

# A Technical Look into Flotera Ransomware

Adrian (Shuai) Li

Purdue University  
li3944@purdue.edu

## 1 Introduction

Ransomware has been wreaking havoc since the mid-2000s. 2016 was the year when ransomware reigned, with 247 new ransomware families [15]. All can render the victim's system unusable by encrypting important files and then asking the user to pay a ransom to revert the damage. Improved ransomware variants continue to be seen since then, often with devastating results. Today's ransomware families encrypt the files and communicate with Command-and-Control servers for possible data exfiltration. This report will cover a behavioral analysis done on a strain named "Flotera". The "Flotera" strain is evolved from the "Polski" and "Vortex" strains ("Polski" was the first on the scene). This report will cover one of the samples of Flotera, which was available through HybridAnalysis. The goal of this report is to present the static and dynamic behavior analysis of Flotera ransomware. More specifically, this report will answer the following questions.

- Does Flotera use any techniques for defense evasion?
- Does Flotera use any techniques that establish a foothold onto the system?
- Does Flotera target certain types of files in certain folders?
- What encryption scheme does Flotera use?
- Does Flotera communicate with its C2 servers?

## 2 Related work:

***Ransomware detection and analysis:*** As the new variants easily evade signature-based mechanisms, researchers try to develop alternative approaches based on analyzing ransomware behaviors. The key insight is that ransomware must tamper with a user's file to mount a successful attack. Kharaz et al. [10] proposed the UNVEIL system that automatically creates an artificial user environment and monitors how ransomware interacts with that environment. Researchers are also looking into the detection of ransomware using machine learning [1, 11, 13]. These approaches select the relevant features that characterize the ransomware behavior and then classify each newly-installed application through a machine learning algorithm.

Today, an important enabler for behavior-based ransomware detection is dynamic analysis. The ransomware sample is launched in an isolated, controlled environment, and its behavior is observed during run time. The study of Flotera adopts other ransomware analysis, such as the one done on "Maze" ransomware by Bitdefender [3]. This research will be based on the previous ransomware project to build an idea of what behaviors to expect.

***Polski-Vortex-Flotera*** According to Bleeping Computer [4], this ransomware family primarily targeted polish users. First on the scene was the Polski ransomware, which was first detected in late January 2017. The Polski used a Sigaint email address for payments, and because the Sigaint email service went down in Feb 2017, the attacker created a new strain named Vortex in March. Vortex used the same ransom note but dropped the ransom from \$249 to \$199. Flotera strain was detected in live infection later in April.

The polish security researchers from Zaufana Trzecia Strona (ZTS) [8] claim that the ransomware was not spread via email campaigns. Instead, the attacker used the remote access trojan vjworm to access the victim’s computer and then install the ransomware by hand.

Mainly three individual security researchers analyzed the ransomware family:

1. Adam Haertle: Haertle published a comprehensive report about “Vortex” strain on the polish website ZTS [8]. The analysis covered five areas: registry key modification for persistence, session key generation, C2 communications, file scanning, and the encryption process mechanism. He stated that the ransomware would encrypt files in a list of folders. The ransomware encrypts the files using AESxWin with an AES key obtained from a publicly available key generation service. The encrypted files are renamed with the aes extension. The progress of its activity is written in a log file in the victim’s system. After getting all the data, the ransomware sends the data to its C2 server, including the encryption key. The report did not look at the newest strain, Flotera.
2. Catalin Cimpanu: Haertle’s report was written in Polish. An English version of the original article was published on BleepingComputer by Cimpanu [4]. It also explains the timeline of Polski, Vortex, and Flotera strains briefly.
3. Michael Gillespie: Gillespie had created a decrypter for the Vortex ransomware that he offered the decrypter to victims in private [5].

### 3 Methodology

This report uses the Flotera sample through HybridAnalysis. The SHA256 hash value of the sample is “8ad4d1f7b46b5f6d28f3e4206a1263bdb88808f39061602bcca4d4e9e61170d0”. The analysis was performed on a designated lab environment consisting of two isolated virtual hosts running on an access-restricted physical device. Section 3.1 will go over the environment setup in detail. Approaches used in the analysis include static analysis and reverse engineering, dynamic analysis with debuggers. The lab environment, including the sample and any installed applications, will be destroyed to prevent the unexpected spread of the ransomware.

#### 3.1 Lab environment

The setup uses Parallels hypervisors to build two hosts-only machines and configures them in the same LAN (Figure 1). Parallels supports system snapshots

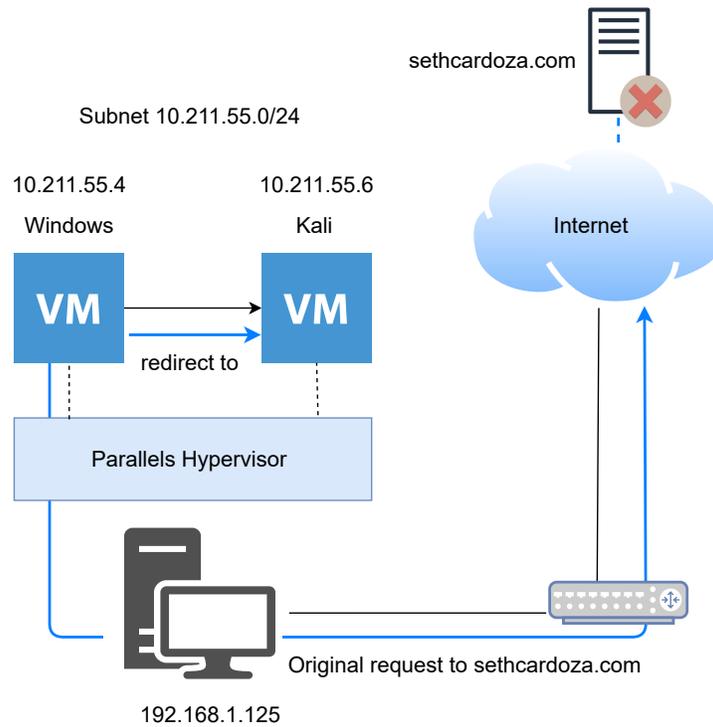


Fig. 1: Analysis lab environment

where the state of the VM is captured. This feature enables us to roll back the VM to a clean state after each analysis round. In analysis round 4, the hypervisor was set to the shared networking mode, and the host-only mode was used for round 5. In the shared networking mode, each virtual machine has full internet access. It is necessary for the analysis because the ransomware will call a public password generator service. In the host-only network mode, each VM can only communicate to other VMs.

To run the sample, one VM was installed with Windows 10 Home 64-bit Edition. The Windows Security features were all turned off, including virus & threat protection, ransomware protection. Even so, Windows can still detect the Flotera ransomware during dynamic analysis. This issue was resolved by manually add the Flotera ransomware into the allowed threats in Windows security. A fake user space was set up with fake user files of different sizes.

A second VM is built using Kali Linux 2021.1 release, which includes additional implementation and will act as a server that provides a required service by Flotera. Section 3.3 will explain why a second VM is need and how the service is implemented on this VM.

## 3.2 Analysis

*Round 1:* First, a snapshot of the system was taken before the first run. After running the sample, a CMD terminal opened, and a single process was noticed. However, no files were encrypted. The ransomware was executed again in a clean system state with admin privileges. Still, the sample was not able to encrypt the files. A possible reason could be that the sample was corrupted. It is also likely that the sample may require an internet connection for proper execution. Further investigation is needed to account for the encryption failure.

*Round 2:* File property analysis takes place in this round. Initially, the sample was loaded into Detect it Easy (DIE) tool [9] to retrieve general information about the sample. The results show that the sample was written in C# on .NET platforms. No obfuscation or packer use was identified. It means that the binary executive can be effectively decompiled into a higher-level representation similar to the source code.

*Round 3:* In this round, the sample was run through dnSpy debugger [7], which can debug and edit .NET assemblies. As expected, the translation of the binary executable was extremely close to the source code. One goal of this round was to retrieve some details about the code. For example, are there any strings in the code, such as hard-code C2 server URLs or hardcoded encryption key? This round also aimed to identify the flow of function calls and prepare for dynamic analysis. The sample was not executed in this round.

*Round 4:* In this round, dnSpy was used and set breakpoints at function calls. Debugging the sample allowed us to control the execution of the ransomware and monitor the variable content and the stack view. In this round, the debugging process helped to demystify the encryption failure in Round 1. The ransomware obtains a 120-character password from a public random number generation service hosted at domain sethcardoz.com. Then, the ransomware transforms this password into a key, which is later used for file encryption.

The problem is that this service is no longer accessible for some unknown reasons. As a result, the ransomware will terminate and throw a "server not found" exception, making it impossible to debug and observe the entire runtime behavior. Cicala suggested building up a replicated server providing the same service and directing the messages to the replica using DNS spoofing attacks. However, this option only works if we can find out the exact implementation of the original server. Luckily, the owner of the domain sethcardoz.com posted the server implementation on his Github. The replicated server was set up on the Kali VM. The research could redirect the traffic from the sethcardoz.com to the replicated server using DNS spoofing attacks.

*Round 5:* In this round, debuggers were used again to analyze the encryption scheme. File deletion, encryption, and modification were observed in the final round.

```

adrian@Adrian:~/Downloads
File Actions Edit View Help

---(adrian@Adrian)-[~/Downloads]
└─$ sudo bettercap
[sudo] password for adrian:
bettercap v2.30.2 (built for linux amd64 with go1.15.8) [type 'help' for a list of commands]

10.211.55.0/24 > 10.211.55.6  * set arp.spoof.targets 10.211.55.4;arp.spoof on
10.211.55.0/24 > 10.211.55.6  * [00:40:25] [sys.log] [inf] netspoofer starting net.recon as a requirement for arp.spoof
10.211.55.0/24 > 10.211.55.6  * [00:40:25] [sys.log] [inf] netspoofer arp spoofer started, probing 1 targets.
10.211.55.0/24 > 10.211.55.6  * [00:40:25] [endpoint.new] endpoint 10.211.55.2 detected as 8e:85:90:62:5f:64.
10.211.55.0/24 > 10.211.55.6  * [00:40:25] [endpoint.new] endpoint 10.211.55.4 detected as 00:1c:42:48:f3:3b (Parallels, Inc.)
0.
10.211.55.0/24 > 10.211.55.6  * set dns.spoof.domains sethcardoza.com; set dns.spoof.address 10.211.55.6; dns.spoof on;
00:40:55] [sys.log] [inf] dns.spoof sethcardoza.com -> 10.211.55.6
10.211.55.0/24 > 10.211.55.6  * [00:41:20] [sys.log] [inf] dns.spoof sending spoofed DNS reply for sethcardoza.com (->10.211.55.6) to 10.211.55.4 : 00:1c:42:48:f3:3b (Parallels, Inc.).
10.211.55.0/24 > 10.211.55.6  * [00:41:31] [sys.log] [inf] dns.spoof sending spoofed DNS reply for www.sethcardoza.com (->10.211.55.6) to 10.211.55.4 : 00:1c:42:48:f3:3b (Parallels, Inc.).
10.211.55.0/24 > 10.211.55.6  * [00:51:56] [sys.log] [inf] dns.spoof sending spoofed DNS reply for www.sethcardoza.com (->10.211.55.6) to 10.211.55.4 : 00:1c:42:48:f3:3b (Parallels, Inc.) - SHUAI99DD.local.
10.211.55.0/24 > 10.211.55.6  * [00:51:56] [sys.log] [inf] dns.spoof sending spoofed DNS reply for www.sethcardoza.com (->10.211.55.6) to 10.211.55.4 : 00:1c:42:48:f3:3b (Parallels, Inc.) - SHUAI99DD.local.
10.211.55.0/24 > 10.211.55.6  * [07:05:37] [sys.log] [inf] dns.spoof sending spoofed DNS reply for www.sethcardoza.com (->10.211.55.6) to 10.211.55.4 : 00:1c:42:48:f3:3b (Parallels, Inc.) - SHUAI99DD.local.
10.211.55.0/24 > 10.211.55.6  * [07:05:40] [sys.log] [inf] dns.spoof sending spoofed DNS reply for www.sethcardoza.com (->10.211.55.6) to 10.211.55.4 : 00:1c:42:48:f3:3b (Parallels, Inc.) - SHUAI99DD.local.
10.211.55.0/24 > 10.211.55.6  *

```

Fig. 2: Redirecting the traffic from sethcardoza.com to 10.211.55.6

### 3.3 DNS spoofing

DNS translates domain names to IP 'addresses' so browsers can load Internet resources. The process of translation/DNS resolution involves interaction with DNS servers. DNS spoofing is an attack of subverting the resolution of DNS queries so that the DNS server responds with a different IP address pointing to a site controlled by the attacker. There are mainly two ways of achieving this,

- MITM: Attackers intercept communication between a user and a DNS server and provide a different IP address pointing to a malicious site.
- Rogue DNS server: Attacks hack a DNS server and change DNS records to return a different IP address pointing to a malicious site.

Consequently, when the user intends to visit a legitimate website such as google.com, they will be redirected to a malicious site such as google.attacker.com.

Recall in Section 3.2: the research identified the issue of server no longer found during ransomware execution. More specifically, the ransomware calls URL “http://www.sethcardoza.com/api/rest/tools/random\_password\_generator/length:120/complexity:alphaNumeric” to get the password. Although this URL does not work anymore, the owner of the sethcardoza.com domain published the implementation of its backend API on their Github [12]. It implements a function of random strings generation over an alphabet consisting of English letters, 0-9 numbers, and special characters in PHP.

The idea is to set up a fake server running the random string generation function and redirect the traffic from the original URL to our fake server using DNS spoofing. There are many tools available for DNS spoofing attacks. The research uses bettercap [2], and it performs DNS spoofing attacks with the MITM approach. The remaining steps are straight forward: a PHP server was set up and running at IP address 10.211.55.6(Kali VM), the path of the server is “10.211.55.6/api/rest/tools/random\_password\_generator/length:120/complexity:

Table 1: Registry keys created by Flotera

Registry Key	Value Name	Value Data (String)
Directory\Background \shell\AESxWin	Empty icon	Encrypt\Decrypt with AESxWin Application.ExecutablePath
Directory\Background \shell\AESxWin\command	Empty	Application.ExecutablePath\“%V\”
Directory\shell\AESxWin	Empty icon	Encrypt\Decrypt with AESxWin Application.ExecutablePath
Directory\shell\AESxWin \command	Empty	Application.ExecutablePath\“%1\”

alphaNumeric”. The bettercap was installed on the Kali machine. Bettercap requires three parameters for spoofing attack: the target IP (Windows VM IP: 10.211.55.4), the spoofing domain (sethcardoza.com), and the IP address in response to DNS query (PHP server IP: 10.211.55.6). Figure 2 shows the activity of bettercap dns.spoof in the presence of DNS query sent from the ransomware during runtime.

## 4 Analysis results

### 4.1 Persistence

At startup, the ransomware registers itself in Windows Registry under HKEY\_CLASSES\_ROOT\DIRECTORY. HKEY\_CLASSES\_ROOT is the top-level system key that holds information about installed applications and their associate file extensions. The ransomware achieves two goals by creating the registry keys shown in Table 1. First, the system loads the setting of the registry into memory whenever it restarts. Second, the ransomware checks the stored registry keys to find its configuration setting at startup.

### 4.2 File scanning and backup deletion

Before enumerating files, any existing Windows backups are destroyed, namely the Volume Shadow Copies. This is done using windows utilities called VSSADMIN.EXE.

The following folders are searched for files to encrypt. These folders are added to **StartPaths** list.

- Environment.SpecialFolder.Personal
- Environment.SpecialFolder.Recent
- Environment.SpecialFolder.MyPictures
- Environment.SpecialFolder.MyMusic
- Environment.SpecialFolder.MyVideos

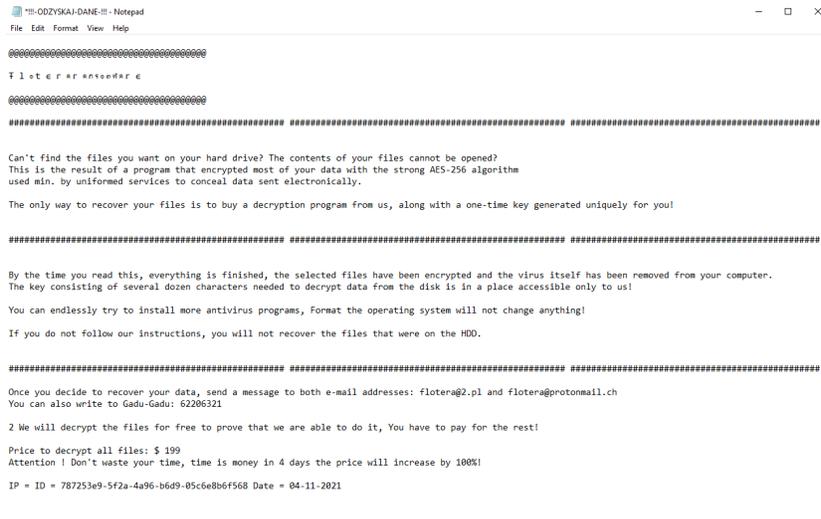


Fig. 3: Ransom note

- Environment.SpecialFolder.Favorites
- Environment.SpecialFolder.CommonDocuments
- Environment.SpecialFolder.CommonPictures
- Environment.SpecialFolder.CommonMusic
- Environment.SpecialFolder.CommonVideos
- Environment.SpecialFolder.CommonDesktopDirectory
- and all drives RootDirectory

The following folders are excluded from encryption:

- Application.ExecutablePath
- Environment.SpecialFolder.ApplicationData
- Environment.SpecialFolder.LocalApplicationData
- Environment.SpecialFolder.CommonApplicationData
- "C:\\Program Files\\Common Files"

The ransomware first skips the folder where the executable is located. The reason why the rest folders get skipped can be because the attacker wants the system to keep function properly.

### 4.3 Encryption

For each directory in StartPaths, the ransomware will encrypt files in the directory recursively except those that end with .aes or unsupported extensions. The details of the target extension list are not included due to the page limit. The original file is replaced by the encrypted data in place. For this, the data is read from the file, data is encrypted, then the encrypted data is written back to the

file, and finally, the file is closed and its name is appended with .aes extension. The ransomware drops a ransom note named !!!-ODZYSKAJ-DANE-!!! in each encrypted folder. The ransom note includes the victim's IP and current time. Figure 3 shows the English translation of the original ransom note in Polish.

#### 4.4 File encryption

Flotera uses SharpAESCrypt library [14], which is a C# implementation of file encryption and decryption using AESCrypt file format [6].

The AESCrypt file format is shown below. This was validated with the encrypted files through Hex editor.

- 3 Octets - 'AES'.
- 1 Octet - 0x02 (Version).
- n Octet - extension block section. This is where some user-defined "tags" that may be inserted as plaintext into the encrypted file.
- 16 Octets - Initialization Vector ( $IV_1$ ) used for encrypting the  $IV_2$  and symmetric key ( $Key_2$ ) that is actually used to encrypt the plaintext file.
- 48 Octets - Encrypted  $IV_2$  and 256-bit  $Key_2$  using AES  $Key_1$ . The first 16 octets are  $IV_2$ , followed by 32 octets of  $Key_2$ .
- 32 Octets -  $HMAC(Encrypted\ IV_2\ and\ Key_2, Key_1)$ .
- mn Octets - Encrypted message.
- 1 Octet - File size modulo 16 in least significant bit positions.
- 32 Octets -  $HMAC(Encrypted\ message, Key_2)$ .

The HMAC values protect data integrity. The following section shows how Flotera generates these keys and initial vectors for encryption.

#### 4.5 Encryption keys

Figure 4 shows the keys generation and file encryption. New IVs and keys are generated for different files.  $IV_1$  is generated by DigestRandomByte(byte[] byte, int repetitions) function, which performs repeated hashing of the data in the Bytes() combined with random data. The time and the victim's MAC address are components in the input. AES  $Key_1$  is generated by hashing the concatenation of  $IV_1$  and password 8192 times. The password parameter is the random string obtained from sethcardoz.com.

$IV_2$  is randomly generated using DigestRandomByte() with a predefined IV value. AES  $Key_2$  is randomly generated using DigestRandomByte() with a predefined key value.

The  $Key_2$  is written in the processed file encrypted with the  $Key_1$ , using AES 256. The HMAC of the encrypted  $Key_2$  is stored and calculated using HMAC 256 with  $Key_1$ . The file is then encrypted using AES 256 with  $Key_2$ . Afterward, the HMAC of the encrypted content is stored and calculated with  $Key_2$ .

File decryption can be performed if  $Key_1$  is revealed. Theoretically, this can be done by hashing the password and  $IV_1$  8192 times. The research has identified the password value by examining the memory during debugging process. Decrypting the  $Key_2$  and further recovering the encrypted file is possible.

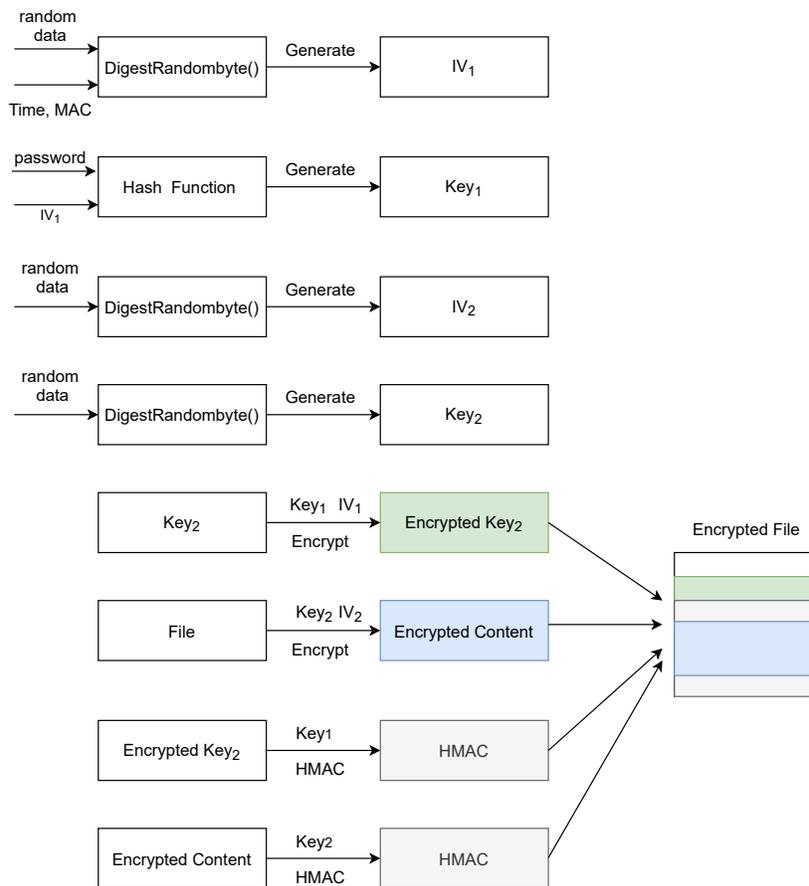


Fig. 4: Flotera Encryption

#### 4.6 Network connections

Besides calling a benign public service, the malware tries to connect to a C2 host for further instructions and possible data exfiltration. The list of contacted hosts was found in the binary.

The following outbound connections were seen from this sample: POST <http://aktualizacja.tk/api/>

The data sent to the C2 hosts is the computer fingerprint and the password. The data is sent over plain text and looks like this:

IP: 787253e9-5f2a-4a96-b6d9-05c6e8b6f568

ComputerID: 787253e9-5f2a-4a96-b6d9-05c6e8b6f568

Date: 04-11-2021

password: AzL887(jl5a3ZvUiscza\*6yWEW@WGLEI03M3gmb4)i4ICEEJ8Jj4Ê

9F\$rV)7RVl0ln(Dr)4-K0L&k\*va1PeKDJs2GCLw10(u5IrU-wPSoJ4u9TRK5xw  
&&zzC

#### 4.7 Event logging

The progress of its activity is written to a file, including the first four characters of the password, the encryption folder, the encrypted file, and exceptions during execution. The log is located at: C:\ProgramData\Keyboard. A log file example is included in Appendix A.

## 5 Conclusion

As part of our research, related work on Vortex by other individuals was used to compare our analysis on Flotera. Flotera inherits some behaviors from its predecessor, Vortex. Therefore, the research has identified some characteristics of Flotera, which match what Haertle and Cimpanu covered in their analysis.

- They both call the same publicly available internet service to request the password.
- The fact that they both encrypt files in specific folders.
- Flotera and Vortex both use AES-256 encryption.
- The same .aes extension is added to the encrypted file names.
- The same amount of ransom is charged.
- Both malware tries to connect to a C2 host. They both send the same types of data (IP, ComputerID, Date, password) to the C2 host.

On the contrary, the research has identified how Vortex evolved to be Flotera,

- Flotera uses a different encryption model, cryptographic library, and key generation strategy. Vortex uses simple AES encryption with the AESxWin library. There is no protection on the encryption key. No prior work indicates whether Vortex used different keys for different files.
- Flotera extends the lists of folders to be encrypted.
- Flotera skips system and application data folders.
- The outbound connections flow to a different C2 host.
- Flotera adds different registry keys for persistence.

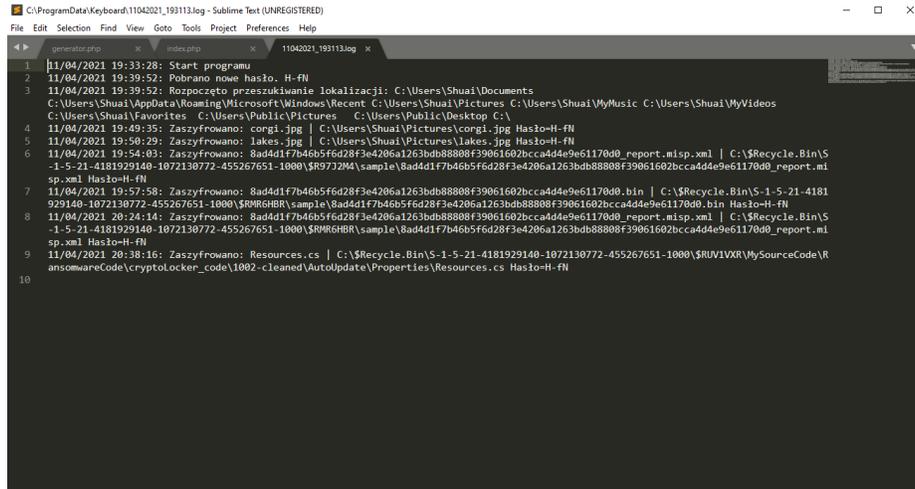
Due to time constraints, the analysis of process activities was not conducted. With the help of the process analysis before, during, and after execution, one can investigate more details on the running processes.

## References

1. Arabo, A., Dijoux, R., Poulain, T., Chevalier, G.: Detecting ransomware using process behavior analysis. *Procedia Computer Science* **168**, 289–296 (2020)
2. bettercap: Bettercap release v2.31.0. <https://www.bettercap.org/> (2021), Accessed on April 2021

3. Bitdefender: A technical look into maze ransomware. <https://download.bitdefender.com/resources/files/News/CaseStudies/study/318/Bitdefender-TRR-Whitepaper-Maze-creat4351-en-EN-GenericUse.pdf> (2020), Accessed on Feb 2021
4. Cimpanu, C.: The polski-vortex-flotera ransomware connection. <https://www.bleepingcomputer.com/news/security/the-polski-vortex-flotera-ransomware-connection/> (2017), Accessed on April 2021
5. Cimpanu, C.: Author of polski, vortex, and flotera ransomware families arrested in poland. <https://www.bleepingcomputer.com/news/security/author-of-polski-vortex-and-flotera-ransomware-families-arrested-in-poland/> (2018), Accessed on April 2021
6. Crypt, A.: Aes file format. [https://www.aescrypt.com/aes\\_file\\_format.html](https://www.aescrypt.com/aes_file_format.html) (2021), Accessed on April 2021
7. dnSpy: dnspy - latest release. <https://github.com/dnSpy/dnSpy> (2021), Accessed on April 2021
8. Haertle, A.: Nieuchwytny polski ransomware z arabskim akcentem w kocu zapany. <https://zaufanatrzeciastrona.pl/post/nieuchwytny-polski-ransomware-z-arabskim-akcentem-w-koncu-zlapany/> (2017), Accessed on April 2021
9. horsicq: Detect-it-easy. <https://github.com/horsicq/Detect-It-Easy>, Accessed on April 2021
10. Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E.: {UNVEIL}: A large-scale, automated approach to detecting ransomware. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 757–772 (2016)
11. Morato, D., Berrueta, E., Magaña, E., Izal, M.: Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and computer Applications* **124**, 14–32 (2018)
12. sethcardoza: random-password-generator/random-password-generator.php. <https://github.com/sethcardoza/random-password-generator/blob/master/random-password-generator.php> (2017), Accessed on April 2021
13. Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C.: Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020 (2016)
14. SharpAESCrypt: Sharpaescript. <https://github.com/kenkendk/sharpaescript> (2019), Accessed on April 2021
15. TrendMicro: Ransomware past, present, and future. <https://documents.trendmicro.com/assets/wp/wp-ransomware-past-present-and-future.pdf> (2017), Accessed on April 2021

## A Event logging example



```
C:\ProgramData\Keyboard\11042021_193113.log - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
11/04/2021 19:33:28: Start programu
11/04/2021 19:39:52: Pobrano nowa haslo. H-FN
11/04/2021 19:39:52: Rozpoczeto przeszukiwanie lokalizacji: C:\Users\Shuai\Documents
C:\Users\Shuai\AppData\Roaming\Microsoft\Windows\Recent C:\Users\Shuai\Pictures C:\Users\Shuai\Music C:\Users\Shuai\Videos
C:\Users\Shuai\Favorites C:\Users\Public\Pictures C:\Users\Public\Desktop C:\
11/04/2021 19:49:35: Zasyfrowano: corgi.jpg | C:\Users\Shuai\Pictures\corgi.jpg Haslo=H-FN
11/04/2021 19:50:29: Zasyfrowano: lakes.jpg | C:\Users\Shuai\Pictures\lakes.jpg Haslo=H-FN
11/04/2021 19:54:03: Zasyfrowano: 8add41f7b46b5f6d28f3e4206a1263bdb88808f39061602bccad4e9e61170d0_report.misp.xml | C:\$Recycle.Bin\S
-1-5-21-4181929140-1072130772-455267651-1000\SRMRGHR\sample\8add41f7b46b5f6d28f3e4206a1263bdb88808f39061602bccad4e9e61170d0_report.mi
sp.xml Haslo=H-FN
11/04/2021 19:57:58: Zasyfrowano: 8add41f7b46b5f6d28f3e4206a1263bdb88808f39061602bccad4e9e61170d0_bin | C:\$Recycle.Bin\S-1-5-21-4181
929140-1072130772-455267651-1000\SRMRGHR\sample\8add41f7b46b5f6d28f3e4206a1263bdb88808f39061602bccad4e9e61170d0_bin Haslo=H-FN
11/04/2021 20:24:14: Zasyfrowano: 8add41f7b46b5f6d28f3e4206a1263bdb88808f39061602bccad4e9e61170d0_report.misp.xml | C:\$Recycle.Bin\S
-1-5-21-4181929140-1072130772-455267651-1000\SRMRGHR\sample\8add41f7b46b5f6d28f3e4206a1263bdb88808f39061602bccad4e9e61170d0_report.mi
sp.xml Haslo=H-FN
11/04/2021 20:38:16: Zasyfrowano: Resources.cs | C:\$Recycle.Bin\S-1-5-21-4181929140-1072130772-455267651-1000\SRUV1VXR\MySourceCode\R
ansomwareCode\cryptoLocker_code\1002-cleaned\Autolupdate\Properties\Resources.cs Haslo=H-FN
10
```

Fig. 5: Log file example