Picking Locky 🔓

dissectingmalwa.re/picking-locky.html

Tue 30 July 2019 in Ransomware

Back in 2016 Locky was (one of) the first to commercialize the "art" of holding data for ransom. I picked this strain because I would like a bit more of a challenge in terms of obfuscation and anti-disassembly techniques, so strap in for this OG Ransomware

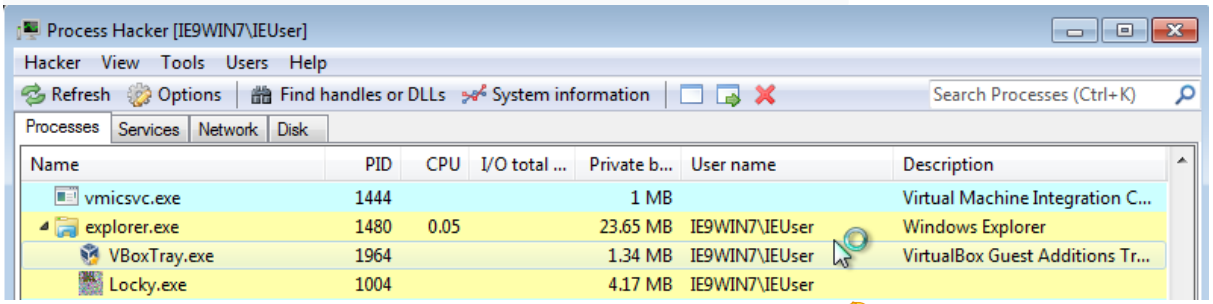
Locky (at least the first few versions) is said to be created by the makers of the Dridex/Cridex banking trojans. For example the spreading mechanism via macros in Word or Excel documents sent out via carefully crafted spear-phishing emails is exactly the same for both of these strains.

A general disclaimer as always: downloading and running the samples linked below will lead to the encryption of your personal data, so be f$cking careful. Also check with your local laws as owning malware binaries/ sources might be illegal depending on where you live.

Todays samples are brought to you by:

Locky #1 available @ https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/Ransomware.Locky `sha256` `bc98c8b22461a2c2631b2feec399208fdc4ecd1cd2229066c2f385caa958daa3`

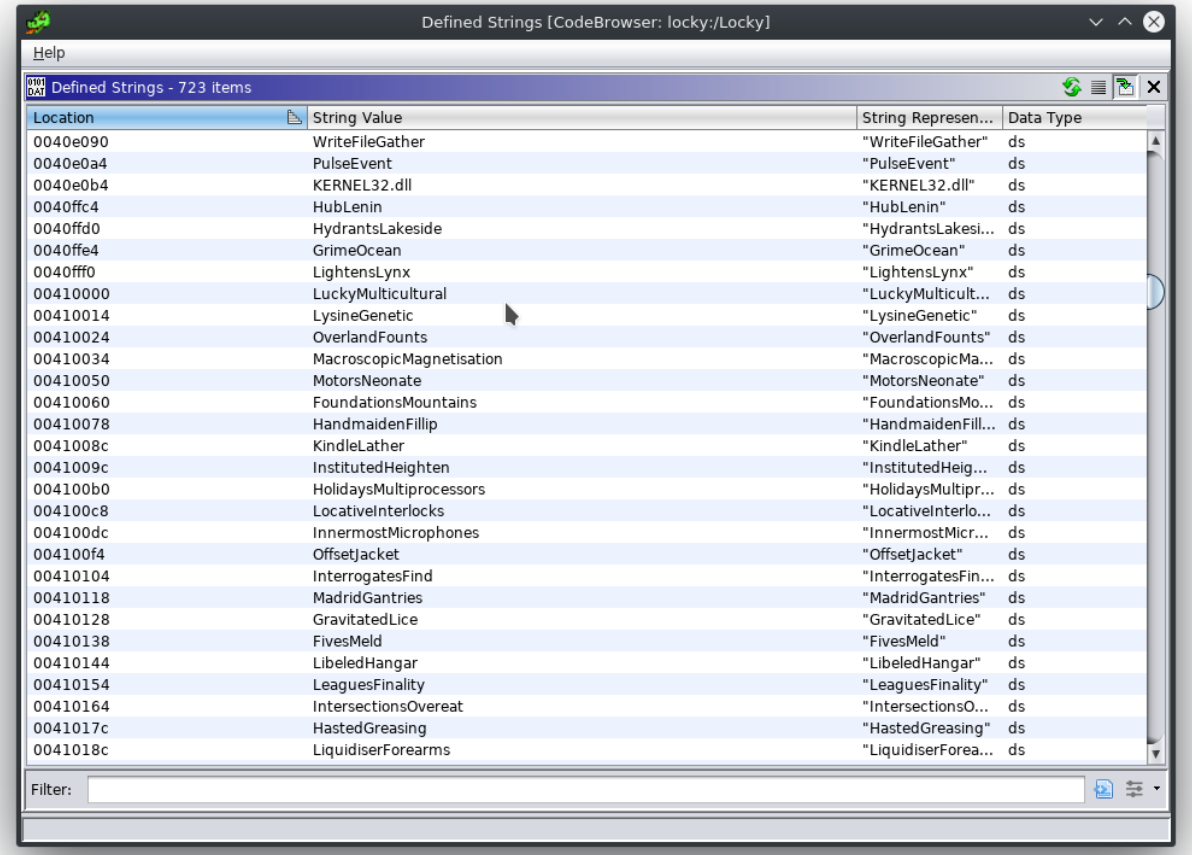
Locky.AZ available @ https://dasmalwerk.eu/ `sha256` `2e4319ff62c03a539b2b2f71768a0cfc0adcaedbcca69dbf235081fe2816248b`



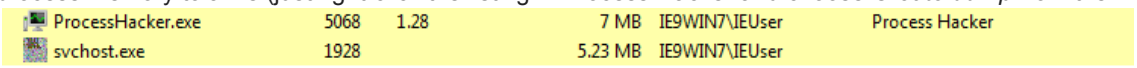
Running this first Locky Sample was pretty unspectacular since nothing really happend 🤔. Let's take a look at the binary first:


Import Results Summary
Project File Name: Locky
Last Modified: Tue Jul 30 00:07:36 CEST 2019
Readonly: false
Program Name: Locky
Language ID: x86:LE:32:default (2.8)
Compiler ID: windows
Processor: x86
Endian: Little
Address Size: 32
Minimum Address: 00400000
Maximum Address: 007d00c7
# of Bytes: 3988554
# of Memory Blocks: 6
# of Instructions: 0
# of Defined Data: 830
# of Functions: 0
# of Symbols: 183
# of Data Types: 5
# of Data Type Categories: 3
CompanyName: FileSee.com
Compiler: visualstudio:unknown
Created With Ghidra Version: 9.0.4
Date Created: Tue Jul 30 00:07:35 CEST 2019
Executable Format: Portable Executable (PE)
Executable Location: /home/f0wl/Malware/Locky/Locky
Executable MD5: b06d9dd17c69ed2ae75d9e40b2631b42
FSRL: file:///home/f0wl/Malware/Locky/Locky?MD5=b06d9dd17c69ed2ae75d9e40b2631b42
FileVersion: 0.37.213.27
LegalCopyright: Intend (C) 2013
ProductName: Lipreading Fenced
ProductVersion: 0.144.212.113
Relocatable: false
SectionAlignment: 4096
Additional Information
----- Loading /home/f0wl/Malware/Locky/Locky -----
Locky: failed to create IconResource at 007b9630: Conflicting data exists at address 007b8df0 to 007b
Locky: failed to create IconResource at 007ba6b0: Conflicting data exists at address 007b9e70 to 007b
Locky: failed to create MenuResource at 007b7b78: Conflicting data exists at address 007b7b28 to 007b
Locky: failed to create GroupIconResource at 007b9618: Conflicting data exists at address 007b8df0 to
Locky: failed to create GroupIconResource at 007ba698: Conflicting data exists at address 007b9e70 to
Searching for referenced library: USER32.DLL ...
Unable to find external library: USER32.DLL
Searching for referenced library: RASAPI32.DLL ...
OK

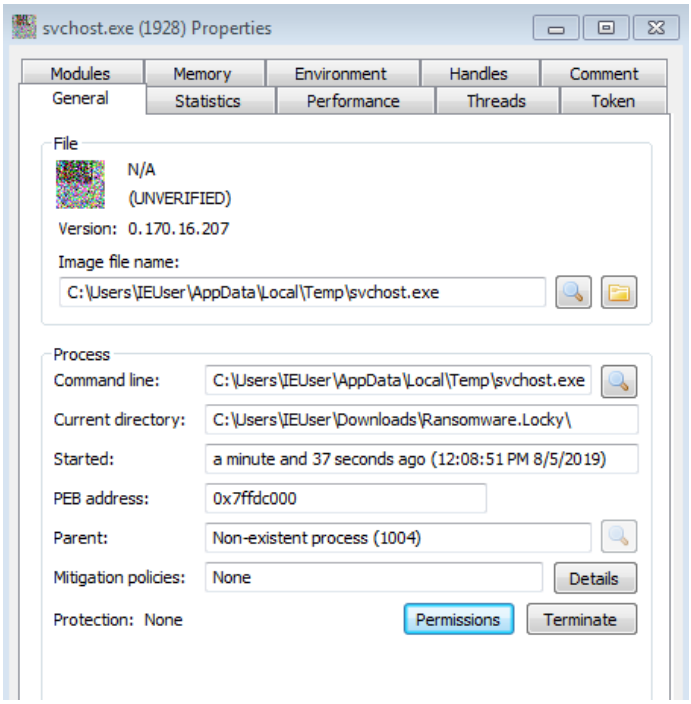
Would you look at that! We found ourselves some poor mans obfuscation :D A whole bunch of random strings to make the analyst's life just a little bit harder. We'll come back to this later to see if we can simplify our strings output a bit.



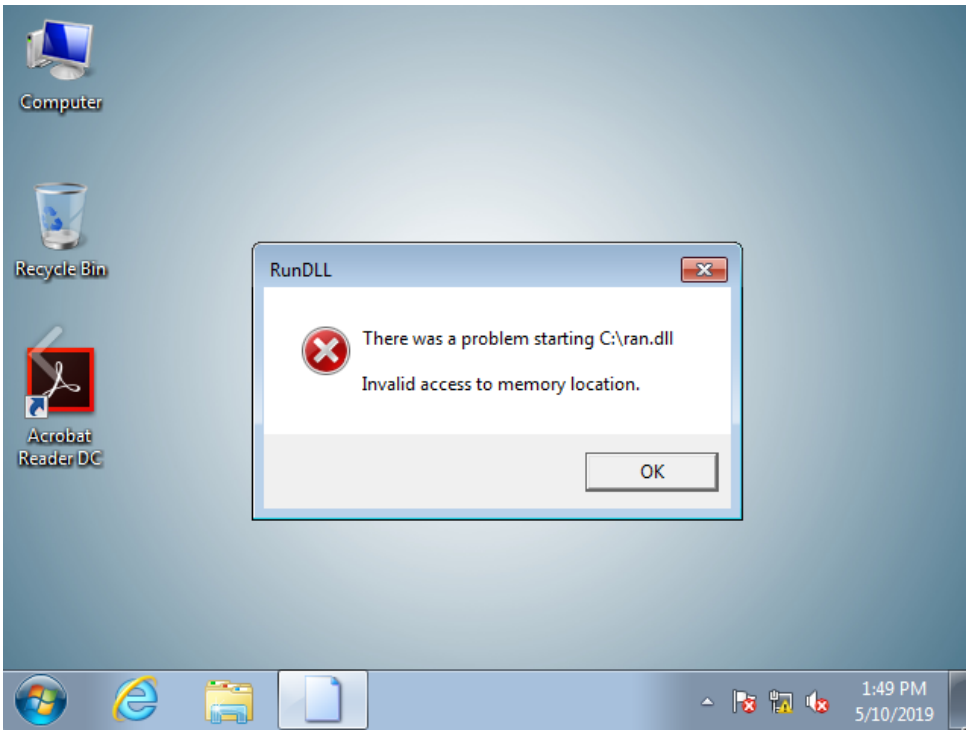
After a couple of seconds it spawns a new svchost.exe process with the same icon as Locky.exe had previously. Of course we'll dump the process memory to a file (just right-click the listing in Process Hacker and choose *Create dump* from the context menu).



Looking at the properties of the new *svchost.exe* process we can see that it is actually run from C:\Users\IEUser\AppData\Local\Temp\ and it's unsigned as well.



Trojan.Ransom.Locky.AZ



https://www.hybrid-analysis.com/sample/2e4319ff62c03a539b2b2f71768a0cfc0adcaedbcca69dbf235081fe2816248b/5cd5813d028838383d3ab408

This article is a work in progress, updates going to follow soon

IOCs

Locky (SHA256)

```
2e4319ff62c03a539b2b2f71768a0cfc0adcaedbcca69dbf235081fe2816248b
5ed2f09e648dca8f0ca75466b1442f6e599afddc80777e0559fb6881c6cd9ff3
3b60fde281d91cc3e7ea3e343ee5b13a31def564903c0136ae928f70e25c3c02
6afc78b5630726c907a69d62a6c8a7d86326e21383fe3aae1efc715342238e02
```