Authenticate against a MySQL server without knowing the cleartext password

Andrea Cardaci — 22 December 2018

This post is based on the README of the **mysql-unsha1** (March 2017) project that also contains the handshake sniffer and a patch for the MySQL client, go **here** if you just need the tools.

This has also been featured on the SANS Internet Storm Center **podcast**.

Abstract

This PoC shows how it is possible to authenticate against a MySQL server under certain circumstances without knowing the cleartext password when the **Secure Password Authentication** authentication plugin (aka `mysql_native_password`, the default method) is used.

Preconditions are:

- to obtain a read-only access to the `mysql.user` table in the target database in order to fetch the hashed password for a given user;
- to be able to intercept a successful authentication handshake performed by the aforementioned user (i.e., **authentication via SSL** would nullify this attempt).

The above are quite strong propositions, but this attack could provide an alternative way to obtain database access in a completely stealth way.

MySQL server passwords

By default, passwords are stored in the `mysql.user` table and are hashed using the `PASSWORD` function which is just a two-stage SHA1 digest:

```
mysql> SELECT DISTINCT password FROM mysql.user WHERE user = 'root';
*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19

mysql> SELECT PASSWORD('password');
*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19

mysql> SELECT SHA1(UNHEX(SHA1('password')));
2470c0c06dee42fd1618bb99005adca2ec9d1e19
```

The handshake

After the TCP connection phase, initiated by the client, the MySQL authentication **handshake** continues as follows (simplified):

1. the server sends a `Server Greeting` packet containing a *salt* (`s`);
2. the client replies with a `Login Request` packet containing the session password (`x`), computed as follows:

    ```
    x := SHA1(password) XOR SHA1(s + SHA1(SHA1(password)))
    ```

where `password` is the cleartext password as provided by the user and `+` is a mere string concatenation operator;

3. the server can verify the *challenge* and authenticate the client if:

```
SHA1(x XOR SHA1(s + SHA1(SHA1(password)))) = SHA1(SHA1(password))
```

where `SHA1(SHA1(password))` is the two-stage SHA1 digest of the password, stored in the `mysql.user` table; the server does not know the cleartext password nor its SHA1 digest.

Computing the hashed password

With enough information an attacker is able to obtain `SHA1(password)` and therefore to solve the server challenge without the knowledge of the cleartext password.

Let:

- `h` be the hashed password obtained from the `mysql.user` table (i.e., `SHA1(SHA1(password))`);
- `s` and `x` be the salt and the session password respectively obtained from the intercepted handshake.

The first-stage SHA1 can be obtained as follows:

```
SHA1(password) = x XOR SHA1(s + h)
```

Tools

`mysql-unsha1` comes with two tools:

- a simple sniffer to extract and check the handshake information either live or offline from a PCAP file;
- a patch for MySQL client which allows to treat the prompted passwords as SHA1 digests instead of cleartexts.

Refer to the **GitHub repository** for compilation and usage instructions.