Free Tool: LooCipher Decryptor

marcoramilli.com/2019/07/13/free-tool-loocipher-decryptor/

View all posts by marcoramilli

July 13, 2019

There are many ways to fight cyber-crime, but what we used to do in Yoroi is Malware analysis and Incident response by using special and proprietary technologies. Often analyses are enough to temporary block cyber-criminals by sharing to everybody IOC allowing National and International players (ISP, AV vendors, CERTs and so on) to block connections or to trash files. But sometimes when a ransomware hits a victim, the ultimate

desire is to be able to decrypt those files and to restore the last consistent data set. Today we realized this ultimate scope and we want to release it public, to everybody needs a decryptor for LooCipher.

At the beginning of July Yoroi Z-Lab Team publicly released a quite exhaustive report about LooCipher (available here). The initial vector (through Microsoft Office Macro) was foreshadowing an important spread over the next few days. Indeed few days later Fortinet researchers released a nice report on LooCipher (available here) mostly focused on the encryption algorithm, where they discussed it through a python POC.

> "LooCipher starts its encryption routine by generating a 16-byte data block with random characters chosen from the following predefined characters, using the current system time as seed. "
>
> *From Fortinet report*

Most of the LooCipher technical features are described as follows:

- The ransomware spreads using weaponized Word document.
- The Command and Control is hosted on the TOR Network, at the following onion address "hxxp://hcwyo5rfapkytajg[.]onion" .
- The attackers leverage several Tor2Web proxy services to easily allow the access to the Tor C2.
- The binary can work both as cryptor and decryptor.
- The C2 dynamically generates a different Bitcoin address for each infection.

The Fortinet researches spent time in describing the used algorithm (AES-256-ECB) and portrayed a decryption code as showed in the following image. By focusing on how to recover the obfuscated key – which was retrievable either via network or via memory dumping Fortinet researchers gave us the right reading key to be able to write our own decryptor code.

```
import sys
import re
import os

keychars = ['!','@','#','%','&','?','+','-','*','/','=','_','0','1','2','3','4','5','6','7','8','9','A','B','C','D','E',\
            'F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','a','b','c','d','e','f',\
            'g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

reps =     ['00','01','02','03','28','60','46','48','21','15','53','38','62','68','31','13','24','55','42','50','06','11',\
            '22','45','14','08','30','58','70','44','43','35','32','66','09','54','12','04','19','64','17','40','71','25',\
            '36','69','10','49','67','05','26','51','07','34','56','61','20','65','41','37','57','33','29','63','47','16',\
            '39','27','52','18','59','23','72','73']

def main():
    if sys.argv.__len__() == 2 and sys.argv[1].__len__() == 32:
        print('k: %s' % sys.argv[1])
        lst_encoded_key = re.findall('.{1,2}', sys.argv[1])
        key = []
        for ch_encoded in lst_encoded_key:
            indx = reps.index(ch_encoded)
            key.append(keychars[indx])
        print('LooCipher AES-EBC key: %s' % ''.join(key))

    else:
        print('usage: %s <k-value>' % os.path.basename(sys.argv[0]))


if __name__ == '__main__':
    main()
```

From Fortinet Report

The turning point was on the way the key was encoded. The obfuscation method was quite trivial. It consists in a simple find-and-replace of each key characters with a pre-defined double-digit number, belonging to the following set:

```
'00', '01', '02', '03', '28', '60', '46', '48', '21', '15', '53', '38', '62', '68', '31', '13', '24', '55',
'42', '50','06', '11', '22', '45', '14', '08', '30', '58', '70', '44', '43', '35', '32', '66', '09', '54', '12',
'04', '19', '64','17', '40', '71', '25', '36', '69', '10', '49', '67', '05', '26', '51', '07', '34', '56', '61',
'20', '65', '41', '37','57', '33', '29', '63', '47', '16', '39', '27', '52', '18', '59', '23', '72', '73'
```

Decoding Matrix

So once retrieved the obfuscated key it was possible to reconstruct the original key to decrypt all involved files.

The master key is available directly in memory into LooCipher segments. So please remember to not kill the process even if you have been infected or to not reboot your windows box. If you kill the process or reboot your system, ZLab Team decrypter is not going to work there. You can download it **HERE** and use it for free.

1. Find the Process ID of the LooCipher ransomware
2. Open cmd with the Administrator privileges in the path where is downloaded the tool
3. In cmd prompt: ZLAB_LOOCIPHER_DECRYPTION_TOOL.exe <PID>

Happy recovery !