SQLite database inside a secure Intel SGX enclave

Peng Xu

July 1, 2019
SQLite database

1. What is SQLite?
2. Advantages
   - Lightweight
   - Cross-platform
   - Highly efficient queries
   - High query speed, less memory
   - Embedding system
3. Disadvantages
   - SQL standard support is not fully
   - Discomfort for large insert, update, etc.
1. Multiple programming language interfaces
   ▶ C/C++: Straightforward to use
   ▶ Java: SQLite’s JDBC driver
   ▶ Python: Pysqlite
   ▶ Ruby: Sqlite-ruby
   ▶ ......

2. The SQLite Amalgamation
   ▶ The SQLite library consists of 102 files of C code
   ▶ Of the 102 main source files, about 75% are C code and about 25% are C header files
   ▶ The amalgamation is “sqlite3.c”, that contains all C code for the core SQLite library
1. Basic Usage
2. Database
   ▶ Create database:  `CREATE DATABASE databasename`
   ▶ Drop database:  `DROP DATABASE databasename`
   ▶ Backup database:  `BACKUP DATABASE databasename TO DISK = 'filepath'`;
1. Basic Usage

2. Database
   - Create database: `CREATE DATABASE databasename`
   - Drop database: `DROP DATABASE databasename`
   - Backup database: `BACKUP DATABASE databasename TO DISK = 'filepath'`

3. Table
   - Create table: `CREATE TABLE table_name ( column1 datatype, column2 datatype, column3 datatype, .... );`
   - Drop table: `DROP TABLE table_name;`
   - Alter table: `ALTER TABLE table_name ADD column_name datatype;`
1. Entry

- Select: `SELECT column1, column2, ... FROM table_name;`
- Update: `UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;`
- Insert: `INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);`
- Delete: `DELETE FROM table_name WHERE condition;`
- ......
SGX_SQLite

1. SQLite database inside a secure Intel SGX enclave (Linux)
2. Execute SQL statements securely
3. https://github.com/yerzhan7/SGX_SQLite
1. App
   - App.cpp
   - ocalls.c

2. Enclave
   - Enclave.cpp
   - Enclave.edl
   - Enclave_private.pem
   - Configures
   - Ocall_interface.c
   - Sqlite3.h/.c

3. Ocall_types.h

4. Makefile
SGX.SQLite (cont.)

1. Advantages?
2. Disadvantages?
3. How to improve this work?
Dynamic taint analysis

1. Valgrind + taintgrind https://github.com/wmkhoo/taintgrind

2. Steps:
   - labeling the sensitive data
   - tracing the taint propagation
   - finding the functions and statements relative with labeled sensitive data

3. Example
   - tests/sign32.c
   - \texttt{TNT\_TAINT(&a, sizeof(a));}
   - \texttt{valgrind \textendash tool=taintgrind tests/sign32}
   - \texttt{valgrind \textendash tool=taintgrind tests/sign32 2>&1 \textendash python log2dot.py > sign32.dot}
   - \texttt{gcc -g}
Partitioning C program

1. Getting tainted information
   ▶ Tainted files
   ▶ Tainted functions
2. Pinpointing the tainted files and functions
3. Splitting program on the source code level
4. Generating splitted source code
   ▶ Bare-metal system
   ▶ SDK-based
5. Compiling and linking to binary
Partitioning C program

1. How can we partition SGX_SQLite with tainted variables?
2. Try to amalgamate openssl’s source code
   ▶ https://github.com/openssl/openssl
   ▶ https://github.com/vinniefalco/Amalgamate
   ▶ https://github.com/rindeal/Amalgamate
3. Is it possible to create SGX_FreeType and SGX_TagLib like SGX_SQLite?
   ▶ FreeType: https://github.com/vinniefalco/FreeTypeAmalgam
   ▶ TagLib: https://github.com/vinniefalco/TagLibAmalgam
Question?

Questions?