Creating Search Guard TLS certificates with OpenSSL

Jochen Kressin · 2019-01-04

SearchGuard security heavily relies on TLS certificates. For proof-of-concept instances, they can be generated by our demo installer or using our online TLS Generator. For production environments we have created an offline TLS tool. In this article, we describe how to create certificates from scratch based on OpenSSL command line. This allows more custom setup when necessary.

SearchGuard distinguishes three different certificate types:

Node certificates

- used to identify and secure traffic between Elasticsearch nodes on the transport layer
  Client certificates
- used to identify Elasticsearch clients on the REST and transport layer.
  Admin certificates
- which basically are client certificates that have elevated rights to perform administrative tasks.

Generating Root CA

Root CA (Certificate Authority) is a certificate that will be used to sign all other certificates within a system. In other words, Root CA is an issuer of *node*, *client* and *admin* certificates. We start with changing certificates' directory permissions to *600*. Then we can create config file *cert*config_:

```
[ req ]
prompt             = no
distinguished_name = demo-ca

[ demo-ca ]
commonName = demo-root-ca
countryName = DE
organizationName = Demo-SearchGuard
organizationalUnitName = demo-cluster
stateOrProvinceName = Brandenburg
```

Based on this, Root CA can be generated with a command:

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out root-ca.pem -days 365 -nodes -config
cert_config
```

The whole certificate can be viewed with a command:

```
openssl x509 -in root-ca.pem -text -noout
```

or the following command just to present subject:

```
openssl x509 -in root-ca.pem -subject -noout
```

Generating Node Certificates

Next, we generate node certificate issued by the Root CA. We start with a similar config file (country and organisational unit fields may be copied) which additionally contains:

- Extended Key Usage (EKU)
- Service Alternative Name (SAN entries) that may be used for hostname verification

```
[ req ]
prompt             = no
distinguished_name = node
req_extensions     = req_ext

[ node ]
commonName = demo-node
countryName = DE
organizationName = Demo-SearchGuard
organizationalUnitName = demo-cluster
stateOrProvinceName = Brandenburg

[ req_ext ]
subjectAltName = @alt_names
extendedKeyUsage = serverAuth,clientAuth

[alt_names]
DNS.1 = localhost
IP.1 = 127.0.0.1
```

Then the following two commands will create a certificate as specified and issued by Root CA:

```
openssl req \
  -newkey rsa:4096 \
  -keyout key.pem \
  -out node-cert.csr \
  -days 365 \
  -nodes \
  -config <<node-config-file>>

openssl x509 \
  -req \
  -in node-cert.csr \
  -CA <<root-ca-certificate>>.pem \
  -CAkey <<root-ca-key>>/.pem \
  -CAcreateserial \
  -out node-cert.pem \
  -extensions req_ext \
  -extfile node_cert_config \
  -days 365
```

At this stage it is worth to check if an issuer is correctly set:

```
$ openssl x509 -in node-cert.pem -issuer -noout
issuer= /CN=demo-root-ca/C=DE/O=Demo-SearchGuard/OU=demo-cluster/ST=Brandenburg
```

X509 extensions part of a certificate should look like this:

```
X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:localhost, IP Address:127.0.0.1
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
```

This means we have successfully added Subject Alternative Names and Extended Usages that are necessary for SearchGuard.

Generating Admin Certificate

Generating admin certificate can be done exactly the same way and only *commonName* has to be modified.

Configuring SearchGuard

We ensure that the following keys in *elasticsearch.yml* target to certificate and key files we have created before:

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

and

```
searchguard.ssl.http.pemcert_filepath: node-cert.pem
searchguard.ssl.http.pemkey_filepath: node-key.pem
searchguard.ssl.http.pemtrustedcas_filepath: root-ca.pem
```

Note that Elasticsearch requires key and certificate files to be located within its *config* directory. Make sure to change directory permissions to *600.*

We also define which certificate (Distinguished Name of subject) will be used for admin requests:

```
searchguard.authcz.admin_dn:
 - "ST=Brandenburg,OU=demo-cluster,O=Demo-SearchGuard,C=DE,CN=demo-admin"
```

Demo

We restart the cluster and run *sgadmin*, to see if it works:

```
./sgadmin.sh -w
  -cacert <<root-ca>>.pem
  -cert <<admin-cert>>.pem
  -key <<admin-key>>.pem
  -icl
```

The *-w* option will print *whoami* of admin user, so it may be good for testing.

The command results with a response:

```
Connected as ST=DE,OU=demo-cluster,O=Demo-SearchGuard,C=DE,CN=demo-admin
{
  "whoami" : {
    "dn" : "ST=DE,OU=demo-cluster,O=Demo-SearchGuard,C=DE,CN=demo-admin",
    "is_admin" : true,
    "is_authenticated" : true,
    "is_node_certificate_request" : false
  }
}
```

which assures our setup works correctly.

Where to go next:

- Read about the Search Guard TLS configuration options in our documentation
- If you have problems configuring TLS, read our TLS troubleshooting guide or ask for help on our forum.

Image: shutterstock / Profit Image

This article was published on the Search Guard blog: https://search-guard.com/blog/elasticsearch-tls-certificates-openssl/
Search Guard is Security and Alerting for Elasticsearch and Kibana — search-guard.com · Start a free trial