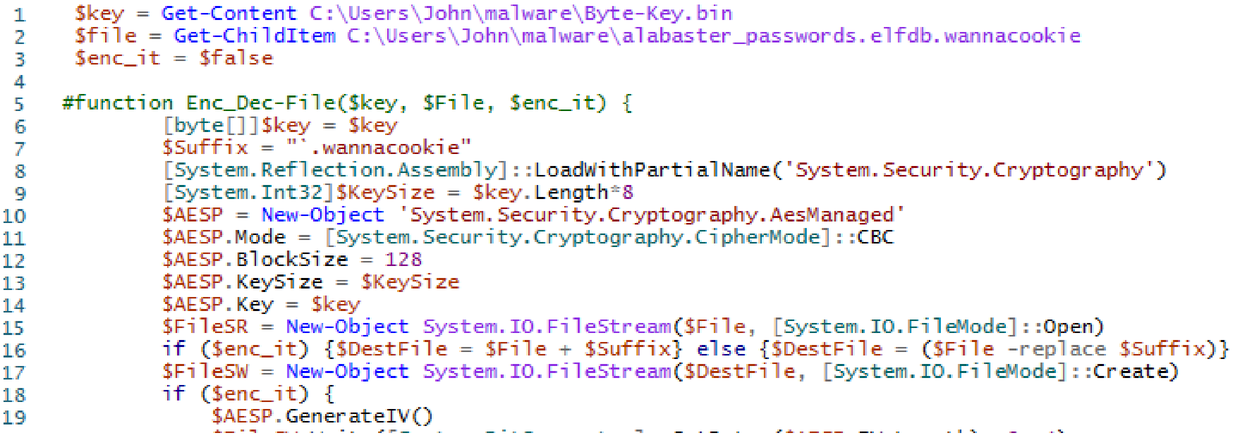
Objective--Recover Alabaster’s Password

(Part 6)

Decrypting Alabaster’s file

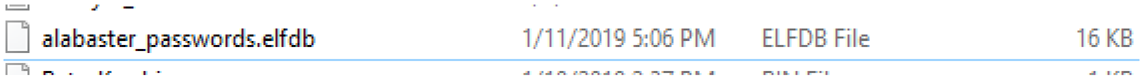
The code to decrypt Alabaster’s file is shown here. The key is the file we saved in the last lesson. Note that `$file` is the path to the wannacookie file, not the content; that’s why the line uses `Get-Childitem` (`dir` or `ls`) and not `Get-Content`. The variable `$enc_it` is set to False to cause the file to be decrypted.



```
1    $key = Get-Content C:\Users\John\malware\Byte-Key.bin
2    $file = Get-ChildItem C:\Users\John\malware\alabaster_passwords.elfdb.wannacookie
3    $enc_it = $false
4
5   #function Enc_Dec-File($key, $File, $enc_it) {
6           [byte[]]$key = $key
7           $Suffix = "`.wannacookie"
8           [System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography')
9           [System.Int32]$KeySize = $key.Length*8
10          $AESP = New-Object 'System.Security.Cryptography.AesManaged'
11          $AESP.Mode = [System.Security.Cryptography.CipherMode]::CBC
12          $AESP.BlockSize = 128
13          $AESP.KeySize = $KeySize
14          $AESP.Key = $key
15          $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open)
16          if ($enc_it) {$DestFile = $File + $Suffix} else {$DestFile = ($File -replace $Suffix)}
17          $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create)
18          if ($enc_it) {
19              $AESP.GenerateIV()
```

The rest of the file is unchanged, except that the final “}” is commented out to match the one in line 5.

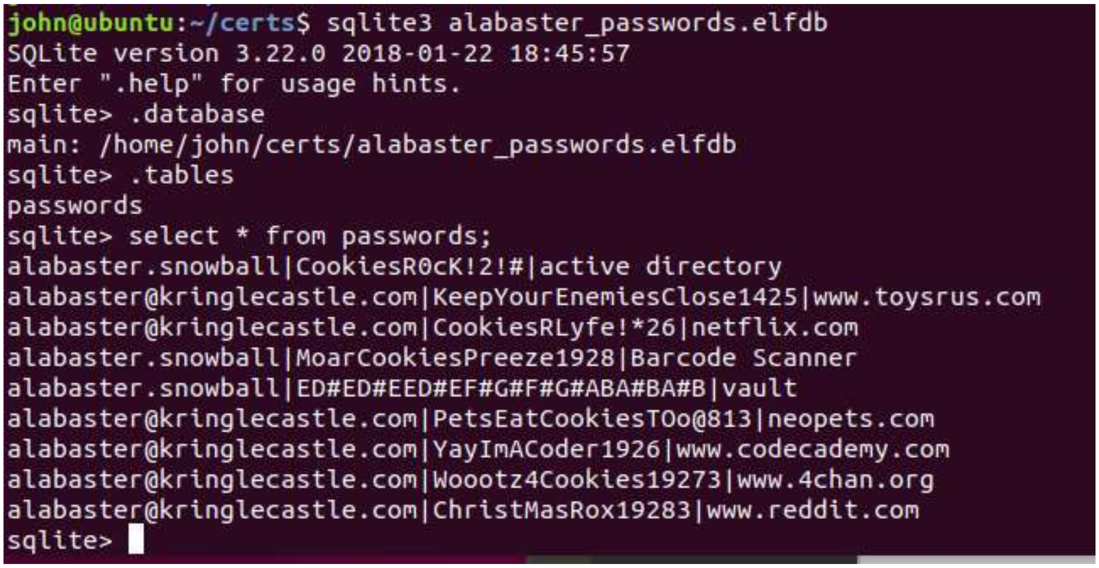
When we look in the directory where alabaster_passwords.elfdb.wannacry used to be, we find it has been replaced by alabaster_passwords.elfdb. Whew!



```
alabaster_passwords.elfdb          1/11/2019 5:06 PM     ELFDB File          16 KB
```

Exploring the Database

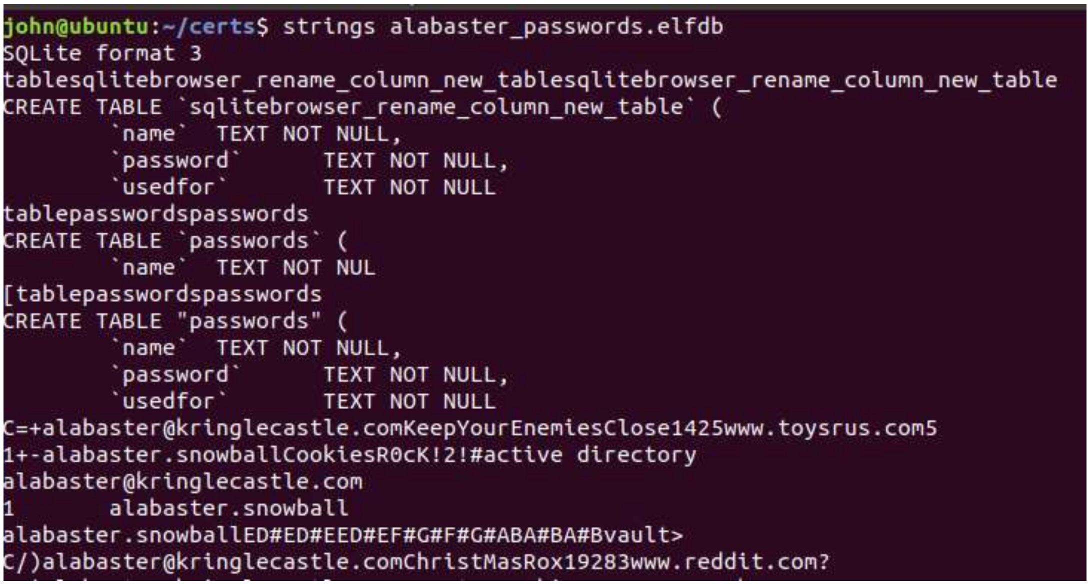
It’s less work to paste Alabaster’s database file into a Linux VM that already has sqlite3 than to install sqlite3 on Windows, so that is what we will do. Then we can open the database.



```
john@ubuntu:~/certs$ sqlite3 alabaster_passwords.elfdb
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .database
main: /home/john/certs/alabaster_passwords.elfdb
sqlite> .tables
passwords
sqlite> select * from passwords;
alabaster.snowball|CookiesR0cK!2!#|active directory
alabaster@kringlecastle.com|KeepYourEnemiesClose1425|www.toysrus.com
alabaster@kringlecastle.com|CookiesRLyfe!*26|netflix.com
alabaster.snowball|MoarCookiesPreeze1928|Barcode Scanner
alabaster.snowball|ED#ED#EED#EF#G#F#G#ABA#BA#B|vault
alabaster@kringlecastle.com|PetsEatCookiesTOo@813|neopets.com
alabaster@kringlecastle.com|YayImACoder1926|www.codecademy.com
alabaster@kringlecastle.com|Woootz4Cookies19273|www.4chan.org
alabaster@kringlecastle.com|ChristMasRox19283|www.reddit.com
sqlite> 
```

Alabaster’s vault password is ED#ED#EED#EF#G#F#G#ABA#BA#B.

We could also have used brute force. The string command works, it is just harder to read.



```
john@ubuntu:~/certs$ strings alabaster_passwords.elfdb
SQLite format 3
tablesqlitebrowser_rename_column_new_tablesqlitebrowser_rename_column_new_table
CREATE TABLE `sqlitebrowser_rename_column_new_table` (
        `name`  TEXT NOT NULL,
        `password`      TEXT NOT NULL,
        `usedfor`       TEXT NOT NULL
tablepasswordspasswords
CREATE TABLE `passwords` (
        `name`  TEXT NOT NUL
[tablepasswordspasswords
CREATE TABLE "passwords" (
        `name`  TEXT NOT NULL,
        `password`      TEXT NOT NULL,
        `usedfor`       TEXT NOT NULL
C=+alabaster@kringlecastle.comKeepYourEnemiesClose1425www.toysrus.com5
1+-alabaster.snowballCookiesR0cK!2!#active directory
alabaster@kringlecastle.com
1       alabaster.snowball
alabaster.snowballED#ED#EED#EF#G#F#G#ABA#BA#Bvault>
C/)alabaster@kringlecastle.comChristMasRox19283www.reddit.com?
```