How to Lose Your Bitcoins: Part 2 (Cracking Bitcoin Core wallet.dat Files)

05 Feb 2018

This is part two in a series of blogs on cryptocurrencies and security. The goal is to show how passwords can be recovered for encrypted Bitcoin Core (or Satoshi Client) wallets.

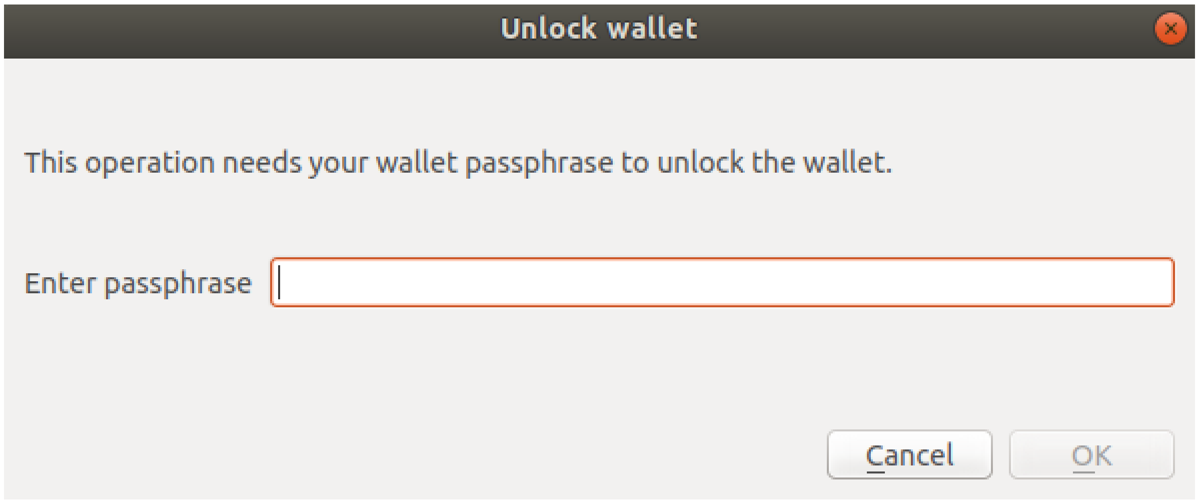
Please note: This is an information security blog. The intent is to help people learn about hacking, point out vulnerabilities that we encounter every day, and ultimately help people make better decisions about security. STEALING SOMEONE'S BITCOINS WOULD BE A DICKHEAD MOVE. DON'T DO THAT.

The Target

We're making progress! Following the instructions in part one, you've gained access to an encrypted USB drive. Looking around, you notice a some interesting files - perhaps in a hidden folder called `.bitcoin`. Inside that folder, look for a file called `wallet.dat` - that contains all of the information needed to access a bitcoin wallet and its associated funds.

You can also search for other `.dat` files, as these can be manual backups created by the user.

Let's assume you've already tried to import the file into a wallet application and are prompted to enter a password. If you haven't yet tried this - do it now. You might get lucky.


Unlock wallet
This operation needs your wallet passphrase to unlock the wallet.
Enter passphrase
Cancel
OK

Nah, that would have been too easy!

Tools

Cracking these wallets can be fairly hardware-intensive, especially when using really long wordlists. For this tutorial, we are using a Linux-based computer with an Nvidia GPU.

You'll need a few things ready to go:

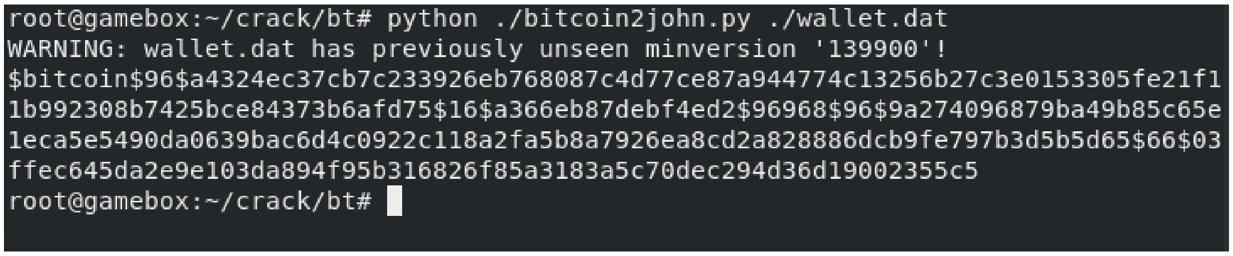
- Bitcoin2john. This is a fork of pywallet modified to extract the password hash in a format that hashcat can understand.
- Hashcat. Get the newest version from this link, some Linux package managers are woefully behind on this stuff.
- A text file that contains your encryption passphrase. While you are practicing, just make a short text file with 10 lines in it, one of them being the passphrase you set on your practice wallet. Here are some good resources for cracking the real stuff:
  - My passphrase wordlist. This is a work in progress aimed at collecting common short phrases.
  - Crackstation's wordlist. This holds more common passwords, as most people aren't really using phrases yet.
- For advanced cracking, a rule list. I like these:
  - Hob0Rules.
  - OneRule.

First Up - Extract the Password Hash

In this step, we'll extract a hash that can be used to crack the password. Place both `bitcoin2john.py` and `wallet.dat` in the same folder. Inside the terminal, run the python script as follows:

```
python ./bitcoin2john.py ./wallet.dat
```

Take the line that starts with `$bitcoin` and place it in a file called `hash.txt` in the working directory.



```
root@gamebox:~/crack/bt# python ./bitcoin2john.py ./wallet.dat
WARNING: wallet.dat has previously unseen minversion '139900'!
$bitcoin$96$a4324ec37cb7c233926eb768087c4d77ce87a944774c13256b27c3e0153305fe21f1
1b992308b7425bce84373b6afd75$16$a366eb87debf4ed2$96968$96$9a274096879ba49b85c65e
1eca5e5490da0639bac6d4c0922c118a2fa5b8a7926ea8cd2a828886dcb9fe797b3d5b5d65$66$03
ffec645da2e9e103da894f95b316826f85a3183a5c70dec294d36d19002355c5
root@gamebox:~/crack/bt# 
```

Note: If you're reading this because you've forgotten your own password and are unable to crack it yourself, you can share this hash with a reuptable wallet recovery service. Cracking this hash will not allow them to access your Bitcoins unless they also have access to your `wallet.dat` file. However, it you've re-used this password elsewhere, it may allow them to log in to those services and otherwise make your life hell.

Get Cracking - Standard Dictionary Attack

Now comes the fun. Cracking passwords is an art, and consistent success requires really fine tuning your approach. If you want to learn some advanced methods, Google around a bit or check out this great book.

Assumptions:

- You've downloaded hashcat and placed the files into `/opt/hashcat`.
- You've configured your drivers as recommended here. This is important.
- You've created a short text file, with one potential password per line, your password being one of them. That text file is at called `wordlist.txt` and is in the working folder with your `hash.txt` file.

First, we are going to run a straight-up dictionary attack. This means that password has to be found in your wordlist exactly - with correct case, special characters, etc.

Here we go:

```
# Try it this way first, with some hardware optimization parameters:
/opt/hashcat/hashcat64.bin -a 0 -m 11300 ./hash.txt ./wordlist.txt -O -w 3

# If that doesn't work, try this:
/opt/hashcat/hashcat64.bin -a 0 -m 11300 ./hash.txt ./wordlist.txt
```

Press the `S` key at any time to see that status of your cracking session.

If your session completes successfully, you should see an output with your password. If the session completed and you aren't sure it was successful, running the command as follows will show you all successfully cracked passwords for a given target:

```
/opt/hashcat/hashcat64.bin -a 0 -m 11300 ~/hash.txt --show
```

If the output of the above command is blank, the password has not yet been cracked.

Getting Tricky - Rule-Based Attacks

As humans, we are pretty dumb when it comes to making passwords. We think doing things like adding an `!` or replacing an `S` with a `$` makes them more secure.

When it comes to password cracking, this may slow us down a bit but certainly doesn't stop us. We can use a pre-defined set of rules for transforming files in a wordlist to many possible permutations.

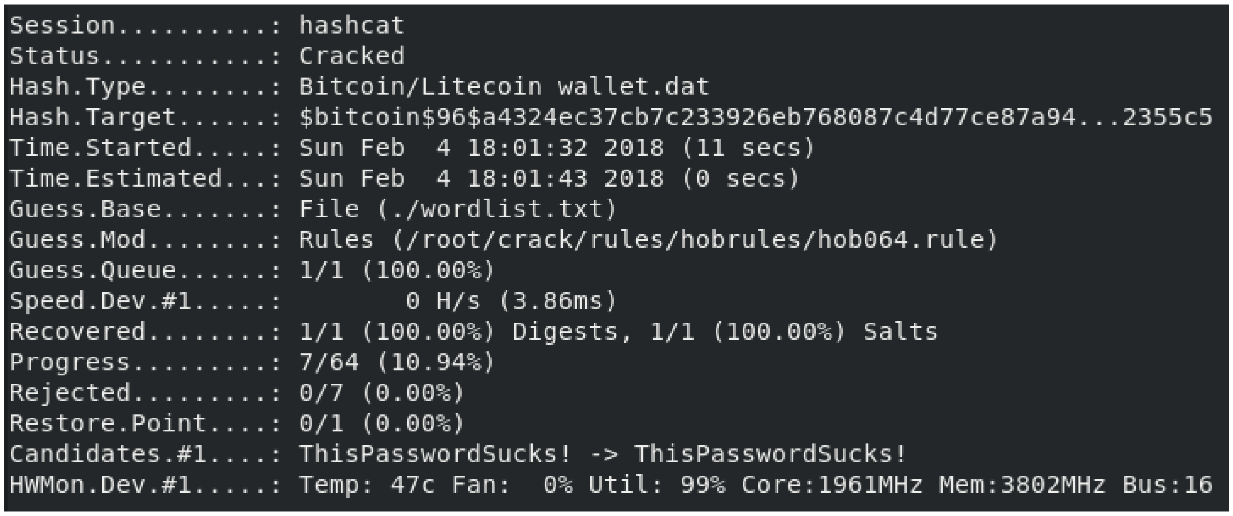
If you'd like to test this out, use a password like `ThisPasswordSucks!` for your `wallet.dat`, but in the wordlist you create enter only `ThisPasswordSucks`. This isn't really stressing your rule list, but you get the idea.

Assumptions:

- You've downloaded Hob0Rules and placed it in `/opt/rules/`.

Follow the same process as before, but alter the command to use one of the rule sets as follows:

```
/opt/hashcat/hashcat64.bin -a 0 -m 11300 ./hash.txt ./wordlist.txt  -r /opt/rules/Hob0R
```



```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: Bitcoin/Litecoin wallet.dat
Hash.Target......: $bitcoin$96$a4324ec37cb7c233926eb768087c4d77ce87a94...2355c5
Time.Started.....: Sun Feb  4 18:01:32 2018 (11 secs)
Time.Estimated...: Sun Feb  4 18:01:43 2018 (0 secs)
Guess.Base.......: File (./wordlist.txt)
Guess.Mod........: Rules (/root/crack/rules/hobrules/hob064.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:        0 H/s (3.86ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 7/64 (10.94%)
Rejected.........: 0/7 (0.00%)
Restore.Point....: 0/1 (0.00%)
Candidates.#1....: ThisPasswordSucks! -> ThisPasswordSucks!
HWMon.Dev.#1.....: Temp: 47c Fan:  0% Util: 99% Core:1961MHz Mem:3802MHz Bus:16
```

As you can see, this password sucks. I hope the hash above didn't get you too excited.

Happy Hacking!

Have fun with this! If you have any questions or need any help, feel free to reach out.