How to Encrypt Replication Traffic in MariaDB Galera Cluster on Ubuntu

Previously, we discussed how to set up MariaDB Galera multi-master cluster on Ubuntu. By default, replication traffic in MariaDB Galera cluster is unencrypted. This tutorial will be showing you how to Enable TLS encryption so nodes in the cluster can communicate with each other securely over the public Internet. There are 2 types of replication traffic in Galera cluster:

- State transfer
- Write-set replication

What is State Transfer?

State transfer is the process of copying a database from one node to another. The node sending data is called a donor. The node receiving data is called a joiner. There are two kinds of state transfer:

- Snapshot state transfer (SST): copy entire database
- Incremental state transfer (IST): copy data modifications only

SST, aka node provisioning, is used when a new node joins the cluster because it contains no data. IST is used when a disconnected node rejoins the cluster.

Understanding SST

There are 5 SST methods:

- wsrep_sst_rsync: default method
- wsrep_sst_mysqldump: slowest method
- wsrep_sst_mariabackup: non-blocking method
- wsrep_sst_xtrabackup
- wsrep_sst_xtrabackup-v2

Rsync is the default SST method and it's the fastest. It's recommended that you do not use **xtrabackup** or **xtrabackup-v2** method because they don't work with MariaDB 10.3 + and they don't support GTID and data at rest encryption.

Personally, I prefer **mariabackup** because it doesn't block the donor node during the state transfer, which is an advantage inheritated from the xtrabackup method. Rsync and mysqldump require the donor made read-only during the transfer, via the `flush tables with read lock` command. The mariabackup SST method is a fork of the xtrabackup-v2 SST method and it relies on the `mariabackup` and `socat` command-line utility.

To set **mariabackup** as the SST method, first install `socat` and `mariabackup` on each node in the cluster and the joiner node.

```
sudo apt install socat
```

If you installed the latest version of MariaDB from mariadb.org repository, then you need to install `mariabackup` with the following command.

```
sudo apt install mariadb-backup
```

Then add the following two lines in the `[galera]` unit of the `/etc/mysql/mariadb.conf.d/60-galera.cnf` file, on each node in the cluster and the joiner node.

```
[galera]
...
...
wsrep_sst_method = mariabackup
wsrep_sst_auth = mariabackup:your_password
```

The first line specifies that `mariabackup` will be used for SST method and the second line defines the username and password, which is needed for authentication.

Next, log into MariaDB monitor on a node that's already in the cluster and create the database user and grant necessary permissions. (Note that you do not need to run the following commands on each node. The data will be replicated to other nodes automatically.)

```
create user 'mariabackup'@'localhost' identified by 'your_password';
grant reload, process, lock tables, replication client on *.* to 'mariabackup'@'localhost';
```

Flush privilege tables and exit;

```
flush privileges;


exit;
```

Restart MariaDB server on each node one by one and then restart the MariaDB server on the joiner node.

```
sudo systemctl restart mariadb
```

Encrypting Mariabackup SST

You can use openssl to create a self-signed TLS certificate, but I'm going to use the existing Let's Encrypt TLS certificate configured for my web server. To enable TLS encryption on the mariabackup SST, open the MariaDB main configuration file (`/etc/mysql/mariadb.conf.d/50-server.cnf` or `/etc/mysql/my.cnf`), and add the following lines **at the end of the file**. Replace the path name as necessary.

```
[sst]
encrypt=4
tkey=/etc/letsencrypt/live/linuxbabe.com/privkey.pem
tcert=/etc/letsencrypt/live/linuxbabe.com/cert.pem
tca=/etc/letsencrypt/live/linuxbabe.com/chain.pem
```

The `encrypt` variable has 5 possible values: 0, 1, 2, 3, 4. `0` means encryption is disabled. `1`, `2` and `3` doesn't work any more. Save and close the file. The `mysql` user needs permission to access the above SSL files, so you need to grant read permission with the following commands.

```
sudo setfacl -R -m "u:mysql:rx" /etc/letsencrypt/archive/ /etc/letsencrypt/live/
```

Note that you need to use the same TLS certificate and private key on all your nodes in the cluster and then restart the entire Galera Cluster for the changes to take effect.

Note that during SST, extra files in `/var/lib/mysql/` directory added by user will be deleted.

Understanding IST

IST is used when a disconnected node rejoins the cluster. All nodes in the cluster maintain a Galera cache (GCache), aka write-set cache, which contains the write sets committed by a node. When a disconnected node rejoins the cluster, it requests an incremental state transfer from a donor's GCache. IST does not lock the donor.

Obviously, IST is faster than SST, because it transfers modifications only. However, if the donor's GCache isn't big enough to store all the write sets the joiner needs, an SST will be initiated instead. To avoid SST, you can increase the size for GCache. The default value is 128MB, which you can check with the following command at the MariaDB monitor.

```
show variables like 'wsrep_provider_options'\G
```

GCache related values are as follows:

```
gcache.dir = /var/lib/mysql/; gcache.keep_pages_size = 0; gcache.mem_size = 0; gcache.name = /var/lib/mysql//galera.cache; gcache.page_size = 128M; gcache.recover = no; gcache.size = 128M;
```

The GCache file is located at `/var/lib/mysql/galera.cache` and its size is preallocated. To increase the size of GCache, you need to add the following line in `[mysqld]` unit of MariaDB configuration file.

```
wsrep_provider_options="gcache.size = 1G"
```

Replace 1G with your preferred cache size. Make sure your server has enough RAM. Setting a big GCache size with a small amount of RAM can cause your MariaDB server to fail to start. It's also recommended to set the `gcache.recover` parameter to `yes`, so a node can attempt to recover its Gcache on startup and continue serving IST to other joining nodes.

```
wsrep_provider_options="gcache.size = 1G; gcache.recover = yes"
```

Apply the same configuration on each node in the cluster and restart them one at a time.

Encrypting IST

I tried to use Let's Encrypt TLS certificate to encrypt IST traffic, but I always get the following error in the error log file when a node joins the cluster.

```
[ERROR] WSREP: handshake with remote endpoint ssl://xx.xx.xx.xx:4567 failed:
asio.ssl:337047686: 'certificate verify failed' ( 337047686: 'error:1416F086:SSL
routines:tls_process_server_certificate:certificate verify failed')
```

So I created a self-signed server certificate and private key. The steps are as follows.

Create a directory to hold SSL files.

```
sudo mkdir /etc/ssl/mysql/
```

Change directory.

```
cd /etc/ssl/mysql/
```

Generate the CA key file:

```
sudo openssl genrsa 2048 > ca-key.pem
```

Generate the CA certificate file. We use a very long expiration day (100 years) to avoid shutting down the cluster.

```
sudo openssl req -new -x509 -nodes -days 36500 -key ca-key.pem -out ca.pem
```

You will need to answer a few questions. I just entered the country code and the organization name.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example CA
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Generate the server key file. You will also need to answer a few questions and enter a passphrase to protect the key.

```
sudo openssl req -newkey rsa:2048 -days 36500 -nodes -keyout server-key.pem -out server-req.pem
```

Remove the passphrase.

```
sudo openssl rsa -in server-key.pem -out server-key.pem
```

Generate the server certificate file:

```
openssl x509 -req -in server-req.pem -days 36500 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem
```

Then copy all the files to other nodes in the cluster with `scp` utility and it's recommended to store them in the same directory (`/etc/ssl/mysql/`). Also make sure the mysql user has permission to read all the SSL files.

```
sudo chown mysql:mysql /etc/ssl/mysql/ -R
sudo chmod 400 /etc/ssl/mysql/*
sudo chmod 700 /etc/ssl/mysql/
```

After that, open the MariaDB main configuration file (`/etc/mysql/mariadb.conf.d/50-server.cnf` or `/etc/mysql/my.cnf`), and add the following lines in the `[mysqld]` unit.

```
wsrep_provider_options="socket.ssl_key=/etc/ssl/mysql/server-key.pem;socket.ssl_cert=/etc/ssl/mysql/server-cert.pem;socket.ssl_ca=/etc/ssl/mysql/ca.pem"
```

If you have other wsrep provider options, they need to be combined together like below.

```
wsrep_provider_options="gcache.size = 1G; gcache.recover = yes;socket.ssl_key=/etc/ssl/mysql/server-key.pem;socket.ssl_cert=/etc/ssl/mysql/server-cert.pem;socket.ssl_ca=/etc/ssl/mysql/ca.pem"
```

Configure each node in the cluster and restart the entire cluster for the changes to take effect. If everything goes well, your node will be able to join the cluster and you can see message like below in your MariaDB log (`sudo journalctl -eu mariadb`), which indicates IST is encrypted. (IST uses TCP port 4568.)

```
2019-03-19 14:40:03 1 [Note] WSREP: IST sender using ssl
2019-03-19 14:40:03 0 [Note] WSREP: async IST sender starting to serve ssl://xx.xx.xx.xx:4568
```

```
sending 166613-167529
```

Encrypting Write-set Replication Traffic

Write-set replication is the normal synchronous replication whereby a node sends data modifications to all other nodes. Actually, write-set replication uses the same encryption machenism as IST. If you enabled TLS encryption for IST, write-set replication is also encrypted and you will see messages like below in MariaDB log (`sudo journalctl -eu mariadb`), which indicates write-set replication is secured with TLS. (Write-set replication uses TCP port 4567.)

```
SSL handshake successful, remote endpoint ssl://xx.xx.xx.xx:4567 local endpoint
ssl://xx.xx.xx.xx:37118 cipher: TLS_AES_256_GCM_SHA384 compression: none
```

Revision #1
Created 11 July 2021 19:11:53 by Admin
Updated 11 July 2021 19:12:13 by Admin