Enabling Security

The following steps enable security for the cluster, wire-level encryption for the platform and ecosystem components, authentication for all network-based connections, and optionally, data at rest encryption on the cluster.

When you set up a cluster, run the `configure.sh (../ReferenceGuide/configure.sh.html)` script on each node that you want to add to the cluster. After you enable security, review the System Behavior Changes After Enabling Security (System-Behavior-Changes.html).

Basic Procedure

To enable security for the cluster, follow these steps in order:

1. If the cluster is running, shut it down (../AdministratorGuide/ShutDownCluster.html).
2. Delete the `ssl_keystore` , `ssl_truststore` , `ssl_truststore.p12` , and `ssl_truststore.pem` files on a machine where wire-level security is not enabled because the `configure.sh` script fails if you already have these files in the directory.

   Note: The `ssl_keystore` , `ssl_keystore.p12` , `ssl_keystore.pem` , `ssl_truststore` , `ssl_truststore.p12` , and `ssl_truststore.pem` files are generated during installation of the Web server even if you did not enable security.

   For example, to delete the files, run the following commands:

   ```
   cd /opt/mapr/conf
   /bin/rm -f ssl_keystore ssl_keystore.p12 ssl_keystore.pem ssl_truststore ssl_truststore.p12
    ssl_truststore.pem
   ```

   If you are re-running the script due to an invocation error from a previous run, the `cldb.key` and `maprserverticket` files may also have been created. Delete these files because the script fails if you already have these files in the directory. For example, run the following command to delete these files:

   ```
   cd /opt/mapr/conf/bin/rm -f cldb.key maprserverticket ssl_keystore ssl_keystore.p12 ssl_key
   store.pem ssl_truststore ssl_truststore.p12 ssl_truststore.pem
   ```

3. Run the `configure.sh` script with the `-secure -genkeys -dare` options on the first CLDB node in your cluster:

   ```
   /opt/mapr/server/configure.sh -secure -dare -genkeys -Z <Zookeeper_node_list> -C <CLDB_node
   _list> -N <cluster_name>
   ```

   where both `<Zookeeper_node_list>` and `<CLDB_node_list>` have the form `hostname[:port_no] [,hostname[:port_no]...]` and `-N <cluster_name>` specifies the cluster name. The `-dare` option is required only if you wish to enable data at rest encryption at the cluster-level.

   Important: You must run `configure.sh -dare -genkeys` only *once* on one CLDB node, since the resulting files must be copied to other nodes.

   This command generates nine files in the `/opt/mapr/conf` directory:

   - `cldb.key`

- dare.master.key
- maprserverticket
- ssl_keystore
- ssl_keystore.p12
- ssl_keystore.pem
- ssl_truststore
- ssl_truststore.p12
- ssl_truststore.pem

Note: The dare.master.key file is generated only if data at rest encryption is enabled on the cluster.

4. Copy the cldb.key file to any node that has the CLDB or Zookeeper service installed and copy the dare.master.key file to all the other CLDB nodes on the cluster.

   Note: Copy the dare.master.key file only if you are enabling data at rest encryption on the cluster.

5. Copy the maprserverticket , ssl_keystore , ssl_keystore.p12 , ssl_keystore.pem , ssl_truststore , ssl_truststore.p12 , and ssl_truststore.pem files to the /opt/mapr/conf directory of *every* node in the cluster.

6. Verify that the files from the previous step are owned by the user that runs cluster services. This user is mapr by default. Also, the maprserverticket , ssl_keystore , ssl_keystore.p12 , and ssl_keystore.pem files must have their UNIX permission-mode bits set to 600 , and the ssl_truststore , ssl_truststore.p12 , and ssl_truststore.pem files must be readable to all users.

7. Run configure.sh on each existing node in the cluster using the same arguments as in Step 3 but without the -genkeys option.

   ```
   /opt/mapr/server/configure.sh -secure -dare -Z <Zookeeper_node_list> -C <CLDB_node_list> -N
    <cluster_name>
   ```

   The -secure option indicates that security must be enabled on the node where the command is run and the -dare option indicates that data at rest encryption must be enabled on the node and must be specified only if it was specified in Step 3.

   Important: You must also do this on any nodes that you add to the cluster in the future.

8. Copy the ssl_truststore , ssl_truststore.p12 , and ssl_truststore.pem files to any client nodes outside the cluster.

   Important: If you run configure.sh -secure on a node *before* you copy the necessary files to that node, the command fails.

9. Log in as the mapr superuser using the maprlogin (ThemaprloginUtility.html) command: maprlogin password (in this command, password is literal text).

10. If clients will connect to multiple secure clusters, merge the ssl_truststore files with the /opt/mapr/server/manageSSLKeys.sh tool. See Setting Up the Client (../AdvancedInstallation/SettingUptheClient.html) for more information on MapR clients.

Advanced Procedure

In certain situations, you may opt for variant of the basic procedure. Such situations include, but are not limited to the following:

- You are running the script on a host that is configured without a domain name.
- You have a cluster where all the machines do not have the same domain name.
- You wish to import your own custom certificates instead of the self-signed certificates generated by the `configure.sh` utility.

Running on Hosts with no Configured Domain Name

When used without the `-certdomain` argument, the `configure.sh` script discovers the domain name of the node on which it is being executed using the `hostname -d` command and then creates a 100-year self-signed certificate using the PKCS#1 v1.5 with SHA-512 hash function (SHA512withRSA) with a wildcard certificate with the common name (CN) `*.<domain>` . For example, if `hostname -d` returns the domain name `mycompany.com` , then the CN of the certificate is `*.mycompany.com` . This certificate works for all machines within the `mycompany.com` domain and can therefore be copied to all cluster nodes as specified in Step 5 in the Basic Procedure (Enable-Security.html#basic).

Certificate generation fails if the host that you are running the script from is configured without a domain name. To fix this, modify your machine configuration so that `hostname -d` returns a non-empty string and then run the `configure.sh` script.

Alternatively, re-run the script with the `-certdomain` option as shown in Step 3 of the Basic Procedure (Enable-Security.html#basic):

```
/opt/mapr/server/configure.sh -secure -genkeys -certdomain <domain_name> -Z <Zookeeper_node_list>
-C <CLDB_node_list> [ -N <cluster_name> ]
```

Securing Clusters with Multiple Domain Names

Generally, all machines within a cluster should belong to the same domain. In the unusual case where you have a cluster with different machines belonging to different domains, applications that perform hostname verification can fail if you run the `configure.sh` script (as described in Step 3 of the Basic Procedure (Enable-Security.html#basic)) to generate a single-domain wildcard certificate. In this case, you must provide your own multi-domain wildcard certificate and import the custom certificate into the keystore as described in Importing a Certificate Authority Signed (CA Signed) SSL Certificate Into a MapR Cluster (https://mapr.com/support/s/article/Importing-a-Certificate-Authority-Signed-CA-Signed-SSL-Certificate-Into-a-MapR-Clusters).

Using Custom Certificates

To import your own custom certificates into the keystore instead of using the self-signed certificates, see Importing a Certificate Authority Signed (CA Signed) SSL Certificate Into a MapR Cluster (https://mapr.com/support/s/article/Importing-a-Certificate-Authority-Signed-CA-Signed-SSL-Certificate-Into-a-MapR-Clusters).