Unreleased RaaS analysis- CashRansomware - TEHTRIS

By Katuiscia Benloukil

Published: 2024-05-31 · Archived: 2026-04-05 19:02:43 UTC

Introduction

Our experts managed to intercept a pre-release sample of a new Ransomware-as-a-Service (RaaS) variant called “CashRansomware” which is currently in active development. This RaaS is highly anticipated and already being advertised on a store for malicious software, Cashout. It is also branded on the Telegram channel of the group, dedicated to Mint Stealer, another product sold on the cashout store. We managed to have this early access to the malware thanks to advanced threat intelligence and network monitoring techniques. It will allow us to give you a unique analysis of its evolving capabilities and design strategies before it will even be used by threat actors./

In this technical analysis, we will delve into the intricate workings of a ransomware. Dissecting its architecture, attack vectors, and payload delivery mechanisms. Our objective is to provide a detailed examination of the life cycle of a ransomware attack, from the initial infiltration to the execution and propagation. By analyzing specific ransomware families and case studies, we aim to uncover common patterns and unique characteristics that define modern ransomware threats.

Ransomware-as-a-Service: one of the current biggest threats

The fall of some of the biggest ransomware families has led to the multiplication of small actors, which isn’t good news, as it multiplies the potential threats. Ransomware has become one of the most damaging cyber threats in the digital landscape. This malicious software, designed to encrypt a victim’s data or lock access to their system until a ransom is paid, is a big challenge for individuals, businesses, and government agencies worldwide. The rapid evolution of ransomware tactics and the increasing sophistication of attacks create an urgent need for comprehensive technical analysis to understand, mitigate, and ultimately neutralize these threats.

Ransomware-as-a-Service (RaaS) is a cybercrime model that allows individuals with little to no technical expertise to launch ransomware attacks. It operates similarly to legitimate Software-as-a-Service (SaaS) platforms, providing a ready-to-use ransomware toolkit and support infrastructure in exchange of a part of the profits. RaaS platforms typically offer user-friendly interfaces, customization options, and even customer service, making it easier for cybercriminals to deploy sophisticated ransomware campaigns. This model has made it significantly easier to become a cybercriminal and has led to an increase in the frequency and scale of ransomware attacks worldwide.

CashRansomware: full analysis by our experts

Our experts managed to find a CashRansom.exe sample that was investigating on our TEHTRIS Threat Intelligence feed.

The ransomware sample that was found is written in C# and suffers from poor obfuscation and inadequate cryptographic practices. Thanks to the lack of effective obfuscation, it was even easier for our security researchers to dissect the code and understand its behavior. In addition to that, the flawed cryptographic techniques used by the sample create vulnerabilities and potential weaknesses. These deficiencies undermine the ransomware's effectiveness, but they also enable decryption and mitigation by our teams. We will delve into these cryptographic flaws and their implications in greater detail later in this analysis. The software used by the sample is still in its development phase, so it is not surprising its protection and stealth are weak. The fact that this sample also leaked on open source datasets also raises concerns about their OPSEC process.

This analysis is also an opportunity to highlight the crucial steps to take when an infection occurs: cut the internet connection to stop communicating with the attackers, back up the encrypted files and the memory of the process using tools like ProcDump, then suspend the ransomware process using tools like Process Hacker to prevent further file deletion.

Samples

The analyzed ransomware sample is uniquely identified by the hashes listed below. The other samples are very similar.

Type	Value
File Type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections
DateTimestamp	2024-05-12 03:48:15
SIZE	2.8 MB (pesize = 2.8 MB)
MD5	69cc2e20ea7a51666b8c14be90441073
SHA256	958ccd8e8dcce5e7bac5f891e8edc42ad6c5497d9385c8ae26c328c5f7beda24
SHA512	de565813d0ddfe491c367e78b2a11891a73859a04efd83d8f35a4a6f6a028a29c873750dc863d1dfca9c40f9b4778cb1882bf8c07b9609f8463db22ac912922a

Figure 1: Hash of "CashRansom.exe"

Code details

The sample is developed using the .NET Framework 4.7.2 and is designed for 64-bit architectures. This file does not require any dependencies to run on Windows architectures only. It supports Windows versions from Vista to Windows 10. The debug information is stripped from this binary.

Techniques

The ransomware uses several MITRE ATT&CK techniques to carry out its malicious activities.

Figure 2: MITRE tactics distribution

Context

The pre-release sample of CashRansomware, a new Ransomware-as-a-Service (RaaS) variant intercepted by our experts, turned out to currently still be in active development.

Even if it is not released yet, it is already anticipated on Cashout, the store for malicious software.

Figure 2: Cashout products

It is also already branded on the Telegram channel of the group dedicated to Mint Stealer, another of their malicious software. It seems that there is no release date yet:

Figure 3: Ad for the release

During our investigation into CashRansomware, we delved into various underground websites and Telegram channels to map out malicious activities. This extensive research involved monitoring discussions, exchanges, and transactions among cybercriminals, allowing us to piece together the operational structure and tactics used by the developers and their affiliates. By analyzing these communications, we gained valuable insights into the distribution methods, target selection criteria, and the overall strategy behind CashRansomware's development and deployment, enabling us to anticipate and counteract this emerging threat more effectively. The following information are available in Stix format in the appendices.

Figure 4: Stix schema

The others malicious software are not in the scope of the current article. While there are no ads for this software yet, we can anticipate well-designed marketing efforts like those seen before by the same author. These campaigns are likely to use sophisticated techniques to entice users and spread the ransomware more effectively.

Figure 5: Mint Stealer branding

After an extensive OSINT investigation, it was revealed that the lead malware developer, known by the nickname “Artem,” is both a Russian and French native speaker. Artem is believed to live in the Provence-Alpes-Côte d’Azur (PACA) region of France. The details of the OSINT analysis are classified.

Figure 6: Artem

Telegram channels are used to provide customer support.

Defense

Obfuscation

The data flow and control flow of the sample are obfuscated using Eziriz .NET Reactor. This tool makes the analysis process more complicated. By obscuring the logical structure and execution pathways of the code, it is harder for security researchers to understand and reverse-engineer the ransomware.

You can see an evaluation version of Eziriz .NET Reactor in the screenshot below. This is the demo version of this software which is valid for only 14 days starting from 2024-05-11, which is coherent with the ransomware compilation time

Figure 7: Software Protection

Detection

CashRansomware is programmed to avoid infecting systems located in Russia and other Commonwealth of Independent States (CIS) countries. This selective targeting is achieved through geolocation checks and system language settings, ensuring that the malware only activates in non-Russian environments. This tactic not only reduces the risk of local law enforcement scrutiny but also indicates a potential link to cybercriminal groups operating within these regions, who often use such strategies to evade detection and prosecution by their own governments. It makes a lot of sense as long as CashRansomware is hosted in Russia and doesn’t want its domains seized, as it already happened with their old domain.

Figure 8: Country filter

The malware employs time-stomping techniques to detect the presence of a sandbox environment. By manipulating file timestamps or monitoring system clock discrepancies, it can identify anomalies that are typical of the execution of a sandbox. This allows the malware to evade detection and analysis by delaying its malicious actions until it is confident that it is not being observed within a controlled environment.

Figure 9: Time Stomping Detection

The malware includes a very simple anti-debugging function.

Figure 10: Anti debug

It also detects the presence of sandboxie and any.run sandbox based on those tricks:

```
1  GetModuleHandle("SbieDll.dll")
```

Check that this variable is set:

```
1  "%anyrun%"
```

The malware has anti VM features and can detect if the "Manifacturer" contains "microsoft corporation" or "vmware" by calling:

```
1  Select * from Win32_ComputerSystem
```

Stealth

The sample isn’t stealthy which can be attributed to its conspicuous file extension and process name, likely indicative of its early development stage. The presence of a distinct “cashransomware” file extension and process name raises red flags for security software and knowledgeable users, potentially leading to quicker detection and mitigation efforts. This lack of stealth may suggest that CashRansomware is still undergoing refinement and optimization by its developers, with improvements in evasion techniques and obfuscation measures anticipated in future iterations. The file extension is however configurable in the builder.

Execution

Initial execution

The sample is a .NET PE executable intended to be executed manually. It autonomously encrypts files on the infected system and subsequently displays a pop-up window demanding cryptocurrency payment for decryption. This automated process ensures a swift and effective attack, as the ransomware immediately locks critical data and provides the victim with clear instructions on how to purchase and transfer the required cryptocurrency to regain access to their files.

Figure 11: Ransom pop up window

In addition to encrypting files and displaying a ransom note, CashRansomware also modifies the desktop wallpaper to reinforce its presence and demand for ransom. This alteration is a visual indication of the infection, displaying a customized image or message typically containing instructions on how to pay the ransom.

Figure 12: Wallpaper

Ensuring privileges

The sample detects if it is running in administrative mode. This step is used to make sure that the malware can gain the necessary privileges to modify system files, disable security measures, and fully encrypt the victim's data. If the sample is not launched as root, the following useless command is performed with powershell. The reason behind this is not clear. Note the typo in "recure".

```
1 cmd.exe /c start computerdefaults.exe && powershell.exe Remove-Item Path HKCU:\Software\Classes\ms-settings\shell -Recure
```

Lateral movement

Every device connected to the compromised computer is systematically explored by enumerating drive letters. This process allows the malware to identify and target additional storage devices, such as external hard drives, USB drives, and network shares, to maximize the scope of its encryption campaign. By going through available drive letters, CashRansomware ensures comprehensive coverage of potential data sources, potentially extending its impact beyond the local system.

Persistence

The software copies itself to the Start Menu's Programs Startup folder, automatically executing each time the system boots up. This tactic allows the ransomware to continuously show instruction to pay.

C:/Users/admin/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/ Startup/

Configuration

By analyzing the payload builder leaked in a demonstration on Vimeo, researchers can gain insights into the configuration and operational parameters of CashRansomware. There is an authentication window prior to the builder window backing up the fact that this malware is a Ransomware-as-a-Service.

Figure 13: Cash builder Frontend

In the video, the malware developer leaked the builder console amongst other features.

Figure 14: Cash builder Frontend

Figure 15: Cash builder Builder

Sensitive data

Destruction

The sample is able to delete system restore points as part of its evasion and persistence strategy. By removing these restore points, which are often used by users to revert their system to a previous state before the ransomware infection, CashRansomware further complicates the victim's ability to recover their files without paying the ransom.

The sample doesn't erase the encrypted files after encryption by calling a simple Fileinfo .Delete(). It usually doesn't matter as a lot of the file modification might overwrite ancient file artifact. It is however a bad practice: a secured deletion or in place encryption would have totally wiped the original file with identical performances.

Command and control

Identification

The sample establishes communication with its command and control (C2) server through the encrypted messaging platform Telegram. This choice of communication channel makes it possible for the malware to operate stealthily within network traffic, leveraging the encryption and anonymity provided by the messaging service. By using Telegram, CashRansomware can receive commands, transmit encryption keys, and facilitate ransom negotiations securely, minimizing the risk of detection by security measures that may not scrutinize messaging traffic. This strategic use of a popular messaging platform shows the malware's adaptability and the evolving tactics employed by cybercriminals to evade detection and maintain operational secrecy.

As the ransomware is communicating the new victim status over Telegram, it is possible to intercept the API key from the ransomware and directly communicate with the API.

```
curl -s --socks5-hostname 127.0.0.1:9050 -A "" https://api.telegram.org /bot5990276952:AAHb30fvIHOh_d1GRVKrpfW
```

The channel is not secure enough and leaks some data about the author which uses its personal Telegram account instead of creating an account dedicated to the exchange between victims and ransomware operators.


```
{
   "ok": true,
   "result": {
       "id": 968071618,
       "first_name": "Artem",
       "last_name": "Ey",
       "username": "ArtemDotIcu",
       "type": "private",
       "active_usernames": [
           "ArtemDotIcu"
       ],
       "bio": "Cash-Hosting.PW CEO - Cashout.pw Founder | RAT, Bypassing Defender Crypter, Stealer",
       "business_location": {
           "address": "In a place where feds can't come"
       },
       "photo": {
           "small_file_id": "AQADBAADy6cxG8KZszkACAIAA8KZszkABKQxOquj8qFhNQQ",
           "small_file_unique_id": "AQADy6cxG8KZszkAAQ",
           "big_file_id": "AQADBAADy6cxG8KZszkACAMAA8KZszkABKQxOquj8qFhNQQ",
           "big_file_unique_id": "AQADy6cxG8KZszkB"
       },
       "emoji_status_custom_emoji_id": "4983570692374004583",
       "max_reaction_count": 11,
       "accent_color_id": 3
   }
}
```


Here is the Telegram profile of the developer:

Figure 16: Telegram profile

Commands

CashRansomware communicates with its command and control (C2) server primarily through a beacon containing crucial information about the infected system and the generated decryption keys. This beacon includes details such as the user's system information, unique identifiers for the infected machine, and the encryption keys necessary for decrypting the files. This method of communication ensures that the attackers maintain control over the decryption process and can monitor the status of the infected systems. By centralizing this information, the cybercriminals can manage ransom demands more effectively and provide decryption keys upon payment, reinforcing their extortion strategy. With the message, a screenshot is attached.

Figure 17: New Victim Notification Message

If no network communication is available, CashRansomware adapts by storing the decryption key in the registry, which is encrypted using asymmetric algorithm. Thanks to this, it is made sure that the key remains accessible to the attackers in order to provide it to the victim once the ransom is paid, even if the infected system can’t connect to the command and control (C2) server.

Cryptography

Keys

The encryption key and IV are generated using RNGCryptoServiceProvider which is now depreciated but still pretty secure.

The file encryption keys are then encrypted by a static Key encryption key (KEK). Using symmetric encryption for local storage is a huge vulnerability, as it makes it easier to decrypt. Instead, it should be an asymmetric encryption.

Figure 18: Key encryption routine

Because an RSA mechanism exists in the code and because the URL https://pastebin.com/raw/ azDDWzUg points to the following public key, it can be assumed that a proper key encryption will be performed in the next versions.

```
 -----BEGIN PUBLIC KEY-----
MIGeMA0GCSqGSIb3DQEBAQUAA4GMADCBiAKBgGFPnrvYFsHG3+NAFcVf4czqpdFX Of/eQyyFTUxwm4qjPJGLpm/agh5U3gUS6E5t9QHHSpN6hf3
-----END PUBLIC KEY-----
```

The found RSA key is very weak, making it vulnerable to cryptographic attacks:

```
 openssl rsa -pubin -in /dev/shm/a.asc  -noout -text
Public-Key: (1023 bit)
Modulus:
    61:4f:9e:bb:d8:16:c1:c6:df:e3:40:15:c5:5f:e1:
    cc:ea:a5:d1:57:39:ff:de:43:2c:85:4d:4c:70:9b:
    8a:a3:3c:91:8b:a6:6f:da:82:1e:54:de:05:12:e8:
    4e:6d:f5:01:c7:4a:93:7a:85:fd:e0:f2:a2:0c:81:
```

```
    b9:43:b5:34:a5:b5:4e:2b:99:61:1f:dc:2f:09:a0:
    72:bd:7d:2b:09:7a:8d:89:69:d3:92:a0:98:fc:60:
    14:cf:16:33:93:37:3e:ff:3e:44:b4:17:e4:a7:ff:
    cb:68:ff:26:f3:b7:cb:54:45:e4:fd:e4:5c:62:e0:
    95:a4:27:2c:50:a5:c5:a7
Exponent: 65537 (0x10001)
```

The ransomware includes a feature that allows the decryption of a single file, which indicates that the decryption key remains in the system's memory during this process. By leveraging memory forensics techniques, it is possible to extract this key while it is temporarily stored in volatile memory (RAM). We have developed a tool that can decrypt files encrypted by the ransomware. This tool first identifies and extracts the decryption key from the system's memory, which is temporarily stored there when the ransomware decrypts a single file. The source code is available in appendices.

Figure 19: Decryption process using TEHTRIS' tool

After a ransomware infection, backing up encrypted files and system memory is essential due to potential design flaws in the malware that could be used for file recovery. Ransomware is often developed in a hurry which can

create weaknesses in the encryption algorithm, improper key management, or bugs in the encryption process itself. By preserving the encrypted files, you maintain the original state of the data, enabling you to apply different decryption strategies or updates to decryption tools as they become available. Simultaneously, backing up the system memory captures crucial information, such as the decryption key and operational data of the ransomware, which can be analyzed to uncover vulnerabilities or methods to reverse the encryption. This dual backup approach maximizes the chances of successful data recovery and mitigates the risk of permanent data loss.

Algorithms

The files are encrypted using AES-256 CBC. The IV is a part of the decryption secret and is not included in the encrypted file, which is strange but does not jeopardize the global security level of the ransomware.

Figure 20: Encryption Routine

IOCss

Files and registry

1	HKEY_CURRENT_USER\\SOFTWARE\\Wow64Inject\\([A-F\d]{2}){32}
2	HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Tracing\\.*_RASAPI32\ EnableFileTracing
3	HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Tracing\\.*_RASAPI32\ EnableConsoleTracing
4	HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Tracing\\.*_RASAPI32\ FileTracingMask
5	HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Tracing\\.*_RASAPI32\ ConsoleTracingMask
6	HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Tracing\\.*_RASAPI32\ MaxFileSize
7	HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Tracing\\.*_RASAPI32\ FileDirectory
8	.*\.CashRansomware

Artifacts

Mutex: MVI6MT0qPLmQhQ6j

Similar samples

The following SHA256 are similar samples:

1	958ccd8e8dcce5e7bac5f891e8edc42ad6c5497d9385c8ae26c328c5f7beda24
2	b8f506741843e2c76fb207b41d205530236f4a263a9a5902146cd71a13fdfd23
3	5525d297a346b80912c4f5ec0ac4875e9d49f96d01e52c10df5c064bd803bd79
4	e387c084d5c3b62413743e912ee10776564e7c55ba1dc801990b312b88b61efe
5	39096e9a521ea1c001083d8c82317c8e6dbdd5d705d9a92beb15db102fb87263
6	e1696968ad55e7e03a8334711d90350c4145fb4f60de5fb4a2f5f19187183c05
7	6332e43af93cf00c5bd536e189b30d5a44c0568fb3fdef5e9b020146420d8b15
8	dd7d6dc03dea59e2f48ef92849ec0d03aa6cf4c5e1e6758eba184be311e6fb1f
9	003311f504eec2d0203de5e5e9d8c4213a981a3f6db85141a6d1e84a58e2b6b9
10	91edb2ed65ae31113c3c2f3ba63bcac5d24d48ef3d5765018799863e4717b845

11	b7f5f19864639b32f24e806091bf0a0a7483df73c97eac41fe5eb1d3321cea39
12	c67332d690304dfd5d43fc35dbdf0a832a803ff5f73f6187e3d94ace7433fffd
13	132ef1a933f9d26fb0bb46b0a970dbfe05ad8fe0859ece8eb973b5584a580cc3
14	8040b684f12ac819ba9ecb407f202f01182d25186552093379ad33c86f3a4273

Detection

Yara

```
import "dotnet"

rule CashRansomware {
    meta:
        author = "PEZIER Pierre-Henri. Copyright TEHTRIS 2024"
    strings:
        $str_01 = "cashransom.exe" ascii wide nocase
        $str_02 = "cashransomware" ascii wide nocase
        $func_01 = "get_icons8_code_file_100"
        $func_02 = "get_icons8_document_120"
        $func_03 = "get_icons8_inquiry_100"
        $func_04 = "get_icons8_lock_500"
        $func_06 = "get_icons8_unlocking_a_secure_web_login_for_admin_96"
        $func_07 = "get_icons8_upload_90"
        $func_08 = "get_KadavroSupp"
        $func_09 = "get_key"
        $func_10 = "get_logo_jester_done"
        $func_11 = "get_monero"
        $func_12 = "get_monero_icon_512x512_kqg9n5mp"
        $func_13 = "get_Monero_Logo_svg"
        $func_14 = "get_money_bag_coins_bitcoin_isolated_white_background_106234394_removebg_preview"
        $func_15 = "get_Pinvoke"
        $func_16 = "get_SuppID"
        $func_17 = "get_SuppToken"
        $func_18 = "get_telegram"
        $msg_01 = "Your files are heavily encrypted, and none can be decrypted without the decryption key."
        $msg_02 = "To obtain the decryption key, you need to make a payment to the specified amount to the XMR /
        $msg_03 = "Once you've made the payment, you should contact the attackers via email or Telegram to rece
        $msg_04 = "After receiving the decryption key, you need to input it into the decryption panel in Cash."
        $msg_05 = "Once you hit the decryption button, your files will be decrypted."
        $msg_06 = "The \"trial decrypt file\" function allows you to decrypt one of the"
        $msg_07 = "encrypted files to verify the legitimacy"
        $msg_08 = "After decrypting one of your files, be sure to save it because"
```

```
        $msg_09 = "files you manage to decrypt will be encrypted again."
        $msg_10 = "However, there won't be another \"trial decrypt\""
        $msg_11 = "attempt available, so keep that in mind!"
        $msg_12 = "https://pastebin.com/raw/azDDWzUg"
        $msg_13 = ">regret to inform</span> you that your files have been compromised by the insidious"
        $msg_14 = "encryption algorithm, your files have been ensnared with unbreakable tags and a deadly combi
        $msg_15 = "meticulously chosen by the ransomware's constructors to ensure maximum devastation."
        $msg_16 = "To further fortify its grip on your data, <span class"
        $msg_17 = "employs a hybrid bulletproof encryption technique, rendering any attempts at decryption futi
        $msg_18 = "Files bearing specific extensions have been singled out for priority encryption, ensuring tha
        $msg_19 = "deploys a double-key encryption mechanism, thwarting any attempts at deception or circumvent
        $msg_20 = "In light of this harrowing situation, we implore you to refrain from taking any actions that
        $msg_21 = "with conventional means will only serve to alert its creators, potentially triggering further
        $msg_22 = "Do not disconnect from the network: Isolation will not shield you from the relentless reach o
        $msg_23 = "instead, it may hinder potential avenues of negotiation or resolution."
        $msg_24 = "Do not reboot your systems: Restarting your devices could disrupt ongoing encryption processe
        $msg_25 = "We understand the gravity of your situation and stand ready to assist you in navigating this
        $msg_26 = "However, time is of the essence, and decisive action is imperative to mitigate the extent of
        $msg_27 = "https://i.ibb.co/djp6D7n/logo-jester-done.png"
        $msg_28 = "\" alt=\"Cash Support\" />"
    condition:
        dotnet.is_dotnet
        and (
            all of ($str*)
            or 5 of ($func*)
            or any of ($msg*)
        )
}
```

snort

Warning: The Telegram API enforces TLS. This rule will not work unless a proxi splits the TLS.

```
 alert http any any -> any any (\
    sid: 110000001;\
    msg:"Cash ransomware";\
    metadata: author PEZIER Pierre-Henri. Copyright TEHTRIS 2024;\
    content:"POST"; content:"CASH RANSOMWARE";
    classtype:bad-unknown;\
    rev:1
)
```

sigma

The following Sigma rule detects the presence of encrypted files:

```
 title: CashRansomware file encryption
id: 1a4f7892-4766-4e2b-ac60-edaa00fcc31f
description: Detects default CashRansomware file extension
author: TEHTRIS - Pezier Pierre-Henri
date: 2024/05/17
tags:
    - detection.threat_hunting
logsource:
    category: file_access
    product: windows
detection:
    encrypted_file:
        FileName|endswith: '.CashRansomware'
    condition: encrypted_file
falsepositives:
    - Unknown
level: critical
```

The following rule detects the local backup of encrypted keys:

```
 title: CashRansomware registry activity
id: d0756305-56fe-4a5d-a06b-8cc447f11e66
description: Detects CashRansomware malicious activity based on registry access
author: TEHTRIS - Pezier Pierre-Henri
date: 2024/05/17
tags:
    - detection.threat_hunting
logsource:
    category: registry_event
    product: windows
detection:
    selection:
        TargetObject|endswith: 'SOFTWARE\Wow64Inject\SysWow64'
        Details|re: '(?ims)(([a-f\d]{2}\s?){8}(\r\n)?){4}'
    condition: selection
falsepositives:
    - Unknowns
level: critical
```

Appendix

decrypt.cpp

The following code leverages a decryption key and algorithm to reverse the encryption process enforced by the ransomware, restoring the original files to their accessible state if the ransomware is still running.

```
 #include <windows.h>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <string>
#include <cctype>
#include <cassert>
#include <regex>
#include <set>
#include <thread>
#include <experimental/filesystem>
#include "AES.hpp" // github: http://github.com/mrdcvlsc/AES

// build using: cl.exe decrypt.cpp /std:c++latest /EHsc

#define ENTROPY_THRESHOLD 3.5


#pragma comment(lib, "user32.lib")
wchar_t extension[] = L".CashRansomware"; // The file extensions


/*
* Find the ransomware window based on its window name. Returns the window ID if it exists, 0 if not
* Do not hesitate to patch the window by identifying window title. Autoit editor can be helpful identifying this
*/
HWND find_window(void)
{
         char window_text[1024] = {0};
         char window_class[1024] = {0};

         HWND curr_win =  GetTopWindow(NULL);
         do {
                  RealGetWindowClass(curr_win, window_class, sizeof(window_class));
                  GetWindowText(curr_win, window_text, sizeof(window_text));
                  if(strstr(window_text, "ansomware")) {
                           return curr_win;
                  }
                  curr_win = GetNextWindow(curr_win, 2);
         } while(curr_win);
         return NULL;
}


/*
* Egg hunting to extract keys from memory. Having the right extension is critical. Patch if needed.
```

```
*/
std::string find_candidate_mem_page(HWND window_id) {
	DWORD pid = 0;
	SYSTEM_INFO si;
	MEMORY_BASIC_INFORMATION mbi;
	GetWindowThreadProcessId(window_id, &pid);
	HANDLE process_handle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);
	if(process_handle == NULL) {
		return "";
	}
	GetSystemInfo(&si);
	uint8_t *ptr = (uint8_t *)si.lpMinimumApplicationAddress;
	while(ptr < si.lpMaximumApplicationAddress) {
		if(!VirtualQueryEx(process_handle, ptr, &mbi, sizeof(mbi))) {
			ptr += si.dwPageSize;
			continue;
		}
		if(mbi.Protect == PAGE_READWRITE && mbi.Type == MEM_PRIVATE && mbi.State == MEM_COMMIT) {
			uint8_t *data = (uint8_t *)malloc(mbi.RegionSize);
			if(!data) {
				CloseHandle(process_handle);
				return "";
			}
			size_t read;
			if(ReadProcessMemory(process_handle, ptr, data, mbi.RegionSize, &read)) {
				std::string data_str((char *)data, read);
				std::string tofind((char *)extension, sizeof(extension));
				size_t pos = 0, ext_count = 0;
				while((pos = data_str.find(tofind, pos)) < data_str.length()) {
					ext_count += 1;
					pos += tofind.length();
				}
				if(ext_count > 1) {
					CloseHandle(process_handle);
					free(data);
					return data_str;
				}
			}
			free(data);
		}
		ptr += mbi.RegionSize;
	}
	CloseHandle(process_handle);
	return "";
}
```

```
/*
 * Shanon entropy of a std::string
 */
float shannon_entropy(const std::string & s)
{
    int counts[256] = {};
    for (unsigned char c: s)
    {
        counts[c]++;
    }

    float entropy = 0;
    float length = (float)s.size();
    for (int count: counts)
    {
        if (count == 0)
            continue;
        float p = (float)count / length;
        entropy -= p * std::log2f(p);
    }
    return entropy;
}



/*
 * Retrieve potential key and IV based on string entropy to improve performance
 */
bool get_key_and_iv(std::set<std::string> *IVs, std::set<std::string> *keys, std::string &data)
{
         int progress = -1;
         std::cout << "progress (%): ";
         for(size_t i=1; i<data.length() - 33/*0x91358*/; i++) {
                  int curr_progress = (100 * i) / data.length();
                  if(curr_progress > progress) {
                           std::cout << curr_progress << " ";
                           progress = curr_progress;
                  }
                  if((int)data.at(i - 1) == 0) {
                           std::string candidate_iv = data.substr(i, 16);
                           std::string candidate_key = data.substr(i, 32);
                           if((int)data.at(i + 33) == 0 && shannon_entropy(candidate_key) > ENTROPY_THRESHOLD) {

                                    keys->insert(candidate_key);
                           }
                           if((int)data.at(i + 17) == 0 && shannon_entropy(candidate_iv) > ENTROPY_THRESHOLD) {
                                    IVs->insert(candidate_iv);
                           }
```

```
                }
        }
        std::cout << std::endl;
        return IVs->size() > 0 && keys->size() > 0;
}


/*
 * Extract every file enctypted by the ransomware. They are wide encoded and ends with the encryption extension
 */
std::set<std::wstring> extract_file_names(std::string &data)
{
        std::set<std::wstring> retval;
        size_t min_string_len = (sizeof(extension) + 1) * sizeof(wchar_t);
        for(size_t i=0; i<data.length() - min_string_len; i++) {
                if(data.c_str()[i] != '\x00') {
                        size_t j;
                        for(j=1; i+j < data.length() && data.c_str()[i + j] == '\x00' && data.c_str()[i + j -
                        if(j > min_string_len) {
                                std::wstring ws(
                                        (wchar_t *)&data.c_str()[i]
                                );
                                if(ws.length() == 2 && ws.c_str()[i]) {
                                        break;
                                }
                                i += ws.length() * 2;
                                if(ws.length() > sizeof(extension) / sizeof(wchar_t) &&  ws.substr(ws.length
                                        retval.insert(ws);
                                }
                        }
                }
        }
        return retval;
}

/*
 * Decrypt the first chunk of encrypted payload and xor it with the IV
 */
std::string decrypt(std::string &IV, std::string &key, std::string &chunk)
{
        unsigned char block[16];
        chunk.copy((char *)block, 16);

        Cipher::Aes<256> aes((unsigned char *)key.c_str());
        aes.decrypt_block((unsigned char *)block);

        for(size_t i=0; i<16; i++) {
```

```
                block[i] ^= IV.at(i);
        }
        return std::string((char *)block, 16);
}



/*
 * Entropy based key validation. We try to decrypt using every couple IV/KEY found and accept if entropy of dec
 * Return False if no key has been found
 */
bool bruteforce_keys(std::wstring &encrypted_file_path, std::set<std::string> &IVs, std::set<std::string> &keys,
{
        std::ifstream input_file(encrypted_file_path, std::ios::binary);
        if(!input_file.is_open()) {
                return FALSE;
        }
        unsigned char block[16];
        input_file.read((char *)block, sizeof(block));
        input_file.close();
        std::string encrypted_chunk((char *)block, sizeof(block));
        std::cout << "progress (%): ";
        int progress = -1;

        size_t counter = 0;
        for(std::string key: keys) {
                int curr_progress = (100 * counter) /  keys.size();
                if(curr_progress > progress) {
                        std::cout << curr_progress << " ";
                        progress = curr_progress;
                }
                counter += 1;
                for(std::string IV: IVs) {
                        std::string cleartext = decrypt(IV, key, encrypted_chunk);
                        if(shannon_entropy(cleartext) < 2.5) { // Arbitrary threshold to determine that we fo
                                *out_iv += IV;
                                *out_key += key;
                                std::cout << std::endl;
                                return TRUE;
                        }
                }
        }
        std::cout << std::endl;
        return FALSE;
}



/*
```

```
 * Decrypt a file. The destination file will be the stem concatenate with ".decrypted" extension
 * Returns the file name if decryption is sucessfull, empty path if not
 */
std::experimental::filesystem::path decrypt_file(std::wstring &encrypted_file_path, std::string &key, std::strin
{
	std::experimental::filesystem::path path(encrypted_file_path);
	std::experimental::filesystem::path destination = path.parent_path() / path.stem();
	destination += ".decrypted";
	if(std::experimental::filesystem::exists(destination)) {
		std::cout << "[+] File: " << destination << " already exists" << std::endl;
		return "";
	}
	std::ofstream out_file(destination, std::ios::binary);
	if(!out_file.is_open()) {
		return "";
	}
	std::ifstream in_file(encrypted_file_path, std::ios::binary);
	if(!in_file.is_open()) {
		out_file.close();
		return "";
	}

	Cipher::Aes<256> aes((unsigned char *)key.c_str());

	unsigned char diffuser[16];
	iv.copy((char *)diffuser, 16);

	while(!in_file.eof()) {
		unsigned char block[16], deciphered[16];
		in_file.read((char *)block,16);
		memcpy(deciphered, block, 16);

		aes.decrypt_block(deciphered);

		for(size_t i=0; i<16; i++) { // quick CBC implem
			deciphered[i] ^= diffuser[i];
		}
		out_file.write((char *)deciphered, 16);

		memcpy(diffuser, block, 16);
	}
	in_file.close();
	out_file.close();

	return destination;
}
```

```
int main(void)
{
	std::cout << "CashRansomware decryptor. Copyright TEHTRIS 2024" << std::endl;
	HWND window = find_window();
	if(window) {
		std::cout << "[+] Found ransomware process window" << std::endl;
	} else {
		std::cerr << "[-] Cannot find cash ransomware windows. Is the ransomware running? If window cl
		return 1;
	}
	std::string data = find_candidate_mem_page(window);
	if(data.length() > 0) {
		std::cout << "[+] Found candidate memory page." << std::endl;
	} else {
		std::cerr << "[-] Could not find any valid memory page. If file extention changed, please comm
		return 1;
	}
	std::set<std::wstring> encrypted_file_list = extract_file_names(data);
	if(encrypted_file_list.size() > 0) {
		std::cout << "[+] found: " << encrypted_file_list.size() << " encrypted file(s)" << std::endl;
	} else {
		std::cerr << "[-] No encrypted file found. If file extention changed, please commit in the sou
		return 1;
	}
	std::set<std::string> IVs, keys;
	std::cout << "[+] Looking for candidate keys and ivs. This can take a long time" << std::endl;
	if(get_key_and_iv(&IVs, &keys, data)) {
		std::cout << "[+] Found: " << std::dec << keys.size() << " candidate keys and " << IVs.size()
	} else {
		std::cerr << "[-] Could not extract valid keys and IVs" << std::endl;
		return 1;
	}
	std::string real_iv, key;
	for(std::wstring encrypted_file_path: encrypted_file_list) {
		std::wcout << L"[+] Bruteforcing key on file: " << encrypted_file_path << std::endl;
		if(bruteforce_keys(encrypted_file_path, IVs, keys, &real_iv, &key)) {
			std::cout << "[+] Found couple KEY, IV" << std::endl;
			break;
		} else {
			std::cerr << "[-] Failed to extract couple KEY, IV. Continuing with next file" << std
		}
	}
	if(real_iv.length() == 0 || key.length() == 0) {
		std::cerr << "[-] Failed to extract couple KEY, IV." << std::endl;
		return 1;
	}
```

```
        for(std::wstring encrypted_file_path: encrypted_file_list) {
                std::experimental::filesystem::path decrypted_file = decrypt_file(encrypted_file_path, key, re
                if(std::experimental::filesystem::exists(decrypted_file)) {
                        std::cout << "[+] File " << decrypted_file << " successfully decrypted" << std::endl;
                } else {
                        std::wcerr << L"[-] Failed to decrypt file: " << encrypted_file_path << std::endl;
                }
        }
        system("pause"); // Do not close console
}
```

Stix2.1 json source

```
{
 "type": "bundle",
 "id": "bundle--b9d25a93-f002-4fef-b115-38f9f7205516",
 "objects": [
         {
             "type": "threat-actor",
             "spec_version": "2.1",
             "id": "threat-actor--af63e750-a86a-4f61-afff-d90c14291b0f",
             "created_by_ref": "identity--8f89679b-f5f9-4773-8980-dc409c74b548",
             "threat_actor_types": [ "crime-syndicate"],
             "name": "Cash LLC",
             "description": "Cash out and Cash Hosting malware as a service entity",
             "aliases": ["cash out", "cash hosting"],
             "goals": ["Steal Money"],
             "sophistication": "advanced",
             "resource_level": "team",
             "primary_motivation": "organizational-gain"
         },
         {
             "type": "identity",
             "spec_version": "2.1",
             "id": "identity--023d105b-752e-4e3c-941c-7d3f3cb15e9e",
             "name": "Artem Ey",
             "identity_class": "individual"
         },
         {
             "type": "relationship",
             "spec_version": "2.1",
             "id": "relationship--8a2c59b2-990e-4f05-b2d9-25b0da6bc064",
             "relationship_type": "attributed-to",
             "source_ref": "threat-actor--af63e750-a86a-4f61-afff-d90c14291b0f",
             "target_ref": "identity--023d105b-752e-4e3c-941c-7d3f3cb15e9e"
         },
```

```
        {
            "type":"infrastructure",
            "spec_version": "2.1",
            "id":"infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
            "name":"Cash out malware store",
            "infrastructure_types": ["hosting-malware"]
        },
        {
            "type":"infrastructure",
            "spec_version": "2.1",
            "id":"infrastructure--b285b5b0-d757-418f-ba5b-4df23a4fe8e7",
            "name":"Cash hosting",
            "infrastructure_types": ["hosting-target-lists"]
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--621277c3-198e-4c9a-b91b-ed54eacd33de",
            "relationship_type": "hosts",
            "source_ref": "threat-actor--af63e750-a86a-4f61-afff-d90c14291b0f",
            "target_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--2765d8df-a6d5-4d2a-b042-20d7450a0396",
            "relationship_type": "hosts",
            "source_ref": "threat-actor--af63e750-a86a-4f61-afff-d90c14291b0f",
            "target_ref": "infrastructure--b285b5b0-d757-418f-ba5b-4df23a4fe8e7"
        },
        {
            "type": "domain-name",
            "spec_version": "2.1",
            "id": "domain-name--faf0609d-2a3d-4706-925a-a6f7699e385a",
            "value": "cashout.pw"
        },
        {
            "type": "domain-name",
            "spec_version": "2.1",
            "id": "domain-name--1e42e57c-269a-4446-8557-95c4f807a91b",
            "value": "cash-hosting.pw"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--280c98d0-3ac8-4eb0-b333-3b3e0309f9cc",
            "relationship_type": "consists-of",
```

```
        "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
        "target_ref": "domain-name--faf0609d-2a3d-4706-925a-a6f7699e385a"
    },
    {
        "type": "relationship",
        "spec_version": "2.1",
        "id": "relationship--b4d5e3fa-cab0-426c-84b8-f6bd02e96feb",
        "relationship_type": "consists-of",
        "source_ref": "infrastructure--b285b5b0-d757-418f-ba5b-4df23a4fe8e7",
        "target_ref": "domain-name--1e42e57c-269a-4446-8557-95c4f807a91b"
    },
    {
        "type": "malware",
        "spec_version": "2.1",
        "id": "malware--b793b2ed-9c02-46ef-8e8a-039ecc983a1b",
        "name": "CashRansomware",
        "description": "A not release yet ransomware",
        "malware_types": ["ransomware"],
        "is_family": false
    },
    {
        "type": "malware",
        "spec_version": "2.1",
        "id": "malware--d511b55c-35b2-425d-b04d-e2e417421198",
        "name": "Mint Stealer",
        "description": "A dotnet stealers",
        "malware_types": ["spyware"],
        "is_family": false
    },
    {
        "type": "malware",
        "spec_version": "2.1",
        "id": "malware--d877eb07-94c1-429b-bf99-d5719148b1e0",
        "name": "Cash Rat",
        "description": "A csharp RAT",
        "malware_types": ["remote-access-trojan"],
        "is_family": false
    },
    {
        "type": "malware",
        "spec_version": "2.1",
        "id": "malware--f35e38e0-681b-481f-9ca8-86eda1c128c8",
        "name": "Cash Crypter",
        "description": "A PE crypter",
        "malware_types": ["unknown"],
        "is_family": false
    },
```

```
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--e7b46831-cb79-4528-b8c5-3a31eb373757",
            "relationship_type": "hosts",
            "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
            "target_ref": "malware--b793b2ed-9c02-46ef-8e8a-039ecc983a1b"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--0c133ffa-fd72-46f3-865e-e9e8d6222766",
            "relationship_type": "hosts",
            "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
            "target_ref": "malware--d511b55c-35b2-425d-b04d-e2e417421198"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--8cd09dc7-a106-4aa9-8589-7b81fe2ba972",
            "relationship_type": "hosts",
            "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
            "target_ref": "malware--d877eb07-94c1-429b-bf99-d5719148b1e0"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--048581fc-c12e-4f7f-ae90-2ff7a2a3e39f",
            "relationship_type": "hosts",
            "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
            "target_ref": "malware--f35e38e0-681b-481f-9ca8-86eda1c128c8"
        },
        {
            "type": "location",
            "spec_version": "2.1",
            "id": "location--a7c4f414-3fab-4940-86d0-f4370ce2c7e50",
            "country": "RU",
            "city": "moscow"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--d9a4c1f2-e9ff-48cf-bdc4-5a27f2a077ef",
            "relationship_type": "located-at",
            "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
            "target_ref": "location--a7c4f414-3fab-4940-86d0-f4370ce2c7e50"
        },
```

```
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--6b530293-0275-4862-98dc-69d452da8542",
            "relationship_type": "located-at",
            "source_ref": "infrastructure--b285b5b0-d757-418f-ba5b-4df23a4fe8e7",
            "target_ref": "location--a7c4f414-3fab-4940-86d0-f4370ce2c7e50"
        },
        {
            "type": "location",
            "spec_version": "2.1",
            "id": "location--c5c1eacf-cd9f-44ed-9515-9aa0401e7067",
            "country": "FR",
            "region": "PACA"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--ce8bff7b-3aa3-4475-8a01-6bdc4e967a2b",
            "relationship_type": "located-at",
            "source_ref": "identity--023d105b-752e-4e3c-941c-7d3f3cb15e9e",
            "target_ref": "location--c5c1eacf-cd9f-44ed-9515-9aa0401e7067"
        },
        {
            "type": "campaign",
            "spec_version": "2.1",
            "id": "campaign--7cdc2b62-3f35-43eb-ae6d-11145036876d",
            "name": "CashRansom Development",
            "description": "Active development of CashRansom"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--c5f677d4-55ee-43ad-90b5-d19552731625",
            "relationship_type": "uses",
            "source_ref": "campaign--7cdc2b62-3f35-43eb-ae6d-11145036876d",
            "target_ref": "malware--b793b2ed-9c02-46ef-8e8a-039ecc983a1b"
        },
       {
            "type": "file",
            "spec_version": "2.1",
            "id": "file--75f6adba-3c71-4815-9763-3aef55b9f174",
            "hashes": {
              "SHA-256": "e1696968ad55e7e03a8334711d90350c4145fb4f60de5fb4a2f5f19187183c05"
            },
            "name": "CashRansom.exe"
        },
```

```
{
     "type": "file",
     "spec_version": "2.1",
     "id": "file--36c8fdaa-6b8f-4112-acb4-1095d75beee8",
     "hashes": {
       "SHA-256": "39096e9a521ea1c001083d8c82317c8e6dbdd5d705d9a92beb15db102fb87263"
     },
     "name": "CashRansom.exe"
 },
{
     "type": "file",
     "spec_version": "2.1",
     "id": "file--034a9dbf-083e-464a-a055-509a4c9a330b",
     "hashes": {
       "SHA-256": "958ccd8e8dcce5e7bac5f891e8edc42ad6c5497d9385c8ae26c328c5f7beda24"
     },
     "name": "CashRansom.exe"
 },
{
     "type": "file",
     "spec_version": "2.1",
     "id": "file--6ca2758f-2586-40b9-b712-fcd22a70873e",
     "hashes": {
       "SHA-256": "b8f506741843e2c76fb207b41d205530236f4a263a9a5902146cd71a13fdfd23"
     },
     "name": "CashRansom.exe"
 },
{
     "type": "file",
     "spec_version": "2.1",
     "id": "file--f084f3da-b64a-468c-8733-2230751ef02a",
     "hashes": {
       "SHA-256": "003311f504eec2d0203de5e5e9d8c4213a981a3f6db85141a6d1e84a58e2b6b9"
     },
     "name": "CashRansom.exe"
 },
{
     "type": "file",
     "spec_version": "2.1",
     "id": "file--85975011-d8f3-43b0-982a-30f2286a3dfb",
     "hashes": {
       "SHA-256": "c67332d690304dfd5d43fc35dbdf0a832a803ff5f73f6187e3d94ace7433fffd"
     },
     "name": "CashRansom.exe"
 },
{
     "type": "file",
```

```
      "spec_version": "2.1",
      "id": "file--fe389703-22f3-4950-93ae-07f5d0d76775",
      "hashes": {
        "SHA-256": "6332e43af93cf00c5bd536e189b30d5a44c0568fb3fdef5e9b020146420d8b15"
      },
      "name": "CashRansom.exe"
    },
    {
      "type": "file",
      "spec_version": "2.1",
      "id": "file--485a2105-4905-48dc-bfc3-f7bc4bfa87e0",
      "hashes": {
        "SHA-256": "5525d297a346b80912c4f5ec0ac4875e9d49f96d01e52c10df5c064bd803bd79"
      },
      "name": "CashRansom.exe"
    },
    {
      "type": "file",
      "spec_version": "2.1",
      "id": "file--c9e4aa04-9366-48a3-a086-3bc5f0eff111",
      "hashes": {
        "SHA-256": "8040b684f12ac819ba9ecb407f202f01182d25186552093379ad33c86f3a4273"
      },
      "name": "CashRansom.exe"
    },
    {
      "type": "file",
      "spec_version": "2.1",
      "id": "file--c61fd93b-06c6-4e9b-b6da-fb4bd9f7f24a",
      "hashes": {
        "SHA-256": "b7f5f19864639b32f24e806091bf0a0a7483df73c97eac41fe5eb1d3321cea39"
      },
      "name": "CashRansom.exe"
    },
    {
      "type": "file",
      "spec_version": "2.1",
      "id": "file--406008db-228c-45b8-adaf-bc3d9a8592f1",
      "hashes": {
        "SHA-256": "dd7d6dc03dea59e2f48ef92849ec0d03aa6cf4c5e1e6758eba184be311e6fb1f"
      },
      "name": "CashRansom.exe"
    },
    {
      "type": "file",
      "spec_version": "2.1",
      "id": "file--8ceda5a1-a4a4-402d-8a6f-9f7c2df4c733",
```

```
            "hashes": {
              "SHA-256": "91edb2ed65ae31113c3c2f3ba63bcac5d24d48ef3d5765018799863e4717b845"
            },
            "name": "CashRansom.exe"
         },
        {
            "type": "file",
            "spec_version": "2.1",
            "id": "file--7b12438f-63cc-4c44-8618-61f18aea1955",
            "hashes": {
              "SHA-256": "132ef1a933f9d26fb0bb46b0a970dbfe05ad8fe0859ece8eb973b5584a580cc3"
            },
            "name": "CashRansom.exe"
         },
        {
            "type": "file",
            "spec_version": "2.1",
            "id": "file--defa04d0-7f9b-4901-b029-7b094bdc45e9",
            "hashes": {
              "SHA-256": "e387c084d5c3b62413743e912ee10776564e7c55ba1dc801990b312b88b61efe"
            },
            "name": "CashRansom.exe"
         },
        {
            "type": "observed-data",
            "spec_version": "2.1",
            "id": "observed-data--fa04cdeb-623c-46f3-afa1-bc76b219aaba",
            "first_observed": "2024-05-10T00:00:00Z",
            "last_observed": "2024-05-10T00:00:00Z",
            "number_observed": 1,
            "object_refs": ["campaign--7cdc2b62-3f35-43eb-ae6d-11145036876d", "file--75f6adba-3c71-4815-9763-3
         }
  ]
}
```

Source: https://tehtris.com/en/blog/unreleased-raas-analysis-cashransomware/