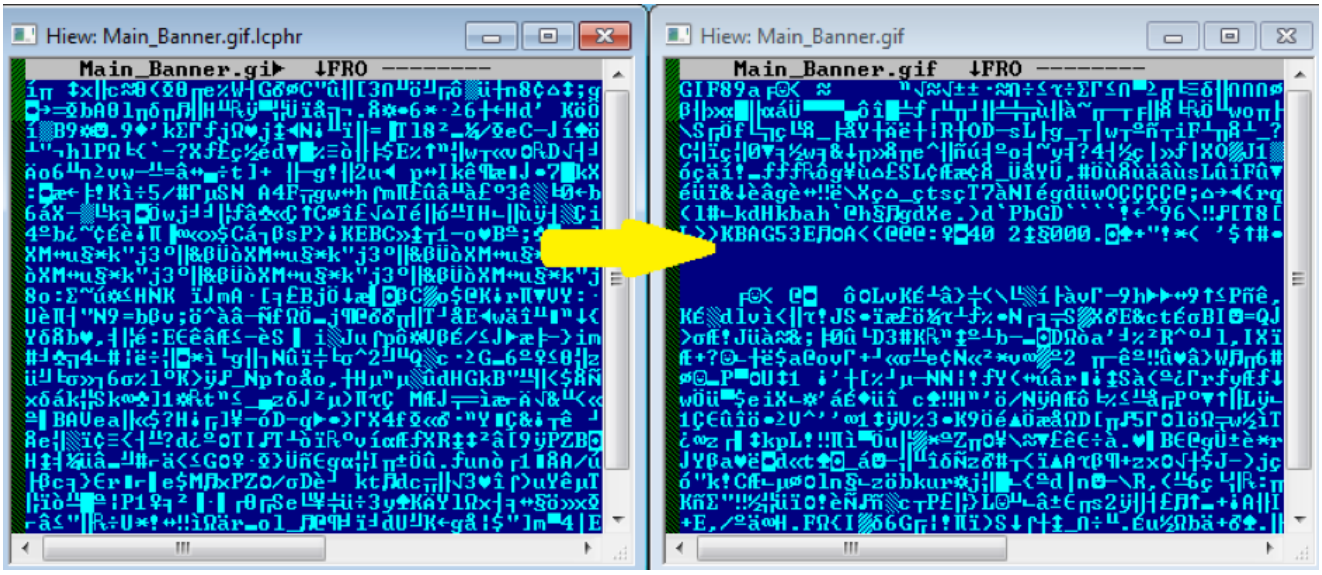
LooCipher Ransomware: Can Encrypted Files Be Recovered?

fortinet.com/blog/threat-research/loocipher-can-encrypted-files-be-recovered.html

July 10, 2019



LooCipher is a file-encrypting ransomware distributed in the wild. While there have been articles discussing its main behavior, how this type of ransomware is spread, and how it communicates with its command and control server to send victim machine information, this blog will focus on LooCipher's file encryption mechanism and take a look at the possibility of decrypting affected files without paying the ransom.

Despite the suggestive implication of its being hellish (the name sounds like Lucifer), this malware is actually pretty straightforward. For example, it doesn't use any obfuscation. However, due to its use of high-level libraries such as Crypto++ for its encryption functions, it's a bit more difficult to reverse engineer compared to those ransomware variants that use low-level Windows APIs.

How Does the LooCipher Ransomware Encrypt Files?

Although our FortiGuard Labs team found several encryption codes in the body of the malware – such as DES (Data Encryption Standard), AES (Advanced Encryption Standard), and ECC/ECDSA (Elliptic Curve Cryptography or Elliptic Curve Digital Signature Algorithm) – in our test, only the AES-128 ECB mode (Electronic Codebook) was used to encrypt files. However, it is possible that since LooCipher was only recently discovered it might be still in the initial stage of development, and that those other encryption codes are there for future use.

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.

Fig. 1. Predefined characters used in generating random 16-byte data block
This data block is then shuffled to form the 16-byte key that this ransomware uses for encrypting files with the AES-ECB encryption algorithm. This key is used for all file encryption. This is unlike most types of ransomware, which generate a different key for each file they encrypt.

Fig. 2. Generated 16-byte AES key
The following directories are excluded from its file encryption routine to prevent corrupting critical files used by the Windows operating system to start and work properly.

Fig. 3. Directories exempted from file encryption
It then searches all attached drives to encrypt files with the following extension names.

Fig. 4. LooCipher encrypts files with specific extension names
When a target file has been found, LooCipher creates a file with the original name of the file being encrypted, then adds a .lcphr extension to the name. It then encrypts the content of the file with the AES-128 ECB algorithm using the generated 16-byte random key and writes it to the newly created file with the .lcphr extension and leaving the original file as a 0-byte file.

Fig. 5. Encrypted files have the .lcphr extension name
We validated this finding by creating a simple AES-128 EBC mode decryption code in Python and then using the code to decrypt several files that had been encrypted with the generated key.

Fig. 6. Decrypting an encrypted file using AES-128 ECB mode

Fig. 7. Successfully decrypting the encrypted file

Can Ransomware Encrypted Files Be Recovered?

The version of LooCipher we analyzed only uses the AES-128 ECB mode to encrypt files. Since it is executed in ECB mode, it doesn't need an IV (initial vector) and only uses a 16-byte key, which is randomly generated from the following 74 characters:

Fig. 8. Predefined 74 characters from which a random key is generated
AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. So, in order for us to recover the files, we just needed the key used for the encryption.

This is also unlike other types of ransomware that use both symmetric and asymmetric encryption, where the private key – which is held only by the attacker – is required to decrypt the files.

Since LooCipher only generates a single key for all file encryption, and it only chooses from the 74 characters shown above to generate the key, one might think that it's easier to recover the key using crypto attacks compared to other types of ransomware. But it turns out that this is not the case.

To recover the key using brute force, we must first have the original file to compare to the decrypted file. Then, we must test all possible combinations that can be made from these 74 characters. This requires performing 7416 = 808,551,180,810,136,214,718,004,658,176 (808 Octillion) AES-128 ECB operations, which will take an impractically long period of time, even on a supercomputer.

There is a chosen plaintext attack on AES-128 ECB mode that can be used to decrypt the ciphertext without breaking the key. But this requires a cryptographic oracle that is always running, which in this case should be the LooCipher process. However, LooCipher only performs a single run of its encryption routine, so this isn't possible. Also, this method brute-forces each byte by iterating through all possible values and comparing the outcome to a reference value which, again, will take an enormous amount of time since LooCipher doesn't just encrypt one file.

Key Recovery from C2 Communication Traffic

A capture of network traffic during ransomware attacks can really be very helpful, especially for ransomware like LooCipher that sends the encryption key to a C2 (command and control) server where the threat actors keep a database of these keys.

LooCipher sends a victim ID (*u*), the encoded AES key (*k*), and the IP address (*i*) of the machine to the C2 server.

Fig. 9. Data sent to the C2 server
Since AES is a symmetric-key algorithm, it turns out that we only need to decode the value of *k*. Fortunately, *k* is just encoded with some kind of position encoding. Each character in the key is represented by a value depending on the character's position in the array/string.

Fig. 10. Key representation using position encoding
The python code below shows how to decode the value of *k*.

Fig. 11. Script to decode LooCipher AES key
In the network capture above, the value of *k* is "69604607186414680318386143262470" which is a representation of the original AES key "X?+evRC1%v_hIc4G".

Fig. 12. Output showing the decoded AES key
Using this information, we may be able to decrypt the encrypted files using the following script.

Disclaimer: Please be aware that while all scripts here were written with the intention of helping users recover their encrypted files, you must use them at your own risk.

Fig. 13. Decrypting encrypted files
However, it is unlikely that someone would capture the traffic all the time, so this method may not always be useful.

Key Recovery from Memory of a Running LooCipher Process

For this option to work, the first instance of LooCipher should still be running. If any AV tool has removed LooCipher, and its process has been terminated, the key won't be recovered from memory as LooCipher uses the current time as the seed to generate the key. However, if LooCipher is still running, we can extract the key from its process memory.

We start with using Sysinternals ProcessExplorer to create a full dump of the LooCipher process memory. LooCipher uses a randomly generated-looking process name.

Fig. 14. Creating a full dump of LooCipher's process memory
After dumping the memory, we can use *PowerShell* to search for the generated URLs. Just type the following command:

Select-String –Encoding Unicode –Path <memory dump file> -Pattern '(ttps?://./k.php.*o=[0-9])' –AllMatches | %{$_.Matches} | %{$_.Value}*

Note: If Unicode encoding didn't work, use BigEndianUnicode.

Fig. 15. Searching for the generated URLs using PowerShell
As can be seen above, the command gives us several references to the encoded key.

Sysinternals Strings and *findstr* command can also give us these strings with references to the encoded key.

Fig. 16. Searching for the generated URLs using Strings and FINDSTR
After getting the URL, we can now use the script provided in Fig. 13 to decode the key and decrypt the affected files.

Looking at the Infection Timestamp to Generate Keys

While we believe that it's possible to generate keys based on the infection timestamp, we haven't completed any proof of concept yet. That said, we are looking into this and hope to publish the information as it relates to similar ransomware families that use the current time

as the seed for generating random keys.

Conclusion

LooCipher currently only uses AES-128 with ECB mode, but since it may still be in the initial stages of development, and because those other encryption algorithms are already present in its body, this may only be temporary and those other encryption algorithms are likely to have been put there for future use.

However, until that changes, we have shown that there is still a chance to recover LooCipher encrypted files. AES is a symmetric-key algorithm. With a network capture during the infection, a skilled analyst can extract the key from the data that is being sent to the attacker. If network traffic was not captured but LooCipher is still running, they can extract the key from the memory and then use that key to recover the files using AES-128 ECB.

As it seems like LooCipher is still in the initial development stage, we will keep an eye out for any further developments.

-= FortiGuard Lion Team =-

Solution

Fortinet customers are protected by the following:

- Samples are detected by W32/Filecoder.NWG!tr signature
- FortiSandbox rates the LooCipher's behavior as high risk

IOCs

Sha256

924cc338d5d03f8914fe54f184596415563c4172679a950245ac94c80c023c7d – W32/Filecoder.NWG!tr

C2

hxxps://hcwyo5rfapkytajg.onion.pet
hxxps://hcwyo5rfapkytajg.darknet.t
hxxps://hcwyo5rfapkytajg.onion.sh
hxxps://hcwyo5rfapkytajg.onion.ws
hxxps://hcwyo5rfapkytajg.tor2web.xyz

Learn more about FortiGuard Labs and the FortiGuard Security Services portfolio. Sign up for our weekly FortiGuard Threat Brief.

Read about the FortiGuard Security Rating Service, which provides security audits and best practices.