Configure ESA to Prefer Perfect Forward Secrecy

Contents

Introduction

This document describes how to configure the preference for PFS in TLS encrypted connections on the Email Security Appliance (ESA).

Prerequisites

Requirements

Cisco recommends that you have knowledge of Secure Sockets Layer (SSL)/Transport Layer Security (TLS).

Components Used

The information in this document is based on AsyncOS for Email version 9.6 and above.

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. If your network is live, ensure that you understand the potential impact of any command.

Background Information

ESA supports Perfect Forward Secrecy (PFS). PFS means that data is transferred over a channel that uses symmetric encryption with ephemeral secrets. Even if the private key on one or both hosts is compromised, a previously recorded session cannot be decrypted.

The secret is not transferred through the channel. Instead, the shared secret is derived with the Diffie-Hellman (DH) problem. The secret is stored only in the host Random Access Memory (RAM) during the established session or key regeneration timeout.

ESA supports DH for key exchange.

Configure

INBOUND - ESA Acts as TLS Server

These cipher suites are available on ESA for inbound Simple Mail Transfer Protocol (SMTP) traffic and provide PFS. In this example, the cipher selection allows only cipher suites considered HIGH or MEDIUM, uses Ephemeral Diffie-Hellman (EDH) for key exchange, and prefers TLSv1.2. The cipher selection syntax is after OpenSSL syntax.

Ciphers with Forward Secrecy on AsyncOS 9.6+:

```
"EDH+TLSv1.2:EDH+HIGH:EDH+MEDIUM:!LOW:!EXP:!aNULL:!RC4:!DSS:!SEED:!IDEA:!MD5:!PSK:!3DES:!SRP"

List:
TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any Enc=AESGCM(256) Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=DH       Au=RSA  Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-RSA-AES256-CCM      TLSv1.2 Kx=DH       Au=RSA  Enc=AESCCM(256) Mac=AEAD
DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-CCM      TLSv1.2 Kx=DH       Au=RSA  Enc=AESCCM(128) Mac=AEAD
DHE-RSA-AES256-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA256
DHE-RSA-CAMELLIA256-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=Camellia(256) Mac=SHA256
DHE-RSA-AES128-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA256
DHE-RSA-CAMELLIA128-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA256
DHE-RSA-AES128-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA1
DHE-RSA-CAMELLIA128-SHA SSLv3 Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA1
```

The Kx (key exchange) field shows that DH is used to derive the secret.

ESA supports these ciphers with the default **sslconfig** setting (:ALL), but does not prefer them. To prefer ciphers that offer PFS, modify **sslconfig** and add **EDH** or a combination such as **EDH+<cipher or cipher group name>** to the cipher selection.

1. Review the default configuration.

```
ESA> sslconfig

sslconfig settings:
  Inbound SMTP method:  tlsv1/tlsv1.2
  Inbound SMTP ciphers:
        RC4-SHA
        RC4-MD5
        ALL
```

2. Configure the new inbound cipher preference.

```
ESA> sslconfig

Inbound SMTP method:  tlsv1/tlsv1.2
  Inbound SMTP ciphers:
        EDH+TLSv1.2
        EDH+HIGH
        EDH+MEDIUM
        RC4-SHA
        RC4-MD5
        ALL
```

Note: RC4 as a cipher and MD5 as a MAC are considered weak and legacy. Avoid their use with SSL/TLS, especially for higher data volumes without key regeneration.

Recommended sslconfig Settings for INBOUND

This example shows a recommended configuration that allows only ciphers generally considered strong and secure.

A recommended inbound configuration removes RC4 and MD5, as well as other legacy and weak options such as Export (EXP), Low (LOW), IDEA, SEED, 3DES, DSS certificates, anonymous key exchange (aNULL), pre-shared keys (PSK), and SRP. It also disables Elliptic Curve Diffie-Hellman (ECDH) for key exchange and Elliptic Curve Digital Signature Algorithm (ECDSA).

1. Enter the next cipher string in **sslconfig**.

```
EDH+TLSv1.2:EDH+HIGH:EDH+MEDIUM:HIGH:MEDIUM:!ECDH:!ECDSA:!LOW:!EXP:!aNULL:!RC4:!DSS:!SEED:!IDEA:!MD5:!P
```

The string entered in **sslconfig** results in this list of supported ciphers for INBOUND:

```
TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=DH       Au=RSA  Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-RSA-AES256-CCM      TLSv1.2 Kx=DH       Au=RSA  Enc=AESCCM(256) Mac=AEAD
DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-CCM      TLSv1.2 Kx=DH       Au=RSA  Enc=AESCCM(128) Mac=AEAD
DHE-RSA-AES256-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA256
DHE-RSA-CAMELLIA256-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=Camellia(256) Mac=SHA256
DHE-RSA-AES128-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA256
DHE-RSA-CAMELLIA128-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA256
DHE-RSA-AES128-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA1
DHE-RSA-CAMELLIA128-SHA SSLv3 Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA1
AES256-GCM-SHA384       TLSv1.2 Kx=RSA      Au=RSA  Enc=AESGCM(256) Mac=AEAD
AES256-CCM              TLSv1.2 Kx=RSA      Au=RSA  Enc=AESCCM(256) Mac=AEAD
AES128-GCM-SHA256       TLSv1.2 Kx=RSA      Au=RSA  Enc=AESGCM(128) Mac=AEAD
AES128-CCM              TLSv1.2 Kx=RSA      Au=RSA  Enc=AESCCM(128) Mac=AEAD
```

```
AES256-SHA256           TLSv1.2 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA256
CAMELLIA256-SHA256      TLSv1.2 Kx=RSA      Au=RSA  Enc=Camellia(256) Mac=SHA256
AES128-SHA256           TLSv1.2 Kx=RSA      Au=RSA  Enc=AES(128)  Mac=SHA256
CAMELLIA128-SHA256      TLSv1.2 Kx=RSA      Au=RSA  Enc=Camellia(128) Mac=SHA256
AES256-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA1
AES128-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(128)  Mac=SHA1
CAMELLIA128-SHA         SSLv3 Kx=RSA      Au=RSA  Enc=Camellia(128) Mac=SHA1
```

Note: ESA acting as a TLS server for inbound traffic does not currently support Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) for key exchange or ECDSA certificates.

OUTBOUND - ESA acts as TLS Client

For outbound SMTP traffic, ESA supports ECDHE and ECDSA certificates in addition to the inbound capabilities.

Note: Elliptic Curve Cryptography (ECC) certificates that use ECDSA are not widely adopted.

When an outbound email is delivered, ESA is the TLS client. A TLS client certificate is optional. If the TLS server does not require ESA to provide an ECDSA client certificate, ESA can continue with an ECDSA-secured session. When ESA as the TLS client is asked for its certificate, it provides the configured RSA certificate for the outbound direction.

Caution: The preinstalled Trusted CA Certificate Store (System List) on ESA does not include ECC (ECDSA) root certificates. You can need to manually add trusted ECC root certificates to the Custom List to make the ECC chain of trust verifiable.

To prefer DHE/ECDHE ciphers that offer PFS, modify the **sslconfig** cipher selection.

1. Add this string to the current cipher selection.

```
"EDH+TLSv1.2:ECDH+TLSv1.2:EDH+HIGH:EDH+MEDIUM:ECDH+HIGH:ECDH+MEDIUM"
```

Recommended sslconfig Settings for OUTBOUND

This example shows a recommended configuration that allows only ciphers generally considered strong and secure.

```
EDH+TLSv1.2:ECDH+TLSv1.2:EDH+HIGH:EDH+MEDIUM:ECDH+HIGH:ECDH+MEDIUM:HIGH:MEDIUM:!LOW:!EXP:!aNULL:!RC4:!D
```

The string entered in **sslconfig** results in this list of supported ciphers for OUTBOUND:

```
TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AES(256) Mac=SHA256
DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AES(128) Mac=SHA256
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(256) Mac=SHA384
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(256) Mac=SHA384
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(128) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(128) Mac=AEAD
ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA256
ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA256
DHE-RSA-AES256-SHA SSLv3 Kx=DH Au=RSA Enc=AES(256) Mac=SHA1
DHE-RSA-CAMELLIA256-SHA SSLv3 Kx=DH Au=RSA Enc=Camellia(256) Mac=SHA1
DHE-RSA-AES128-SHA SSLv3 Kx=DH Au=RSA Enc=AES(128) Mac=SHA1
DHE-RSA-CAMELLIA128-SHA SSLv3 Kx=DH Au=RSA Enc=Camellia(128) Mac=SHA1
ECDHE-RSA-AES256-SHA SSLv3 Kx=ECDH Au=RSA Enc=AES(256) Mac=SHA1
ECDHE-ECDSA-AES256-SHA SSLv3 Kx=ECDH Au=ECDSA Enc=AES(256) Mac=SHA1
ECDHE-RSA-AES128-SHA SSLv3 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA1
ECDHE-ECDSA-AES128-SHA SSLv3 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA1
AES256-GCM-SHA384 TLSv1.2 Kx=RSA Au=RSA Enc=AESGCM(256) Mac=AEAD
AES256-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AES(256) Mac=SHA256
AES256-SHA SSLv3 Kx=RSA Au=RSA Enc=AES(256) Mac=SHA1
CAMELLIA256-SHA SSLv3 Kx=RSA Au=RSA Enc=Camellia(256) Mac=SHA1
AES128-GCM-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AESGCM(128) Mac=AEAD
AES128-SHA256 TLSv1.2 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA256
AES128-SHA SSLv3 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA1
CAMELLIA128-SHA SSLv3 Kx=RSA Au=RSA Enc=Camellia(128) Mac=SHA1
```

Verify

Use these steps to verify that ESA prefers PFS cipher suites.

1. Run **sslconfig** and review the configured inbound and outbound cipher strings.
2. Confirm that the cipher selection includes **EDH** for DHE preference and, for outbound traffic, **ECDH** entries where applicable.
3. Review the resulting cipher list shown by **sslconfig** and confirm that DHE or ECDHE cipher suites appear before RSA key exchange cipher suites.
4. Test a TLS connection and confirm that the negotiated cipher suite uses DHE or ECDHE when the remote TLS peer supports it.

Troubleshoot

Use these checks if ESA does not negotiate a PFS cipher suite.

- Verify that the configured **sslconfig** cipher string includes **EDH** for inbound traffic and the required **EDH** or **ECDH** values for outbound traffic.
- Confirm that the remote TLS server or TLS client supports DHE or ECDHE cipher suites. If the peer does not support them, ESA can negotiate a non-PFS cipher suite that is still allowed by the configuration.
- For outbound ECDSA-based sessions, confirm that the peer does not require an ECDSA client certificate. ESA presents the configured RSA certificate for the outbound direction when a client certificate is requested.
- For outbound ECC trust validation, confirm that the required ECC root certificates are present in the Custom List if they are not included in the System List.

Related Information

- OpenSSL ciphers
- Next Generation Cryptography
- Cisco Technical Support and Documentation