Expressway SSL暗号設定のカスタマイズ

内容

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
```

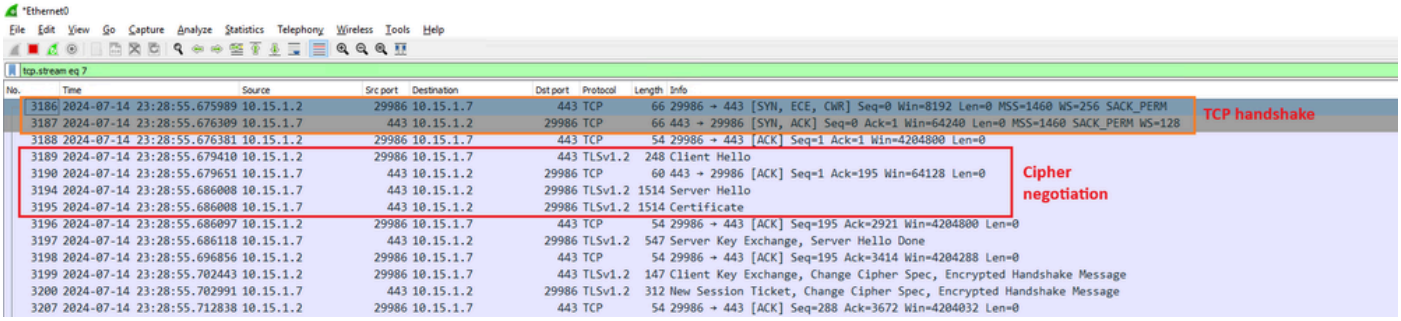
```
0xC0,0x24 - ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(256) Mac=SHA384
0xC0,0x28 - ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(256) Mac=SHA384
0xC0,0x23 - ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA256
0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 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。

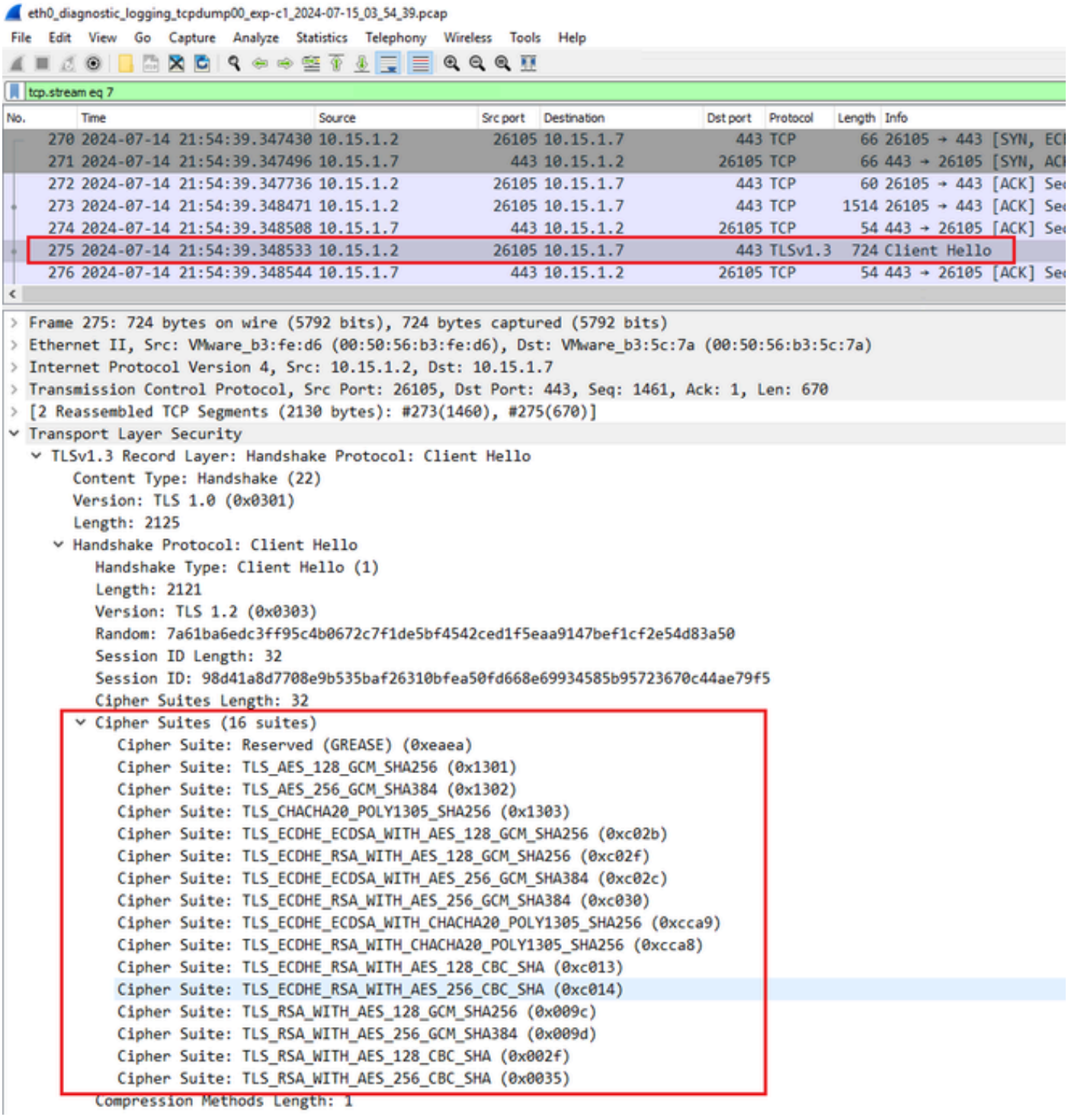
TLSハンドシェイクプロセスには、クライアントデバイスによって送信されたClientHelloパケットが含まれ、接続プロトコルに対して設定された暗号文字列に従って、サポートする暗号のリストが提供されます。サーバはリストを確認し、リストを自身の許可された暗号のリスト（自身の暗号ストリングで決定）と比較して、両方のシステムがサポートする、暗号化されたセッションに使用する暗号を選択します。次に、選択された暗号を示すServerHelloパケットで応答します。TLS 1.2と1.3のハンドシェイクダイアログには重要な違いがありますが、暗号ネゴシエーションのメカニズムでは、両方のバージョンで同じ原理を使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

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。



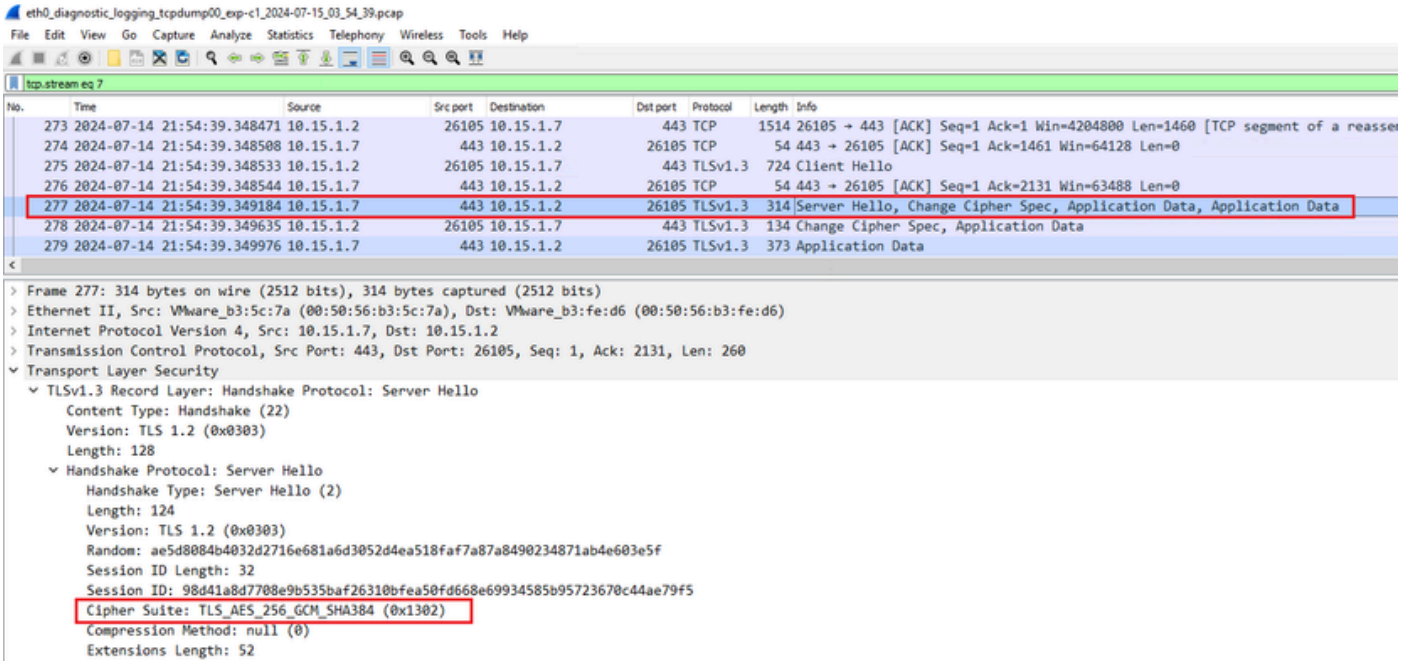
WiresharkでのTLSハンドシェイクの例

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。



WiresharkでのClientHelloパケットの例

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。



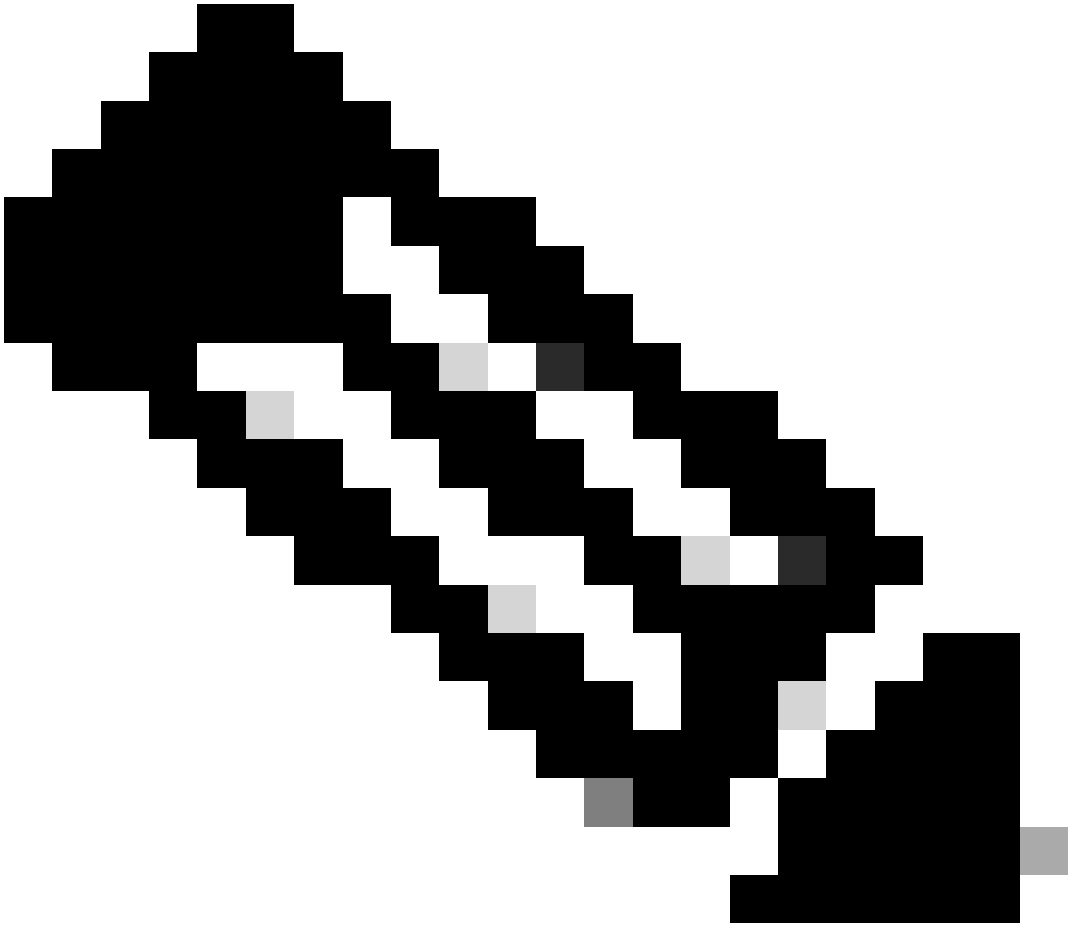
WiresharkでのServerHelloパケットの例

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

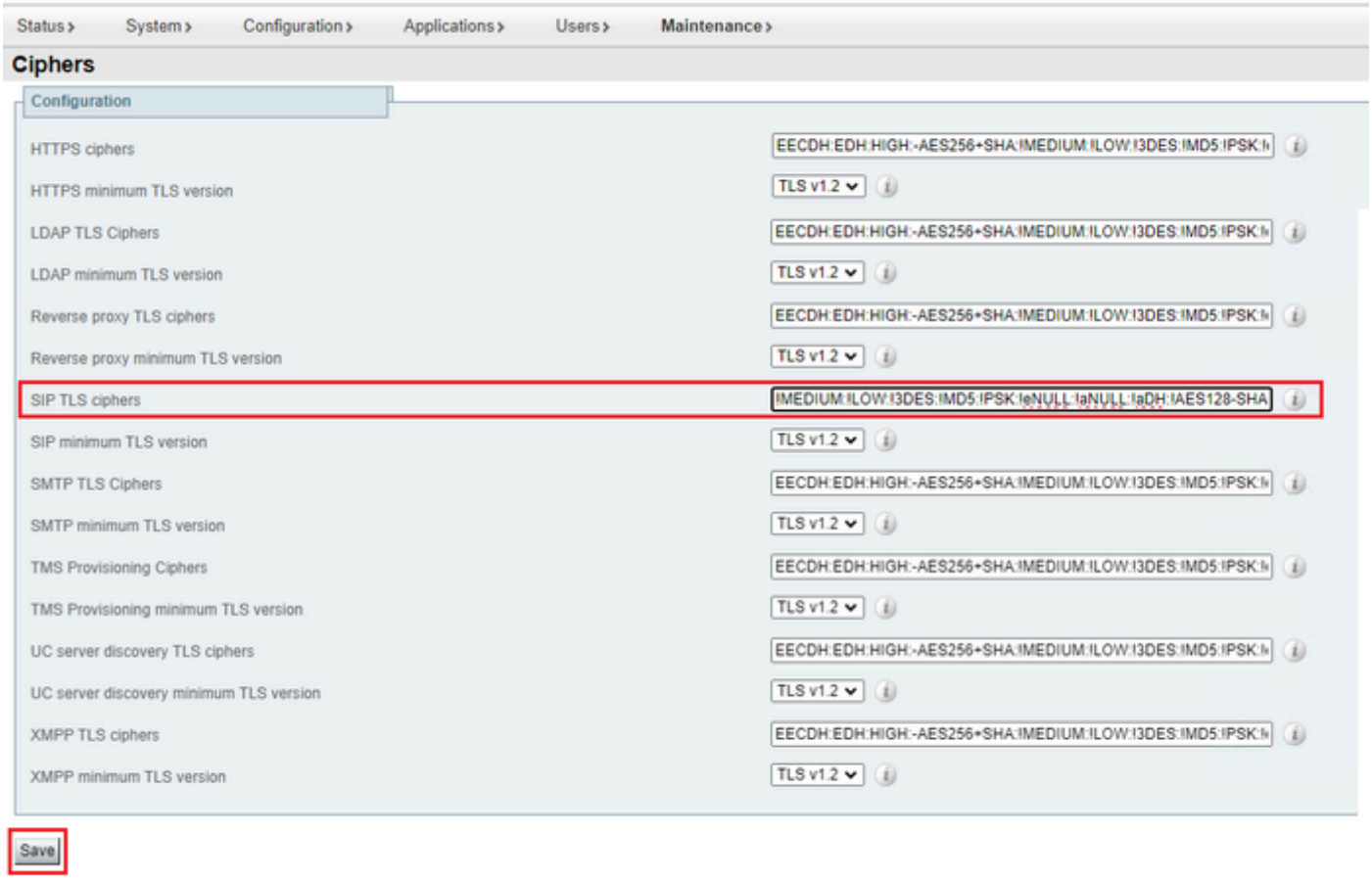
特定の暗号を無効にするには、デフォルト文字列に区切り記号:、!または-、および無効にする暗号名を追加します。暗号名は、OpenSSL Ciphers Manpageで使用可能なOpenSSLの命名形式に従う必要があります。たとえば、SIP接続でAES128-SHA暗号を無効にする必要がある場合は、次のように暗号文字列を設定します。

<#root>

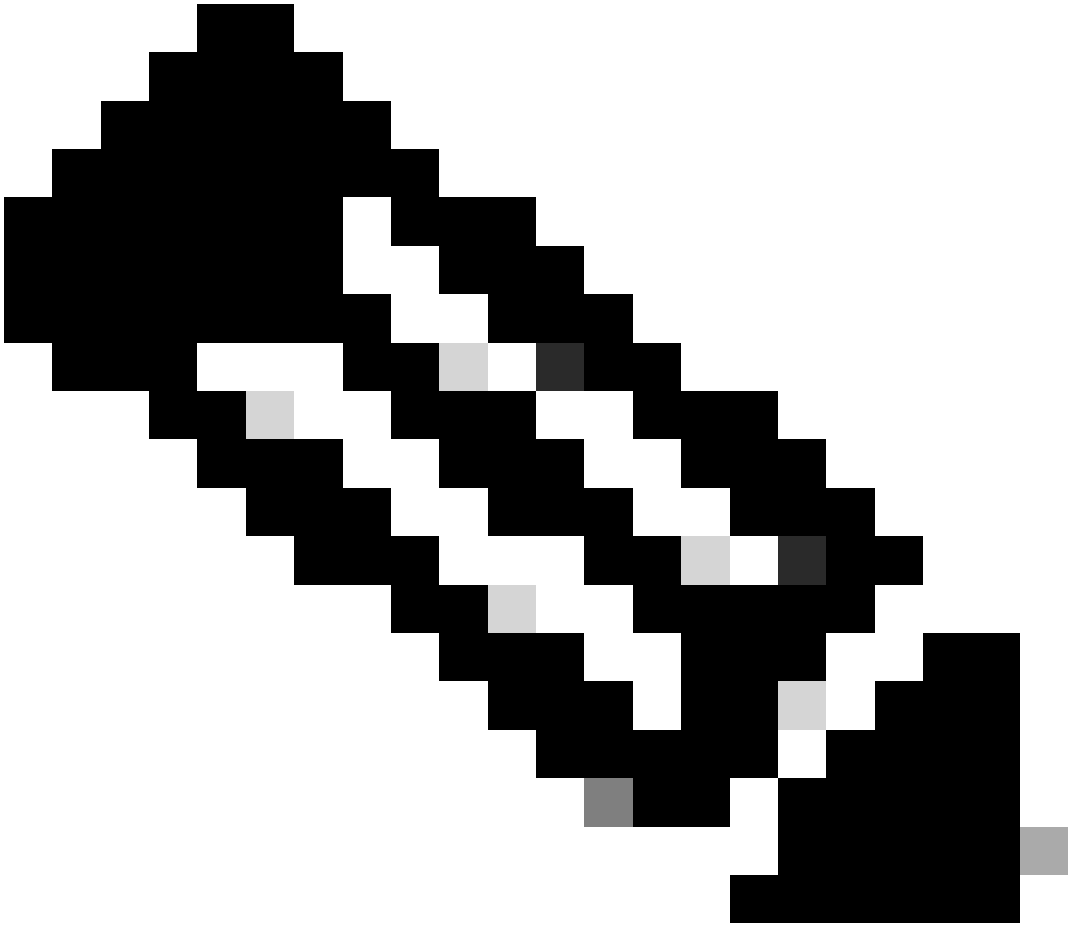
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH

:!AES128-SHA

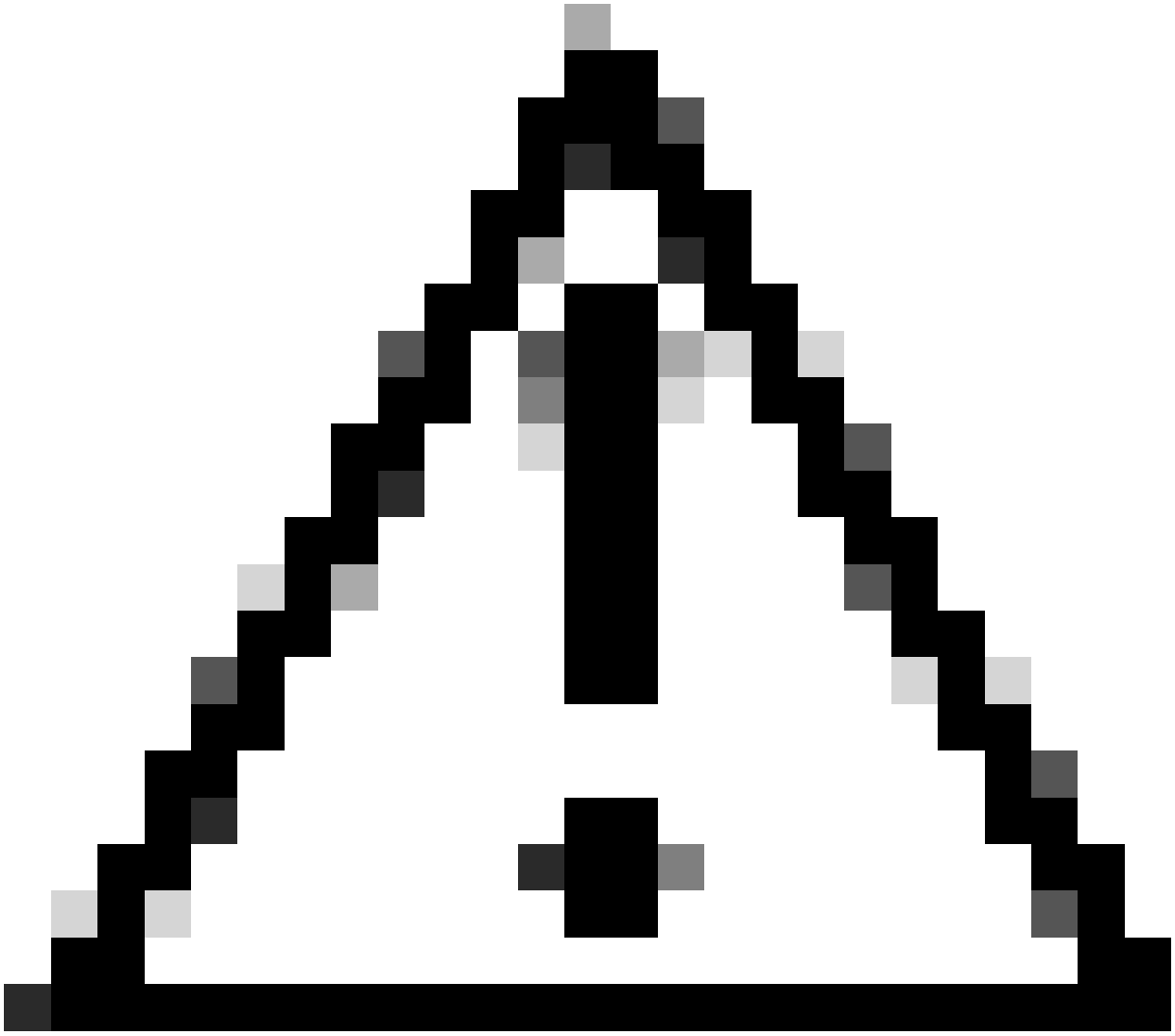
次に、Expressway Web管理ページに移動し、Maintenance > Security > Ciphersの順に選択し、カスタム文字列を必要なプロトコルに割り当て、Saveをクリックします。 新しい設定を適用するには、システムを再起動する必要があります。この例では、カスタム文字列は、SIP TLS ciphersの下のSIPプロトコルに割り当てられます。



Expressway Web管理ポータルの暗号設定ページ



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

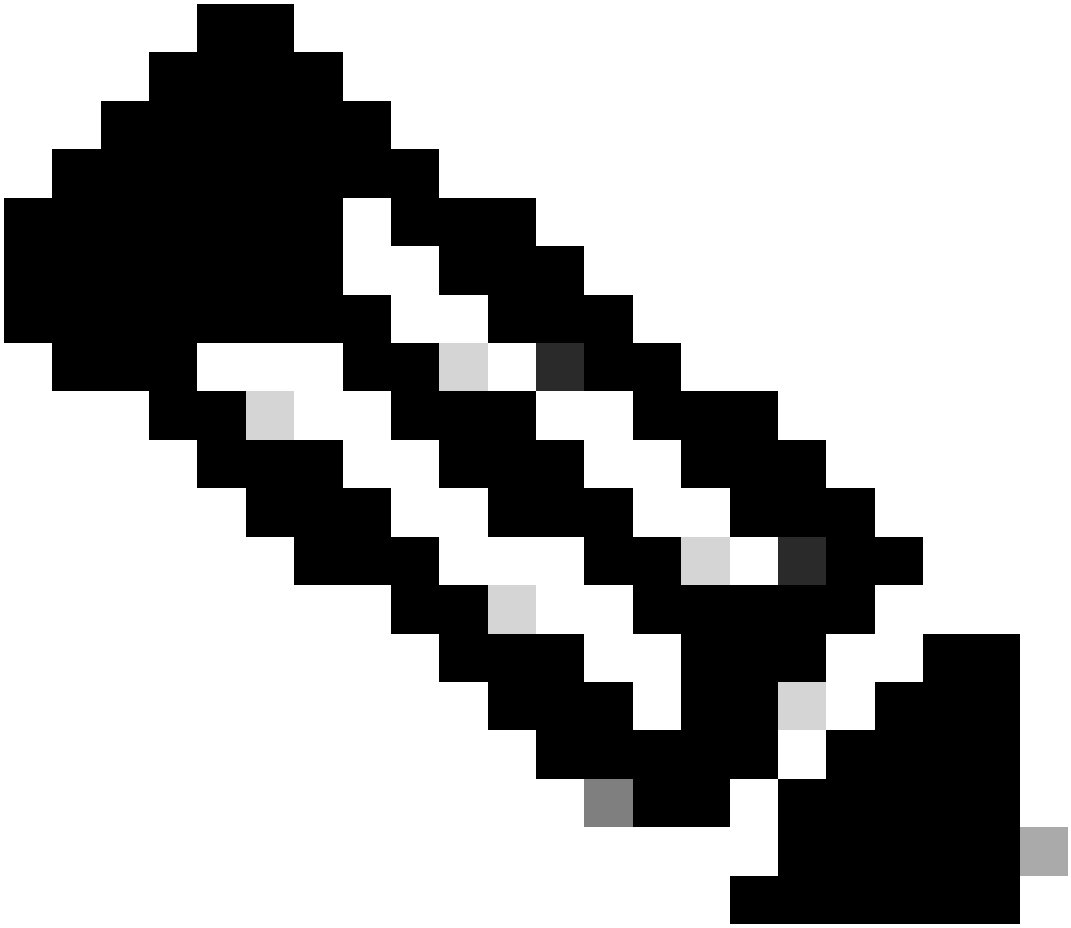
共通のアルゴリズムを使用している暗号のグループを無効にするには、デフォルト文字列に、:セパレータ、!記号または-記号、および無効にするアルゴリズム名を追加します。サポートされるアルゴリズム名は、『OpenSSL Ciphers Manpage』で入手できます。たとえば、DHEアルゴリズムを使用するすべての暗号を無効にする必要がある場合は、次のように暗号ストリングを設定します。

<#root>

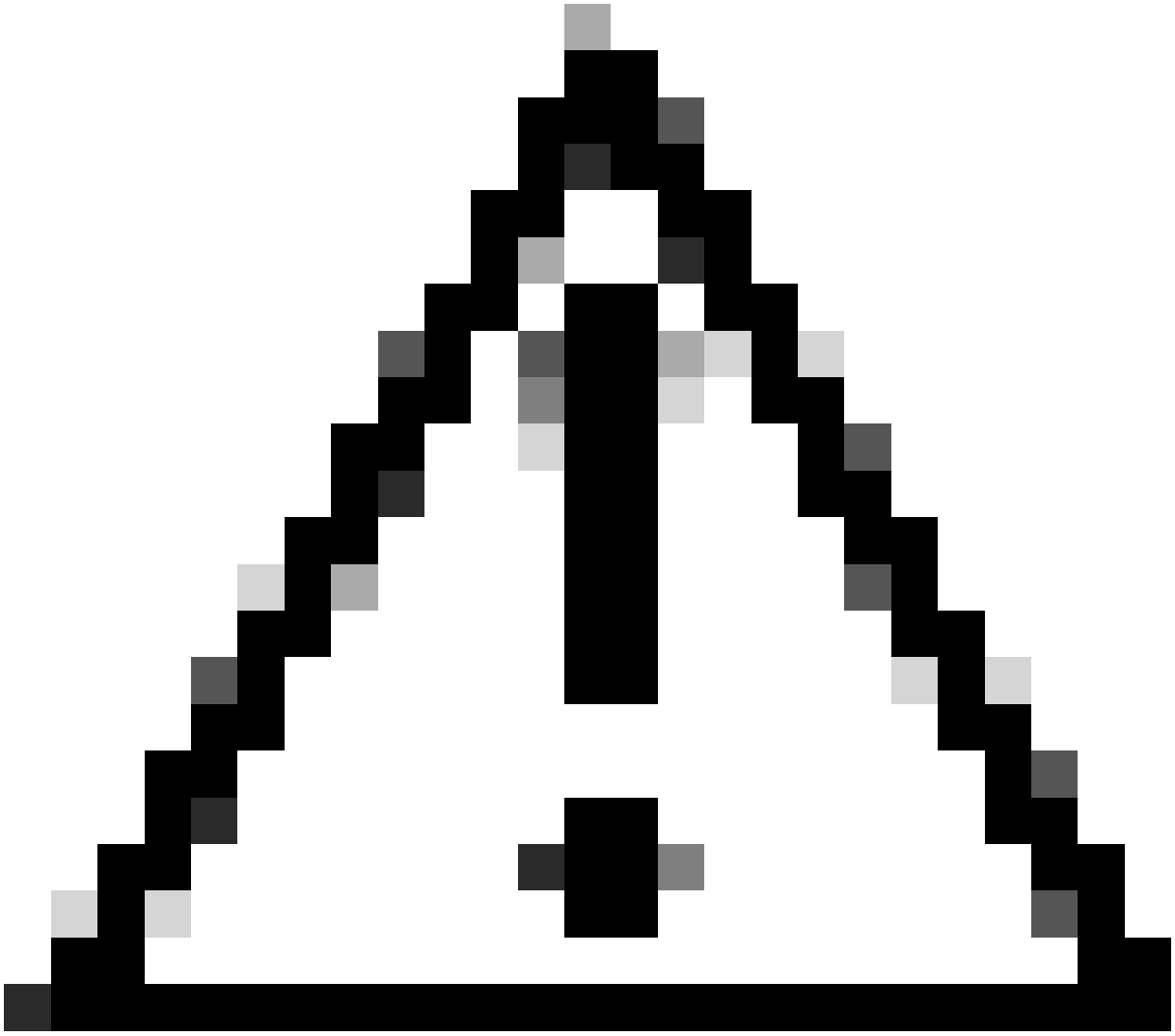
```
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
```

:!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:HIGH:-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

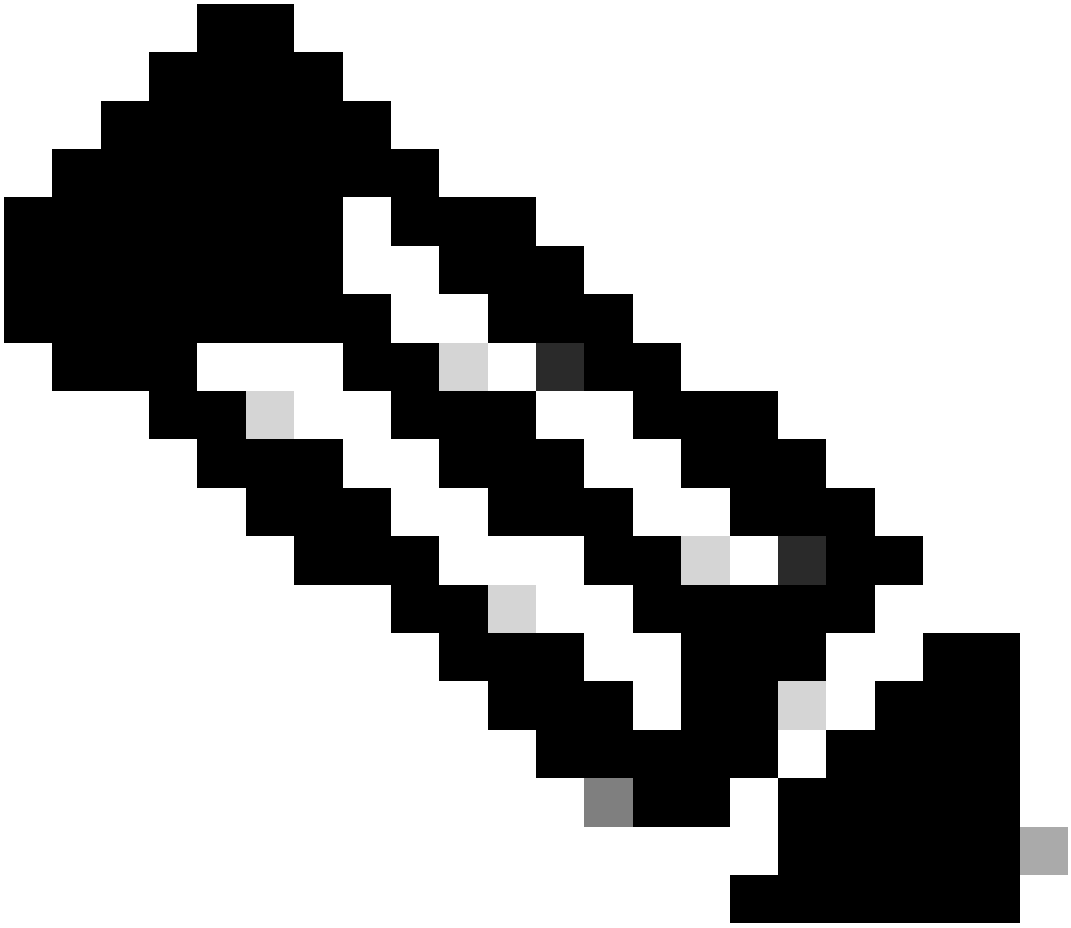
この例では、opensslがインストールされたWindows PCからExpresswayへのTLS接続が試行されます。クライアントであるPCは、DHEアルゴリズムを使用する望ましくないDHE-RSA-AES256-CCM暗号のみをネゴシエートします。

<#root>

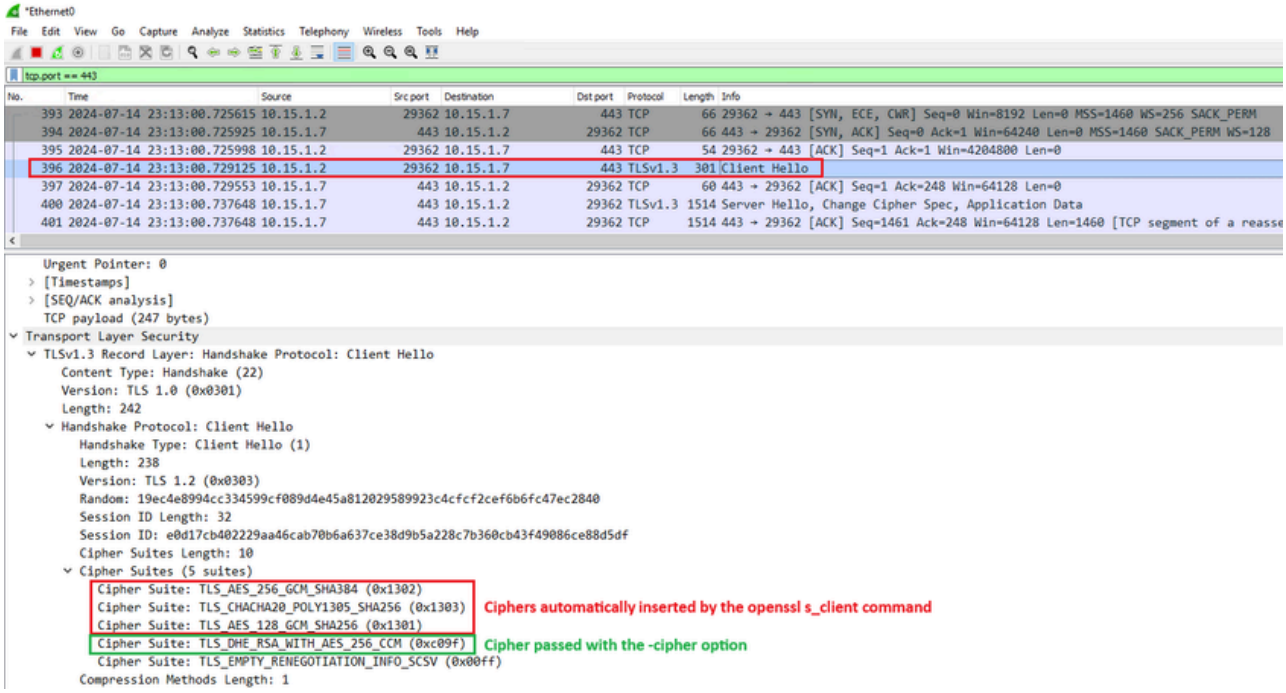
```
C:\Users\Administrator>

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。



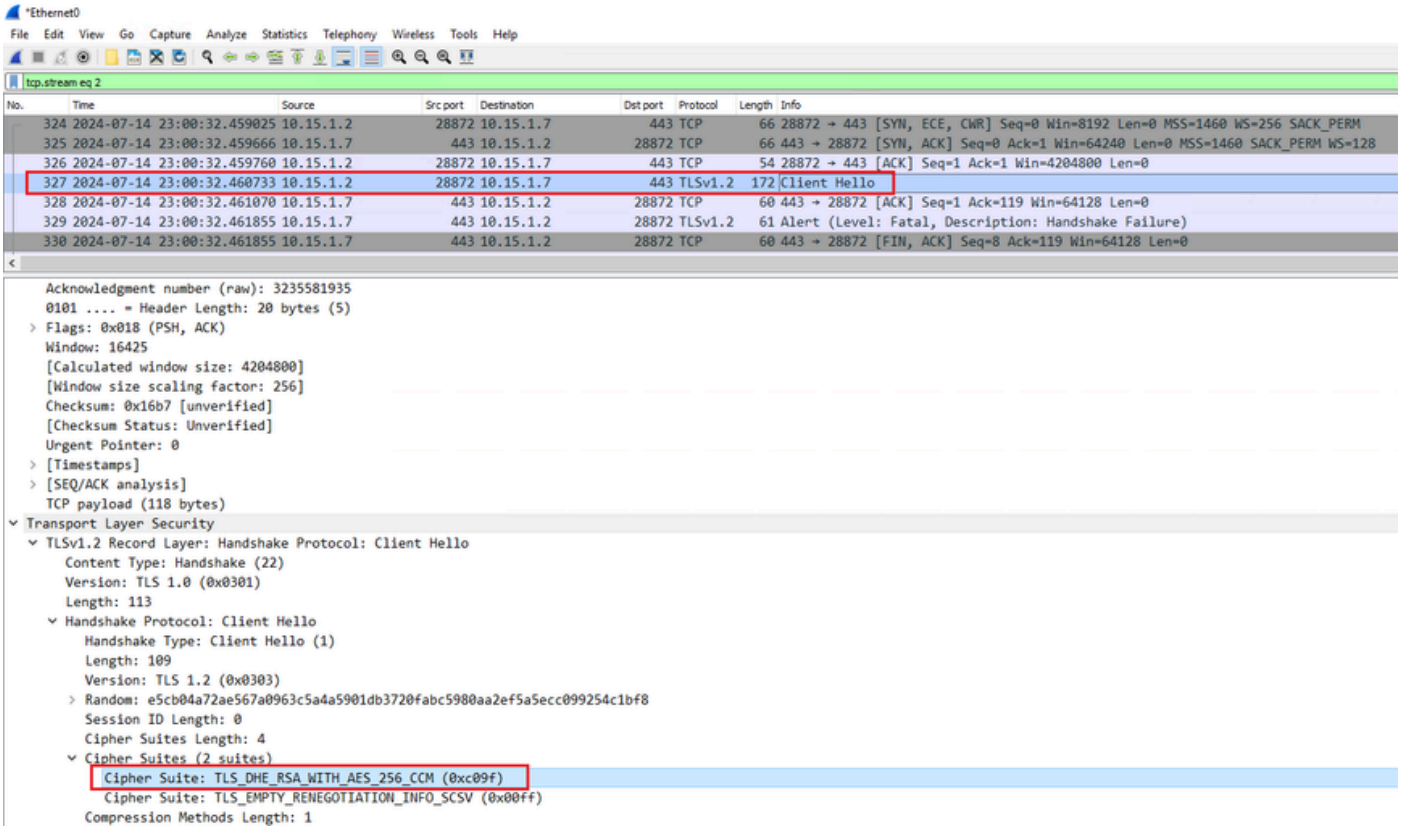
自動的に追加された暗号を含むClientHelloパケット

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。

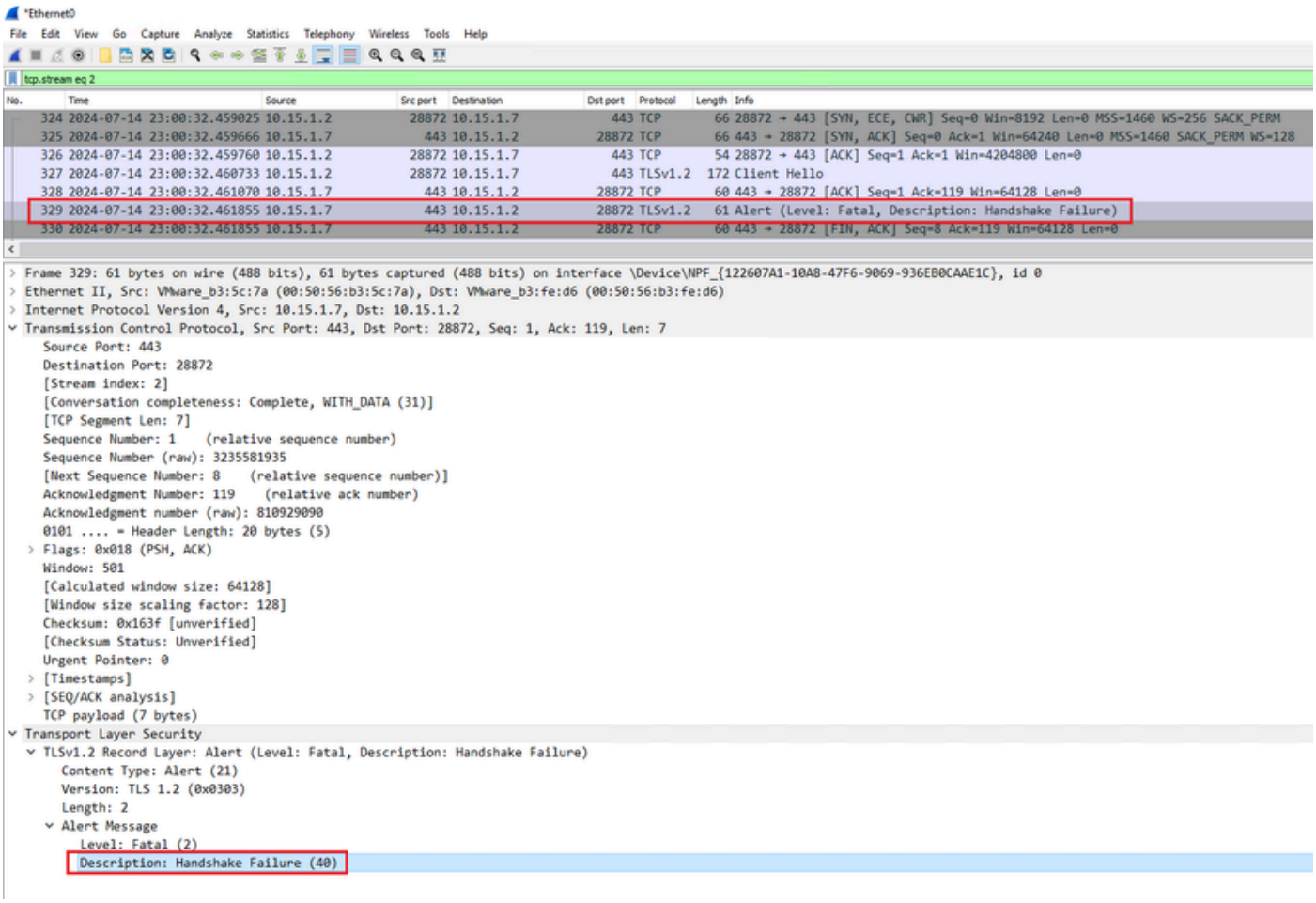
テストデバイスから送信されたClientHelloを見つけます。望ましくないテスト暗号だけをネゴシエートすることを確認します。この例では、DHEアルゴリズムを使用する暗号です。



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。