

Decrypt SSL traffic with certificates and private keys

Published: 2020-03-05

You can decrypt forwarded SSL traffic by uploading the private key and server certificate associated with that traffic. The certificate and key are uploaded over an HTTPS connection from a web browser to the Discover appliance.

After upload, private keys are encrypted and stored on the Discover appliance. To ensure that private keys are not transferable to other appliances, they are encrypted with an internal key that has information specific to the appliance to which it was uploaded.

Separation of privileges is enforced so that only the SSL decryption process on the appliance can access the private keys. While you can add new private keys through the ExtraHop Admin UI, you cannot access stored private keys.



Note: Your traffic must be encrypted with a supported cipher suite. See [Supported SSL cipher suites](#).

Upload a PEM certificate and RSA private key



Tip: You can export a password-protected key to add to your ExtraHop appliance by running the following command on a program such as OpenSSL:

```
openssl rsa -in yourcert.pem -out new.key
```

1. Log into the Admin UI on your Discover appliance.
2. In the System Configuration section, click **Capture**.
3. Click **SSL Decryption**.
4. In the Private Key Decryption section, select the checkbox for **Require Private Keys**.
5. Click **Save**.
6. In the Private Keys section, click **Add Keys**.
7. In the Add PEM Certificate and RSA Private Key section, enter the following information:

Name

A descriptive name to identify this certificate and key.

Enabled

Clear this checkbox if you want to disable this SSL certificate.

Certificate

The public key certificate.

Private Key

The RSA private key.

8. Click **Add**.

Next steps

[Add the encrypted protocols](#) you want to decrypt with this certificate.

Upload a PKCS#12/PFX file

PKCS#12/PFX files are archived in a secure container on the appliance that contain both public and private key pairs and that can only be accessed with a password.



Tip: To export private keys from a Java KeyStore to a PKCS#12 file, run the following command on your server, where `javakeystore.jks` is the path of your Java KeyStore:

```
keytool -importkeystore -srckeystore javakeystore.jks -
destkeystore
pkcs.p12 -srcstoretype jks -deststoretype pkcs12
```

1. Log into the Admin UI on your Discover appliance.
2. In the System Configuration section, click **Capture**.
3. Click **SSL Decryption**.
4. In the Private Key Decryption section, select the checkbox for **Require Private Keys**.
5. Click **Save**.
6. In the Private Keys section, click **Add Keys**.
7. In the Add PKCS#12/PFX File With Password section, enter the following information:

Description

A descriptive name to identify this certificate and key.

Enabled

Clear this checkbox to disable this SSL certificate.

8. Next to PKCS#12/PFX file, click **Choose File**.
9. Browse to the file and select it, then click **Open**.
10. In the Password field, type the password for the PKCS#13/PFX file.
11. Click **Add**.
12. Click **OK**.

Next steps

[Add the encrypted protocols](#) you want to decrypt with this certificate.

Add encrypted protocols

You must add each protocol that you want to decrypt for each uploaded certificate.

1. Log into the Admin UI on your Discover appliance.
2. In the System Configuration section, click **Capture**.
3. Click **SSL Decryption**.
4. In the Protocol to Port Mapping by Key section, click **Add Protocol**.
5. On the Add Encrypted Protocol page, enter the following information:

Protocol

From the drop-down list, select the protocol you want to decrypt.

Key

From the drop-down list, select an uploaded private key.

Port

Type the source port for the protocol. By default this value is set to 443, which specifies HTTP traffic. Specify 0 to decrypt all protocol traffic.

6. Click **Add**.

Supported SSL cipher suites

To decrypt SSL traffic in real time, you must configure your server applications to encrypt traffic with supported ciphers. The following information provides a list of supported cipher suites and the best practices you should consider when implementing SSL encryption.

- Turn off SSLv2 to reduce security issues at the protocol level.
- Turn off SSLv3, unless required for compatibility with older clients.
- Turn off SSL compression to avoid the CRIME security vulnerability.
- Turn off session tickets unless you are familiar with the risks that might weaken Perfect Forward Secrecy.
- Configure the server to select the cipher suite in order of the server preference.

The following cipher suites can be decrypted by the ExtraHop appliance and are listed in from strongest to weakest and by server preference:

- AES256-GCM-SHA384
- AES128-GCM-SHA256
- AES256-SHA256
- AES128-SHA256
- AES256-SHA
- AES128-SHA
- DES-CBC3-SHA

The following list includes some common cipher suites that support Perfect Forward Secrecy (PFS) and can be decrypted by the ExtraHop appliance when session key forwarding is configured. To configure session key forwarding, see [Install the ExtraHop session key forwarder on a Windows server](#) or [Install the ExtraHop session key forwarder on a Linux server](#).

- TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_RC4_128_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-ECDSA-AES256-SHA
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-ECDSA-AES128-SHA
- ECDHE-ECDSA-RC4-SHA
- ECDHE-ECDSA-DES-CBC3-SHA

The following list of cipher suites support Perfect Forward Secrecy (PFS) but cannot be decrypted by the ExtraHop appliance:

- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-ECDSA-AES128-SHA256