

Expressway SSL暗号設定のカスタマイズ

内容

[はじめに](#)

[前提条件](#)

[要件](#)

[使用するコンポーネント](#)

[背景説明](#)

[暗号文字列の検査](#)

[パケットキャプチャによるTLSハンドシェイクでの暗号ネゴシエーションの検査](#)

[設定](#)

[特定の暗号の無効化](#)

[共通アルゴリズムを使用した暗号グループの無効化](#)

[確認](#)

[暗号ストリングによって許可される暗号のリストの検査](#)

[無効な暗号をネゴシエートしてTLS接続をテストする](#)

[無効な暗号を使用したTLSHandshakeのパケットキャプチャの検査](#)

[関連情報](#)

はじめに

このドキュメントでは、Expresswayで事前設定された暗号文字列をカスタマイズする手順について説明します。

前提条件

要件

次の項目に関する知識が推奨されます。

- Cisco ExpresswayまたはCisco VCS
- TLSプロトコル。

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- Cisco ExpresswayバージョンX15.0.2

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認して

ください。

背景説明

Expresswayのデフォルト設定には設定済みの暗号文字列が含まれており、これは互換性を保つために、一部の企業セキュリティポリシーでは脆弱と見なされる可能性のある一部の暗号をサポートできるようにします。各環境の特定のポリシーに合わせて調整するために、暗号ストリングをカスタマイズできます。

Expresswayでは、次の各プロトコルに対して独立した暗号文字列を設定できます。

- HTTPS
- [LDAP]
- 逆プロキシ
- SIP
- SMTP
- TMSプロビジョニング
- UCサーバの検出
- XMPP

暗号文字列は、『[OpenSSL Ciphersのマニュアルページ](#)』に記載されているOpenSSLの形式に従います。現在のExpresswayバージョンX15.0.2には、デフォルト文字列のEECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDHが付属しており、すべてのプロトコルで均等に事前設定されています。Web管理ページのMaintenance > Security > Ciphersで、各プロトコルに割り当てられている暗号文字列を変更して、共通のアルゴリズムを使用している特定の暗号または暗号のグループを追加または削除できます。

暗号文字列の検査

openssl ciphers -V "<cipher string>"コマンドを使用すると、特定の文字列で許可されているすべての暗号を含むリストを出力できます。これは、暗号を視覚的に調べる場合に便利です。次の例は、デフォルトのExpressway暗号文字列を調べた場合の出力を示しています。

```
<#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
~ #

```

パケットキャプチャによるTLSハンドシェイクでの暗号ネゴシエーションの検査

パケットキャプチャでTLSネゴシエーションをキャプチャすることにより、Wiresharkを使用して暗号ネゴシエーションの詳細を検査できます。

TLSハンドシェイクプロセスには、クライアントデバイスによって送信されたClientHelloパケットが含まれ、接続プロトコルに対して設定された暗号文字列に従って、サポートする暗号のリストが提供されます。サーバはリストを確認し、リストを自身の許可された暗号のリスト（自身の暗号ストリングで決定）と比較して、両方のシステムがサポートする、暗号化されたセッションに使用する暗号を選択します。次に、選択された暗号を示すServerHelloパケットで応答します。TLS 1.2と1.3のハンドシェイクダイアログには重要な違いがありますが、暗号ネゴシエーションのメカニズムでは、両方のバージョンで同じ原理を使用しています。

次に、Wiresharkで表示される、Webブラウザとポート443上のExpressway間のTLS 1.3暗号ネゴシエーションの例を示します。

No.	Time	Source	Src port	Destination	Dest 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

最初に、ブラウザはサポートする暗号のリストを含むClientHelloパケットを送信します。

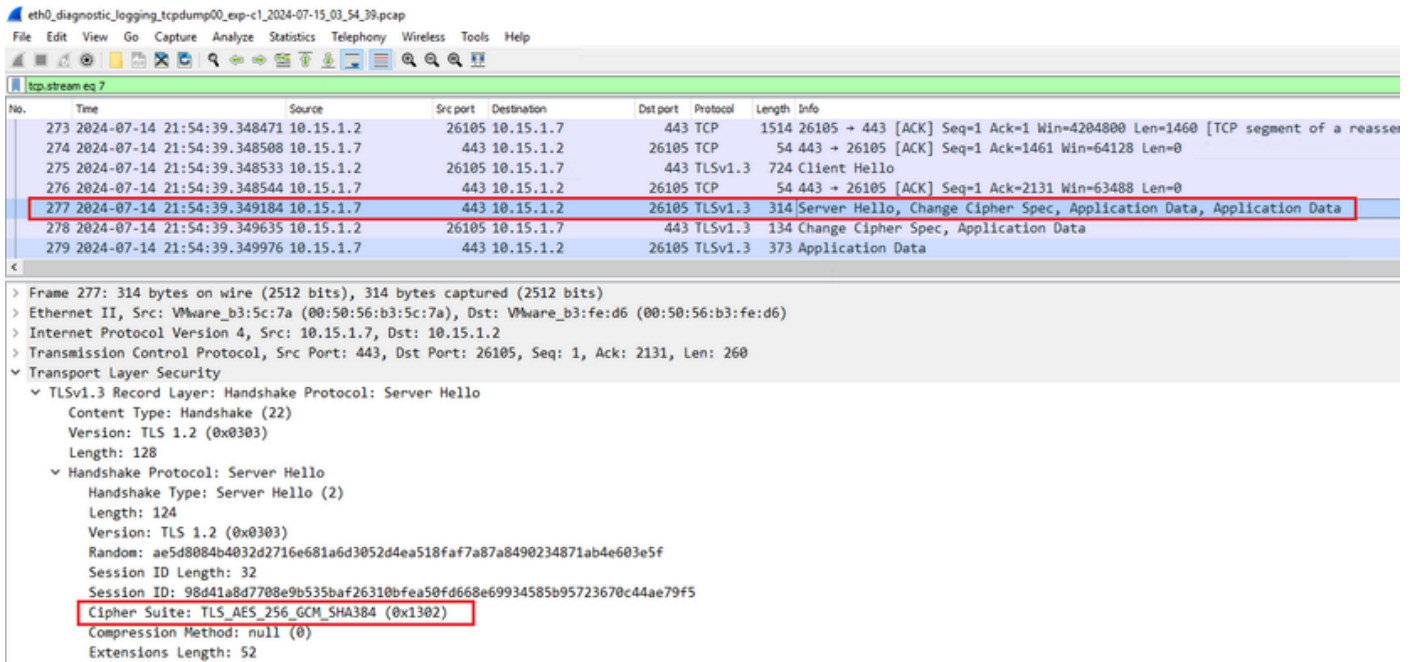
The image shows a Wireshark capture of a network packet. The packet list pane shows a ClientHello packet (No. 275) from 10.15.1.2 to 10.15.1.7 on port 443. The packet details pane is expanded to show the TLSv1.3 Record Layer: Handshake Protocol: Client Hello. The supported cipher suites list is expanded, showing 16 suites. The selected cipher suite, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030), is highlighted in blue.

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 (0xc03a)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc038)
 - 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

Expresswayは、HTTPSプロトコル用に設定されている暗号文字列を確認し、自身とクライアントの両方がサポートする暗号を見つけます。この例では、ECDHE-RSA-AES256-GCM-SHA384暗号が選択されています。Expresswayは、選択された暗号を示すServerHelloパケットで応答します。



```
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
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
  ▼ TLSv1.3 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
```

WiresharkでのServerHelloパケットの例

設定

OpenSSL暗号文字列フォーマットには、特定の暗号や共通コンポーネントを共有する暗号のグループを削除するなどの操作を文字列に対して実行するための特殊文字がいくつか含まれています。これらのカスタマイズの目的は通常、暗号を削除することであるため、これらの例で使用されている文字は次のとおりです。

- -文字。リストから暗号を削除するために使用します。削除された暗号の一部またはすべてが、文字列の後半に表示されるオプションによって再び許可される場合があります。
- !文字。リストから暗号を削除するためにも使用されます。これを使用する場合、削除された暗号は、文字列の後に現れる他のオプションによって再度許可することはできません。
- :文字。リスト内の項目間の区切り文字として機能します。

どちらも文字列から暗号を削除するために使用できますが、!が優先されます。特殊文字の完全なリストについては、『[OpenSSL Ciphers Manpage](#)』を参照してください。



注：OpenSSLサイトでは、!文字を使用する場合、「削除された暗号は、明示的に指定されている場合でもリストに再び表示されることはありません」と規定されています。これは、暗号がシステムから永久に削除されることを意味するのではなく、暗号文字列の解釈の範囲を指します。

特定の暗号の無効化

特定の暗号を無効にするには、デフォルト文字列に区切り記号:、!または-、および無効にする暗号名を追加します。暗号名は、[OpenSSL Ciphers Manpage](#)で使用可能なOpenSSLの命名形式に従う必要があります。たとえば、SIP接続でAES128-SHA暗号を無効にする必要がある場合は、次のように暗号文字列を設定します。

```
<#root>
```

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

```
:!AES128-SHA
```

次に、Expressway Web管理ページに移動し、Maintenance > Security > Ciphersの順に選択し、カスタム文字列を必要なプロトコルに割り当て、Saveをクリックします。新しい設定を適用するには、システムを再起動する必要があります。この例では、カスタム文字列は、SIP TLS ciphersの下のSIPプロトコルに割り当てられます。

Status > System > Configuration > Applications > Users > Maintenance >

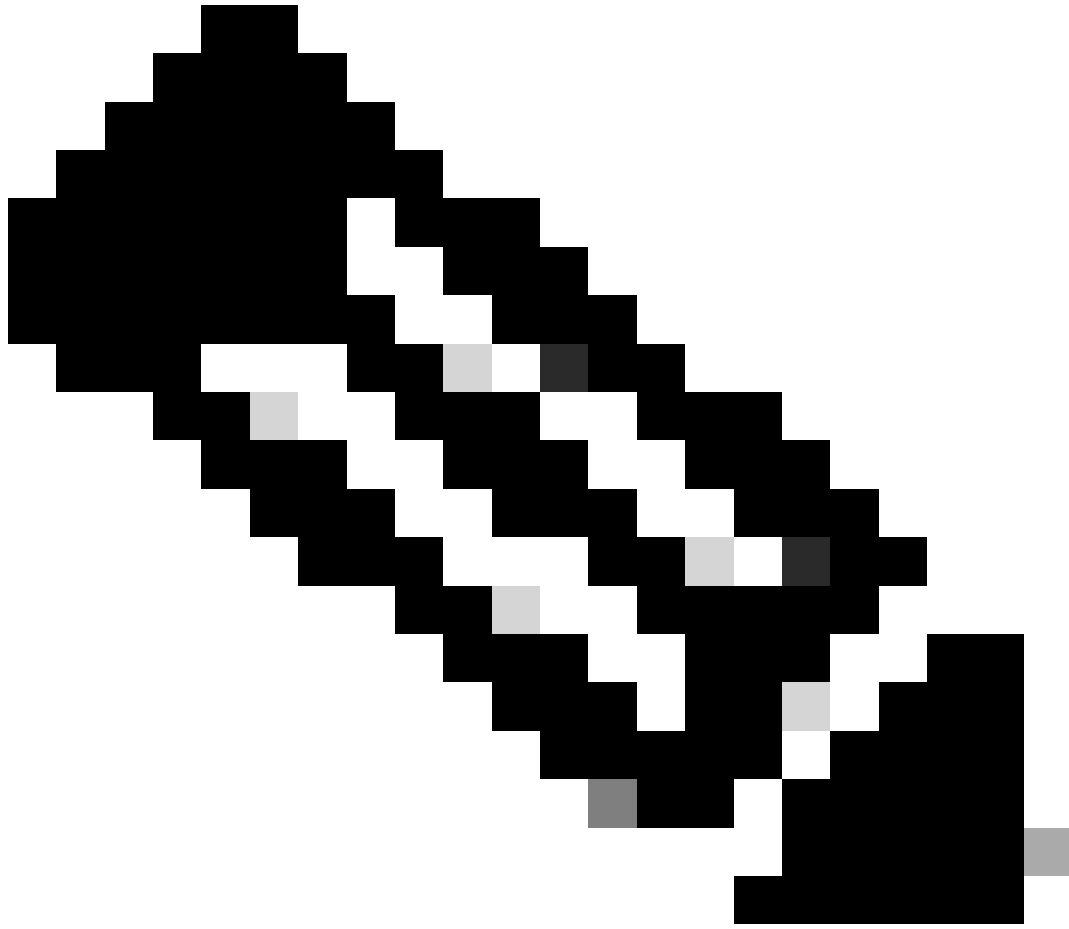
Ciphers

Configuration

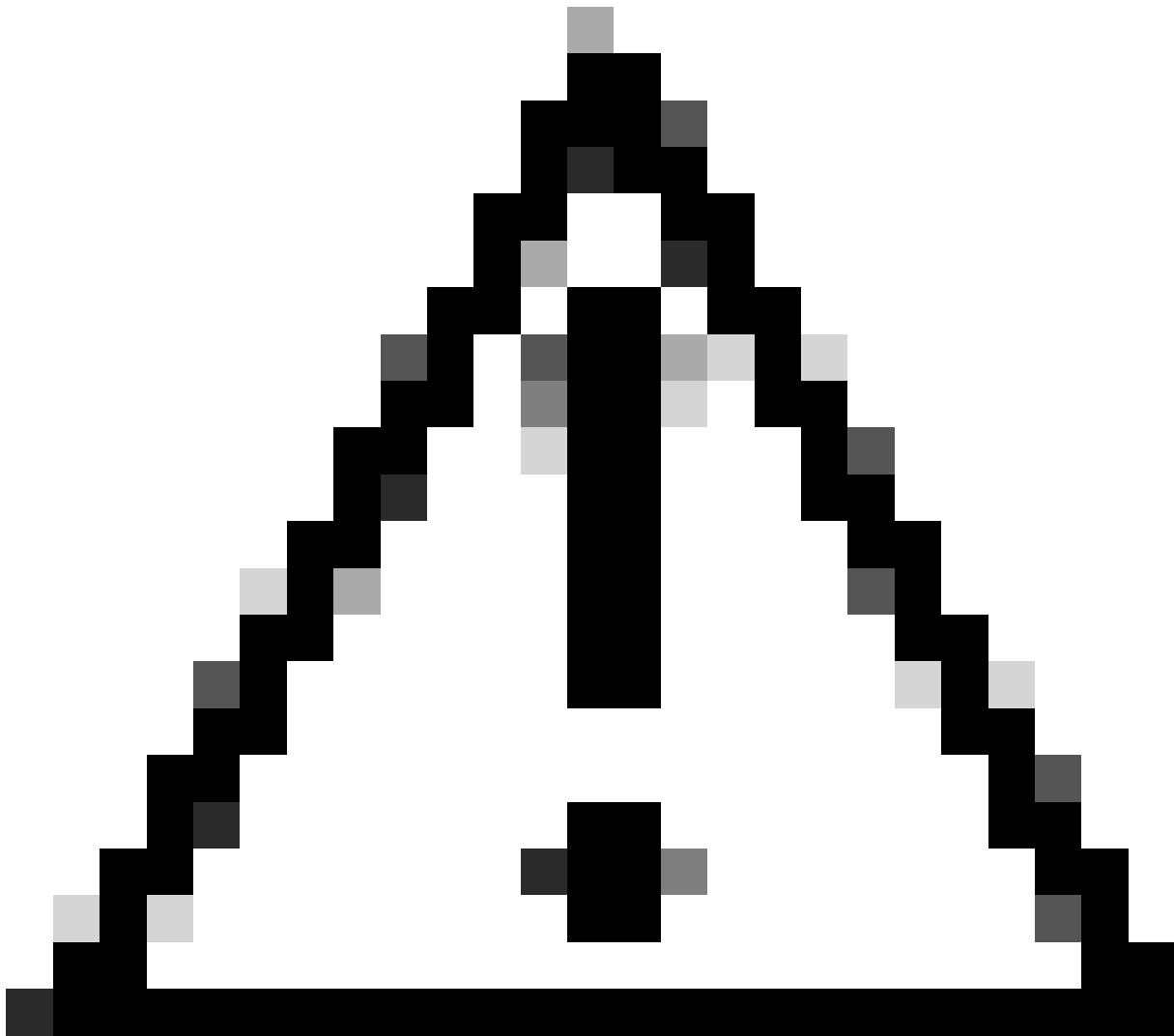
HTTPS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
HTTPS minimum TLS version	TLS v1.2
LDAP TLS Ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
LDAP minimum TLS version	TLS v1.2
Reverse proxy TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
Reverse proxy minimum TLS version	TLS v1.2
SIP TLS ciphers	IMEDIUM:LOW:3DES:1MD5:IPSK:! !eNULL:!!eNULL:!!eDH:IAES128-SHA
SIP minimum TLS version	TLS v1.2
SMTP TLS Ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
SMTP minimum TLS version	TLS v1.2
TMS Provisioning Ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
TMS Provisioning minimum TLS version	TLS v1.2
UC server discovery TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
UC server discovery minimum TLS version	TLS v1.2
XMPP TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:!
XMPP minimum TLS version	TLS v1.2

Save

Expressway Web管理ポータル暗号設定ページ



注：Expresswayクラスタの場合は、プライマリサーバでのみ変更を行います。新しい設定は、残りのクラスタメンバーに複製されます。



注意：『[Cisco Expresswayクラスタの作成およびメンテナンス導入ガイド](#)』に記載されている推奨されるクラスタリブートシーケンスを使用してください。最初にプライマリサーバを再起動し、Webインターフェイスからプライマリサーバにアクセスできるようになるまで待ちます。次に、System > Clusteringで設定されたリストに従って、各ピアで同じ操作を実行します。

共通アルゴリズムを使用した暗号グループの無効化

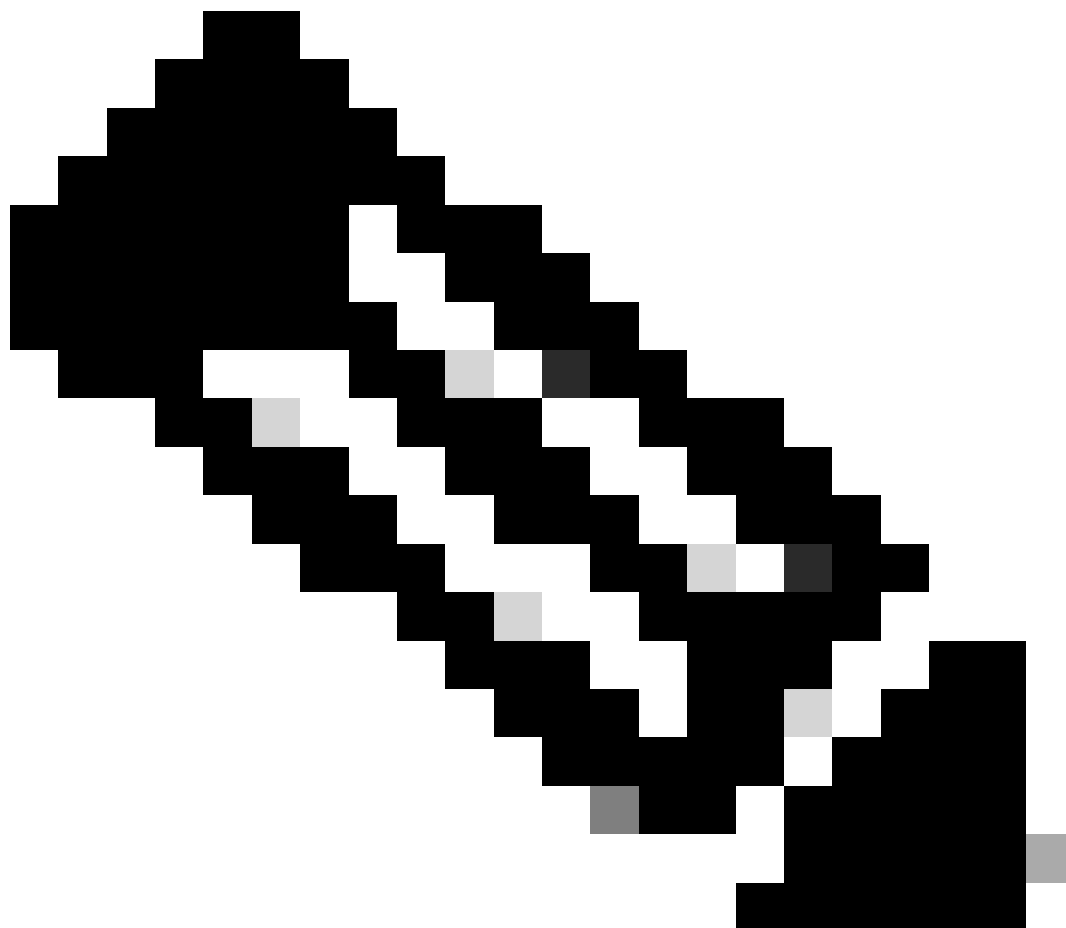
共通のアルゴリズムを使用している暗号のグループを無効にするには、デフォルト文字列に、:セパレータ、!記号または-記号、および無効にするアルゴリズム名を追加します。サポートされるアルゴリズム名は、『[OpenSSL Ciphers Manpage](#)』で入手できます。たとえば、DHEアルゴリズムを使用するすべての暗号を無効にする必要がある場合は、次のように暗号ストリングを設定します。

<#root>

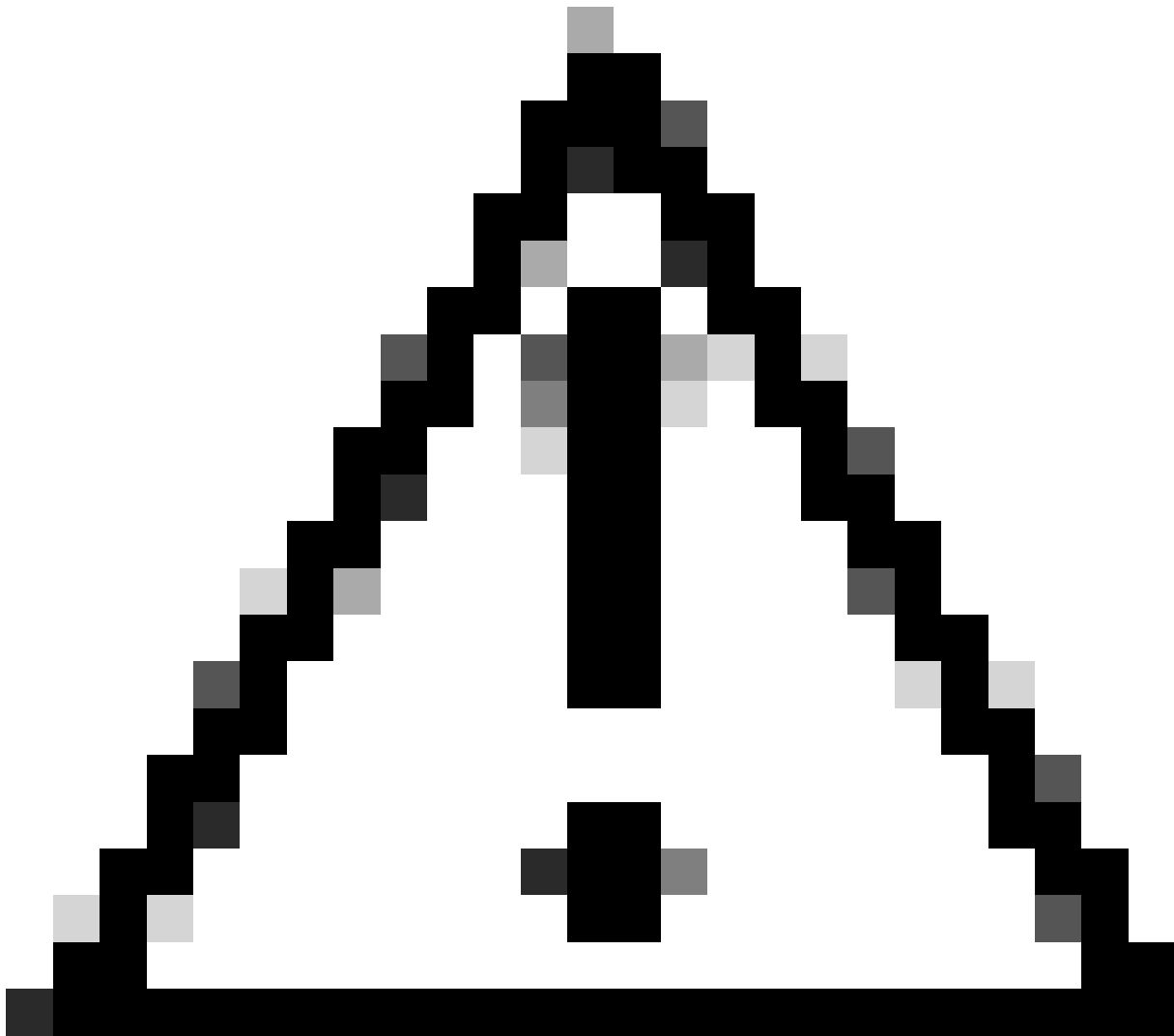
```
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
```

: !DHE

Expressway Web Adminページに移動し、Maintenance > Security > Ciphersの順に選択し、カスタム文字列を必要なプロトコルに割り当て、Saveをクリックします。新しい設定を適用するには、システムを再起動する必要があります。



注：Expresswayクラスタの場合は、プライマリサーバでのみ変更を行います。新しい設定は、残りのクラスタメンバーに複製されます。



注意：『[Cisco Expresswayクラスタの作成およびメンテナンス導入ガイド](#)』に記載されている推奨されるクラスタリブートシーケンスを使用してください。最初にプライマリサーバを再起動し、Webインターフェイスからプライマリサーバにアクセスできるようになるまで待ちます。次に、System > Clusteringで設定されたリストに従って、各ピアで同じ操作を実行します。

確認

暗号ストリングによって許可される暗号のリストの検査

openssl ciphers -V "<cipher string>"コマンドを使用して、カスタマイズした暗号文字列を検査できます。変更後に不要な暗号がリストされていないことを確認するには、出力を確認します。この例では、暗号文字列EECDH:EDH:HIG:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHEが検査されます。このコマンドの出力は、この文字列でDHEアルゴリズムを使用する暗号が許可されていないことを示しています。

```
<#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
~ #
```

無効な暗号をネゴシエートしてTLS接続をテストする

openssl s_clientコマンドを使用すると、無効な暗号を使用した接続の試行が拒否されたことを確認できます。Expresswayのアドレスとポートを指定するには-connectオプションを使用し、TLSハンドシェイク中にクライアントによってネゴシエートされる単一の暗号を指定するには-cipherオプションを使用します。

```
openssl s_client -connect <address>:<port> -cipher <cipher> -no_tls1_3
```

この例では、opensslがインストールされたWindows PCからExpresswayへのTLS接続が試行されます。クライアントであるPCは、DHEアルゴリズムを使用する望ましくないDHE-RSA-AES256-CCM暗号のみをネゴシエートします。

```
<#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>
```

このコマンド出力では、ExpresswayがEECDH:EDH:HIGH:-AES256+SHAを使用するように設定されているため、「ssl/tls alert handshake failure:...\ssl\record\rec_layer_s3.c:865:SSL alert number 40」というエラーメッセージで接続の試行が失敗していることが示されています

HTTPS接続の暗号化ストリング
: !MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHE。DHEアルゴリズムを使用する暗号化を無効にします。



注:openssl s_clientコマンドを使用したテストが説明どおりに機能するためには、-no_tls1_3オプションをコマンドに渡す必要があります。含まれていない場合、クライアントは自動的にTLS 1.3暗号をClientHelloパケットに挿入します。

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 (0xc030)
 - Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
 - Compression Methods Length: 1

Ciphers automatically inserted by the openssl s_client command

Cipher passed with the -cipher option

自動的に追加された暗号を含むClientHelloパケット

ターゲットのExpresswayがそれらの暗号をサポートしている場合は、テストする必要がある特定の暗号の代わりに、そのうちの1つを選択できます。接続は成功し、-cipherオプションを指定してコマンドに渡された無効な暗号を使用することで接続が可能になったと判断できます。

無効な暗号を使用したTLSハンドシェイクのパケットキャプチャの検査

無効になっている暗号のいずれかを使用して接続テストを実行しながら、テストデバイスまたはExpresswayからパケットキャプチャを収集できます。その後、Wiresharkでハンドシェイクイベントをさらに分析するために検査できます。

テストデバイスから送信されたClientHelloを見つけます。望ましくないテスト暗号だけをネゴシエートすることを確認します。この例では、DHEアルゴリズムを使用する暗号です。

The image shows a Wireshark capture of a network session. The packet list pane at the top shows several packets. Packet 327 is highlighted in red and is a Client Hello packet from 10.15.1.2 to 10.15.1.7. The packet details pane below shows the structure of this Client Hello packet, including the TLSv1.2 Record Layer, Handshake Protocol, and Cipher Suites. The Cipher Suites list includes TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, which is highlighted in red.

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

```

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 (0xc09f)
        Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
      Compression Methods Length: 1
  
```

WiresharkでのClientHelloパケットの例

:

Expresswayが致命的なTLSアラートパケットで応答し、接続を拒否することを確認します。この例では、ExpresswayはHTTPSプロトコルに対して設定された暗号文字列に従ってDHE暗号をサポートしていないため、エラーコード40を含む致命的なTLSアラートパケットで応答します。

Wireshark interface showing a network capture on 'Ethernet0'. The packet list pane displays several packets, with packet 329 highlighted in red. The packet details pane for packet 329 is expanded, showing the following information:

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

The packet details pane for packet 329 shows the following structure:

- 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)

WiresharkのTLS致命的アラートパケット

関連情報

- [OpenSSL暗号のマニュアルページ](#)
- [Cisco Expressway管理者ガイド\(X15.0\) – 章：セキュリティの管理 – 最小限のTLSバージョンと暗号スイートの設定](#)

翻訳について

シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。