Cleartext storage of sensitive information in an SQLite database

Name: Cleartext storage of sensitive information in an SQLite database

Description: Storing sensitive information in a non-encrypted database can expose it to an attacker.

ID: cpp/cleartext-storage-database

Kind: problem

Severity: warning

Precision: medium

Sensitive information that is stored in an unencrypted SQLite database is accessible to an attacker who gains access to the database.

Recommendation

Ensure that if sensitive information is stored in a database then the database is always encrypted.

Example

The following example shows two ways of storing information in an SQLite database. In the 'BAD' case, the credentials are simply stored in cleartext. In the 'GOOD' case, the database (and thus the credentials) are encrypted.

```
 1  void bad(void) {
 2    char *password = "cleartext password";
 3    sqlite3 *credentialsDB;
 4    sqlite3_stmt *stmt;
 5
 6    if (sqlite3_open("credentials.db", &credentialsDB) == SQLITE_OK) {
 7      // BAD: database opened without encryption being enabled
 8      sqlite3_exec(credentialsDB, "CREATE TABLE IF NOT EXISTS creds (password TEXT);", NULL,
NULL, NULL);
 9      if (sqlite3_prepare_v2(credentialsDB, "INSERT INTO creds(password) VALUES(?)", -1, &stmt
, NULL) == SQLITE_OK) {
10        sqlite3_bind_text(stmt, 1, password, -1, SQLITE_TRANSIENT);
11        sqlite3_step(stmt);
12        sqlite3_finalize(stmt);
13        sqlite3_close(credentialsDB);
14      }
15    }
16  }
17
18  void good(void) {
19    char *password = "cleartext password";
20    sqlite3 *credentialsDB;
21    sqlite3_stmt *stmt;
22
23    if (sqlite3_open("credentials.db", &credentialsDB) == SQLITE_OK) {
24      // GOOD: database encryption enabled:
25      sqlite3_exec(credentialsDB, "PRAGMA key = 'secretKey!'", NULL, NULL, NULL);
26      sqlite3_exec(credentialsDB, "CREATE TABLE IF NOT EXISTS creds (password TEXT);", NULL,
NULL, NULL);
27      if (sqlite3_prepare_v2(credentialsDB, "INSERT INTO creds(password) VALUES(?)", -1, &stmt
, NULL) == SQLITE_OK) {
28        sqlite3_bind_text(stmt, 1, password, -1, SQLITE_TRANSIENT);
29        sqlite3_step(stmt);
30        sqlite3_finalize(stmt);
```

```
31        sqlite3_close(credentialsDB);
32      }
33    }
34  }
```

References

- M. Dowd, J. McDonald and J. Schuhm, *The Art of Software Security Assessment*, 1st Edition, Chapter 2 - 'Common Vulnerabilities of Encryption', p. 43. Addison Wesley, 2006.
- M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd Edition, Chapter 9 - 'Protecting Secret Data', p. 299. Microsoft, 2002.
- Common Weakness Enumeration: CWE-313.