TLS management: Replace root CAs on a running Elasticsearch cluster

Jochen Kressin · 2020-09-03

In the last article, we explained how you can change TLS certificates on the REST or transport layer on a running Elasticsearch cluster without any downtime or a rolling restart. In this article, we explain how to change the root CA.

Why change the root CA?

Changing the root or intermediate CA on a production Elasticsearch cluster is surely not something you need regularly. However, there can be situations when you want to do exactly that. For example, you use a self-signed root CA on the transport layer, and the certificate is about to expire.

While changing the root CA requires some more steps than merely changing leaf certificates, Search Guard, of course, supports it. Let's first look at some prerequisites.

Prerequisites

We will now look at how to change the root CA that was used to sign the certificates in transport layer. However, the same approach works on the REST layer, as well.

Our current TLS setup for the transport layer looks like:

```
searchguard.ssl.transport.pemcert_filepath: esnode.pem
searchguard.ssl.transport.pemkey_filepath: esnode-key.pem
searchguard.ssl.transport.pemtrustedcas_filepath: root-ca.pem
```

Changing the root certificate inevitably means changing the leaf certificate as well. As a prerequisite, we generate a node certificate and sign it with our new root CA.

You can use your companies PKI, our offline TLS generator, or tools like OpenSSL.

We also make sure that the new node certificate has the same DN as the current one.

This leaves us with a new root CA (*root-ca-new.pem*), a new node certificate (*esnode-new.pem*), and a new private key (*esnode-key-new.pem*).

Handling certificate compatibility

Since we are operating at the node level, we have to ensure that all nodes can talk to each other while changing the certificates.

Therefore we cannot only place the new root CA on our node and remove the current one. Doing so would break communication since the new root CA cannot verify certificates from nodes still running the old root CA.

How to solve this issue? Luckily, a PEM file can contain more than one certificate. So we can just add our new root CA to the existing *root-ca-new.pem* file and our node will be able to verify certificates signed by both the old and new CA.

PEM files are text files, so we open the **root-ca.pem** with a text editor and copy and paste the contents of the **root-ca-new.pem** to this file. The file should now look like this:

```
-----BEGIN CERTIFICATE-----
MIID/jCCAuagAwIBAgIBATANBgkqhkiG9w0BAQsFADCBjzETMBEGCgmSJomT8ixk
ARkWA2NvbTEXMBUGCgmSJomT8ixkARkWB2V4YW1wbGUxGTAXBgNVBAoMEEV4YW1w
...
BU36ZAa80W/8TBnqZTkNnqZV
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDuDCCAqCgAwIBAgIBATANBgkqhkiG9w0BAQsFADBtMR0wGwYDVQQKDBRFeGFt
cGxlIERTRyBJbmMuIDEuMDElMCMGA1UECwwcRXhhbXBsZSBEU0cgSW5jLiAxLjAg
...
bRM0+c02BwwUwZH35xl18LeJnp03hPlKhL/kNrQTz1X4wRA5MIStjE1inU4=
-----END CERTIFICATE-----
```

As you can see, the file now contains two root CAs and can be used to validate certificates signed by either the current or new CA.

Changing the certificates

Now we are ready to change the actual certificates on our node. We upload our combined root CA PEM file, the new node certificate, and the new private key to our node.

Note that for the hot-reload to work, we need to keep the filenames as configured in our elasticsearch.yml:

```
searchguard.ssl.transport.pemcert_filepath: esnode.pem
searchguard.ssl.transport.pemkey_filepath: esnode-key.pem
searchguard.ssl.transport.pemtrustedcas_filepath: root-ca.pem
```

We can for example use scp, overwriting the current *esnode.pem* file like:

```
scp ./esnode-new.pem username@node0.example.com:/etc/elasticsearch/esnode.pem
```

We do this for all files, always keeping the old filename.

Reloading the certificates

Now it's time to trigger the actual certificate reload. As described in the [previous article](https://search-guard.com/elasticsearch-tls-exchange/), we use the Search Guard REST API with an admin TLS certificate to trigger the actual reload:

For transport (node-to-node) certificates, use:

```
POST /_searchguard/api/ssl/transport/reloadcerts/
```

If you use curl, your command will look something like this:

```
curl --cert kirk.pem --key kirk.key.pem -XPOST
"https://node0.example.com:9200/_searchguard/api/ssl/transport/reloadcerts/"
```

```
If this succeeds and the Elasticsearch log file is clean, perform these steps for all nodes in the cluster.

## Cleaning up

Now that we have exchanged all certificates on all nodes, it's time for a bit of cleanup: Our *root-ca.pem* file contains the new root CA and also the now unused old root CA. While not strictly required, we should remove the unused root-CA from the *root-ca.pem* file and upload that file again to all nodes. After that, we again trigger the reload command on all nodes.

## Where to go next

* Read the previous article on how to [reload leaf certificates](/elasticsearch-tls-change-runtime/)
* Read about [certificate revocation lists](https://search-guard.com/elasticsearch-tls-certificate-revocation/) to manage invalid certificates
* Dig into the [documentation on TLS hot reload](https://docs.search-guard.com/latest/hot-reload-tls)
```