自定义Expressway SSL密码配置

目录

简介

本文档介绍在Expressway上自定义预配置密码字符串的步骤。

先决条件

要求

Cisco 建议您了解以下主题：

- Cisco Expressway或Cisco VCS。
- TLS协议。

使用的组件

本文档中的信息基于以下软件和硬件版本：

- Cisco Expressway版本X15.0.2。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

默认Expressway配置包括预配置的密码字符串，出于兼容性的原因，这些字符串支持某些在某些企业安全策略下可能被视为薄弱的密码。可以对密码字符串进行自定义，以便根据每个环境的特定策略对其进行微调。

在Expressway中，可以为以下每种协议配置独立的密码字符串：

- HTTPS
- LDAP
- 反向代理
- SIP
- SMTP
- TMS调配
- UC服务器发现
- XMPP

密码字符串遵守OpenSSL Ciphers Manpage中介绍的OpenSSL格式。当前Expressway版本X15.0.2随附默认字符串EECDH：EDH：HIGH：-AES256+SHA：！MEDIUM：！LOW：！3DES：！MD5：！PSK：！eNULL：！aNULL：！aDH，该字符串对所有协议都进行了同等程度的预配置。在Web管理页面的维护 > 安全 > 密码下，您可以修改分配给每个协议的密码字符串，以使用通用算法添加或删除特定密码或密码组。

检查密码字符串

通过使用openssl ciphers -V“<cipher string>”命令，您可以输出包含特定字符串允许的所有密码的列表，这对于视觉检查密码非常有用。此示例显示检查默认Expressway密码字符串时的输出：

<#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
```

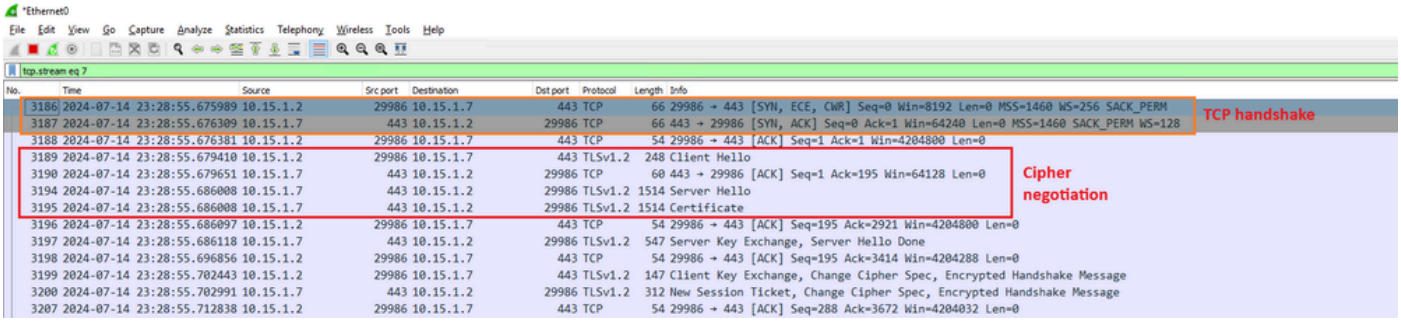
```
0x00,0xA2 - DHE-DSS-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AESGCM(128) Mac=AEAD
0x00,0x9E - DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(128) Mac=AEAD
0xC0,0x9E - DHE-RSA-AES128-CCM TLSv1.2 Kx=DH Au=RSA Enc=AESCCM(128) Mac=AEAD
0x00,0x6B - DHE-RSA-AES256-SHA256 TLSv1.2 Kx=DH Au=RSA 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
~ #
```

使用数据包捕获检查TLS握手中的密码协商

通过在数据包捕获中捕获TLS协商，您可以使用Wireshark检查密码协商的详细信息。

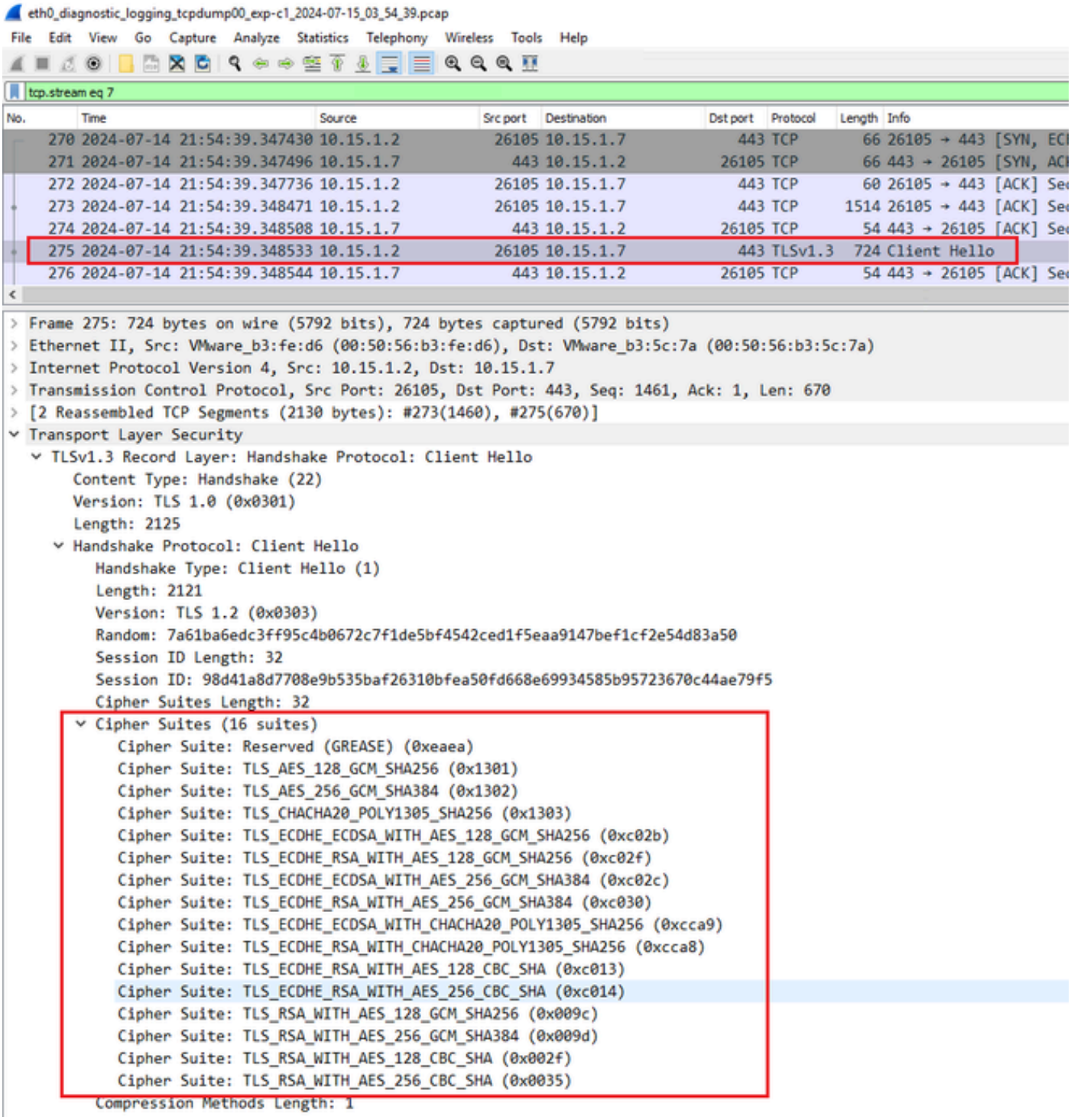
TLS握手过程包括由客户端设备发送的ClientHello数据包，根据为连接协议配置的密码字符串提供其支持的密码列表。服务器审核该列表，将其与自己的允许密码列表（由其自己的密码字符串确定）进行比较，并选择两个系统都支持的密码，以用于加密会话。然后，它以指示所选密码的ServerHello数据包做出响应。TLS 1.2和1.3握手对话之间存在重要区别，但是密码协商机制在两个版本中均使用相同的原理。

以下是Web浏览器与端口443上的Expressway之间的TLS 1.3密码协商示例（如Wireshark所示）：



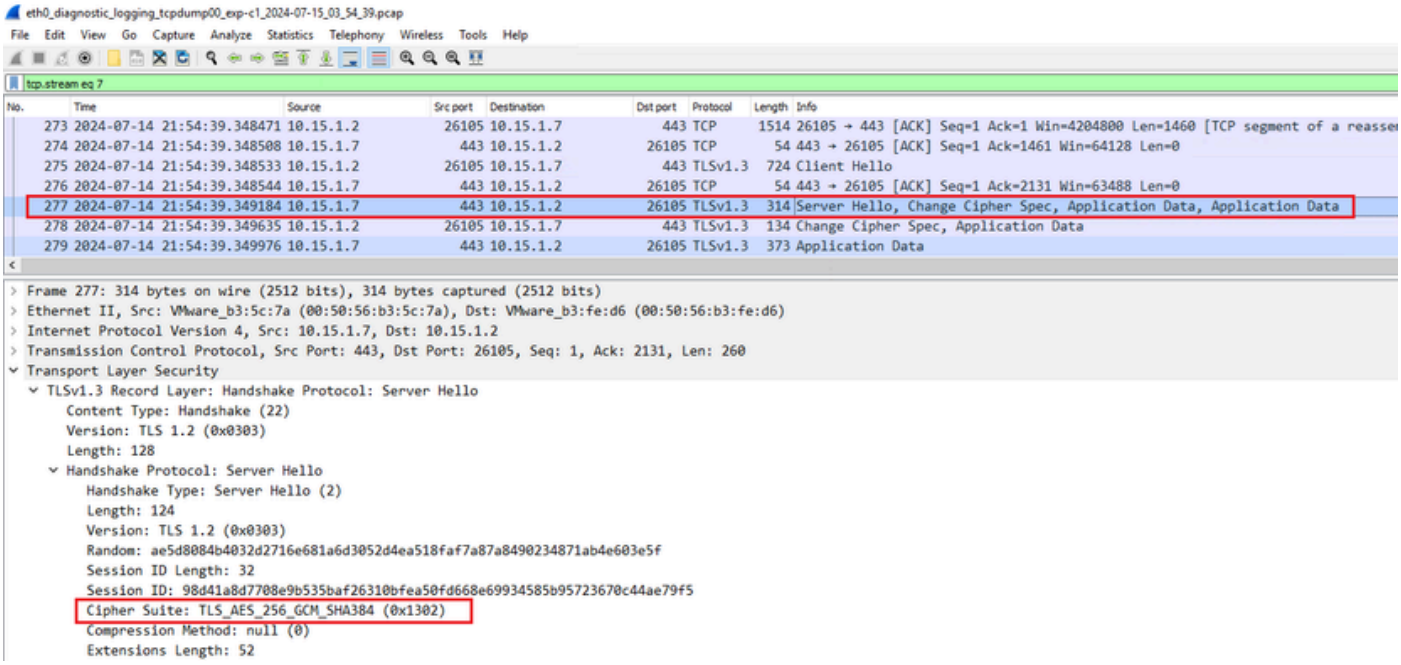
Wireshark中的TLS握手示例

首先，浏览器发送一个带有其支持的密码列表的ClientHello数据包：



Wireshark中的ClientHello数据包示例

Expressway会检查其为HTTPS协议配置的密码字符串，并找到自身和客户端都支持的密码。在本示例中，选择ECDHE-RSA-AES256-GCM-SHA384密码。Expressway使用其ServerHello数据包做出响应，其中指示所选的密码：



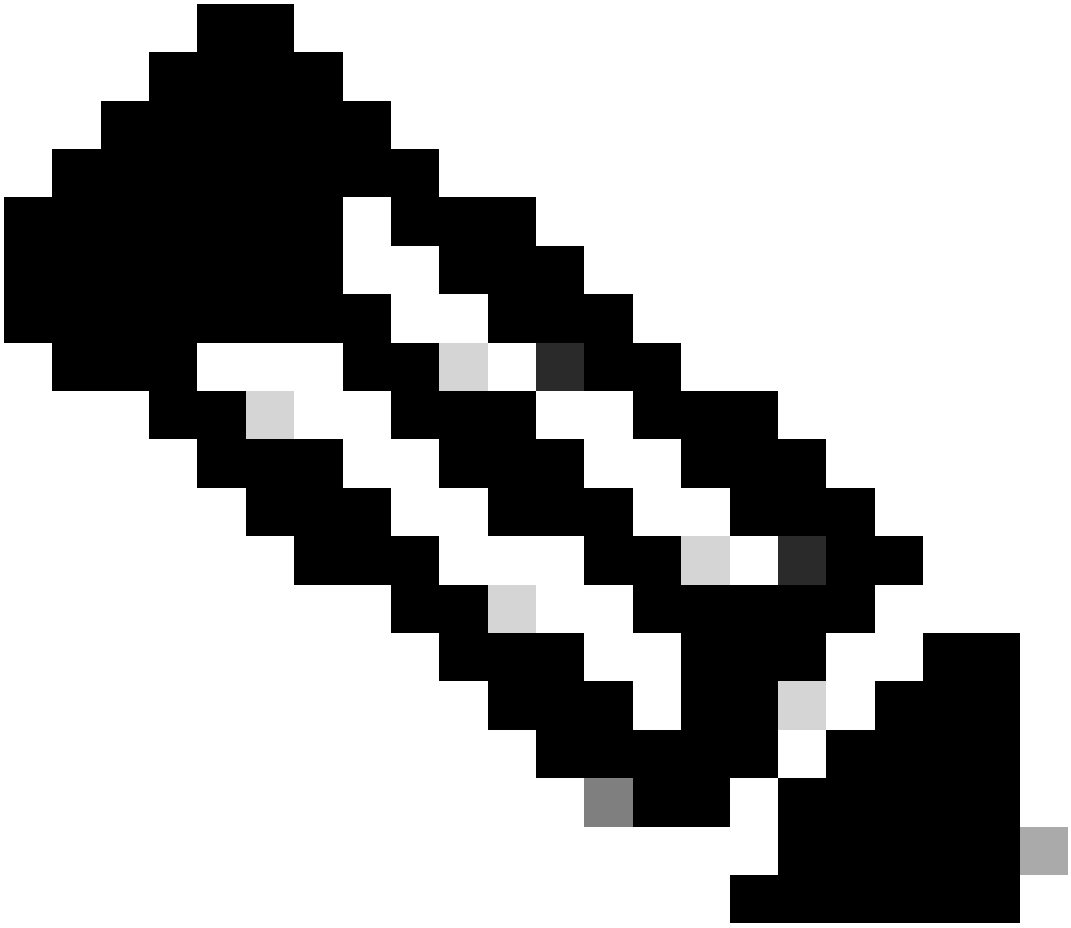
Wireshark中的ServerHello数据包示例

配置

OpenSSL密码字符串格式包括几个特殊字符，以便对字符串执行操作，例如删除特定密码或共享公共组件的密码组。由于这些自定义的目的通常是删除密码，因此这些示例中使用的字符包括：

- -字符，用于从列表中删除密码。通过字符串中稍后出现的选项，可以再次允许部分或全部删除的密码。
- ！字符，也用于从列表中删除密码。使用它时，字符串中稍后出现的任何其他选项均不允许删除的密码。
- ：字符，充当列表中各项之间的分隔符。

两者都可用于从字符串中删除密码，但是！是首选。有关特殊字符的完整列表，请查看OpenSSL Ciphers Manpage。



注意：OpenSSL站点声明，使用！字符时，“删除的密码即使明确声明，也绝不会重新出现在列表中”。这并不意味着密码从系统中永久删除，而是指对密码字符串的解释范围。

禁用特定密码

要禁用特定密码，请在默认字符串后追加要禁用的分隔符：、 ！或-符号和密码名称。密码名称必须遵循OpenSSL命名格式，在OpenSSL Ciphers Manpage中提供了此格式。例如，如果需要禁用SIP连接的AES128-SHA密码，请配置如下所示的密码字符串：

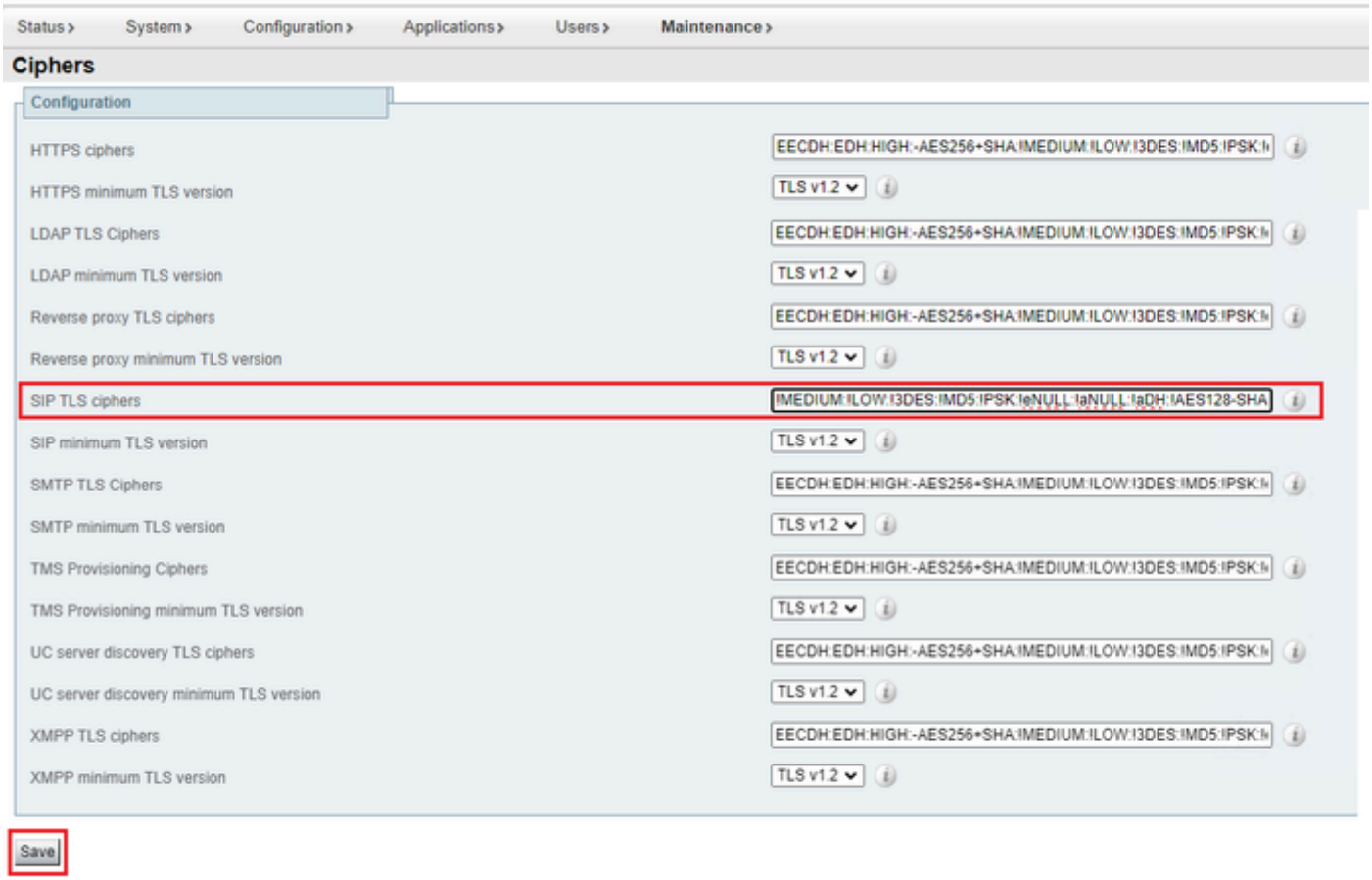
<#root>

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

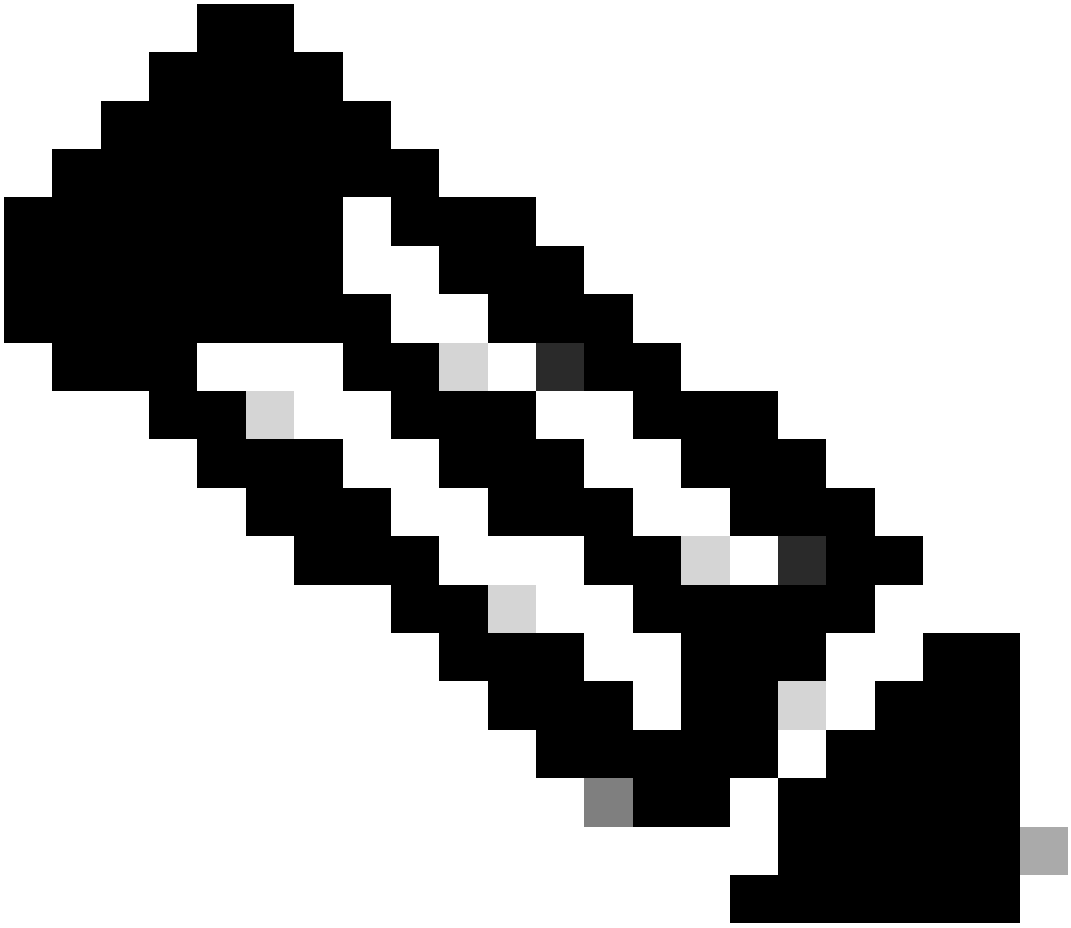
:!AES128-SHA

然后，导航到Expressway Web管理页面，导航到维护>安全>密码，将自定义字符串分配到所需协

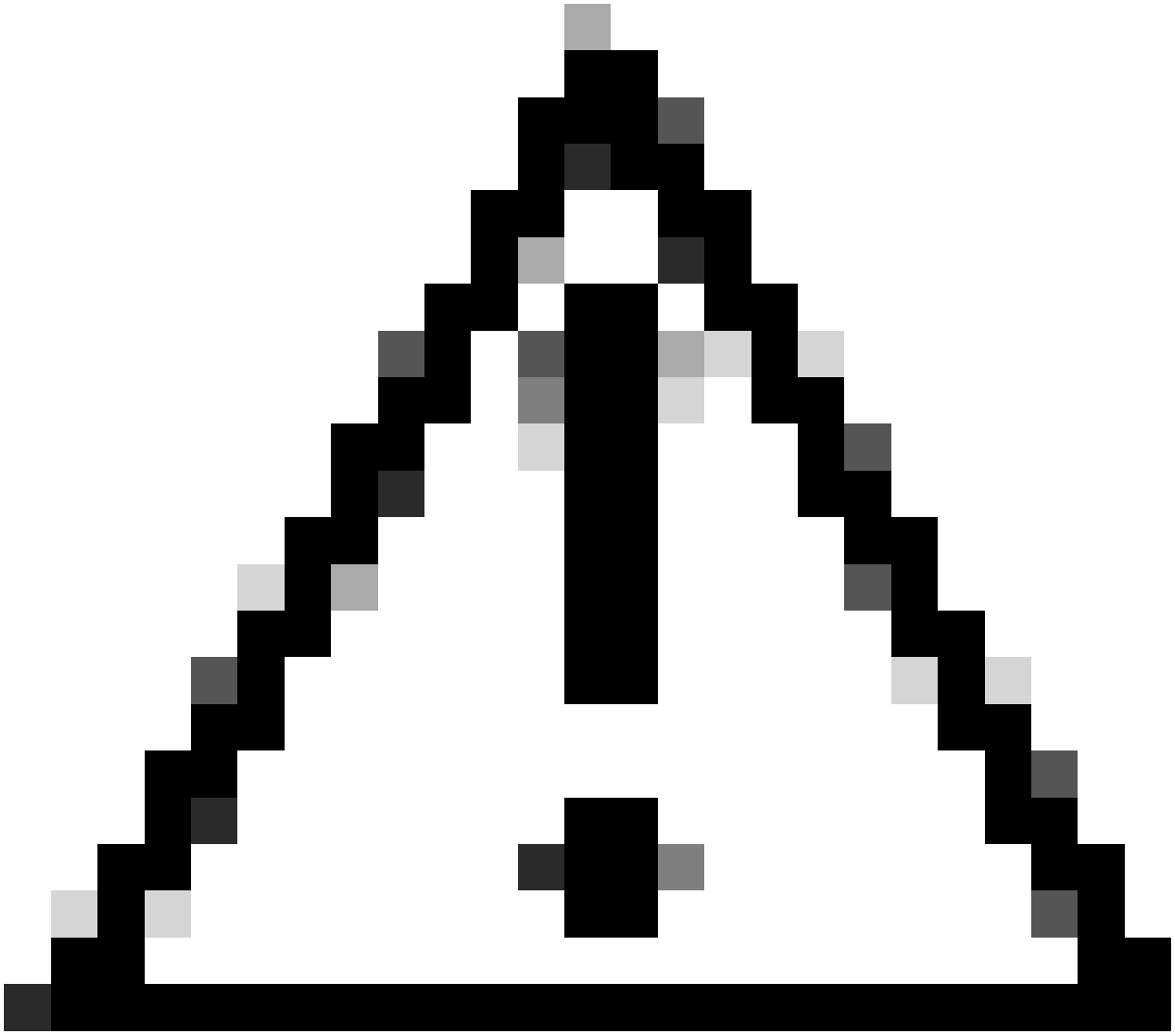
议，然后点击保存。 要应用新配置，需要重新启动系统。在本示例中，在SIP TLS密码下将自定义字符串分配给SIP协议：



Expressway Web管理员门户上的密码设置页面



注意：如果是Expressway集群，请仅在主服务器上进行更改。新配置将复制到其余集群成员。



注意：使用Cisco Expressway集群创建和维护部署指南中提供的推荐集群重新引导顺序。首先重新启动主服务器，等待其可通过Web界面访问，然后根据System > Clustering下配置的列表对每台对等体执行相同的操作。

使用常用算法禁用一组密码

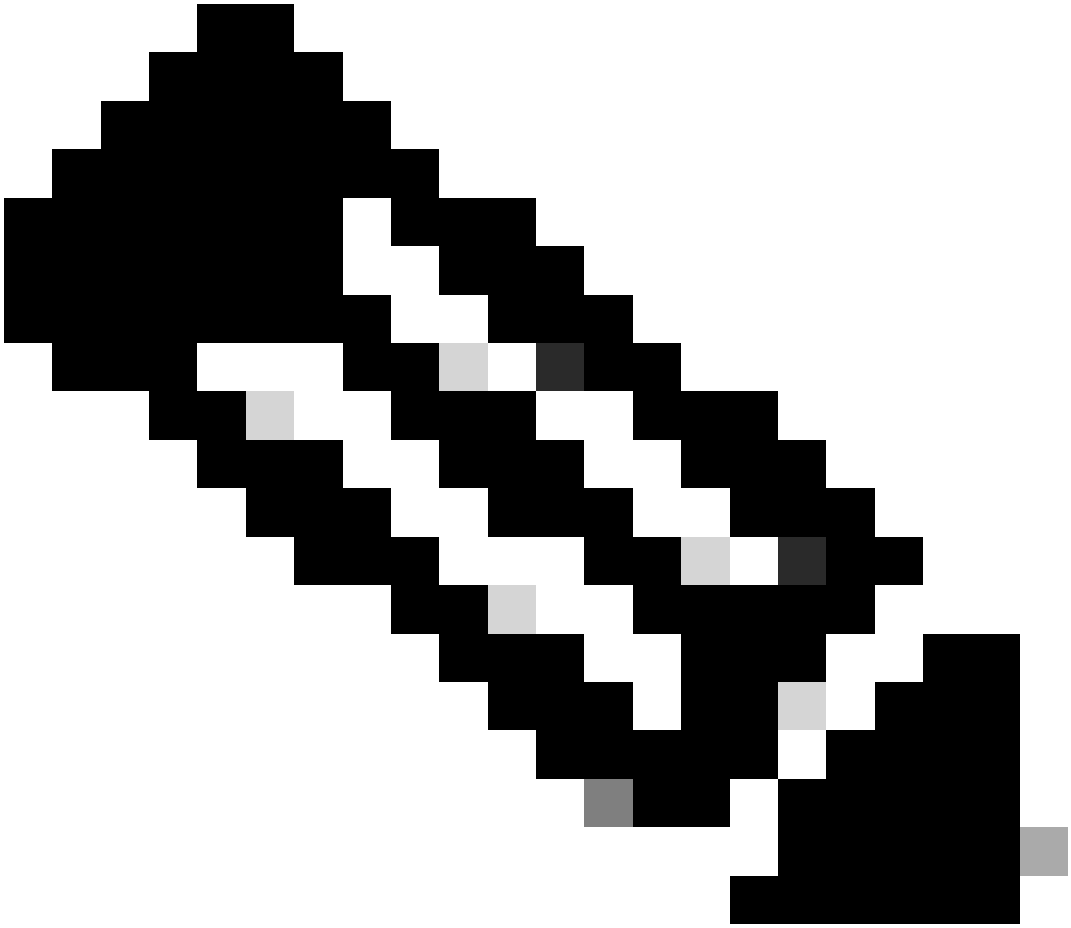
要使用常用算法禁用一组密码，请在默认字符串后附加要禁用的分隔符：、！或-符号和算法名称。在OpenSSL Ciphers Manpage中可以找到支持的算法名称。例如，如果需要禁用使用DHE算法的所有密码，请配置如下所示的密码字符串：

<#root>

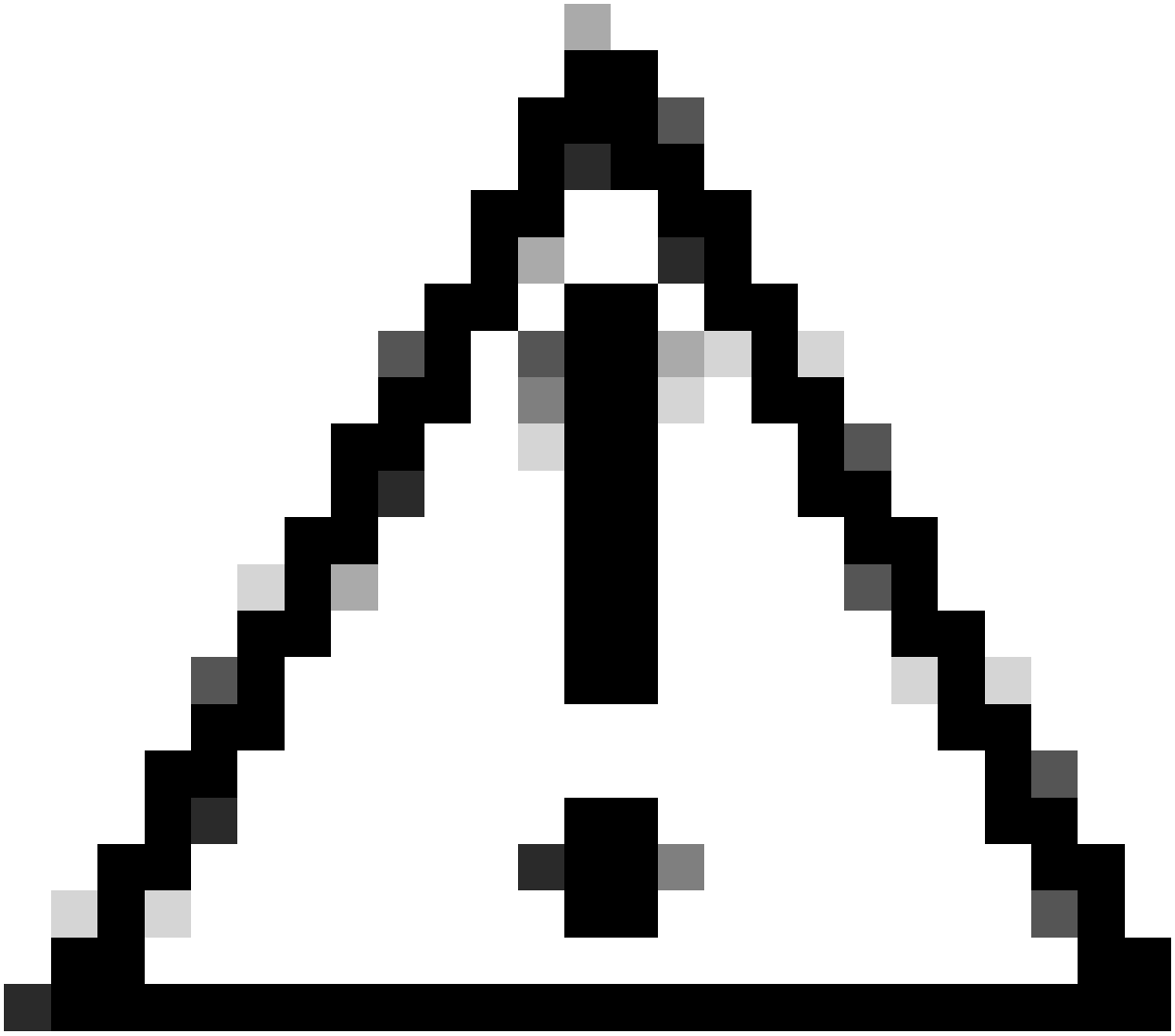
EECDH:EDH:HIGH:-AES256+SHA:!MEDIUM:!LOW:!3DES:!MD5:!PSK:!eNULL:!aNULL:!aDH

:!DHE

导航到Expressway Web管理页面，导航到维护>安全>密码，将自定义字符串分配到所需协议，然后点击保存。 要应用新配置，需要重新启动系统。



注意：如果是Expressway集群，请仅在主服务器上进行更改。新配置将复制到其余集群成员。



注意：使用Cisco Expressway集群创建和维护部署指南中提供的推荐集群重新引导顺序。首先重新启动主服务器，等待其可通过Web界面访问，然后根据System > Clustering下配置的列表对每台对等体执行相同的操作。

验证

检查密码字符串允许的密码列表

您可以使用openssl ciphers -V“<cipher string>”命令检查自定义的密码字符串。查看输出以确认更改后不再列出不需要的密码。在本示例中，检查EECDH：EDH：HIGH：-AES256+SHA：！MEDIUM：！LOW：！3DES：！MD5：！PSK：！eNULL：！aDH：！DHE密码字符串。命令输出确认该字符串不允许使用DHE算法的任何密码：

<#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
~ #
```

通过协商已禁用的密码测试TLS连接

您可以使用openssl s_client命令来验证是否已拒绝使用禁用的口令进行连接尝试。使用-connect选项指定您的Expressway地址和端口，并使用-cipher选项指定客户端在TLS握手期间要协商的单个密码：

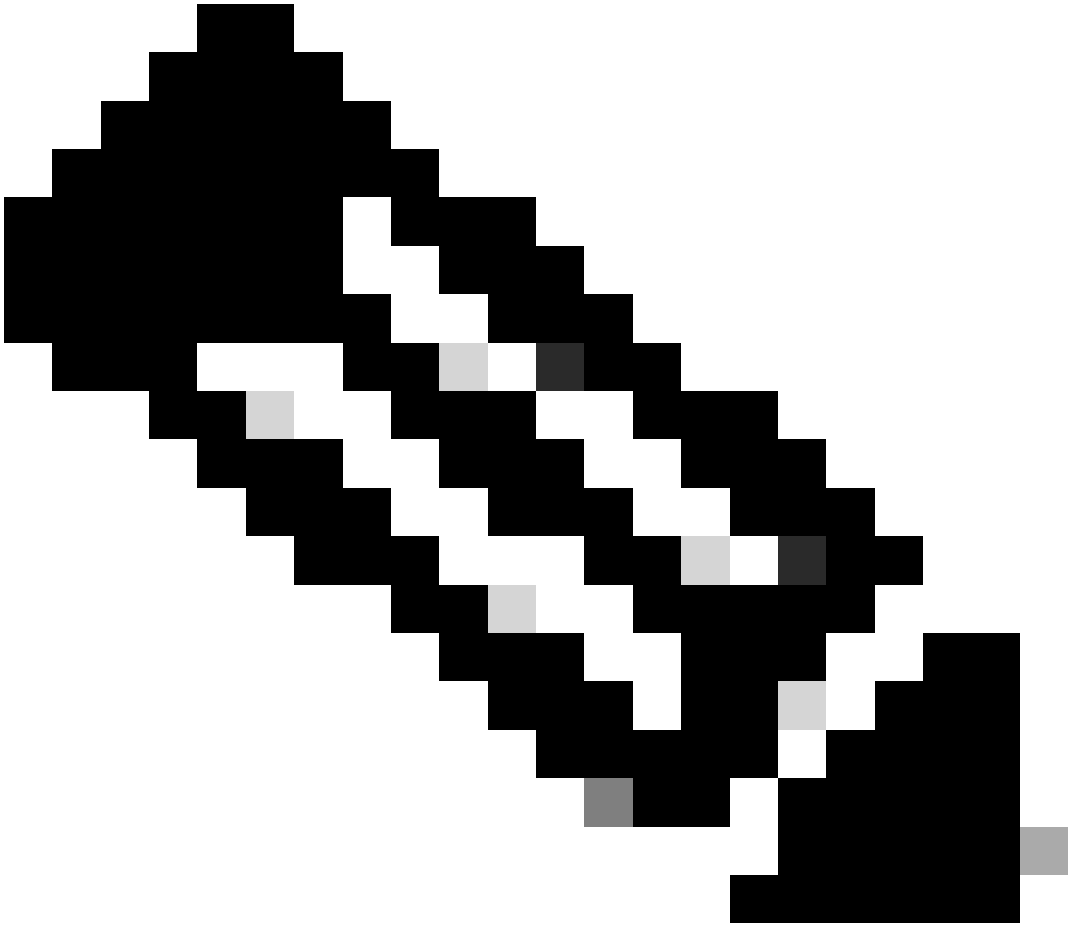
openssl s_client -connect <地址>：<端口> -cipher <密码> -no_tls1_3

在本示例中，从安装了openssl的Windows PC尝试与Expressway建立TLS连接。作为客户端，PC仅协商不必要的DHE-RSA-AES256-CCM密码，该密码使用DHE算法：

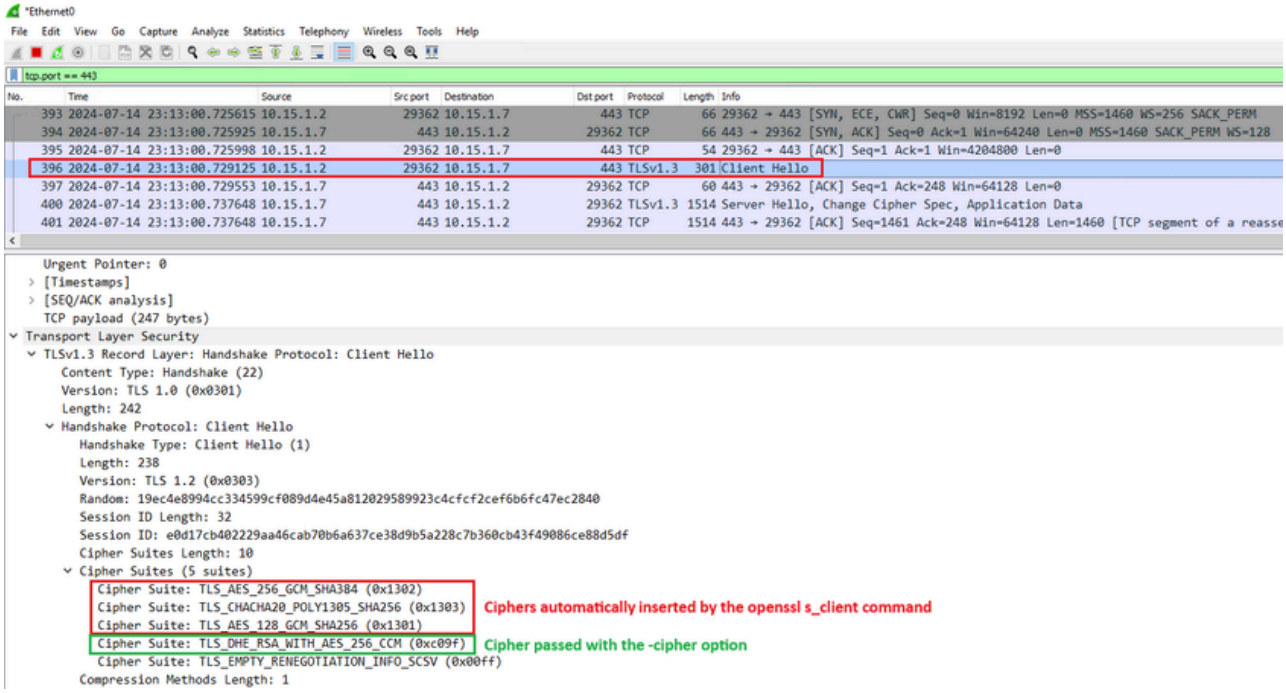
```
<#root>

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>
```

命令输出显示连接尝试失败，并显示“ssl/tls alert handshake failure：..\ssl\record\rec_layer_s3.c：865：SSL alert number 40”错误消息，因为Expressway配置为使用EECDH：EDH：HIGH：-AES256+SHA：！MEDIUM：！LOW：！3DES：！MD5：！PSK：！eNULL：！aDH：！DHE密码字符串进行HTTPS连接，这将禁用使用DHE算法的密码。



注意：要使使用openssl s_client命令的测试按说明工作，需要将-no_tls1_3选项传递给该命令。如果不包括，客户端会自动在ClientHello数据包中插入TLS 1.3密码：



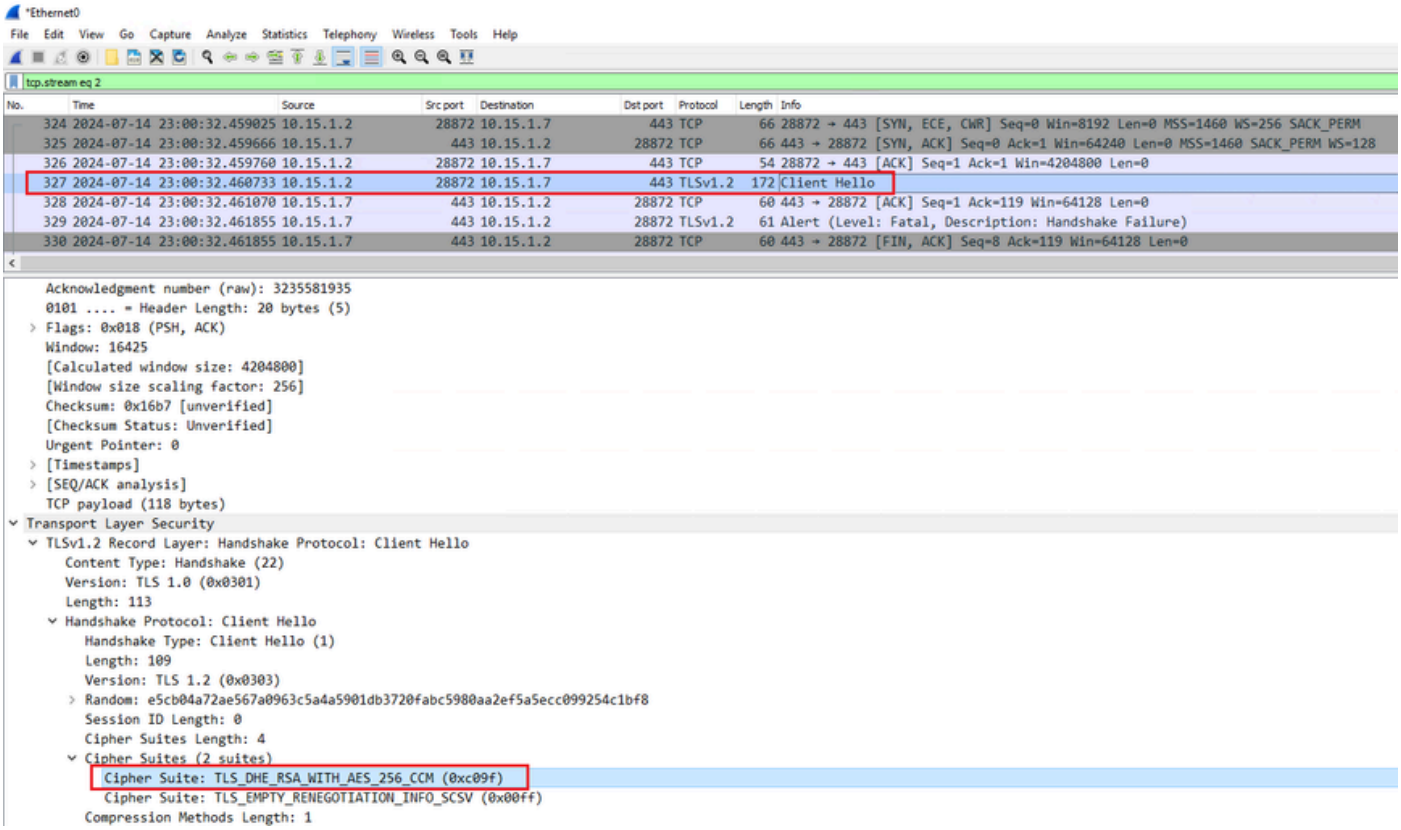
带有自动添加密码的ClientHello数据包

如果目标Expressway支持这些密码，则可以选择一个密码，而不是您需要测试的特定密码。连接成功，这可以让您相信，通过使用与-cipher选项一起传递给命令的已禁用密码，可以建立连接。

检查使用已禁用密码的TLS握手的数据包捕获

在使用其中一个禁用的密码执行连接测试时，您可以从测试设备或Expressway收集数据包捕获。然后，可以使用Wireshark对其进行检查，以进一步分析握手事件。

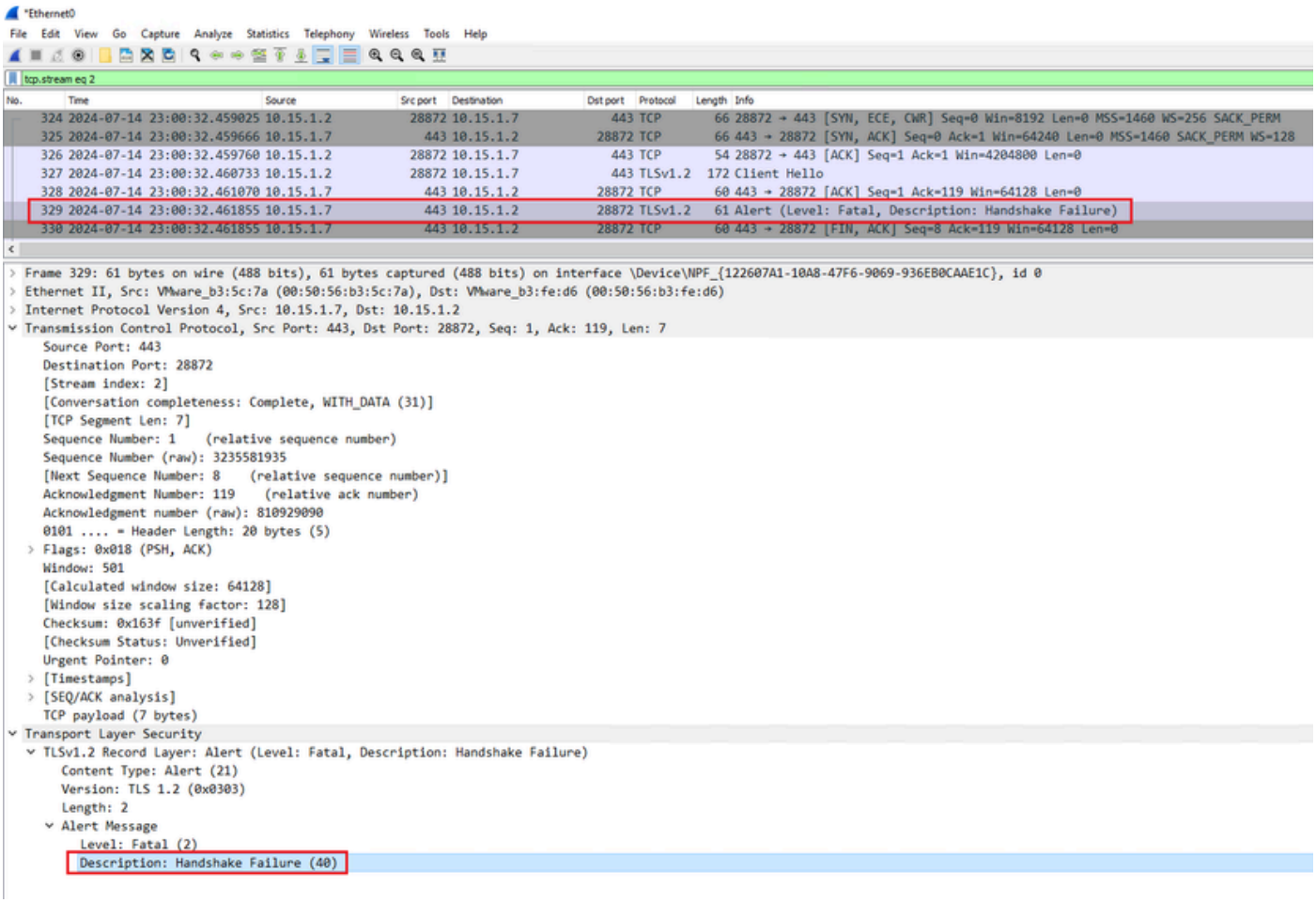
查找测试设备发送的ClientHello。确认它只协商不需要的测试密码，在本例中是使用DHE算法的密码：



Wireshark中的ClientHello数据包示例

：

确认Expressway使用致命的TLS警报数据包做出响应，拒绝连接。在本示例中，由于Expressway不支持按其HTTPS协议配置的密码字符串使用DHE密码，因此它使用包含故障代码40的严重TLS警报数据包进行响应。



Wireshark中的TLS严重警报数据包

相关信息

- [OpenSSL密码手册页]
- [思科Expressway管理员指南(X15.0) -章节：管理安全性-配置最低TLS版本和密码套件]

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。