

Personnaliser la configuration du chiffrement SSL d'Expressway

Table des matières

[Introduction](#)

[Conditions préalables](#)

[Exigences](#)

[Composants utilisés](#)

[Informations générales](#)

[Vérifier la chaîne de chiffrement](#)

[Inspecter la négociation de chiffrement dans la connexion TLS avec une capture de paquets](#)

[Configurer](#)

[Désactiver un chiffrement spécifique](#)

[Désactiver un groupe de chiffrements à l'aide d'un algorithme commun](#)

[Vérifier](#)

[Examinez la liste des chiffrements autorisés par la chaîne de chiffrement](#)

[Tester une connexion TLS en négociant un chiffrement désactivé](#)

[Inspecter une capture de paquet d'un TLSHandshake à l'aide d'un chiffrement désactivé](#)

[Informations connexes](#)

Introduction

Ce document décrit les étapes pour personnaliser les chaînes de chiffrement préconfigurées sur Expressway.

Conditions préalables

Exigences

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Cisco Expressway ou Cisco VCS.
- Protocole TLS.

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Cisco Expressway version X15.0.2.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

Informations générales

La configuration par défaut d'Expressway inclut des chaînes de chiffrement préconfigurées, qui, pour des raisons de compatibilité, permettent la prise en charge de certains chiffrements qui peuvent être considérés comme faibles dans le cadre de certaines politiques de sécurité d'entreprise. Il est possible de personnaliser les chaînes de chiffrement afin de les affiner pour les adapter aux politiques spécifiques de chaque environnement.

Dans Expressway, il est possible de configurer une chaîne de chiffrement indépendante pour chacun de ces protocoles :

- HTTPS
- LDAP
- Mandataire inverse
- SIP
- SMTP
- Provisionnement TMS
- Détection des serveurs UC
- XMPP

Les chaînes de chiffrement obéissent au format OpenSSL décrit dans la page [OpenSSL Chiphers Manpage](#). La version actuelle d'Expressway X15.0.2 est fournie avec la chaîne par défaut ECDH : EDH : HIGH : -AES256+SHA : !MEDIUM : !LOW : !3DES : !MD5 : !PSK : !eNULL : !aNULL : !aDH préconfigurée pour tous les protocoles de la même manière. Sur la page d'administration Web, sous Maintenance > Security > Ciphers, vous pouvez modifier la chaîne de chiffrement attribuée à chaque protocole, afin d'ajouter ou de supprimer des chiffrements ou des groupes de chiffrements spécifiques à l'aide d'un algorithme commun.

Vérifier la chaîne de chiffrement

En utilisant la commande `openssl ciphers -V "<chaîne de chiffrement>"`, vous pouvez sortir une liste avec tous les chiffrements qu'une certaine chaîne autorise, ce qui est utile pour inspecter visuellement les chiffrements. Cet exemple montre le résultat lors de l'inspection de la chaîne de chiffrement par défaut d'Expressway :

```
<#root>
```

```
~ #
```

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

```

Inspecter la négociation de chiffrement dans la connexion TLS avec une capture de paquets

En capturant une négociation TLS dans une capture de paquets, vous pouvez inspecter les détails de la négociation de chiffrement à l'aide de Wireshark.

Le processus d'échange TLS inclut un paquet ClientHello envoyé par le périphérique client, fournissant la liste des chiffrements qu'il prend en charge en fonction de sa chaîne de chiffrement configurée pour le protocole de connexion. Le serveur examine la liste, la compare à sa propre liste de chiffrements autorisés (déterminés par sa propre chaîne de chiffrement) et choisit un chiffrement pris en charge par les deux systèmes, à utiliser pour la session chiffrée. Il répond ensuite avec un paquet ServerHello indiquant le chiffre choisi. Il existe des différences importantes entre les dialogues d'échange TLS 1.2 et 1.3, cependant le mécanisme de négociation de chiffrement utilise le même principe dans les deux versions.

Voici un exemple de négociation de chiffrement TLS 1.3 entre un navigateur Web et Expressway sur le port 443, comme illustré dans Wireshark :

No.	Time	Source	Src port	Destination	Dest port	Protocol	Length	Info
3186	2024-07-14 23:28:55.675909	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

Exemple de connexion TLS dans Wireshark

Tout d'abord, le navigateur envoie un paquet ClientHello avec la liste des chiffrements qu'il prend en charge :

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

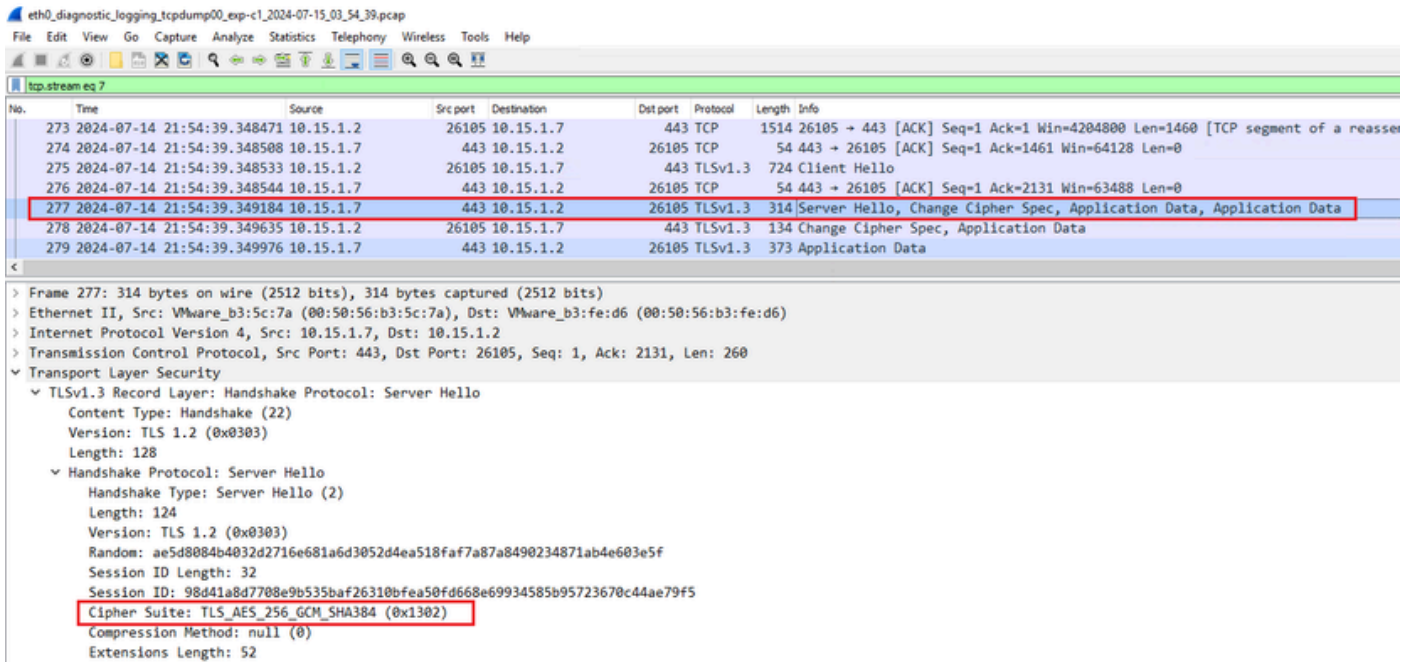
▼ Cipher Suites (16 suites)

- 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

Exemple de paquet Hello client dans Wireshark

Expressway vérifie sa chaîne de chiffrement configurée pour le protocole HTTPS, et trouve un chiffre que lui-même et le client prennent en charge. Dans cet exemple, le chiffre ECDHE-RSA-AES256-GCM-SHA384 est sélectionné. Expressway répond avec son paquet ServerHello indiquant le chiffre sélectionné :



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

Exemple de paquet Hello de serveur dans Wireshark

Configurer

Le format de chaîne de chiffrement OpenSSL comprend plusieurs caractères spéciaux afin d'effectuer des opérations sur la chaîne, telles que la suppression d'un chiffrement spécifique ou d'un groupe de chiffrements partageant un composant commun. Étant donné que ces personnalisations ont généralement pour objectif de supprimer les chiffrements, les caractères utilisés dans ces exemples sont les suivants :

- Caractère -, utilisé pour supprimer les chiffres de la liste. Certains ou tous les chiffrements supprimés peuvent être autorisés à nouveau par des options apparaissant plus loin dans la chaîne.
- Le caractère !, également utilisé pour supprimer les chiffres de la liste. Lors de son utilisation, les chiffrements supprimés ne peuvent pas être autorisés à nouveau par d'autres options apparaissant plus tard dans la chaîne.
- Le caractère :, qui sert de séparateur entre les éléments de la liste.

Les deux peuvent être utilisés pour supprimer un chiffre de la chaîne, cependant ! est préférable. Pour obtenir la liste complète des caractères spéciaux, consultez la page [OpenSSL Chiphers Manpage](#).



Remarque : le site OpenSSL indique que lorsque vous utilisez le caractère !, "les chiffrements supprimés ne peuvent jamais réapparaître dans la liste même s'ils sont explicitement indiqués". Cela ne signifie pas que les chiffrements sont supprimés définitivement du système, il se réfère à la portée de l'interprétation de la chaîne de chiffrement.

Désactiver un chiffrement spécifique

Afin de désactiver un chiffre spécifique, ajoutez à la chaîne par défaut le séparateur :, le signe ! ou- et le nom du chiffre à désactiver. Le nom de chiffrement doit obéir au format d'attribution de noms OpenSSL, disponible dans la page [OpenSSL Ciphers Manpage](#). Par exemple, si vous devez désactiver le chiffrement AES128-SHA pour les connexions SIP, configurez une chaîne de chiffrement comme suit :

```
<#root>
```

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

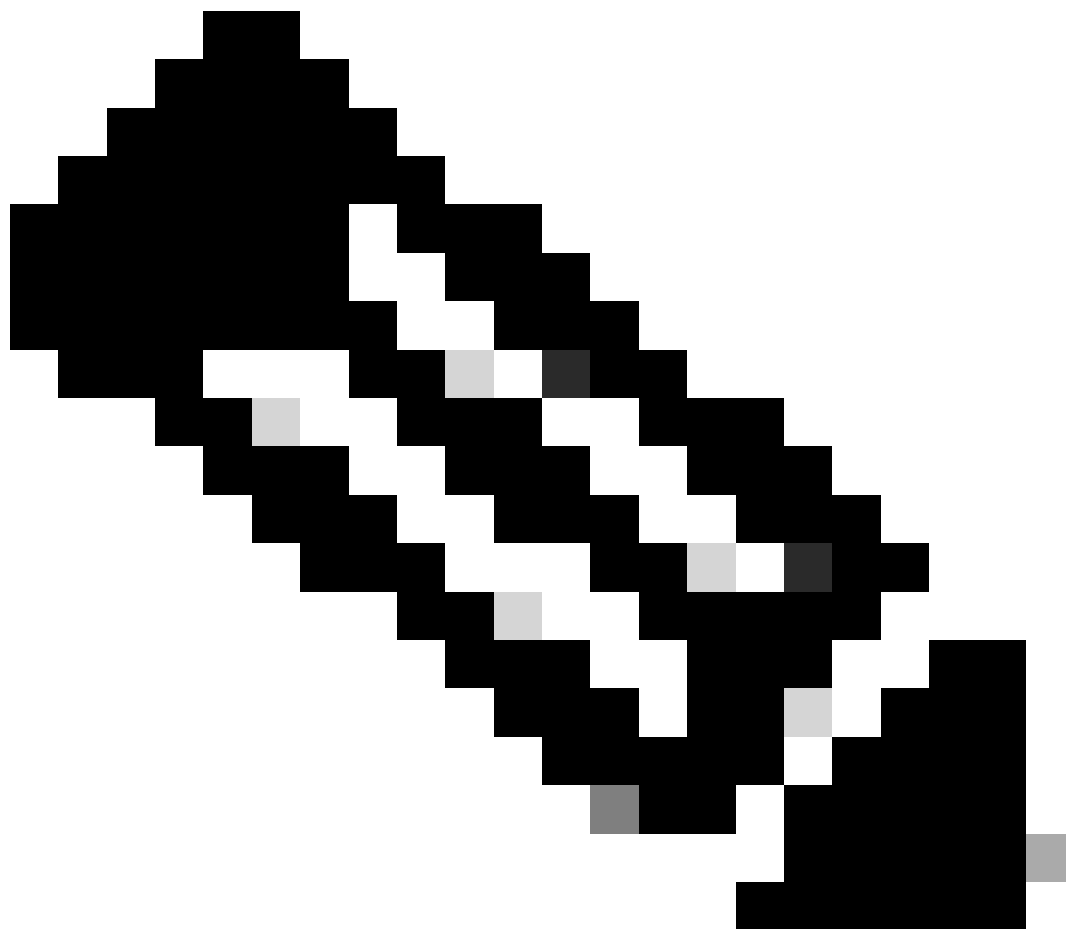
: !AES128-SHA

Ensuite, accédez à la page Web Admin d'Expressway, accédez à Maintenance > Sécurité > Chiffres, attribuez la chaîne personnalisée au(x) protocole(s) requis, puis cliquez sur Enregistrer. Pour que la nouvelle configuration soit appliquée, un redémarrage du système est nécessaire. Dans cet exemple, la chaîne personnalisée est attribuée au protocole SIP sous Chiffres SIP TLS :

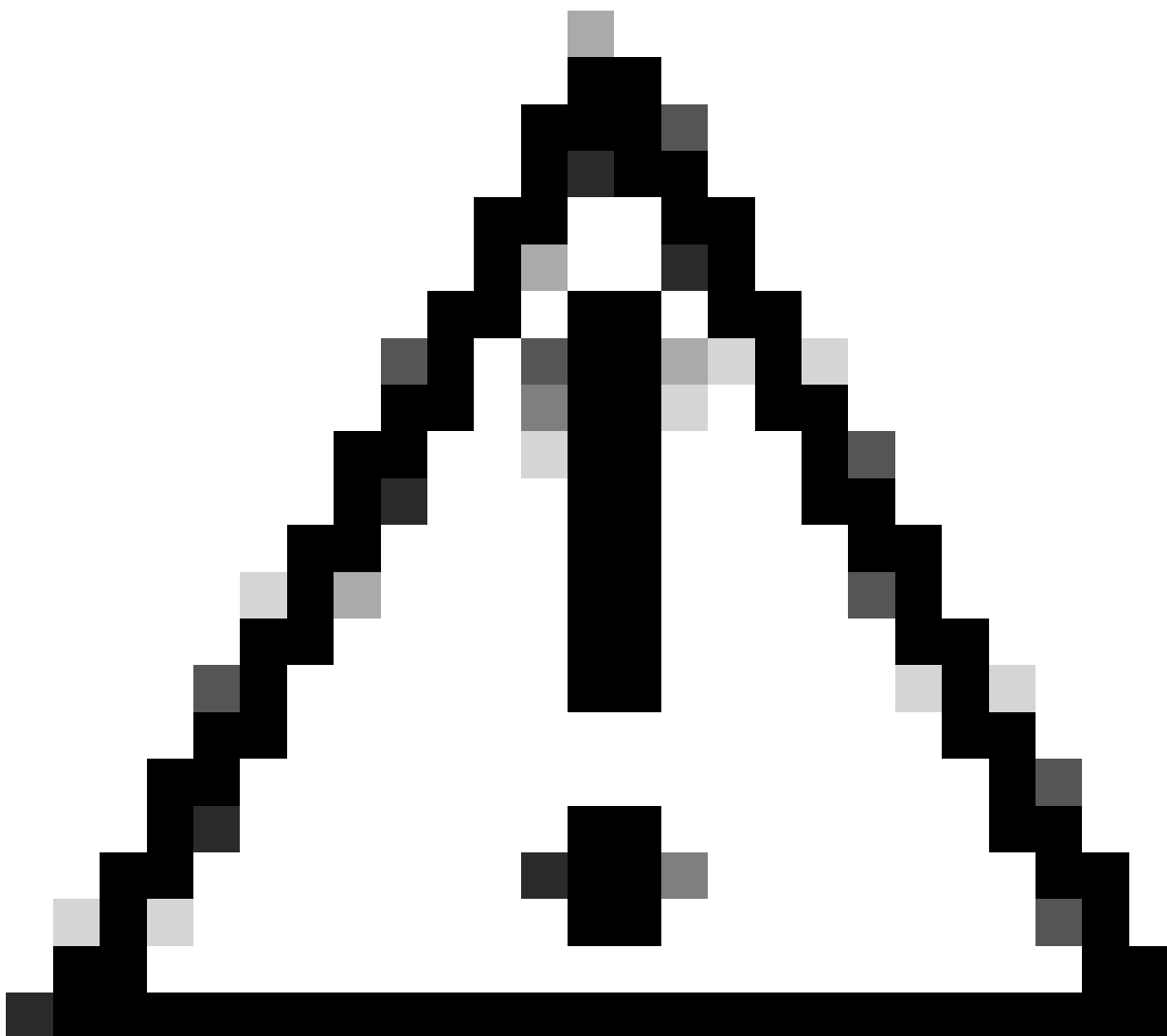
The screenshot shows the 'Ciphers' configuration page in the Expressway Web Admin interface. The page has a navigation bar at the top with 'Maintenance' selected. Below the navigation bar, the 'Ciphers' section is active, and a 'Configuration' tab is selected. The configuration is presented as a list of settings, each with a text input field and an information icon. The 'SIP TLS ciphers' setting is highlighted with a red box and contains the value '!AES128-SHA'. At the bottom left of the configuration area, there is a 'Save' button, also highlighted with a red box.

Setting	Value
HTTPS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
HTTPS minimum TLS version	TLS v1.2
LDAP TLS Ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
LDAP minimum TLS version	TLS v1.2
Reverse proxy TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
Reverse proxy minimum TLS version	TLS v1.2
SIP TLS ciphers	!AES128-SHA
SIP minimum TLS version	TLS v1.2
SMTP TLS Ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
SMTP minimum TLS version	TLS v1.2
TMS Provisioning Ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
TMS Provisioning minimum TLS version	TLS v1.2
UC server discovery TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
UC server discovery minimum TLS version	TLS v1.2
XMPP TLS ciphers	EECDH:EDH:HIGH:-AES256+SHA:IMEDIUM:LOW:3DES:1MD5:IPSK:h
XMPP minimum TLS version	TLS v1.2

Page des paramètres de chiffrement sur le portail d'administration Web Expressway



Remarque : dans le cas d'un cluster Expressway, effectuez les modifications sur le serveur principal uniquement. La nouvelle configuration est répliquée sur les autres membres du cluster.



Attention : utilisez la séquence de redémarrage de cluster recommandée fournie dans le [Guide de déploiement de création et de maintenance de cluster Cisco Expressway](#). Commencez par redémarrer le serveur principal, attendez qu'il soit accessible via l'interface Web, puis faites de même avec chaque homologue dans l'ordre selon la liste configurée sous Système > Clustering.

Désactiver un groupe de chiffrements à l'aide d'un algorithme commun

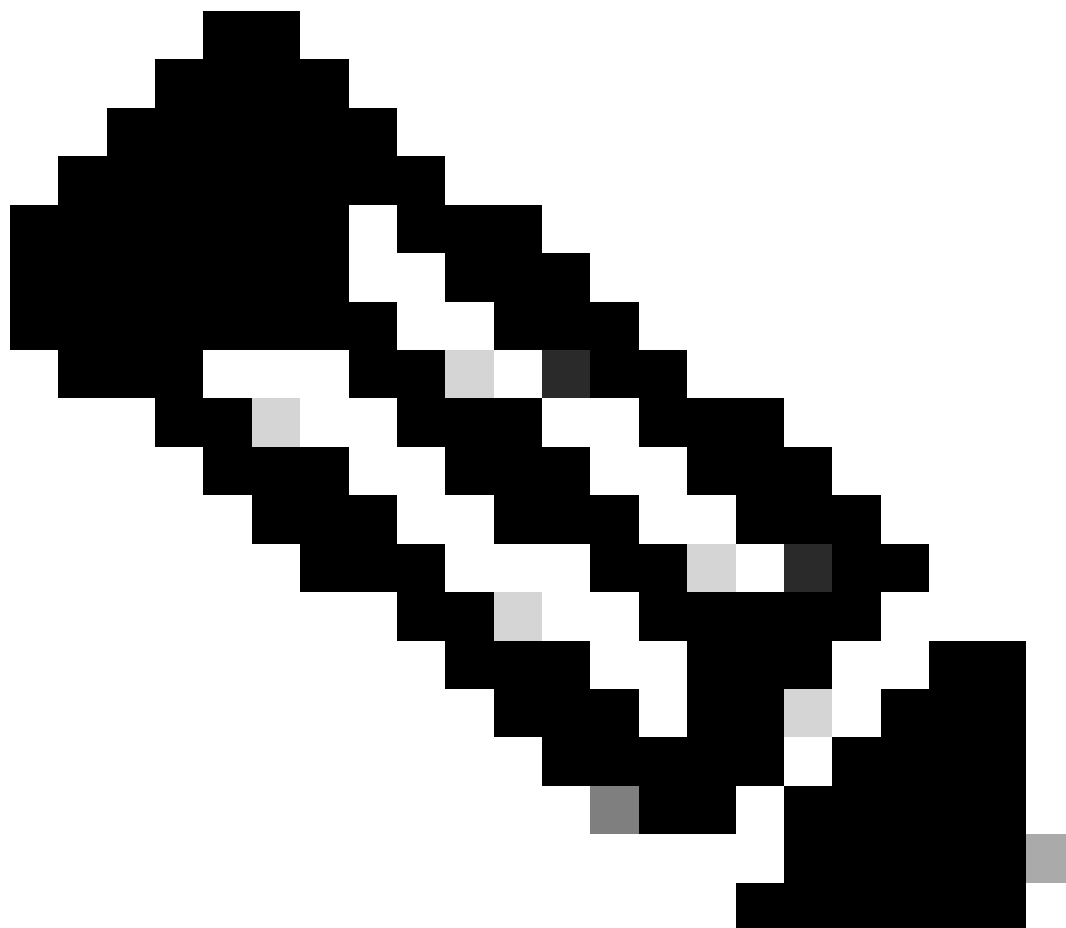
Afin de désactiver un groupe de chiffrements à l'aide d'un algorithme commun, ajoutez à la chaîne par défaut le séparateur :, le signe ! ou - et le nom de l'algorithme à désactiver. Les noms d'algorithme pris en charge sont disponibles dans la page [OpenSSL Ciphers Manpage](#). Par exemple, si vous devez désactiver tous les chiffrements qui utilisent l'algorithme DHE, configurez une chaîne de chiffrement comme ceci :

```
<#root>
```

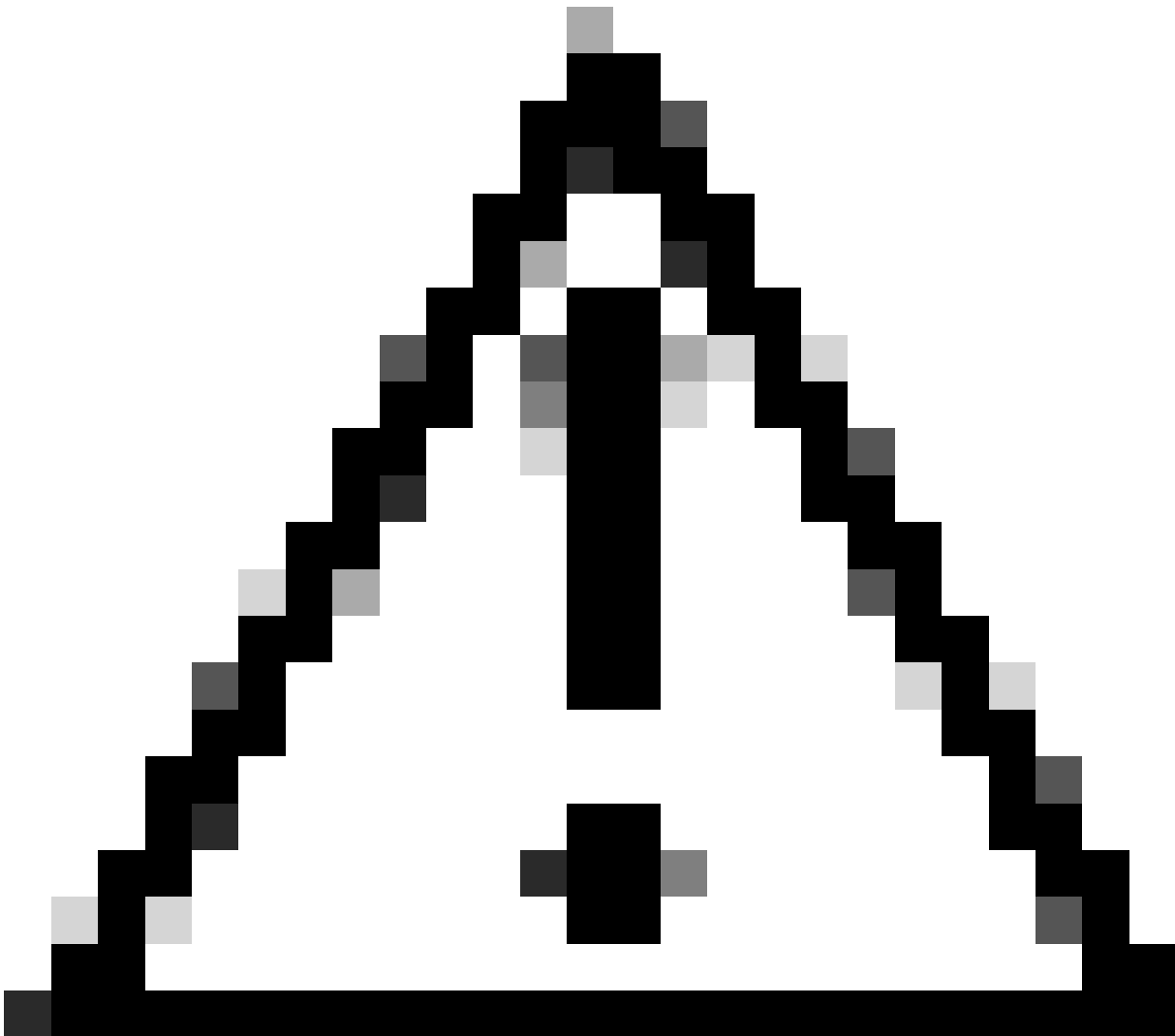
```
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH
```

: !DHE

Accédez à la page Expressway web admin, accédez à Maintenance > Security > Ciphers, attribuez la chaîne personnalisée au(x) protocole(s) requis, et cliquez sur Save. Pour que la nouvelle configuration soit appliquée, un redémarrage du système est nécessaire.



Remarque : dans le cas d'un cluster Expressway, effectuez les modifications sur le serveur principal uniquement. La nouvelle configuration est répliquée sur les autres membres du cluster.



Attention : utilisez la séquence de redémarrage de cluster recommandée fournie dans le [Guide de déploiement de création et de maintenance de cluster Cisco Expressway](#). Commencez par redémarrer le serveur principal, attendez qu'il soit accessible via l'interface Web, puis faites de même avec chaque homologue dans l'ordre selon la liste configurée sous Système > Clustering.

Vérifier

Examinez la liste des chiffrements autorisés par la chaîne de chiffrement

Vous pouvez inspecter la chaîne de chiffrement personnalisée en utilisant la commande `openssl ciphers -V "<chaîne de chiffrement>"`. Vérifiez le résultat afin de confirmer que les chiffrements indésirables ne sont plus répertoriés après les modifications. Dans cet exemple, la chaîne de chiffrement `EECDH:EDH:HIGH:-`

`AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHE` est inspectée. Le résultat de la commande confirme que la chaîne n'autorise aucun des chiffrements qui utilisent

l'algorithme DHE :

```
<#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
~ #
```

Tester une connexion TLS en négociant un chiffrement désactivé

Vous pouvez utiliser la commande `openssl s_client` afin de vérifier qu'une tentative de connexion utilisant un chiffrement désactivé est rejetée. Utilisez l'option `-connect` pour spécifier votre adresse et votre port Expressway, et utilisez l'option `-cipher` pour spécifier le chiffrement unique à négocier par le client pendant la connexion TLS :

```
openssl s_client -connect <adresse>:<port> -cipher <chiffre> -no_tls1_3
```

Dans cet exemple, une connexion TLS vers Expressway est tentée à partir d'un PC Windows sur lequel `openssl` est installé. Le PC, en tant que client, négocie uniquement le chiffrement DHE-RSA-AES256-CCM indésirable, qui utilise l'algorithme DHE :

```
<#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>
```

La sortie de la commande montre que la tentative de connexion échoue avec un message d'erreur « ssl/tls alert handshake failure:..\ssl\record\rec_layer_s3.c:865:SSL alert number 40 », car l'Expressway est configuré pour utiliser la chaîne de chiffrement EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH:!DHE pour les connexions HTTPS, ce qui désactive les chiffrements qui utilisent l'algorithme DHE.



Remarque : pour que les tests avec la commande `openssl s_client` fonctionnent comme expliqué, l'option `-no_tls1_3` doit être passée à la commande. S'il n'est pas inclus, le client insère automatiquement des chiffrements TLS 1.3 dans le paquet ClientHello :

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 (0xc09f)
 - 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

Paquet ClientHello avec chiffrement ajouté automatiquement

Si l'Expressway cible prend en charge ces chiffrements, l'un d'eux peut être choisi au lieu du chiffre spécifique que vous devez tester. La connexion a réussi, ce qui peut vous faire croire qu'une connexion a été possible en utilisant le chiffrement désactivé passé à la commande avec l'option -cipher.

Inspecter une capture de paquet d'une connexion TLS à l'aide d'un chiffrement désactivé

Vous pouvez collecter une capture de paquets, à partir du périphérique de test ou de l'Expressway, tout en effectuant un test de connexion à l'aide de l'un des chiffrements désactivés. Vous pouvez ensuite l'inspecter avec Wireshark afin d'analyser plus en détail les événements de connexion.

Recherchez ClientHello envoyé par le périphérique de test. Confirmez qu'il négocie uniquement le chiffrement test indésirable, dans cet exemple un chiffrement utilisant l'algorithme DHE :

The image shows a Wireshark capture of a network packet. The packet list pane at the top shows a series of packets. Packet 327 is highlighted in blue and has a red box around it. It is a TLSv1.2 Client Hello packet from source 10.15.1.2 to destination 10.15.1.7. The packet details pane below shows the structure of the packet. It is a Transport Layer Security (TLS) record of type Handshake Protocol: Client Hello. The details include the Handshake Type: Client Hello (1), Version: TLS 1.2 (0x0303), and a list of Cipher Suites. The first cipher suite is TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030), which is highlighted with a red box. The second cipher suite is TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff). The packet also includes a Random value, Session ID Length: 0, and Compression Methods Length: 1.

Exemple de paquet Hello client dans Wireshark

:

Confirmez qu'Expressway répond par un paquet d'alerte TLS fatal en refusant la connexion. Dans cet exemple, comme Expressway ne prend pas en charge les chiffrements DHE par sa chaîne de chiffrement configurée pour le protocole HTTPS, il répond avec un paquet d'alerte TLS fatal contenant le code d'échec 40.

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 shows the structure of this packet, including a fatal TLS alert.

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)

Un paquet d'alerte TLS fatal dans Wireshark

Informations connexes

- [Page de manuel OpenSSL Ciphers](#)
- [Guide de l'administrateur de Cisco Expressway \(X15.0\) - Chapitre : Gestion de la sécurité - Configuration de la version TLS minimale et des suites de chiffrement](#)

À propos de cette traduction

Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.