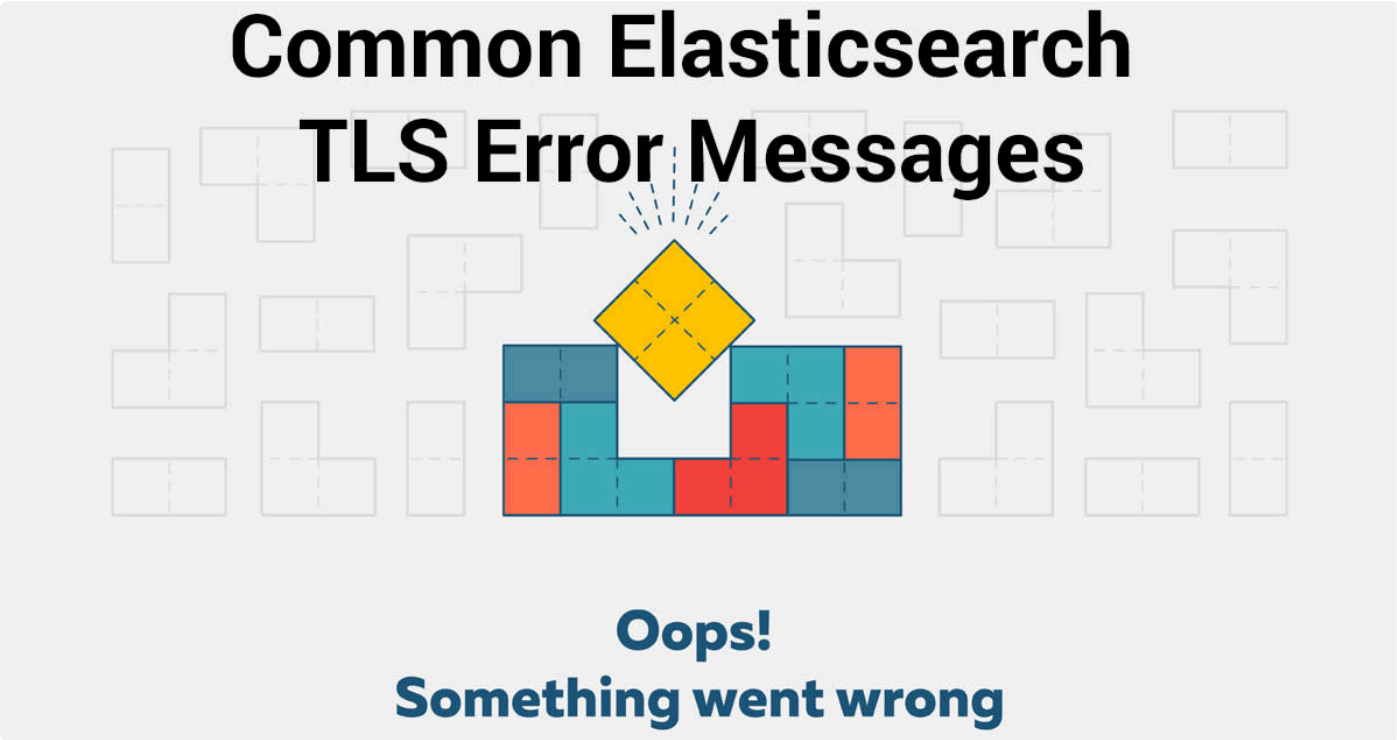
Common Elasticsearch TLS error messages

Jochen Kressin · 2020-09-22



When configuring TLS for Elasticsearch, you can run into a lot of different issues. TLS is undoubtedly the best way to secure data and traffic in Elasticsearch and Kibana. Still, it can also get tricky from time to time. This article is about the most common TLS error messages and how to fix them.

Server-side or client-side? Look at both.

It is essential to understand that TLS always includes two parties, the server, and the client. Error messages can show up on both sides. Usually, when you see an error message on one side, there will be a corresponding message on the other side. It often helps to look at both log files to pinpoint the issue. For example, if Kibana cannot connect to Elasticsearch because of a TLS issue, look at the Elasticsearch logs as well. This will give you a better understanding of the issue.

In any case, check out or TLS Troubleshooting Guide for detailed step-by-step instructions on how to debug TLS problems.

"Someone speaks plaintext instead of ssl, will close the channel"

This indicates that a client like Kibana, Beats, or Logstash is trying to connect to Elasticsearch. But the client is using HTTP instead of HTTPS. Make sure you have configured your client to use HTTPS. For example, in Kibana, instead of:

```
elasticsearch.hosts: "http://elasticsearch.example.com:9200"
```

Use:

```
elasticsearch.hosts: "https://elasticsearch.example.com:9200"
```

"File does not contain valid private key"

This error can have two reasons:

Password not correct

If your key file is password protected, you need to specify the password in elasticsearch.yml like:

```
searchguard.ssl.transport.pemkey_password: ...
searchguard.ssl.http.pemkey_password: ...
```

Vice versa, if your key file is not password protected, be sure to omit those lines.

Key is in PKCS#1 format

Search Guard supports private keys in PKCS#8 format. If your key is in PKCS#1 format, you will also see the above error message.

To check whether your key is in PKCS#1 format, open the key file with a text editor. The key is in PKCS#1 format if the file begins with:

```
-----BEGIN RSA PRIVATE KEY-----
```

Use OpenSSL to convert it to PKCS#8 format like:

```
openssl pkcs8 -topk8 -inform pem -in pkcs1-key.pem -outform pem -nocrypt -out pkcs8-key.pem
```

"java.security.cert.CertificateException: No name matching XYZ found"

This indicates that the hostname in the SAN section of the TLS certificate does not match the Elasticsearch node's hostname. For example, your certificate contains the hostname node-0.example.com, but you try to install it on node-2.exampe.com. Make sure the hostname(s) in your certificate match the hostname of your node. Alternatively, you can also disable hostname verification. This is not recommended for production.

"javax.net.ssl.SSLException: Received fatal alert: certificate_unknown"

"sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target"

This indicates that a certificate that was returned during a TLS handshake is not trusted. The root CA used to sign the certificate is not available. This, for example, happens if you use a self-signed root CA but you forgot to configure it correctly.

For Elasticsearch, make sure you confgured the root CA for both transport and REST layer like:

```
searchguard.ssl.transport.pemtrustedcas_filepath: /path/to/root-ca.pem
searchguard.ssl.http.pemtrustedcas_filepath: /path/to/root-ca.pem
```

For Kibana, configure the root CA like:

```
elasticsearch.ssl.certificateAuthorities: /path/to/root-ca.pem
```

"org.elasticsearch.common.netty.handler.ssl.NotSslRecordException: not an SSL/TLS record"

"java.io.StreamCorruptedException: invalid internal transport message format"

This indicates that a node in your cluster tries to connect to another node, but not all nodes use TLS. In other words, not all nodes are configured to use encrypted communication. Make sure you have installed Search Guard on all nodes in your cluster.

"sun.security.validator.ValidatorException: PKIX path building failed"

"sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target"

This indicates that the "chain of trust" from your leaf certificate (e.g., a node certificate) to your root CA cannot be established.

If you are **not** using any intermediate certificates (means, your leaf certificate was directly signed with the root CA), ensure that you have configured the correct root CA in elasticsearch.yml.

```
searchguard.ssl.transport.pemtrustedcas_filepath: /path/to/root-ca.pem
searchguard.ssl.http.pemtrustedcas_filepath: /path/to/root-ca.pem
```

If you use intermediate certificates (means, your leaf certificate was signed with an intermediate CA rather than the root CA), ensure that the root CA and all intermediate CAs are available. If you have received your certificates from your companies PKI, the filename is typically called "chain-ca.pem", "ca-chain.pem" or "ca-bundle.pem".

By using a text editor, you can inspect the file content and ensure it contains both the root CA and intermediate CAs. Instead of configuring the root CA, use the CA chain in elasticsearch.yml:

```
searchguard.ssl.transport.pemtrustedcas_filepath: /path/to/ca-chain.pem
searchguard.ssl.http.pemtrustedcas_filepath: /path/to/ca-chain.pem
```

"javax.net.ssl.SSLHandshakeException: no cipher suites in common"

This indicates that the client and the server do not share any cipher suite, and thus no TLS connection can be established. The available cipher suites are mainly dictated by the JVM you use. The Oracle JVM

may support other cipher suited than the IBM JVM.

You have configured Search Guard to only use specific ciphers

The most common source of this error is that you configured Search Guard only to use some specific ciphers, and your browser does not support any of them.

In your elasticsearch.yml, check if you have configured the allowed cipher suites:

```
searchguard.ssl.http.enabled_ciphers:
  - "TLS_DHE_RSA_WITH_AES_256_CBC_SHA"
  - "TLS_DHE_DSS_WITH_AES_128_CBC_SHA256"
```

If this is the case, make sure your browser supports at least one of them.

You use different JVMs for sgadmin and Elasticsearch

If this error surfaces while using sgadmin, check the JVMs you are using.

By default, sgadmin uses the JVM installed on your machine, and it also honors the JAVA_HOME environment variable. Elasticsearch, on the other hand, ships with a JVM that it uses by default. If the JVMs differ, they might not share any cipher suites.

Image: Shutterstock / Kay Mosk

This article was published on the Search Guard blog: https://search-guard.com/blog/elasticsearch-tls-error-messages/