Guy Bruneau – guybruneau@outlook.com @guybruneau

Elasticsearch TLS Encryption HTTPS Communication

Table of Contents

Guy Bruneau – guybruneau@outlook.com @guybruneau

Introduction

This document is a compilation of the various references listed in the PDF document (here as well), it documents step-by-step the processes I have been using to setup TLS encryption within my test network. Using Elasticsearch **elasticsearch-certutil** tool in CA mode, it simplifies the creation of certificates and generates a new certificate authority (CA) to use within the local ELK infrastructure. These steps provide secure communication for Linux and Windows between Elasticsearch nodes, Kibana, logstash and the various beats.

Install CentOS Server

These are the steps to enable TLS encryption and HTTPS communication in Elasticsearch with CentOS (has not been tested with other Linux system but should work) with one or multiple nodes using Elasticsearch self-signed certificate authority. When more than one node is in user (multiple nodes, minimum of 3 suggested), it will be indicated to copy the master certificates via scp to the other nodes.

The following steps assumes you have CentOS and Elasticsearch/Kibana installed on the primary server. If you have not already built the server, I suggest for testing you assign a minimum of 250GB to Elasticsearch partition:

- /var/lib/elasticsearch → Default Elasticsearch database location
- /dev/sdb1 → Suggests minimum of 250 GB for elasticsearch database

Login CentOS Server

Configure Elasticsearch repo

$ sudo su -

Configure yum repo

# vi /etc/yum.repo.d/elasticsearch.repo and add the following:

```
[elasticsearch-7.x]

name=Elasticsearch repository for 7.x packages

baseurl=https://artifacts.elastic.co/packages/7.x/yum

gpgcheck=1

gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch

enabled=1
```

autorefresh=1

type=rpm-md

- Install elasticsearch (yum -y install elasticsearch)

- Install kibana (yum -y install kibana)

Note: Make these directories on all the nodes (master-1, master-2, master-3, etc) to copy the certificates to it.

```
# mkdir -p /etc/elasticsearch/certs

# mkdir -p /etc/kibana/certs
```

TLS X.509 Certificate Authority

TLS requires a X.509 certificate to perform encryption and authentication between server(s) and clients to communicate between the Elasticsearch cluster applications.

```
# /usr/share/elasticsearch/bin/elasticsearch-certutil ca  (assign a password to CA)

Please enter the desired output file [elastic-stack-ca.p12]: -> [ENTER]

Enter password for elastic-stack-ca.p12 :
```

Note: Certificate saved in: /usr/share/elasticsearch/elastic-stack-ca.p12 -> Default P12

Important: If ELK stack has more than one node, scp *elastic-stack-ca.p12* to all the remaining nodes.

```
# scp /usr/share/elasticsearch/elastic-stack-ca.p12 root@node2:/usr/share/elasticsearch/
```

Generate Certificates for Kibana and other Elastic Applications

This activity is performed from the master node only, the primary elasticsearch master (master-1) where Kibana is installed. List all the master nodes (master-2, master-3, etc) and the certificate for each master will be processed here.

Create a file called instance.yml and add all the servers/clients and beats in it to create all the required certificate. In this stage, you list all the current devices you have setup or planning to setup of working certificates. Make sure you use the correct DNS server names; you can install later the master CA certificate in the browser(s).

It is possible to add more nodes/beats later using these steps

```
# mkdir /opt/certs

# vi /opt/certs/instance.yml

instances:

  - name: "master-1"

    ip:

      - "127.0.0.1"

      - "192.168.154.50"

    dns:

      - "elk"

      - "elk.network.ca"

  - name: "master-2"

    ip:

      - "127.0.0.1"

      - "192.168.154.51"

    dns:

      - "elk"

      - "elk.network.ca"

  - name: "kibana-1"

    ip:

      - "127.0.0.1"

      - "192.168.154.50"

    dns:

      - "elk"
      - "elk.network.ca"
  - name: 'beat'
    ip:
      - "127.0.0.1"
      - "192.168.154.50"
    dns:
      - "beat"
  - name: "logstash"
    ip:
      - "127.0.0.1"
      - "192.168.154.50"
    dns:
      - "logstash"
```

Note: Save the configuration according to your devices and network.

Generate all the certificates

This stage creates all the certificates for all the devices previously enumerated above. It is important that option --*keep-ca-key* is included here. If later you require to generate more certificates for additional nodes/beats (devices), you will need both the **ca.crt** and **ca.key.** This is covered later in section Adding a new Host after Initial Setup.

```
# /usr/share/elasticsearch/bin/elasticsearch-certutil cert ca --keep-ca-key --
pem --in /opt/certs/instance.yml --out /opt/certs/certs.zip
```

Distribute cert.zip File to all other Devices

Note: Send a copy of the certs.zip to all other nodes (master-2, master-3, etc) as well as any clients/servers (filebeat, logstash, metricbeat, etc) that will be running an Elastic application.

Replace *node2* with the name or IP associated with the device requiring a copy of the certificates:

```
# scp -r /opt/certs root@node2:/opt
```

Setup Certificate on Nodes and Clients

Configure each nodes and clients using the following steps

```
# cd /opt/certs

# unzip certs.zip

# cp ca/* /etc/kibana/certs

# cp ca/* /etc/elasticsearch/certs

# cp kibana/* /etc/kibana/certs

# cp master-1/* /etc/elasticsearch/certs

# chown -R elasticsearch:elasticsearch /etc/elasticsearch/certs

# chown -R kibana:kibana /etc/kibana/certs

# chmod 755 /opt/certs

# chmod 444 /opt/certs/beat/*

# chmod 444 /opt/certs/ca/*

# chmod 444 /opt/certs/logstash/*

# chmod 444 /opt/certs/kibana/*
```

Copy ca.crt to be trusted by CentOS Server

If a server is going to be running any Elasticsearch applications other than the elasticsearch and kibana service, this will be requiring to authenticate with the server (i.e. elastic-agent, filebeat, etc)

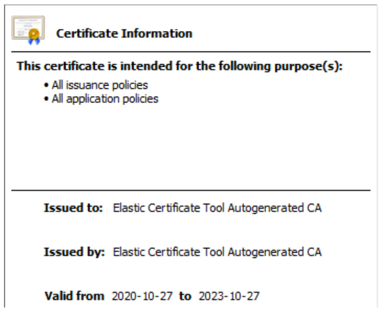
```
# cp /opt/certs/ca/ca.crt /etc/pki/ca-trust/source/anchors/elastic-ca.crt

# update-ca-trust force-enable

# update-ca-trust extract
```

Note: Send a copy to the other nodes/servers/clients as it will be needed to complete installation of agents. You can use WinSCP to transfer to Windows systems.

Install ca.crt to be trusted by Windows/Linux/MacOS Browser

Note: Import the CA certificate located in **/opt/certs/ca/ca.crt** in Windows under *Trusted Root Certification Authority* to authenticate the Kibana webserver.

After the certificate is copied to the Windows server/host, double-click on it to install it in the Local Machine and place it in the Certificate store in the Trusted Root Certification Authorities .


Certificate Information

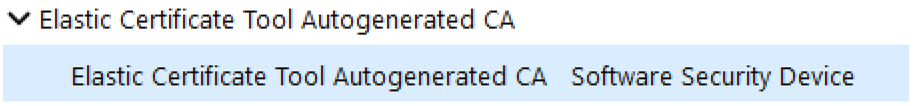
This certificate is intended for the following purpose(s):
- All issuance policies
- All application policies

Issued to: Elastic Certificate Tool Autogenerated CA

Issued by: Elastic Certificate Tool Autogenerated CA

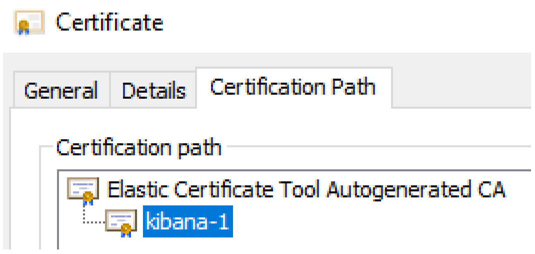
Valid from 2020-10-27 to 2023-10-27

Under Firefox, select Tools → Options → Privacy & Security → Certificates → View Certificates → Authorities → Import the certificate


Elastic Certificate Tool Autogenerated CA

Elastic Certificate Tool Autogenerated CA Software Security Device

After the CA has been installed on the host, opening the Kibana certificate shows the client recognize the Kibana server certificate as valid.



Elastic Certificate Tool Autogenerated CA

kibana-1

If you plan to use the elastic-agent, this certificate must be installed on each host. The elastic-agent uses the CA to authenticate via Elasticsearch and Kibana to send the logs to ELK.

Setup Remaining Nodes

A copy of the certificates was previously copied via scp to the other nodes in the cluster which will require to be configured using these steps:

```
# cd /opt/certs

# unzip certs.zip
```

If more than one Kibana webserver exist, do this:

```
# cp ca/* /etc/kibana/certs
# cp kibana-2/* /etc/kibana/certs
# chown -R kibana:kibana /etc/kibana/certs
```

Just the Elasticsearch Service

```
# cp ca/* /etc/elasticsearch/certs
# cp master-2/* /etc/elasticsearch/certs  (Change master-2 to correct master certificate)
# chown -R elasticsearch:elasticsearch /etc/elasticsearch/certs
# chmod 755 /opt/certs
# chmod 444 /opt/certs/beat/*
# chmod 444 /opt/certs/ca/*
# chmod 444 /opt/certs/logstash/*
# chmod 444 /opt/certs/kibana/*
# cp /opt/certs/ca/ca.crt /etc/pki/ca-trust/source/anchors/elastic-ca.crt
# update-ca-trust force-enable
# update-ca-trust extract
```

Certificates such as beat and logstash are save in /opt/certs and can be copied and used where needed.

Configure Security in elasticsearch.yml

Note: Leave the http.ssl commented out until the username/password have been generated (next step).

For each node, change master-1 to the correct master certificate (master-2, master-3, etc)

```
action.auto_create_index: true
# discovery.seed_hosts: ["192.168.2.12","192.168.2.13","192.168.2.14"]
```

Guy Bruneau – guybruneau@outlook.com @guybruneau

```
# Enable this when starting with only one node or

# when setting up a cluster. Disable discovery.seed_hosts

# When done and ready to enable the cluster, disable discovery.type and

#enable discovery.seed_hosts

discovery.type: single-node


# https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

# Node Roles

node.roles: [ master, data, ingest, ml, ingest, transform , remote_cluster_client ]

# ml: false (except for master -> true)

node.attr.zone: primary

node.attr.zone: ecluster  (for other nodes)


# Enable Snapshot and Restore (chown elasticsearch:elasticsearch /opt/backup)

# path.repo: /opt/backup


# Enable Elastic audit logging

xpack.security.audit.enabled: true


# Exclude the following event type

xpack.security.audit.logfile.events.exclude: access_granted, run_as_granted


# TLS/SSL Reporting (Watcher for Kibana Only)

#xpack.http.ssl.key: certs/master-1.key

#xpack.http.ssl.certificate: certs/master-1.crt

#xpack.http.ssl.certificate_authorities: certs/ca.crt

#xpack.http.ssl.verification_mode: certificate


# TLS/SSL for the HTTP (Rest) interface
```

```
xpack.security.http.ssl.enabled: true

xpack.security.http.ssl.key: certs/master-1.key

xpack.security.http.ssl.certificate: certs/master-1.crt

xpack.security.http.ssl.certificate_authorities: certs/ca.crt

xpack.security.http.ssl.verification_mode: certificate


# General transport TLS/SSL security settings

xpack.security.enabled: true

xpack.security.transport.ssl.enabled: true

xpack.security.transport.ssl.key: certs/master-1.key

xpack.security.transport.ssl.certificate: certs/master-1.crt

xpack.security.transport.ssl.certificate_authorities: certs/ca.crt

xpack.security.transport.ssl.verification_mode: certificate


xpack.security.authc.api_key.enabled: true
```

Start Elasticsearch and Watch the Logs

```
# systemctl start elasticsearch

# systemctl enable elasticsearch

# systemctl status elasticsearch
```

Watch Elasticsearch activity logs (replace *network* with your elasticsearch cluster name -> cluster.name: *network*)

# tail -f /var/log/elasticsearch/*network*.log

Elasticsearch - Generate Passwords

Once elasticsearch service is started (the first node with Kibana), it is time to generate the default passwords for the cluster. Execute this command and follow the instructions:

```
# /usr/share/elasticsearch/bin/elasticsearch-setup-passwords auto
```

This step generates random passwords for the various internal stack user built-in accounts. You can alternatively skip the auto parameter to manually define your passwords using the interactive parameter. *Keep track of these passwords*, we will need them to setup Kibana and login Elasticsearch.

Save the Password List

This will look something like this:

```
Changed password for user apm_system

PASSWORD apm_system = csG5r8QpxWVYofJIfV16


Changed password for user kibana_system

PASSWORD kibana_system = nW2gbLXtESc5OhbTe45Q


Changed password for user kibana

PASSWORD kibana = nW2gbLXtESc5OhbTe45Q


Changed password for user logstash_system

PASSWORD logstash_system = c9BOhIA9oQFryKiZdOxa


Changed password for user beats_system

PASSWORD beats_system = CsdmxJsysSDScjhm4s8Z


Changed password for user remote_monitoring_user

PASSWORD remote_monitoring_user = p7R4ofIhqsQASSd1dAjC
```

Changed password for user elastic

PASSWORD elastic = edgPK6WbYKl5uIvFHGYX

Enable http.ssl in elasticsearch.yml on all master-*.*

Edit *elasticsearch.yml* and add and enable the following in the configuration file:

```
# TLS/SSL Reporting (Watcher)
xpack.http.ssl.key: certs/master-1.key
xpack.http.ssl.certificate: certs/master-1.crt
xpack.http.ssl.certificate_authorities: certs/ca.crt
xpack.http.ssl.verification_mode: certificate
```

Save the file and restart elasticsearch service

```
# systemctl restart elasticsearch
# systemctl status elasticsearch
```

Configure kibana.yml

It is time to configure Kibana to allow authentication with Elasticsearch. Using the previously generated passwords, edit and copy/paste the *kibana_system* password in the configuration file.

```
# vi /etc/kibana/kibana.yml
elasticsearch.username: "kibana_system"
elasticsearch.password: "nW2gbLXtESc5OhbTe45Q"
```

The URLs of the Elasticsearch instances to use for all your queries. Enable this line with the correct hostname or IP address (note that SSL is enable):

```
elasticsearch.hosts: ["https://elastic:9200"]
```

Enables SSL and paths to the PEM-format SSL certificate and SSL key files, respectively. These settings enable SSL for outgoing requests from the Kibana server to the browser.

```
server.ssl.enabled: true
server.ssl.key: /etc/kibana/certs/kibana.key
server.ssl.certificate: /etc/kibana/certs/kibana.crt
elasticsearch.ssl.certificateAuthorities: ["/etc/kibana/certs/ca.crt"]
elasticsearch.ssl.verificationMode: certificate
```

Start Kibana:

```
# systemctl start kibana
# systemctl enable kibana
# systemctl status kibana
```

Configure Firewall Access (CentOS)

Note: Before you can login Kibana, you need to allow access to some ports on the server. Here is a list of firewall ports to add to server where necessary:

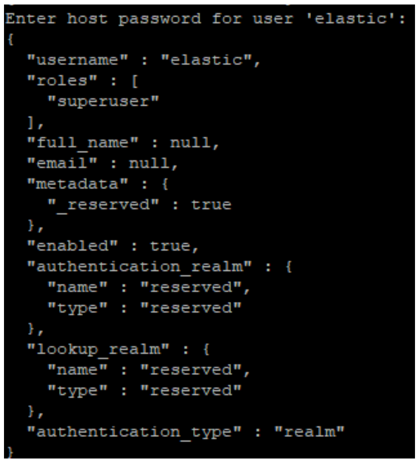
```
firewall-cmd --list-ports
firewall-cmd --permanent --add-port=9200/tcp   -> Elasticsearch
firewall-cmd --permanent --add-port=9300/tcp   -> Elasticsearch Cluster Access
firewall-cmd --permanent --add-port=5601/tcp   -> Kibana
firewall-cmd --permanent --add-port=3002/tcp  -> enterprise-search
firewall-cmd --reload
```

Test Elasticsearch Access and Accessing Kibana

Test to username/password to elasticsearch database. To do this test, user the password previously generated, you can change the password for elastic after you login. Where elastic can also be the IP address if so configured in elasticsearch.yml

```
# curl -u elastic 'https://elastic:9200/_xpack/security/_authenticate?pretty'
```

Enter host password for user 'elastic': edgPK6WbYKl5uIvFHGYX



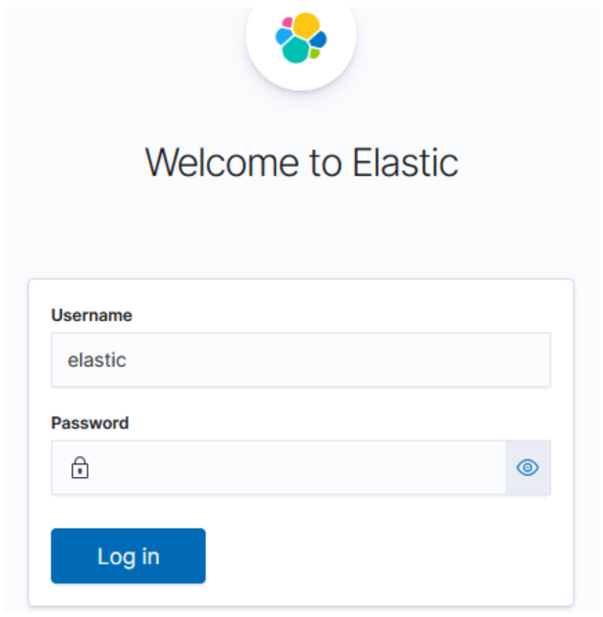
```
Enter host password for user 'elastic':
{
  "username" : "elastic",
  "roles" : [
    "superuser"
  ],
  "full_name" : null,
  "email" : null,
  "metadata" : {
    "_reserved" : true
  },
  "enabled" : true,
  "authentication_realm" : {
    "name" : "reserved",
    "type" : "reserved"
  },
  "lookup_realm" : {
    "name" : "reserved",
    "type" : "reserved"
  },
  "authentication_type" : "realm"
}
```

Login Kibana

Open a browser and login Kibana:

Login Kibana: https://elk:5601

Password: edgPK6WbYKl5uIvFHGYX



Welcome to Elastic

Username: elastic

Password:

Change elastic Account Password

If you want to change the elastic account password, now you can go to:

Stack Management -> Security -> Users

Guy Bruneau – guybruneau@outlook.com @guybruneau

Change the elastic default password (edgPK6WbYKl5uIvFHGYX) to something you prefer using

Note: Verify your new password for elastic works to login Kibana

```
# curl -u elastic 'https://elastic:9200/_xpack/security/_authenticate?pretty'
Enter host password for user 'elastic':
```

Adding a new Host after Initial Setup

This process is to generate additional certificates for either nodes or clients using the CA certificate and key. First is to add another directory under /opt not to overwrite the current keys:

```
# mkdir /opt/certs1
# cd /opt/certs1
```

Next step is to create instance.yml with the new server(s) or client(s) that need to be added to the cluster:

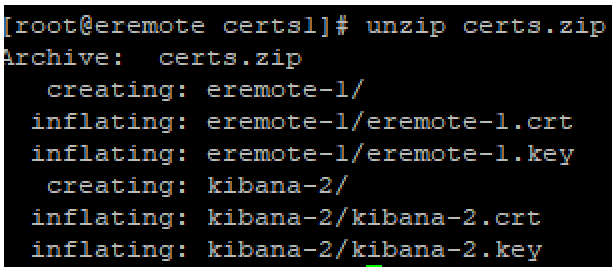
```
# vi instance.yml
```

```
instances:
  - name: 'eremote-1'
    ip:
      - "127.0.0.1"
      - "192.168.25.17"
    dns:
      - "eremote"
      - "eremote.network.ca"
  - name: 'kibana-2'
    ip:
      - "127.0.0.1"
      - "192.168.25.17"
    dns:
      - "eremote"
      - "eremote.network.ca"
```

In the initial setup, the *ca.crt* and *ca.key* was saved in order to be use to create additional certificates as needed. Execute the following command to create the new certificates:

```
# /usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca-cert
/opt/certs/ca/ca.crt --ca-key /opt/certs/ca/ca.key --pem --in
/opt/certs1/instance.yml --out /opt/certs1/certs.zip
```

Unzip the certs.zip File



```
[root@eremote certs1]# unzip certs.zip
Archive:  certs.zip
   creating: eremote-1/
  inflating: eremote-1/eremote-1.crt
  inflating: eremote-1/eremote-1.key
   creating: kibana-2/
  inflating: kibana-2/kibana-2.crt
  inflating: kibana-2/kibana-2.key
```

Proceed using the previous steps to configure the elasticsearch and Kibana certificates on this new node.

```
# cd /opt/certs

# mkdir -p /etc/kibana/certs

# mkdir -p /etc/elasticsearch/certs

# cp ca/* /etc/kibana/certs

# cp ca/* /etc/elasticsearch/certs

# cd /opt/certs1

# unzip certs.zip

# cp kibana-2/* /etc/kibana/certs

# cp eremote-1/* /etc/elasticsearch/certs

# chown -R elasticsearch:elasticsearch /etc/elasticsearch/certs

# chown -R kibana:kibana /etc/kibana/certs

# chmod 755 /opt/certs

# chmod 444 /opt/certs/beat/*
# chmod 444 /opt/certs/ca/*
# chmod 444 /opt/certs/logstash/*
# chmod 444 /opt/certs/kibana-*/*
```

Copy ca.crt to be trusted by CentOS Server

If a server is going to be running any Elasticsearch applications other than the Elasticsearch and kibana service, this will be requiring to authenticate with the server (i.e. elastic-agent, filebeat, etc)

```
# cp /opt/certs/ca/ca.crt /etc/pki/ca-trust/source/anchors/elastic-ca.crt
# update-ca-trust force-enable
# update-ca-trust extract
```

Complete Server Configuration - Starting Services

At this point, we are ready to start the new server services.

```
# systemctl start elasticsearch
# systemctl start kibana
# systemctl enable elasticsearch
# systemctl enable kibana
# systemctl status elasticsearch
# systemctl status kibana
```

Note: If Kibana is enabled on this new server, open the appropriate ports to the firewall to allow access.

Example of Beat SSL Configuration

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["https://elastic:9200"]
  ssl.certificate_authorities: ["/opt/certs/ca/ca.crt"]
  ssl.certificate: "/opt/certs/beat/beat.crt"
  ssl.key: "/opt/certs/beat/beat.key"
# This pipeline applies only to packetbeat.yml
  # pipeline: geoip-info

  # Optional protocol and basic auth credentials.
  #protocol: "https"
  username: "elastic"
  password: "elastic"
```

Example of Logstash SSL Configuration

```
output {
  elasticsearch {
    hosts => ["https://elastic:9200"]
    cacert => '/opt/certs/ca/ca.crt'
    user => "elastic"
    password => "elastic"
```

References

[1] https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-tls.html

[2] https://www.elastic.co/guide/en/elasticsearch/reference/current/certutil.html

[3] https://www.elastic.co/blog/configuring-ssl-tls-and-https-to-secure-elasticsearch-kibana-beats-and-logstash

[4] https://www.elastic.co/guide/en/elasticsearch/reference/master/encrypting-communications-certificates.html

[5] https://www.elastic.co/blog/elasticsearch-security-configure-tls-ssl-pki-authentication

[6] https://techexpert.tips/elasticsearch/elasticsearch-enable-tls-https/