永顺 RP 3.8k 发布于 后台开发
2016-12-15 发布

MySQL 使用 SSL 连接(附 Docker 例子)

查看是否支持 SSL

首先在 MySQL 上执行如下命令, 查询是否 MySQL 支持 SSL:

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
1 row in set (0.02 sec)
```

当 `have_ssl` 为 `YES` 时, 表示此时 MySQL 服务已经支持 SSL 了. 如果是 `DESABLE`, 则需要在启动 MySQL 服务时, 使能 SSL 功能.

使用 OpenSSL 创建 SSL 证书和私钥

首先我们需要使用 openssl 来创建服务器端的证书和私钥. 我使用的 openssl 版本为:

```
>>> /usr/local/Cellar/openssl/1.0.2j/bin/openssl version
OpenSSL 1.0.2j  26 Sep 2016
```

新建一个 **~/temp/cert** 目录, 用于存放生成的证书和私钥

```
mkdir ~/temp/cert
cd ~/temp/cert
```

创建 CA 私钥和 CA 证书

然后, 我们先来生成一个 CA 私钥:

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

当有了一个 CA 私钥, 我们接下来就可以使用这个私钥生成一个新的数字证书:

```
openssl req -sha1 -new -x509 -nodes -days 3650 -key ca-key.pem > ca-cert.pem
```

执行这个命令时, 会需要填写一些问题, 随便填写就可以了. 例如:

```
>>> openssl req -sha1 -new -x509 -nodes -days 3650 -key ca-key.pem > ca-cert.pem

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]:Beijing
Locality Name (eg, city) []:Beijing
```

```
Common Name (e.g. server FQDN or YOUR name) []:xys
Email Address []:yongshun1228@gmail.com
```

执行上述命令后, 我们就有了一个 CA 私钥和一个 CA 证书.

创建服务器端的 RSA 私钥和数字证书

接着, 我们需要创建服务器端的私钥和一个证书请求文件, 命令如下:

```
openssl req -sha1 -newkey rsa:2048 -days 3650 -nodes -keyout server-key.pem > server-req.pem
```

上面这个命令会生成一个新的私钥(server-key.pem), 同时会使用这个新私钥来生成一个证书请求文件(server-req.pem).
上面这个命令同样需要回答几个问题, 随便填写即可. 不过需要注意的是, `A challenge password` 这一项需要为空.
即:

```
>>> openssl req -sha1 -newkey rsa:2048 -days 3650 -nodes -keyout server-key.pem > server-req.pem

Generating a 2048 bit RSA private key
................+++
..+++
writing new private key to 'server-key.pem'
-----
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]:Beijing
Locality Name (eg, city) []:Beijing
Organization Name (eg, company) [Internet Widgits Pty Ltd]:xys
Organizational Unit Name (eg, section) []:xys
Common Name (e.g. server FQDN or YOUR name) []:xys
Email Address []:yongshun1228@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
```

下一步, 我们需要将生成的私钥转换为 RSA 私钥文件格式:

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

最后一步, 我们需要使用原先生成的 CA 证书来生成一个服务器端的数字证书:

```
openssl x509 -sha1 -req -in server-req.pem -days 3650 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 > server-cert.pem
```

上面的命令会创建以服务器端的数字证书文件.

创建客户端的 RSA 私钥和数字证书

和服务器端所执行的命令类似, 我们也需要为客户端生成一个私钥和证书请求文件, 命令如下:

```
openssl req -sha1 -newkey rsa:2048 -days 3650 -nodes -keyout client-key.pem > client-req.pem
```

同样地, 我们需要将生成的私钥转换为 RSA 私钥文件格式:

```
openssl rsa -in client-key.pem -out client-key.pem
```

最后, 我们也需要为客户端创建一个数字证书:

```
openssl x509 -sha1 -req -in client-req.pem -days 3650 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 > client-cert.pem
```

使用工具创建证书与私钥

前面我们介绍了如何使用 OpenSSL 来创建 SSL 连接的私钥和证书文件, 现在我们来看一个更简单的方法.
在 MySQL 5.7 中, 提供了一个名为 `mysql_ssl_rsa_setup` 的工具, 通过它, 我们可以很方便地创建 SSL 连接所需要的各种文件:

```
mkdir ~/temp/cert
cd ~/temp/cert
mysql_ssl_rsa_setup --datadir ./
```

上面的命令中, `--datadir` 表示生成的文件的目录.

当执行了上述命令后, 也会生成八个文件:

```
ca-key.pem
ca.pem
client-cert.pem
client-key.pem
private_key.pem
public_key.pem
server-cert.pem
server-key.pem
```

这些文件和我们使用 OpenSSL 所创建的那八个文件的作用是一样的, 这里就不再详述了.

SSL 配置

在前面的步骤中, 我们已经生成了8个文件, 分别是:

- ca-cert.pem: CA 证书, 用于生成服务器端/客户端的数字证书.
- ca-key.pem: CA 私钥, 用于生成服务器端/客户端的数字证书.
- server-key.pem: 服务器端的 RSA 私钥
- server-req.pem: 服务器端的证书请求文件, 用于生成服务器端的数字证书.
- server-cert.pem: 服务器端的数字证书.
- client-key.pem: 客户端的 RSA 私钥
- client-req.pem: 客户端的证书请求文件, 用于生成客户端的数字证书.
- client-cert.pem: 客户端的数字证书.

接下来我们就需要分别配置服务器端和客户端.

服务器端配置

服务器端需要用到三个文件, 分别是: `CA 证书`, `服务器端的 RSA 私钥`, `服务器端的数字证书`, 我们需要在 `[mysqld]` 配置域下添加如下内容:

```
[mysqld]
ssl-ca=/etc/mysql/ca-cert.pem
ssl-cert=/etc/mysql/server-cert.pem
ssl-key=/etc/mysql/server-key.pem
```

接着我们还可以更改 `bind-address`, 使 MySQL 服务可以接收来自所有 ip 地址的客户端, 即:

```
bind-address = *
```

当配置好后, 我们需要重启 MySQL 服务, 使能配置.

```
GRANT ALL PRIVILEGES ON *.* TO 'ssl_test'@'%' IDENTIFIED BY 'ssl_test' REQUIRE SSL;
FLUSH PRIVILEGES;
```

当配置好后, 使用 root 登录 MySQL, 执行 `show variables like '%ssl%'` 语句会有如下输出:

```
mysql> show variables like '%ssl%';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| have_openssl  | YES             |
| have_ssl      | YES             |
| ssl_ca        | ca.pem          |
| ssl_capath    |                 |
| ssl_cert      | server-cert.pem |
| ssl_cipher    |                 |
| ssl_crl       |                 |
| ssl_crlpath   |                 |
| ssl_key       | server-key.pem  |
+---------------+-----------------+
9 rows in set (0.01 sec)
```

客户端配置

客户端配置相对简单一些. 首先我们需要拷贝 `ca-cert.pem`, `client-cert.pem` 和 `client-key.pem` 这三个文件到客户端主机中, 然后我们可以执行如下命令来使用 SSL 连接 MySQL 服务:

```
mysql --ssl-ca=/path/to/ca-cert.pem --ssl-cert=/path/to/client-cert.pem --ssl-key=/path/to/client-key.pem -h host_name -u ssl_test -p
```

除了上述的使用命令行方式配置 SSL 外, 我们也可以使用配置文件的方式. 即在 `~/.my.cnf` 文件中添加如下内容即可:

```
[client]
ssl-ca=/path/to/ca-cert.pem
ssl-cert=/path/to/client-cert.pem
ssl-key=/path/to/client-key.pem
```

当连接成功后, 我们执行如下指令

```
mysql> \s
--------------
mysql  Ver 14.14 Distrib 5.7.17, for Linux (x86_64) using  EditLine wrapper

Connection id:        14
Current database:
Current user:        ssl_test@172.17.0.4
SSL:            Cipher in use is DHE-RSA-AES256-SHA
Current pager:        stdout
Using outfile:        ''
Using delimiter:    ;
Server version:        5.7.17 MySQL Community Server (GPL)
Protocol version:    10
Connection:        test_db via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
TCP port:        3306
Uptime:            1 hour 2 min 9 sec

Threads: 1  Questions: 23  Slow queries: 0  Opens: 126  Flush tables: 3  Open tables: 0  Queries per second avg: 0.006
--------------
```

如果输出中有 `SSL: Cipher in use is DHE-RSA-AES256-SHA` 之类的信息, 则表示已经使用 SSL 来连接了.

在 Docker 中使能 MySQL SSL 连接

上面我们简单介绍了一下如果使能 MySQL SSL 连接, 那么现在我们使用 Docker 来具体的实战一把吧!

首先拉取最新的 MySQL 镜像:

```
docker pull mysql
```

然后需要准备一下挂载到 Docker 容器的目录结构:

```
>>> cd ~/temp
>>> tree
.
├── cert
│   ├── ca-key.pem
│   ├── ca.pem
│   ├── client-cert.pem
│   ├── client-key.pem
│   ├── private_key.pem
│   ├── public_key.pem
│   ├── server-cert.pem
│   └── server-key.pem
├── config
│   └── my.cnf
└── db

3 directories, 9 files
```

在 temp 目录下有三个子目录:

- `cert` 目录用于存放我们先前生成的证书和私钥信息;
- `config` 目录用于存放 MySQL 服务的配置文件
- `db` 目录是用于存放 MySQL 的数据.

下一步我们需要使用如下命令启动 MySQL 容器:

```
docker run --rm --name test_db -p 10000:3306 -e MYSQL_ROOT_PASSWORD=root -v /Users/xiongyongshun/temp/db:/var/lib/mysql -v
/Users/xiongyongshun/temp/config:/etc/mysql/conf.d -v /Users/xiongyongshun/temp/cert:/etc/mysql/cert mysql:latest
```

我们在上面的命令中, 我们分别挂载了 `cert`, `config`, `db` 这三个宿主机上的目录到 MySQL 容器中.

启动了 MySQL 服务后, 可以先使用 root 帐号登录 MySQL, 来检查 MySQL 服务此时是否已经开启了 SSL 功能:

```
docker run -it --link test_db:test_db --rm mysql sh -c 'exec mysql -u root -p -h test_db'
```

登录成功后, 我们在 MySQL 中执行如下指令:

```
mysql> show variables like '%ssl%';
+---------------+---------------------------------+
| Variable_name | Value                           |
+---------------+---------------------------------+
| have_openssl  | YES                             |
| have_ssl      | YES                             |
| ssl_ca        | /etc/mysql/cert/ca-cert.pem     |
| ssl_capath    |                                 |
| ssl_cert      | /etc/mysql/cert/server-cert.pem |
| ssl_cipher    |                                 |
| ssl_crl       |                                 |
| ssl_crlpath   |                                 |
| ssl_key       | /etc/mysql/cert/server-key.pem  |
+---------------+---------------------------------+
9 rows in set (0.01 sec)
```

有上面的输出后, 表明此时 MySQL 服务已经使用 SSL 功能了.

```
GRANT ALL PRIVILEGES ON *.* TO 'ssl_test'@'%' IDENTIFIED BY 'ssl_test' REQUIRE SSL;
FLUSH PRIVILEGES;
```

上面的命令创建了一个帐号名为 `ssl_test`, 密码为 `ssl_test`, 并且不限制登录主机 ip 的帐号.

这些都配置成功后, 我们再启动一个 MySQL 客户端容器:

```
docker run -it --link test_db:test_db --rm -v /Users/xiongyongshun/temp/cert:/etc/mysql/cert mysql sh -c 'exec mysql --ssl-ca=/etc/mysql/cert/ca-cert.pem --ssl-cert=/etc/mysql/cert/client-cert.pem --ssl-key=/etc/mysql/cert/client-key.pem -h test_db -u ssl_test -p'
```

从上面的这个命令中我们可以看到, 启动 MySQL 客户端容器时, 我们挂载了宿主机的 `cert` 目录到容器内的 `/etc/mysql/cert` 目录, 这样在容器中就可以访问到 SSL 私钥和证书文件了. 接着我们在 MySQL 客户端命令行中, 使用 `--ssl-ca`, `--ssl-cert`, `--ssl-key` 这三个参数来指定 SSL 连接所需要的 CA 证书, RSA 私钥和客户端证书.

登录成功后, 我们执行 `s` 命令:

```
mysql> \s
--------------
mysql  Ver 14.14 Distrib 5.7.17, for Linux (x86_64) using  EditLine wrapper

Connection id:        5
Current database:
Current user:        ssl_test@172.17.0.5
SSL:            Cipher in use is DHE-RSA-AES256-SHA
Current pager:        stdout
Using outfile:        ''
Using delimiter:    ;
Server version:        5.7.17 MySQL Community Server (GPL)
Protocol version:    10
Connection:        test_db via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
TCP port:        3306
Uptime:            6 min 8 sec

Threads: 2  Questions: 10  Slow queries: 0  Opens: 113  Flush tables: 1  Open tables: 106  Queries per second avg: 0.027
--------------
```

输出中有 `SSL: Cipher in use is DHE-RSA-AES256-SHA` 信息则说明我们确实是使用了 SSL 连接的 MySQL 服务器.

你可能感兴趣的

- **Apache 部署SSL数字证书及安全性设置** infiniSign ssl apache
- **docker搭建私有仓库、自签发证书、登录认证** silenceboy linux macos docker

- **OpenSSL 简单思路和函数笔记** amc openssl ssl c linux
- **Working with Docker Hub** adolphlwq docker
- **kubernetes RBAC实战 kubernetes 用户角色访问控制，dashboard访问，kubectl配置生成** fanux golang docker kubernetes
- **使用StartSSL为网站添加SSL** JellyBool ssl证书 ssl
- **浅析TLS 1.2协议** 雾花_小路 tls ssl

评论

默认排序 时间排序

文明社会，理性评论

发布评论