New critical denial of service vulnerability in the SQLCipher SQL command processing (CVE-2021-3119)

Peter Samarin (e-mail: peter.samarin@t-systems.com)
Robert Hörr (e-mail: robert.hoerr@t-systems.com)
(Penetration Testers of the Deutsche Telekom Security GmbH)

A new critical denial of service vulnerability (CVE-2021-3119) in the SQLCipher SQL command processing of the master branch (https://github.com/sqlcipher) was discovered with a self-developed *SQLCipher-FAST* (Fast Automated Software Testing) framework. The versions 4.4.2, 4.4.1, 4.4.0, 4.3.0, 4.2.0, 4.1.0 include the vulnerability, too. The vulnerability cause a segmentation fault and thus a denial of service.

What is the SQLCipher library?
The company Zetetic provides several security frameworks. One of them is the open source SQLCipher library. It is an extension for the open source SQLite database providing complete database encryption. Several companies like Samsung, Motorola and SAP are using this library. According to Zetetic, "*SQLCipher is widely used, protecting data for thousands of apps on hundreds of millions of devices[...]* ." (https://www.zetetic.net/sqlcipher/)

How is the SQLCipher library tested?
The code size of SQLite and SQLCipher library is roughly 250,000 lines of code. It is hardly possible to check all code paths by a manual source code review. Hence, dynamic automated machine testing must be performed. An efficient way to perform this kind of testing is the code coverage fuzzing approach. For that, the *SQLCipher-FAST* framework was developed. This framework unites the strengths of several fuzzing tools and detects issues like buffer overflows.

Which issue was discovered?
The *SQLCipher-FAST* framework detects several issues in the SQL command processing. One of the issues is a read memory access of a variable initialized with NULL. This issue occurs if the following specially crafted SQL command sequence is executed in the API function *`sqlite3_exec(...)`*:

```
select sqlcipher_export(:0);
```

The issue involves the following code in the *`sqlcipher_export(…)`* function:

```
if(targetDb_idx == 0 && sqlite3StrICmp("main", targetDb) != 0) {...}
```

The code tries to match a constant string "main" in the `targetDb` variable that is assumed to be a string. However, with a special query, `targetDb` is `NULL`, which causes a segmentation fault.

How is the discovered issue exploitable?
Exploiting this issue can result in a remote denial of service attack. For example, a SQL injection can be used to execute the specially crafted SQL command sequence. The command sequence causes a segmentation fault that crashes the program, which results in a denial of service.