

Decrypt SSL traffic with certificates and private keys

Published: 2024-03-27

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 ExtraHop system.

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

Separation of privileges is enforced so that only the SSL decryption process on the system can access the private keys. While you can add new private keys through the Administration settings, you cannot access stored private keys.



Note: Your traffic must be encrypted with a [supported cipher suite](#). Learn more about [SSL/TLS decryption](#).

Upload a PEM certificate and RSA private key



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

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

1. Log in to the Administration settings on the ExtraHop system through `https://<extrahop-hostname-or-IP-address>/admin`.
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 ExtraHop system and contains both public and private key pairs, which 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 in to the Administration settings on the ExtraHop system through `https://<extrahop-hostname-or-IP-address>/admin`.
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#12/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 in to the Administration settings on the ExtraHop system through `https://<extrahop-hostname-or-IP-address>/admin`.
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/TLS cipher suites

The ExtraHop system can decrypt SSL/TLS traffic that has been encrypted with PFS or RSA cipher suites. All supported cipher suites can be decrypted by installing the session key forwarder on a server and configuring the ExtraHop system.

Cipher suites for RSA can also decrypt the traffic with a certificate and private key—with or without session key forwarding.

Decryption methods

The table below provides a list of cipher suites that the ExtraHop system can [decrypt](#) along with the supported decryption options.

- **PFS + GPP:** the ExtraHop system can decrypt these cipher suites with session key forwarding and [global protocol to port mapping](#)
- **PFS + Cert:** the ExtraHop system can decrypt these cipher suites with session key forwarding and the [certificate and private key](#)
- **RSA + Cert:** the ExtraHop system can decrypt these cipher suites without session key forwarding as long as you have uploaded the [certificate and private key](#)

| Hex Value | Name (IANA) | Name (OpenSSL) | Supported Decryption |
|-----------|-------------------------------------|-----------------------|------------------------------------|
| 0x04 | TLS_RSA_WITH_RC4_128_MD5 | RC4-MD5 | PFS + GPP PFS + Cert RSA + Cert |
| 0x05 | TLS_RSA_WITH_RC4_128_SHA | RC4-SHA | PFS + GPP PFS + Cert RSA + Cert |
| 0x0A | TLS_RSA_WITH_3DES_EDE_CBC_SHA | 3DES-CBC-SHA | PFS + GPP PFS + Cert RSA + Cert |
| 0x16 | TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA | DHE-RSA-3DES-CBC-SHA | PFS + GPP PFS + Cert |
| 0x2F | TLS_RSA_WITH_AES_128_CBC_SHA | AES128-SHA | PFS + GPP PFS + Cert RSA + Cert |
| 0x33 | TLS_DHE_RSA_WITH_AES_128_CBC_SHA | DHE-RSA-AES128-SHA | PFS + GPP PFS + Cert |
| 0x35 | TLS_RSA_WITH_AES_256_CBC_SHA | AES256-SHA | PFS + GPP PFS + Cert RSA + Cert |
| 0x39 | TLS_DHE_RSA_WITH_AES_256_CBC_SHA | DHE-RSA-AES256-SHA | PFS + GPP PFS + Cert |
| 0x3C | TLS_RSA_WITH_AES_128_CBC_SHA256 | AES128-SHA256 | PFS + GPP PFS + Cert RSA + Cert |
| 0x3D | TLS_RSA_WITH_AES_256_CBC_SHA256 | AES256-SHA256 | PFS + GPP PFS + Cert RSA + Cert |
| 0x67 | TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 | DHE-RSA-AES128-SHA256 | PFS + GPP PFS + Cert |
| 0x6B | TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 | DHE-RSA-AES256-SHA256 | PFS + GPP PFS + Cert |
| 0x9C | TLS_RSA_WITH_AES_128_GCM_SHA256 | AES128-GCM-SHA256 | PFS + GPP PFS + Cert RSA + Cert |

| Hex Value | Name (IANA) | Name (OpenSSL) | Supported Decryption |
|-----------|--|--|------------------------------------|
| 0x9D | TLS_RSA_WITH_AES_256_GCM_SHA384 | TLS_RSA_WITH_AES_256_GCM_SHA384 | PFS + GPP PFS + Cert RSA + Cert |
| 0x9E | TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 | TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 | PFS + GPP PFS + Cert |
| 0x9F | TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 | TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 | PFS + GPP PFS + Cert |
| 0x1301 | TLS_AES_128_GCM_SHA256 | TLS_AES_128_GCM_SHA256 | PFS + GPP PFS + Cert |
| 0x1302 | TLS_AES_256_GCM_SHA384 | TLS_AES_256_GCM_SHA384 | PFS + GPP PFS + Cert |
| 0x1303 | TLS_CHACHA20_POLY1305_SHA256 | TLS_CHACHA20_POLY1305_SHA256 | PFS + GPP PFS + Cert |
| 0xC007 | TLS_ECDHE_ECDSA_WITH_RC4_128_SHA | TLS_ECDHE_ECDSA_WITH_RC4_128_SHA | PFS + GPP |
| 0xC008 | TLS_ECDHE_ECDSA_WITH_CBC3_SHA | TLS_ECDHE_ECDSA_WITH_CBC3_SHA | PFS + GPP |
| 0xC009 | TLS_ECDHE_ECDSA_WITH_CAC128_SHA | TLS_ECDHE_ECDSA_WITH_CAC128_SHA | PFS + GPP |
| 0xC00A | TLS_ECDHE_ECDSA_WITH_CAC256_SHA | TLS_ECDHE_ECDSA_WITH_CAC256_SHA | PFS + GPP |
| 0xC011 | TLS_ECDHE_RSA_WITH_RC4_128_SHA | TLS_ECDHE_RSA_WITH_RC4_128_SHA | PFS + GPP PFS + Cert |
| 0xC012 | TLS_ECDHE_RSA_WITH_CBC3_SHA | TLS_ECDHE_RSA_WITH_CBC3_SHA | PFS + GPP PFS + Cert |
| 0xC013 | TLS_ECDHE_RSA_WITH_CAC128_SHA | TLS_ECDHE_RSA_WITH_CAC128_SHA | PFS + GPP PFS + Cert |
| 0xC014 | TLS_ECDHE_RSA_WITH_CAC256_SHA | TLS_ECDHE_RSA_WITH_CAC256_SHA | PFS + GPP PFS + Cert |
| 0xC023 | TLS_ECDHE_ECDSA_WITH_CAC128_SHA256 | TLS_ECDHE_ECDSA_WITH_CAC128_SHA256 | PFS + GPP |
| 0xC024 | TLS_ECDHE_ECDSA_WITH_CAC256_SHA384 | TLS_ECDHE_ECDSA_WITH_CAC256_SHA384 | PFS + GPP |
| 0xC027 | TLS_ECDHE_RSA_WITH_CAC128_SHA256 | TLS_ECDHE_RSA_WITH_CAC128_SHA256 | PFS + GPP PFS + Cert |
| 0xC028 | TLS_ECDHE_RSA_WITH_CAC256_SHA384 | TLS_ECDHE_RSA_WITH_CAC256_SHA384 | PFS + GPP PFS + Cert |
| 0xC02B | TLS_ECDHE_ECDSA_WITH_CAC128_GCM_SHA256 | TLS_ECDHE_ECDSA_WITH_CAC128_GCM_SHA256 | PFS + GPP |
| 0xC02C | TLS_ECDHE_ECDSA_WITH_CAC256_GCM_SHA384 | TLS_ECDHE_ECDSA_WITH_CAC256_GCM_SHA384 | PFS + GPP |
| 0xC02F | TLS_ECDHE_RSA_WITH_CAC128_GCM_SHA256 | TLS_ECDHE_RSA_WITH_CAC128_GCM_SHA256 | PFS + GPP PFS + Cert |
| 0xC030 | TLS_ECDHE_RSA_WITH_CAC256_GCM_SHA384 | TLS_ECDHE_RSA_WITH_CAC256_GCM_SHA384 | PFS + GPP PFS + Cert |

| Hex Value | Name (IANA) | Name (OpenSSL) | Supported Decryption |
|-----------|---|-----------------------------|----------------------|
| 0xCCA8 | TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 | ECDHE-RSA-CHACHA20-POLY1305 | PSK + GPP PFS + Cert |
| 0xCCA9 | TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 | ECDSA-CHACHA20-POLY1305 | PSK + GPP |
| 0xCCAA | TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 | DHE-RSA-CHACHA20-POLY1305 | PSK + GPP PFS + Cert |