Apple has attempted to make iTunes backup encryption turned on by default as part of the 10.2 upgrade. In addition, they have made the password process complicated to prevent attempts to hack it, which it what we are going to do. Note that this process works on iOS 10 and higher, and iOS 9 and lower with different options. I will notate where the commands change.

The implications for forensics are serious. Even with gaining access to the device, which is not a trivial task, defeating the iTunes encrypted backup adds another level of complexity. Using your favorite extraction tool will get you the data, but result in encrypted contents. So, if you used a court order to compel a suspect's fingerprint, used a deceased person's finger for access, got the passcode from a witness, the data obtained is useless without the password.

A recent post from Patrick Siewart of Pro Digital Forensics Consulting in Virginia (http://prodigital4n6.blogspot.com), demonstrated this in a great walkthrough he did on his blog. Another post on one of the mobile forensic groups talked about using hashcat to crack this encryption. So, in a moment of caffeine filled wonder, I put the two together for this walkthrough.

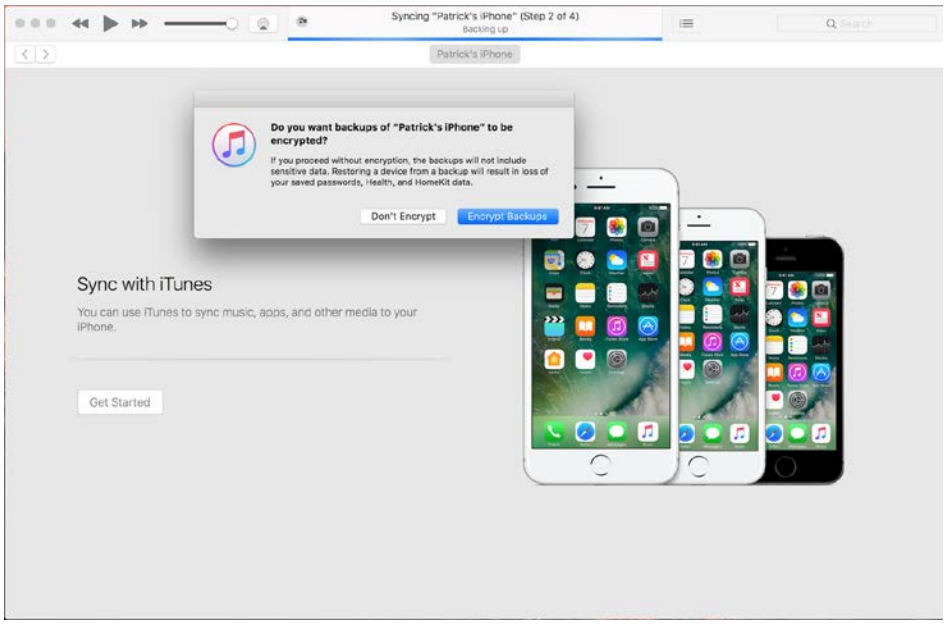


Figure 1 New iTunes Question (courtesy of Patrick Siewart)

First, as I stress in any class I teach, learn to use and love the command line (Linux and Windows). I attempted to make this walkthrough as simple as possible, as not everyone is command line savvy yet, so please keep the following in mind:

- All commands are case sensitive. OK, they may not REALLY be, but please treat as such, so I don't have to spell out which ones are and aren't
- This is for Windows. Yes, Linux and MacOS will work as well, using the correct version and syntax.
- For simplicity, I moved all the files into the hashcat folder. If you are new to CLI please do the same. If you can map a path to your files and don't move them, still works...

- Links to the needed software and test files are located at the **bottom** of this document.

The information we need to crack the password is stored in the Manifest.plist file. This will be in the extraction archive your forensic tool made:

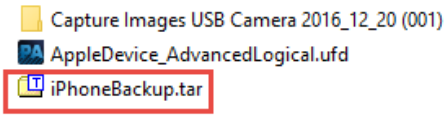


Figure 2 Cellebrite Extraction

Double click the archive and you get the contents of the actual backup:

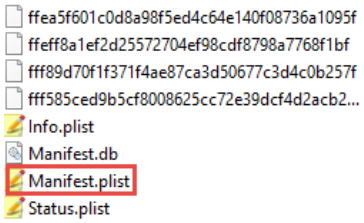
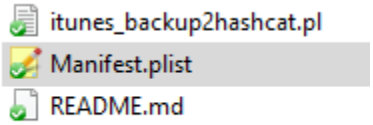


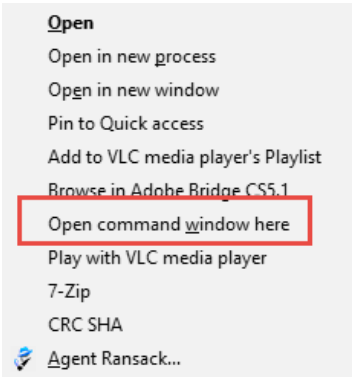
Figure 3 Manifest.plist file location

Export the Manifest.plist file out to a folder. Within the Manifest.plist is the needed information, located in various places. Someone was kind enough to write a script to pull this information from the file. This script requires Perl to be installed on your system to work. Head out, download and install Perl onto your system of choice.

Copy your Manifest.plist file into your backup2hashcat folder.



Launch a command prompt on the folder containing the script and the Manifest file. An easy way to do this within File Explorer is to press and hold SHIFT and RIGHT click on the folder name. Look for the option “Open command window here”.



Run the Perl script with the following command. Use a > to change std. out to a file, in this case, named code.txt.

```
itunes_backup2hashcat.pl Manifest.plist > code.txt
```

Like most things in the command line, when you nothing on the screen it worked!

```
02/06/2017  13:47    <DIR>          .
02/06/2017  13:47    <DIR>          ..
02/06/2017  13:47               198 code.txt
01/27/2017  08:40             5,663 itunes_backup2hashcat.pl
01/28/2017  16:23            46,950 Manifest.plist
01/27/2017  08:40             1,961 README.md
               4 File(s)         54,772 bytes
               2 Dir(s)  341,580,595,200 bytes free
```

Opening the file will give us which version of iTunes encryption was used on the device (9 and under, or 10 or higher). If you aren't sure from the device, look here, as this will tell you which options to run in hashcat.

```
$itunes_backup$*10*4c1adf
```

The example above is using iOS 10 or higher for the encryption.

Copy the code.txt file into your hashcat folder. As of the time of this writing, the hashcat version which supports iOS encryption is in BETA. Ensure you have hashcat 3.3 Beta 55 or higher. Once the final version is released, use that of course.

Hashcat needs a wordlist to crack the password. Think of this as something to compare the hashes from the Manifest file to. For this scenario, I used the rockyou.txt file readily available from the Internet. You can use an index built from a laptop, custom built file, whatever. In our lab we started with RockYou and add to it, so the now file is around 350MB in size. Please keep in mind you won't be using Notepad to open this file and add to it. Notepad++ even chokes on it, though it is possible. You can use Windows command line or, (better) use the Bash shell built into Windows 10 Anniversary edition to use the Linux CLI to easily append to the file.

Using the SHIFT+Right click, open a command prompt at your hashcat beta folder.

Use the following command to begin the process. Keep in mind my wordlist is called rockyou.txt, use whatever you name yours in place of mine. Use the **–m 14700 for iOS 9** and below, use **–m 14800 for iOS 10.**

```
hashcat64 -m 14800 -a 0 --weak-hash-threshold 0 code.txt rockyou.txt
```

Spaces are needed. The RED underline is the name of your file the Perl script made. The BLUE underline file is the name of your wordlist.

Hashcat will put some information on your screen and build the data it needs.

```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

Session..........: hashcat
Status...........: Running
Hash.Type........: iTunes Backup >= 10.0
Hash.Target......: $itunes_backup$*10*4c1adfb376b57bc676c65faaa54eda483176c845350cb17
4b3f1d0252d6d0045423d2145950f84de*10000000*db85e5324101b6b3b149b0b0b29a47b15d20918e
Time.Started.....: Mon Feb 06 14:13:23 2017 (43 secs)
Time.Estimated...: Mon Feb 06 14:14:06 2017 (0 secs)
Input.Base.......: File (rockyou.txt)
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:        0 H/s (2.21ms)
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 0/10 (0.00%)
Rejected.........: 0/0 (0.00%)
Restore.Point....: 0/10 (0.00%)
Candidates.#1....: password -> river
HWMon.Dev.#1.....: Temp: 32c Fan: 33% Util: 94% Core:1341Mhz Mem:3304Mhz Lanes:16
```

You can press “s” when the prompt comes up to see where it is in the process and an estimated time of completion. The times in the above screenshot are NOT accurate, as I used a seeded file for this demo. To run through our rockyou file it took 3 days, 13 hours, and some odd minutes, which it reported accurately.

When it finds the password you get this screen:

```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: iTunes Backup >= 10.0
Hash.Target......: $itunes_backup$*10*4c1adfb376b57bc676c65faaa54eda483176c845350cb
4b3f1d0252d6d0045423d2145950f84de*10000000*db85e5324101b6b3b149b0b0b29a47b15d20918e
Time.Started.....: Mon Feb 06 14:13:23 2017 (1 min, 35 secs)
Time.Estimated...: Mon Feb 06 14:14:58 2017 (0 secs)
Input.Base.......: File (rockyou.txt)
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:        0 H/s (2.11ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 10/10 (100.00%)
Rejected.........: 0/10 (0.00%)
Restore.Point....: 0/10 (0.00%)
Candidates.#1....: password -> river
HWMon.Dev.#1.....: Temp: 33c Fan: 33% Util: 95% Core:1341Mhz Mem:3304Mhz Lanes:16
```

Hashcat64 stores cracked passwords in a potfile. You can display this at the command line or simply locate the hashcat.potfile file in the hashcat folder. Open it with a text editor to view the contents. Scroll all the way to the end of the line (after the last colon) to view the password.

```
e*10000000*db85e5324101b6b3b149b0b0b29a47b15d20918e:boat
```

The password for this file is “boat”. I entered this into Cellebrite’s Physical Analyzer and successfully opened my iOS 10.2.1 backup file.

Hashcat uses the GPU (graphics card) to process the data. Apple has made this process complicated by reiterating the process 100000000 times, purposely to slow this process. You must use a high-end video card, or even better a series of video cards, for this to work even reasonably. On the NVIDIA Titan X ($1000) it topped out at 72 H/s. To put this in perspective, it will crack NTLM hashes as 1.2B H/s. You must have a great video card with the most recent drivers for this to work. Laptops are out, as are standard cards most forensic machines come with. For NVIDIA cards they must support OpenCL 5.0 or

higher, which is going to require a card made in about the last two years. I don't have an AMD card to test on, so I can't speak for those. For our whole list, I didn't think three days was bad, considering.

This process will become very relevant very soon. Thanks to Patrick for the point towards this and the hashcat community for the development.

Det. Ed Michael
Orlando, Florida Police Department
USSS Task Force
Cellebrite Senior Contractor and IACIS Mobile Device Chair

Links are subject to change:

Script: https://github.com/philsmd/itunes_backup2hashcat/ Click the Clone or download button

Hashcat BETA: https://hashcat.net/beta/

Rock You file: https://wiki.skullsecurity.org/index.php?title=Passwords