivieren einer bestimmten Verschlüsselung

Um eine bestimmte Verschlüsselung zu deaktivieren, fügen Sie an die Standardzeichenfolge das Trennzeichen, das ! oder -Zeichen und den zu deaktivierenden Verschlüsselungsnamen an. Der Name der Verschlüsselung muss dem OpenSSL-Namensformat entsprechen, das auf der [OpenSSL Ciphers Manpage](#) verfügbar ist. Wenn Sie z. B. den AES128-SHA-Verschlüsselungscode für SIP-Verbindungen deaktivieren müssen, konfigurieren Sie eine Verschlüsselungszeichenfolge wie diese:

<#root>

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

:!AES128-SHA

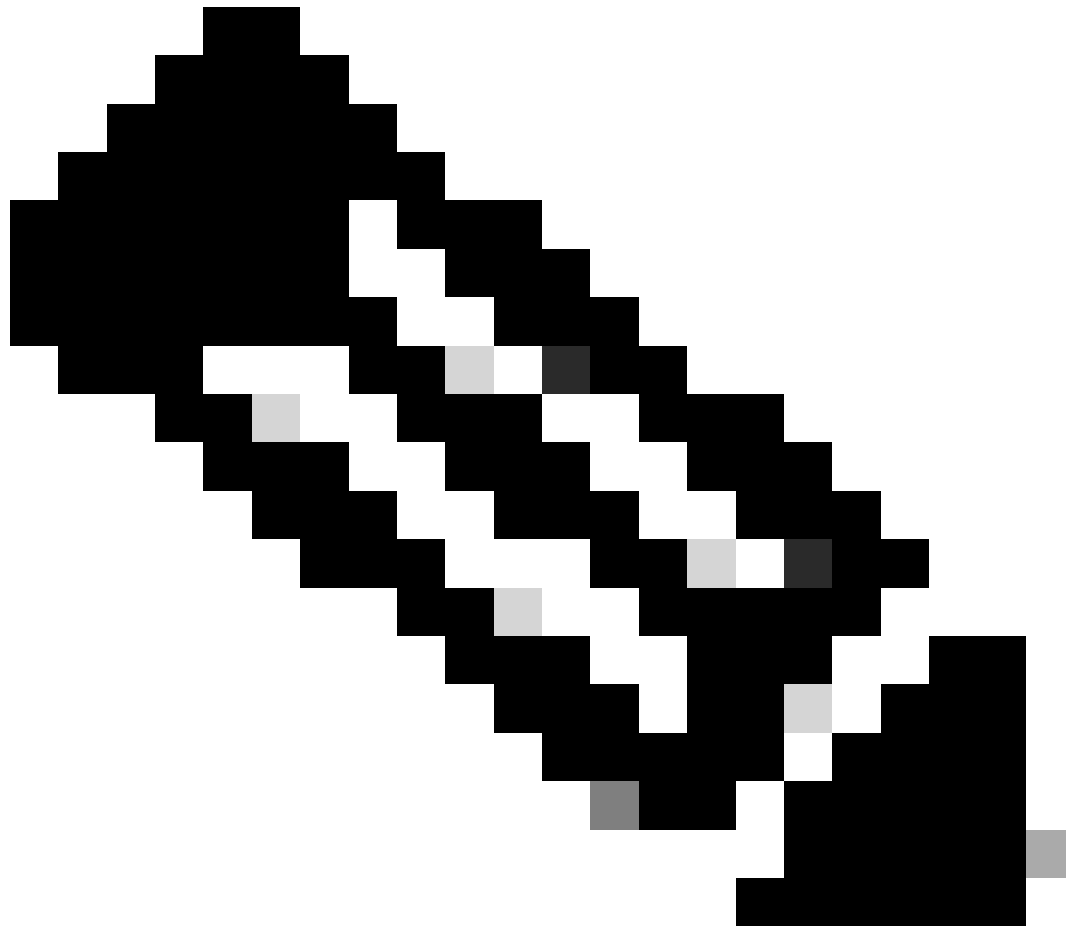
Navigieren Sie anschließend zur Expressway-Web-Admin-Seite, navigieren Sie zu Maintenance > Security > Ciphers, weisen Sie die benutzerdefinierte Zeichenfolge den erforderlichen Protokollen zu, und klicken Sie auf Save (Speichern). Damit die neue Konfiguration angewendet werden kann, muss das System neu gestartet werden. In diesem Beispiel wird die benutzerdefinierte Zeichenfolge dem SIP-Protokoll unter SIP TLS-Verschlüsselungen zugewiesen:

The screenshot shows the 'Ciphers' configuration page in the Expressway-Web-Admin. The page has a navigation bar with 'Status >', 'System >', 'Configuration >', 'Applications >', 'Users >', and 'Maintenance >'. The main title is 'Ciphers' and the sub-section is 'Configuration'. The configuration is organized into a table with two columns: the configuration item name and the value. The 'SIP TLS ciphers' row is highlighted with a red border. The value for 'SIP TLS ciphers' is '!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!AES128-SHA'. Below the configuration table, there is a 'Save' button, also highlighted with a red border.

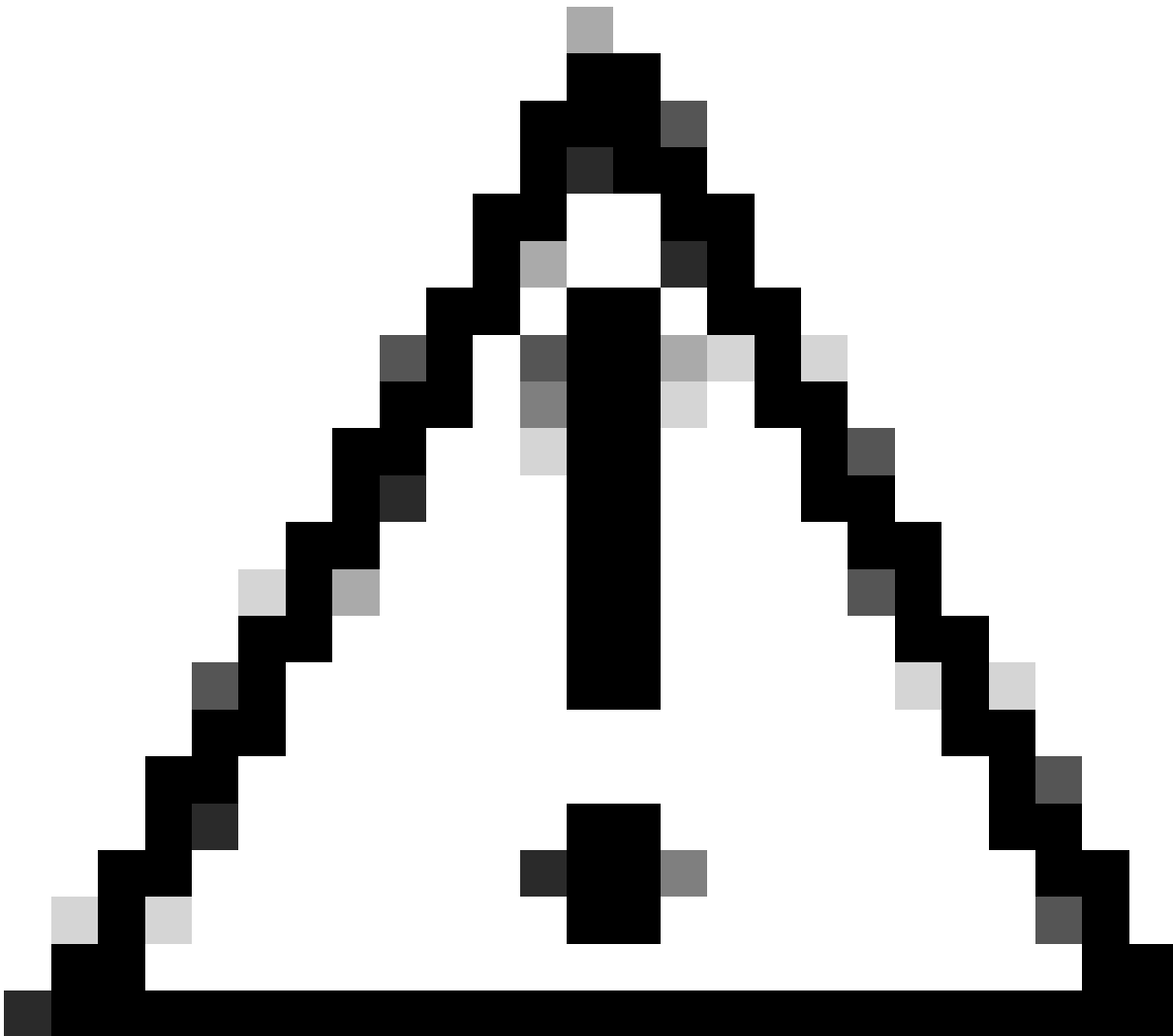
Configuration Item	Value
HTTPS ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
HTTPS minimum TLS version	TLS v1.2
LDAP TLS Ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
LDAP minimum TLS version	TLS v1.2
Reverse proxy TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
Reverse proxy minimum TLS version	TLS v1.2
SIP TLS ciphers	!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!AES128-SHA
SIP minimum TLS version	TLS v1.2
SMTP TLS Ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
SMTP minimum TLS version	TLS v1.2
TMS Provisioning Ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
TMS Provisioning minimum TLS version	TLS v1.2
UC server discovery TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
UC server discovery minimum TLS version	TLS v1.2
XMPP TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
XMPP minimum TLS version	TLS v1.2

Save

Seite "Cipher Settings" im Expressway-Web-Administratorportal



Hinweis: Nehmen Sie bei einem Expressway-Cluster die Änderungen nur auf dem primären Server vor. Die neue Konfiguration wird auf den Rest der Cluster-Mitglieder repliziert.



Vorsicht: Verwenden Sie die empfohlene Cluster-Neustartsequenz aus dem [Cisco Expressway Cluster Creation and Maintenance Deployment Guide](#). Starten Sie zunächst den primären Server neu, warten Sie, bis er über die Webschnittstelle erreichbar ist, und führen Sie dann das Gleiche mit jedem Peer aus, und zwar in der Reihenfolge, in der die Liste unter System > Clustering konfiguriert ist.

Deaktivieren einer Gruppe von Chiffren mithilfe eines allgemeinen Algorithmus

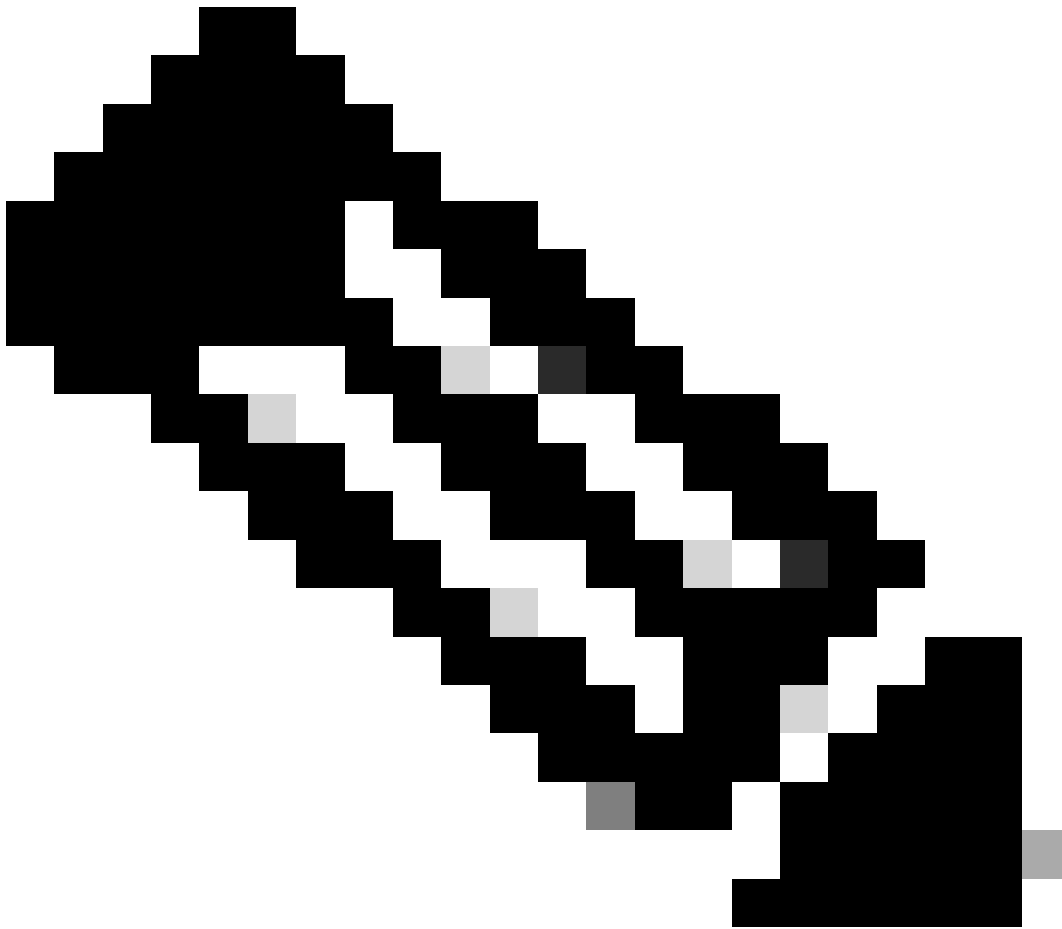
Um eine Gruppe von Chiffren mithilfe eines gemeinsamen Algorithmus zu deaktivieren, fügen Sie an die Standardzeichenfolge die folgenden Zeichen an: Trennzeichen, das ! oder - Zeichen und den zu deaktivierenden Algorithmusnamen. Die unterstützten Algorithmusnamen sind auf der [OpenSSL Ciphers Manpage](#) verfügbar. Wenn Sie z. B. alle Verschlüsselungen deaktivieren müssen, die den DHE-Algorithmus verwenden, konfigurieren Sie eine Verschlüsselungszeichenfolge wie diese:

<#root>

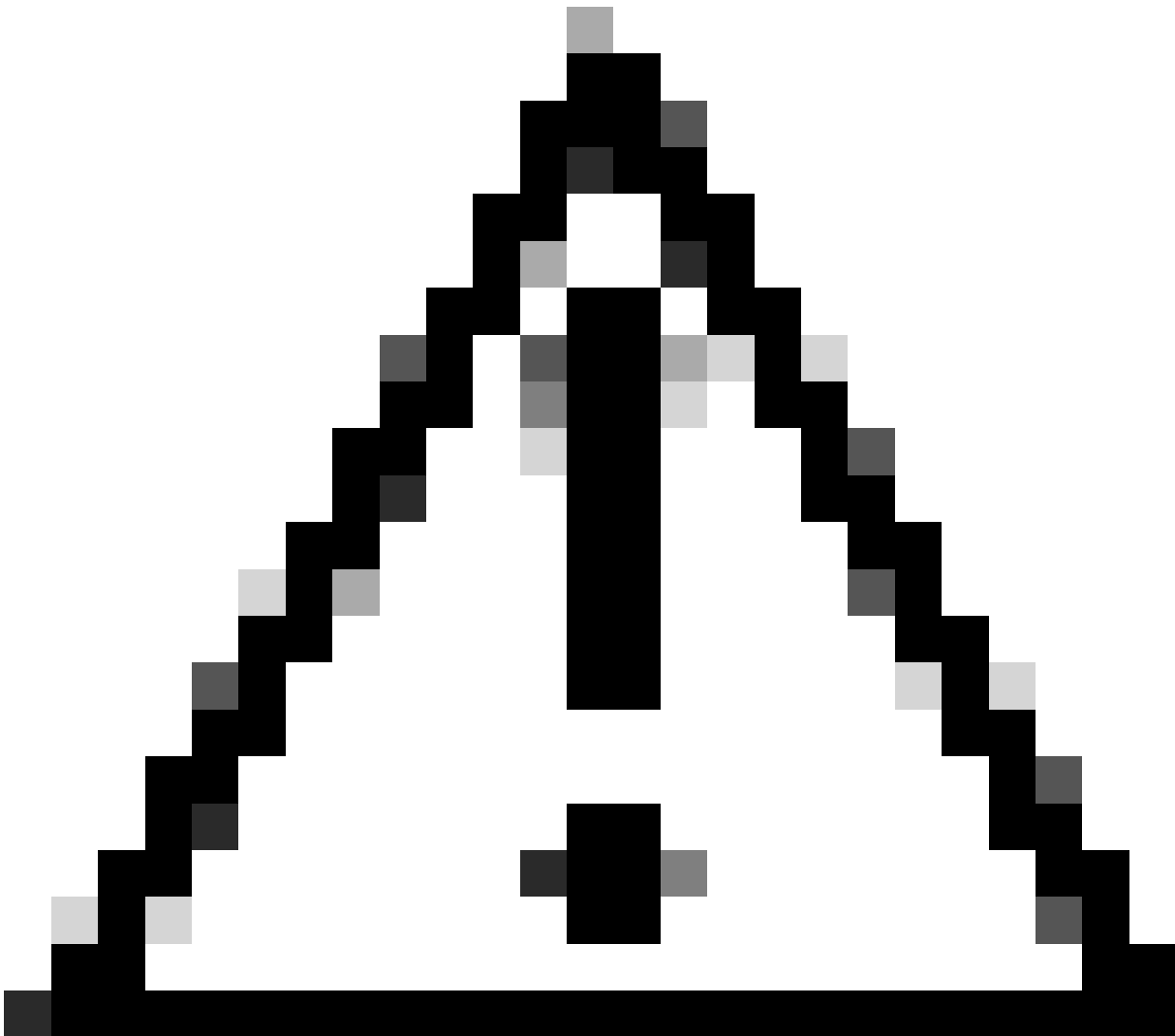
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH

:!DHE

Navigieren Sie zur Expressway-Web-Admin-Seite, navigieren Sie zu Maintenance > Security > Ciphers, weisen Sie die benutzerdefinierte Zeichenfolge den erforderlichen Protokollen zu, und klicken Sie auf Save. Damit die neue Konfiguration angewendet werden kann, muss das System neu gestartet werden.



Hinweis: Nehmen Sie bei einem Expressway-Cluster die Änderungen nur auf dem primären Server vor. Die neue Konfiguration wird auf den Rest der Cluster-Mitglieder repliziert.



Vorsicht: Verwenden Sie die empfohlene Cluster-Neustartsequenz aus dem [Cisco Expressway Cluster Creation and Maintenance Deployment Guide](#). Starten Sie zunächst den primären Server neu, warten Sie, bis er über die Webschnittstelle erreichbar ist, und führen Sie dann das Gleiche mit jedem Peer aus, und zwar in der Reihenfolge, in der die Liste unter System > Clustering konfiguriert ist.

Überprüfung

Überprüfen Sie die Liste der von der Cipher-Zeichenfolge zugelassenen Chiffren.

Sie können die benutzerdefinierte Verschlüsselungszeichenfolge mit dem Befehl `openssl ciphers -V "<cipher string>"` überprüfen. Überprüfen Sie die Ausgabe, um sicherzustellen, dass die unerwünschten Chiffren nach den Änderungen nicht mehr aufgeführt werden. In diesem Beispiel wird die Chiffrierzeichenfolge `EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHE` überprüft. Die Befehlsausgabe bestätigt, dass die Zeichenfolge keine der Chiffren zulässt, die den DHE-

Algorithmus verwenden:

<#root>

```
~ # openssl ciphers -V "ECDH: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
~ #
```

Testen einer TLS-Verbindung durch Aushandeln eines deaktivierten Verschlüsselungsverfahrens

Sie können den Befehl `openssl s_client` verwenden, um zu überprüfen, ob ein Verbindungsversuch mit einer deaktivierten Verschlüsselung abgelehnt wird. Verwenden Sie die Option `-connect`, um Ihre Expressway-Adresse und Ihren Port anzugeben, und die Option `-cipher`, um die einzelne Verschlüsselung anzugeben, die vom Client während des TLS-Handshakes ausgehandelt werden soll:

```
openssl s_client -connect <Adresse>:<Port> -cipher <Chiffre> -no_tls1_3
```

In diesem Beispiel wird von einem Windows-PC mit installiertem `openssl` aus versucht, eine TLS-Verbindung zu Expressway herzustellen. Der PC verhandelt als Client nur die unerwünschte DHE-RSA-AES256-CCM-Verschlüsselung, die den DHE-Algorithmus verwendet:

<#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>
```

In der Befehlsausgabe wird angezeigt, dass der Verbindungsversuch mit der Fehlermeldung "ssl/tls alert handshake failure:...\ssl\record\rec_layer_s3.c:865:SSL alert number 40" fehlgeschlagen ist, da der Expressway für die Verwendung von EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!e NULL:!aNULL:!aDH:!DHE-Verschlüsselungszeichenfolge für HTTPS-Verbindungen, die Verschlüsselungen deaktiviert, die den DHE-Algorithmus verwenden.



Hinweis: Damit Tests mit dem Befehl `openssl s_client` wie beschrieben funktionieren, muss die Option `-no_tls1_3` an den Befehl übergeben werden. Falls nicht enthalten, fügt der Client automatisch TLS 1.3-Chiffren in das ClientHello-Paket ein:

Urgent Pointer: 0

- > [Timestamps]
- > [SEQ/ACK analysis]
- TCP payload (247 bytes)
- Transport Layer Security
 - TLSv1.3 Record Layer: Handshake Protocol: Client Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 242
 - Handshake Protocol: Client Hello
 - Handshake Type: Client Hello (1)
 - Length: 238
 - Version: TLS 1.2 (0x0303)
 - Random: 19ec4e8994cc334599cf889d4e45a812029589923c4cfcf2cef6b6fc47ec2840
 - Session ID Length: 32
 - Session ID: e0d17cb402229aa46cab70b6a637ce38d9b5a228c7b360cb43f49086ce88d5df
 - Cipher Suites Length: 10
 - Cipher Suites (5 suites)
 - Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
 - Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
 - Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0xc09f)
 - Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
 - Compression Methods Length: 1

ClientHello-Paket mit automatisch hinzugefügten Chiffren

Wenn der Ziel-Expressway diese Chiffren unterstützt, kann eine davon ausgewählt werden, anstatt der spezifischen Chiffre, die Sie testen müssen. Die Verbindung ist erfolgreich, was Sie glauben lassen kann, dass eine Verbindung möglich war, indem der deaktivierte Schlüssel an den Befehl mit der Option `-cipher` übergeben wurde.

Überprüfen einer Paketerfassung eines TLS-Handshakes mit einem deaktivierten Cipher

Sie können eine Paketerfassung vom Testgerät oder vom Expressway erfassen, während Sie einen Verbindungstest mit einem der deaktivierten Verschlüsselungscodes durchführen. Sie können es dann mit Wireshark untersuchen, um die Handshake-Ereignisse weiter zu analysieren.

Suchen Sie nach dem ClientHello, das vom Testgerät gesendet wurde. Bestätigen Sie, dass nur die unerwünschte Testchiffre ausgehandelt wird, in diesem Beispiel eine Chiffre mit dem DHE-Algorithmus:

The image shows a Wireshark capture of a network stream on interface 'Ethernet0'. The packet list pane shows several packets, with packet 327 highlighted in red. This packet is a Client Hello from source 10.15.1.2 to destination 10.15.1.7 on port 28872. The packet details pane shows the following structure:

- Acknowledgment number (raw): 3235581935
- 0101 = Header Length: 20 bytes (5)
- Flags: 0x018 (PSH, ACK)
- Window: 16425
- [Calculated window size: 4204800]
- [Window size scaling factor: 256]
- Checksum: 0x16b7 [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- [Timestamps]
- [SEQ/ACK analysis]
- TCP payload (118 bytes)
- Transport Layer Security
 - TLSv1.2 Record Layer: Handshake Protocol: Client Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 113
 - Handshake Protocol: Client Hello
 - Handshake Type: Client Hello (1)
 - Length: 109
 - Version: TLS 1.2 (0x0303)
 - Random: e5cb04a72ae567a0963c5a4a5901db3720fabcf5980aa2ef5a5ecc099254c1bf8
 - Session ID Length: 0
 - Cipher Suites Length: 4
 - Cipher Suites (2 suites)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
 - Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
 - Compression Methods Length: 1

Beispiel für ein ClientHello-Paket in Wireshark

:

Vergewissern Sie sich, dass Expressway mit einem schwerwiegenden TLS-Warnpaket antwortet, und verweigern Sie die Verbindung. Da Expressway in diesem Beispiel keine DHE-Chiffren für die konfigurierte Verschlüsselungszeichenfolge für das HTTPS-Protokoll unterstützt, antwortet es mit einem schwerwiegenden TLS-Warnpaket mit dem Fehlercode 40.

Wireshark interface showing network traffic analysis. The packet list pane displays several packets, with packet 329 highlighted in red, indicating a TLS alert. The packet details pane shows the structure of this alert message.

No.	Time	Source	Src port	Destination	Dst port	Protocol	Length	Info
324	2024-07-14 23:00:32.459025	10.15.1.2	28872	10.15.1.7	443	TCP	66	28872 → 443 [SYN, ECE, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
325	2024-07-14 23:00:32.459666	10.15.1.7	443	10.15.1.2	28872	TCP	66	443 → 28872 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
326	2024-07-14 23:00:32.459760	10.15.1.2	28872	10.15.1.7	443	TCP	54	28872 → 443 [ACK] Seq=1 Ack=1 Win=4204800 Len=0
327	2024-07-14 23:00:32.460733	10.15.1.2	28872	10.15.1.7	443	TLSv1.2	172	Client Hello
328	2024-07-14 23:00:32.461070	10.15.1.7	443	10.15.1.2	28872	TCP	60	443 → 28872 [ACK] Seq=1 Ack=119 Win=64128 Len=0
329	2024-07-14 23:00:32.461855	10.15.1.7	443	10.15.1.2	28872	TLSv1.2	61	Alert (Level: Fatal, Description: Handshake Failure)
330	2024-07-14 23:00:32.461855	10.15.1.7	443	10.15.1.2	28872	TCP	60	443 → 28872 [FIN, ACK] Seq=8 Ack=119 Win=64128 Len=0

Packet 329 details:

- Frame 329: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface \Device\NPF_{122607A1-10A8-47F6-9069-936EB0CAAE1C}, id 0
- Ethernet II, Src: VMware_b3:5c:7a (00:50:56:b3:5c:7a), Dst: VMware_b3:fe:d6 (00:50:56:b3:fe:d6)
- Internet Protocol Version 4, Src: 10.15.1.7, Dst: 10.15.1.2
- Transmission Control Protocol, Src Port: 443, Dst Port: 28872, Seq: 1, Ack: 119, Len: 7
 - Source Port: 443
 - Destination Port: 28872
 - [Stream index: 2]
 - [Conversation completeness: Complete, WITH_DATA (31)]
 - [TCP Segment Len: 7]
 - Sequence Number: 1 (relative sequence number)
 - Sequence Number (raw): 3235581935
 - [Next Sequence Number: 8 (relative sequence number)]
 - Acknowledgment Number: 119 (relative ack number)
 - Acknowledgment number (raw): 810929090
 - 0101 = Header Length: 20 bytes (5)
 - Flags: 0x018 (PSH, ACK)
 - Window: 501
 - [Calculated window size: 64128]
 - [Window size scaling factor: 128]
 - Checksum: 0x163f [unverified]
 - [Checksum Status: Unverified]
 - Urgent Pointer: 0
 - [Timestamps]
 - [SEQ/ACK analysis]
 - TCP payload (7 bytes)
- Transport Layer Security
 - TLSv1.2 Record Layer: Alert (Level: Fatal, Description: Handshake Failure)
 - Content Type: Alert (21)
 - Version: TLS 1.2 (0x0303)
 - Length: 2
 - Alert Message
 - Level: Fatal (2)
 - Description: Handshake Failure (40)

Ein schwerwiegendes TLS-Warnpaket in Wireshark

Zugehörige Informationen

- [OpenSSL-Verschlüsselungs-Manpage](#)
- [Cisco Expressway Administrator Guide \(X15.0\) - Kapitel: Sicherheitsmanagement - Konfigurieren der minimalen TLS-Version und der Cipher Suites](#)

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.