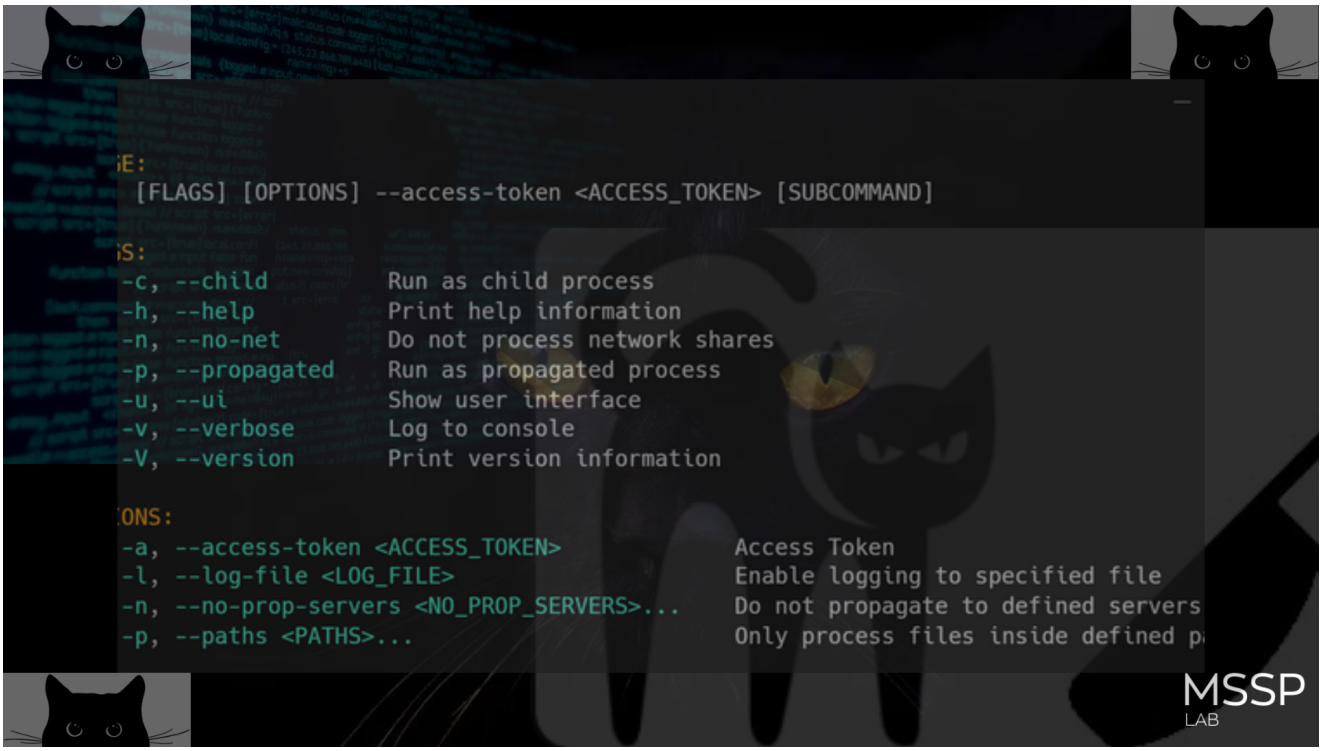
Malware analysis report: BlackCat ransomware

mssplab.github.io/threat-hunting/2023/07/13/malware-analysis-blackcat.html

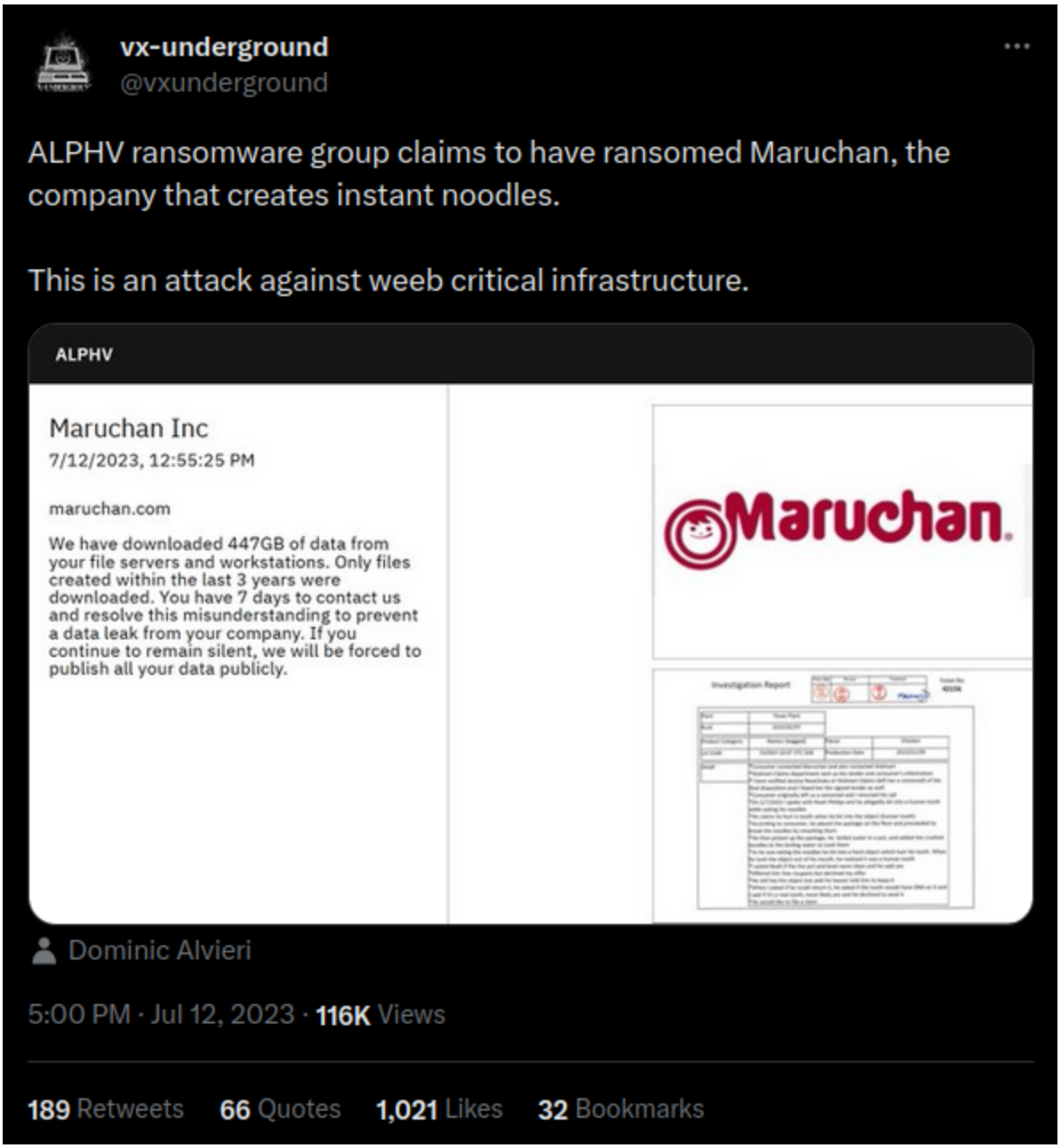
July 13, 2023

10 minute read

BlackCat is Rust-based ransomware distributed via the *Ransomware-as-a-Service (RaaS)* model. BlackCat was observed for the first time in November 2021 and has since been used to target multiple sectors and organizations in numerous countries and regions in Africa, the Americas, Asia, Australia, and Europe.



This ransomware and group caught our attention after this interesting news: *"ALPHV ransomware group claims to have ransomed Maruchan, the company that creates instant noodles."*:



> The name “BlackCat” was mentioned first by MalwareHunterTeam

https://twitter.com/vxunderground/status/1679128724489289728

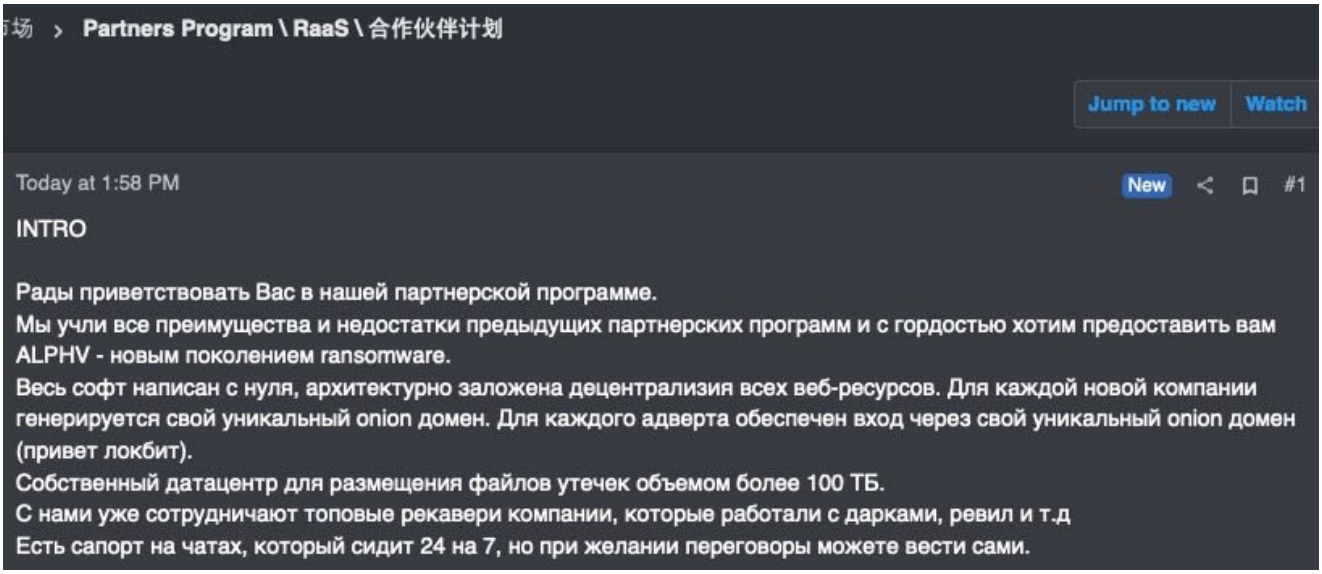
Technical summary

This ransomware encrypts the data of business users and corporate networks using the algorithms `AES-128` (CTR mode) and `RSA-2048`, and then demands a hefty ransom payment in BTC or Monero to decrypt the files. Instead of `AES`, the `ChaCha20` algorithm can be used. The configuration file is consulted to retrieve the global public key used to encrypt local credentials. Original title: *ALPHV-ng RaaS*. A striking example of using the Rust programming language. eSXI is capable of encrypting data on Windows, Linux, and VMWare systems.

Threat Actor

Most of threat hunting labs, also MSSP Lab has observed one of these RaaS providers, ALPHV (also known as BlackCat ransomware), gathering traction since late 2021, actively recruiting new affiliates and targeting organizations in a variety of industries across the globe. The organization actively recruits former REvil, BlackMatter, and DarkSide operators. A campaign to attract new affiliates started to be advertised on underground forums:

```
INTRO

Рады приветствовать Вас в нашей партнерской программе.
Мы  учли все преимущества и недостатки предыдущих партнерских программ и с гордостью хотим
предоставить вам ALPHV - новым поколением ransomware.
Весь софт написан с нуля, архитектурно заложена децентрализия всех веб-ресурсов. Для каждой новой
компании генерируется свой уникальный onion домен. Для каждого адверта обеспечен вход через свой
уникальный onion домен (привет локбит).
Собственный датацентр для размещения файлов утечек объемом более 100 ТБ.
С нами уже сотрудничают топовые рекавери компании, которые работали с дарками, ревил и т.д
Есть сапорт на чатах, который сидит 24 на 7, но при желании переговоры можете вести сами.

SECURITY

Мы всячески готовы к существованию в современных условиях, соответствуя всем требованиям к
безопасности инфраструктуры и адвертов. В партнерской программе архитектурно исключены все возможные
связи с форумами(привет ревил), заложены алгоритмы само удаления данных по истечению срока давности,
интегрирован встроенный миксер с настоящим разрывом цепочки(не путать с Wasabi, BitMix и
прочими),т.к. Вы получаете совершенно чистые монеты с иностранных бирж. Кошельки на которые были
отправлены Ваши монеты неизвестны для нашего бекенда. Инфраструктура раздроблена на т.н. ноды,
которые связанны между собой через целую сеть прокладок в пределах сети onion и находятся за NAT+FW.
Даже при получении полноценного cmdshell атакующий не сможет раскрыть реальный ip адрес сервера.
(привет конти)
```



Posted December 4, 2021 (edited)

Нужны опытные пентестеры, такого уровня вы еще не видели, чтобы узнать все подробности пишите по контакту ниже.
TOX: 3488458145EB62D7D3947E3811234F4663D9B5AEEF6584AB08A2099A7F946664BBA2B0D30BFC
TOX: 16BF03E7266A1859E5032203EB546C1DFD1AF6D72A23A863B0100198354C9F7D330C2001EA1B
JAB: username01@thesecure.biz

Edited December 4, 2021 by alphv

Identification

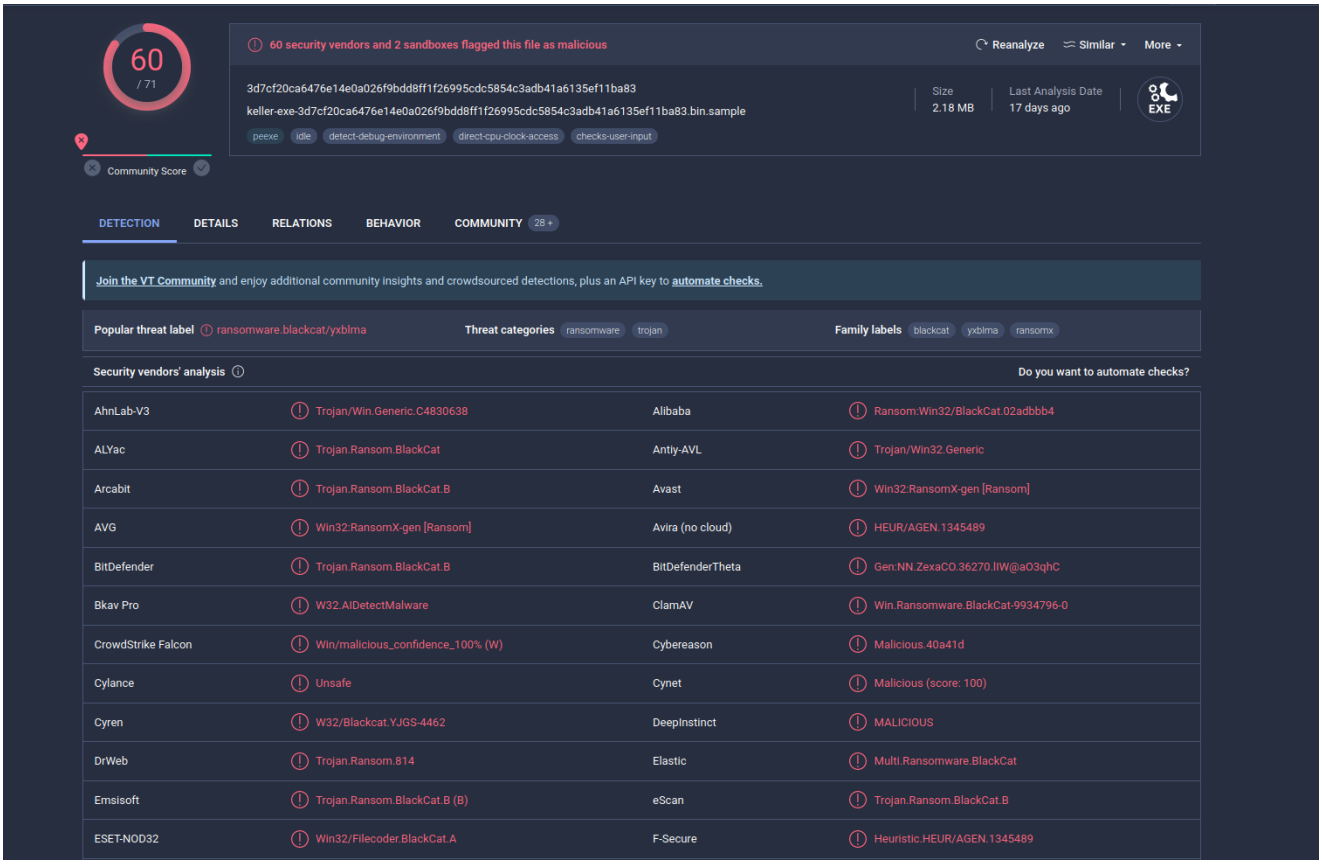
Samples is being investigated:

sample.exe:

File size: `2281472 bytes`
MD5 sum: `aea5d3cced6725f37e2c3797735e6467`
SHA-1 sum: `087497940a41d96e4e907b6dc92f75f4a38d861a`
SHA-256 sum: `3d7cf20ca6476e14e0a026f9bdd8ff1f26995cdc5854c3adb41a6135ef11ba83`

First of all, check our sample via VirusTotal:

https://www.virustotal.com/gui/file/3d7cf20ca6476e14e0a026f9bdd8ff1f26995cdc5854c3adb41a6135ef11ba83/details



As we can see, 60 of 71 AV engines detect our sample as malicious.

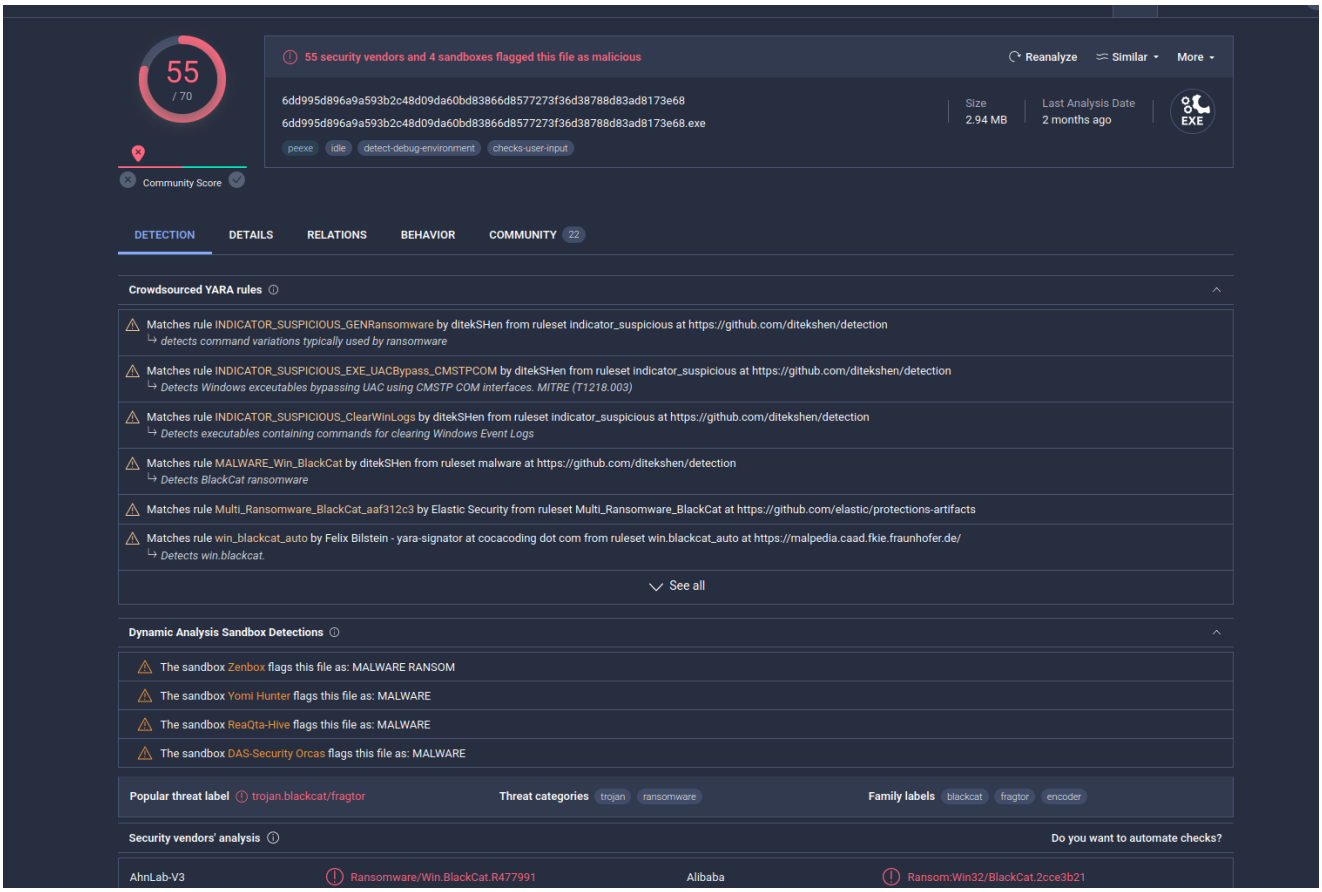
and the most interesting sample written in Rust:

sample2.exe:

File size: `2281472 bytes`
MD5 sum: `701b4b004eecb69046c210237846d46d`
SHA-1 sum: `8c70191b12f14eed594388c8fbe05efe6ebaa564`
SHA-256 sum: `6dd995d896a9a593b2c48d09da60bd83866d8577273f36d38788d83ad8173e68`

which also checked via VirusTotal:

https://www.virustotal.com/gui/file/6dd995d896a9a593b2c48d09da60bd83866d8577273f36d38788d83ad8173e68



As we can see, 55 of 70 AV engines detect our sample as malicious.

More of the detect it as `Win.Ransomware.BlackCat-9974801-0`

Static analysis

sample.exe

The specified sample is a `32-bit` PE file:

```
file <sample.exe>
```

```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ file sample.exe
sample.exe: PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows
```

```
hexdump -C <sample.exe>
```

```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ hexdump -C sample.exe | head
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 80 00 00 00  |................|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |........!..L.!Th|
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.......|
00000080  50 45 00 00 4c 01 08 00  2c 25 96 61 00 00 00 00  |PE..L...,%.a....|
00000090  00 00 00 00 e0 00 2f 03  0b 01 02 1e 00 fc 18 00  |....../.........|
```

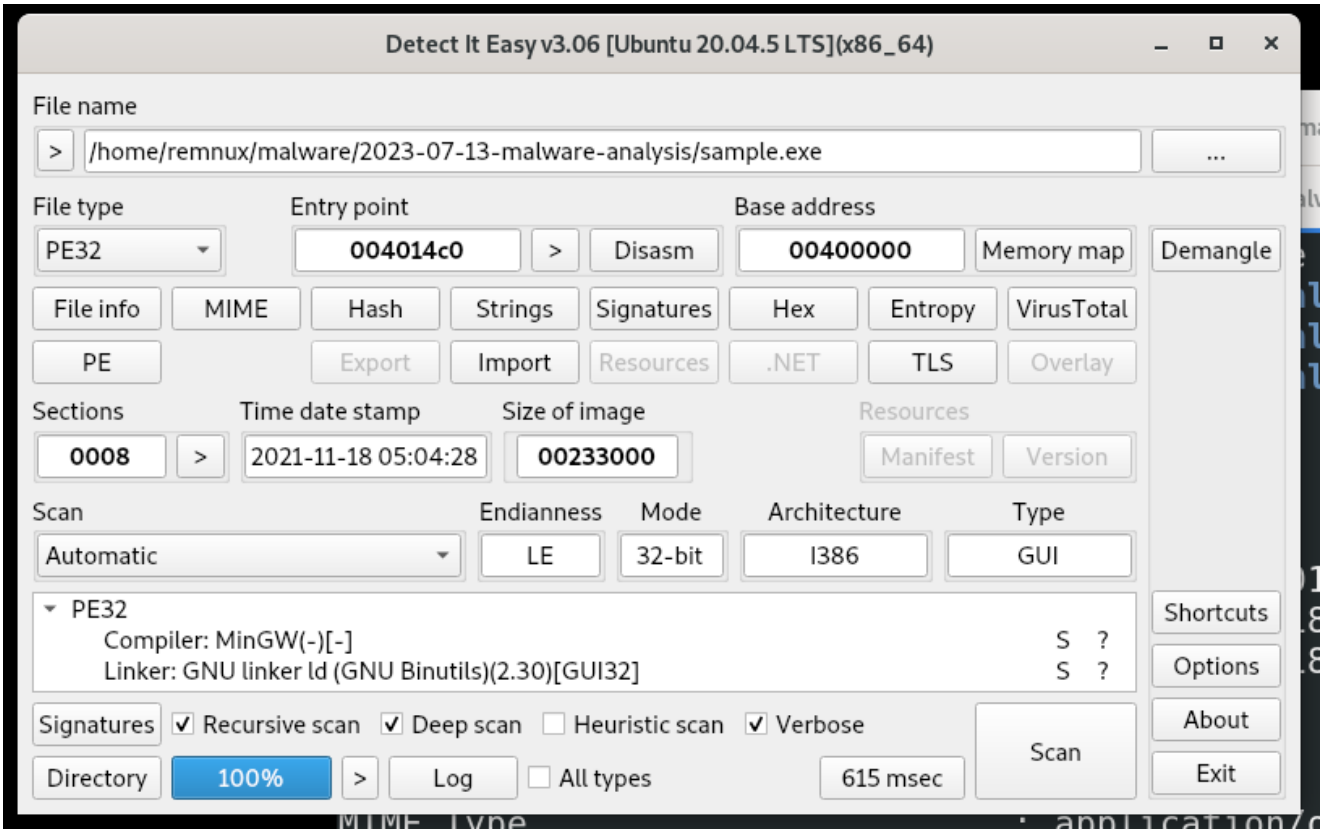
Use `exiftool` for looking metadata:

```
exiftool <sample.exe>
```

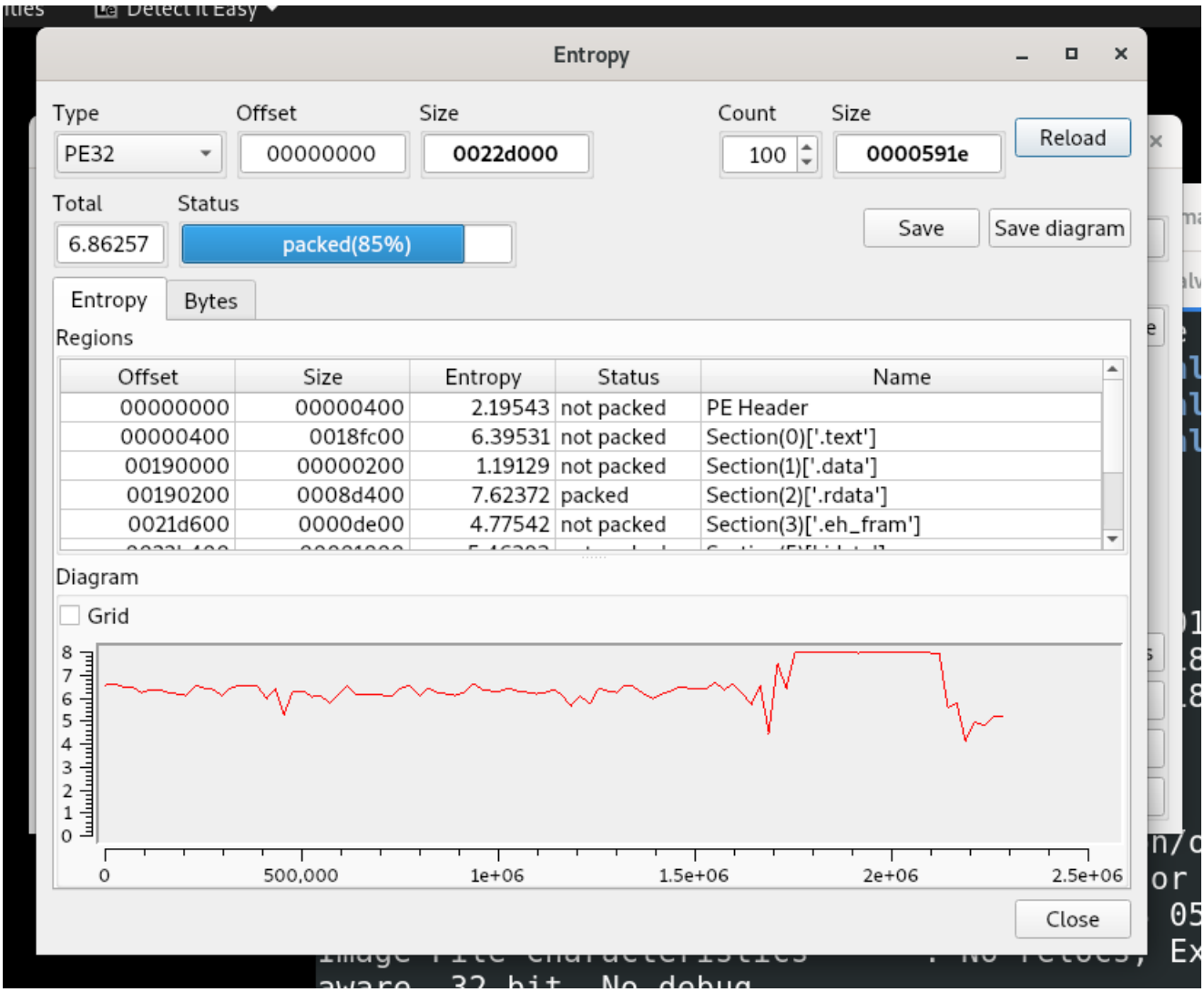
```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ exiftool sample.exe
ExifTool Version Number         : 12.50
File Name                       : sample.exe
Directory                       : .
File Size                       : 2.3 MB
File Modification Date/Time     : 2023:04:22 01:10:28-04:00
File Access Date/Time           : 2023:07:13 18:49:57-04:00
File Inode Change Date/Time     : 2023:07:13 18:49:53-04:00
File Permissions                : -rw-rw-r--
File Type                       : Win32 EXE
File Type Extension             : exe
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2021:11:18 05:04:28-05:00
Image File Characteristics      : No relocs, Executable, No line numbers, No symbols, Large address
aware, 32-bit, No debug
PE Type                         : PE32
Linker Version                  : 2.30
Code Size                       : 1637376
Initialized Data Size           : 2280448
Uninitialized Data Size         : 1536
Entry Point                     : 0x14c0
OS Version                      : 4.0
Image Version                   : 1.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
remnux@remnux:~/malware/2023-07-13-malware-analysis$
```

File timestamp is `2021:11:18 05:04:28-05:00`

Compiled via `MinGW`:



Shannon entropy:



sample2.exe

The specified sample is a `32-bit` PE file:

```
file <sample2.exe>
```

```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ file sample2.exe
sample2.exe: PE32 executable (console) Intel 80386 (stripped to external PDB), for MS Windows
```

```
hexdump -C <sample2.exe>
```

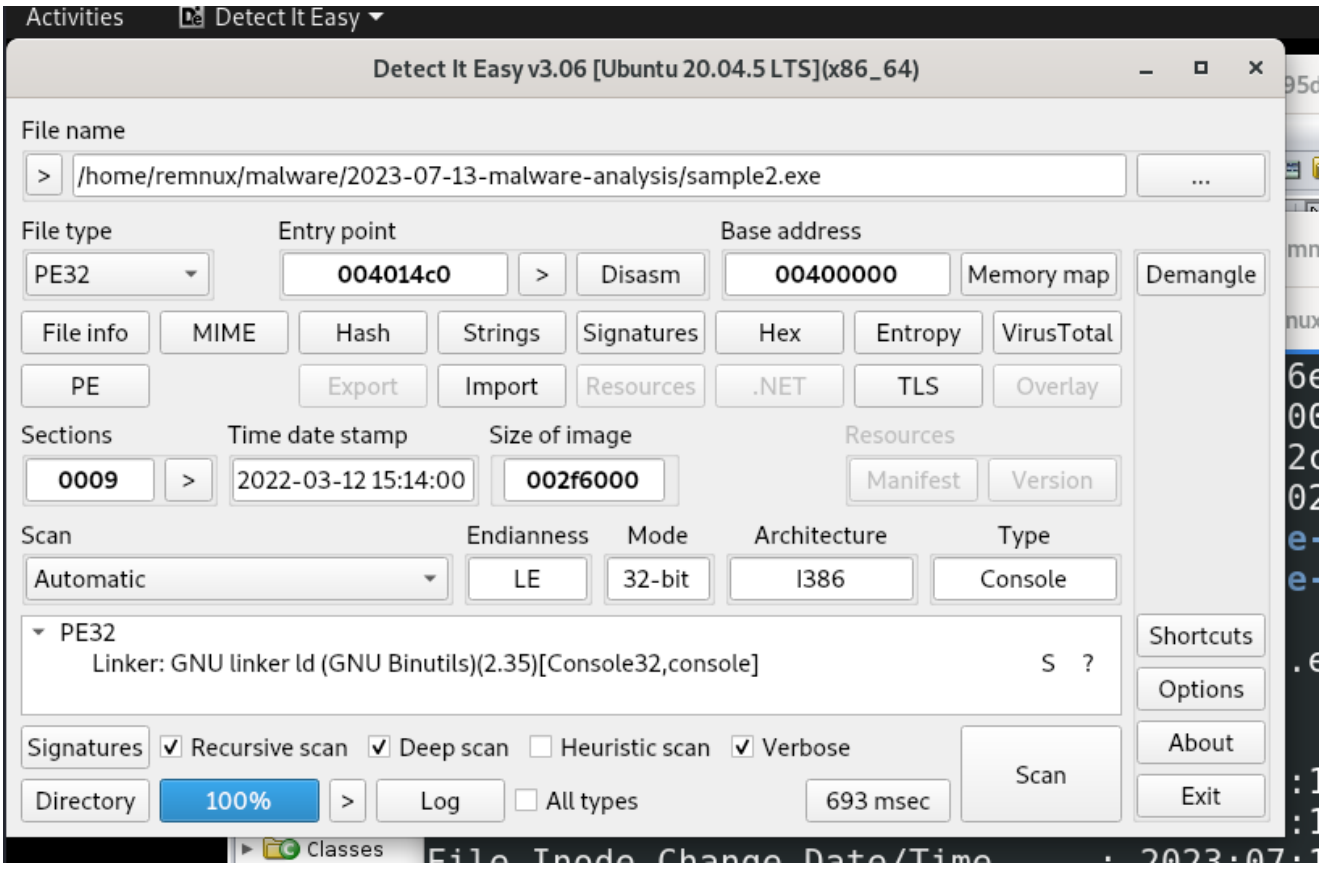
```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ hexdump -C sample2.exe | head
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 80 00 00 00  |................|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |........!..L.!Th|
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.......|
00000080  50 45 00 00 4c 01 09 00  08 ff 2c 62 00 00 00 00  |PE..L.....,b....|
00000090  00 00 00 00 e0 00 2e 03  0b 01 02 23 00 f4 1e 00  |...........#....|
```

Use `exiftool` for looking metadata:

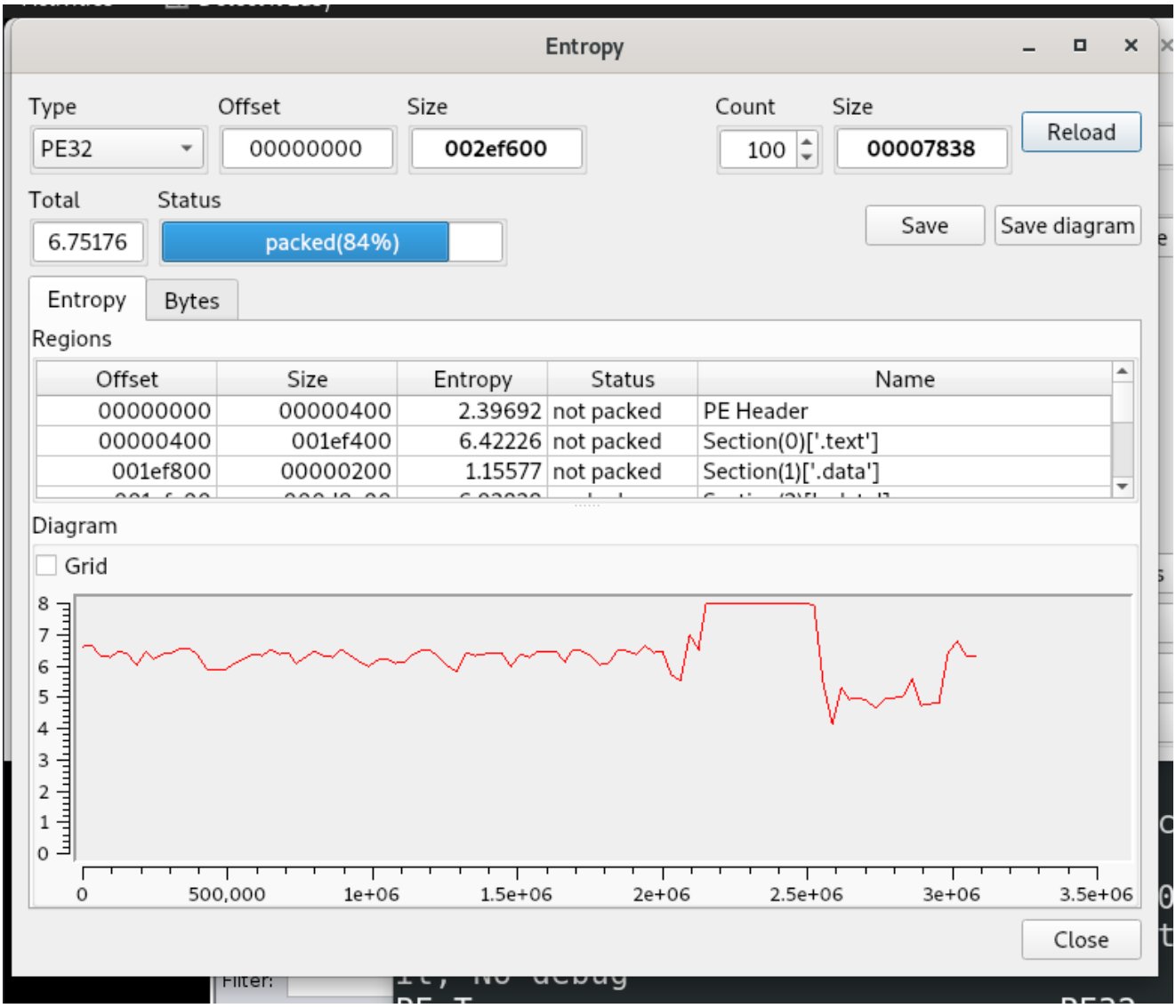
```
exiftool <sample2.exe>
```

```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ exiftool sample2.exe
ExifTool Version Number         : 12.50
File Name                       : sample2.exe
Directory                       : .
File Size                       : 3.1 MB
File Modification Date/Time     : 2023:07:13 20:50:27-04:00
File Access Date/Time           : 2023:07:13 20:50:42-04:00
File Inode Change Date/Time     : 2023:07:13 20:50:27-04:00
File Permissions                : -rw-r--r--
File Type                       : Win32 EXE
File Type Extension             : exe
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2022:03:12 15:14:00-05:00
Image File Characteristics      : Executable, No line numbers, No symbols, Large address aware, 32-b
it, No debug
PE Type                         : PE32
Linker Version                  : 2.35
Code Size                       : 2028544
Initialized Data Size           : 3076608
Uninitialized Data Size         : 1024
Entry Point                     : 0x14c0
OS Version                      : 4.0
Image Version                   : 1.0
Subsystem Version               : 4.0
Subsystem                       : Windows command line
Warning                         : Error processing PE data dictionary
```

Linker information `GNU binutils`:



Entropy:


Entropy
Type
PE32
Offset
00000000
Size
002ef600
Count
100
Size
00007838
Reload
Total
6.75176
Status
packed(84%)
Save
Save diagram
Entropy
Bytes
Regions
Offset
Size
Entropy
Status
Name
00000000
00000400
2.39692
not packed
PE Header
00000400
001ef400
6.42226
not packed
Section(0)['.text']
001ef800
00000200
1.15577
not packed
Section(1)['.data']
Diagram
Grid
8
7
6
5
4
3
2
1
0
0
500,000
1e+06
1.5e+06
2e+06
2.5e+06
3e+06
3.5e+06
Close

```
remnux@remnux:~/malware/2023-07-13-malware-analysis$ python3 entropy.py -f sample2.exe
.text
        virtual address: 0x1000
        virtual size: 0x1ef26c
        raw size: 0x1ef400
        entropy: 6.422260919037174
.data
        virtual address: 0x1f1000
        virtual size: 0x118
        raw size: 0x200
        entropy: 1.1529522754724144
.rdata
        virtual address: 0x1f2000
        virtual size: 0xd88e8
        raw size: 0xd8a00
        entropy: 6.928279453909548
.eh_fram
        virtual address: 0x2cb000
        virtual size: 0x12df4
        raw size: 0x12e00
        entropy: 4.831238290004092
.bss
        virtual address: 0x2de000
        virtual size: 0x254
        raw size: 0x0
        entropy: 0.0
.idata
        virtual address: 0x2df000
        virtual size: 0x1b24
        raw size: 0x1c00
```

```
.CRT
        virtual address: 0x2e1000
        virtual size: 0x58
        raw size: 0x200
        entropy: 0.7034519812709853
.tls
        virtual address: 0x2e2000
        virtual size: 0x8
        raw size: 0x200
        entropy: 0.0
.reloc
        virtual address: 0x2e3000
        virtual size: 0x123a4
        raw size: 0x12400
        entropy: 6.701597134441917
remnux@remnux:~/malware/2023-07-13-m
```

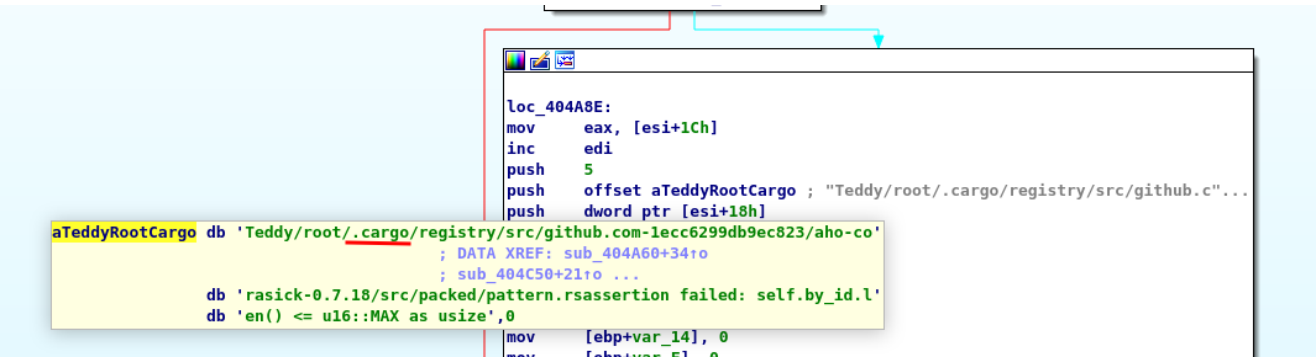
Dynamic analysis

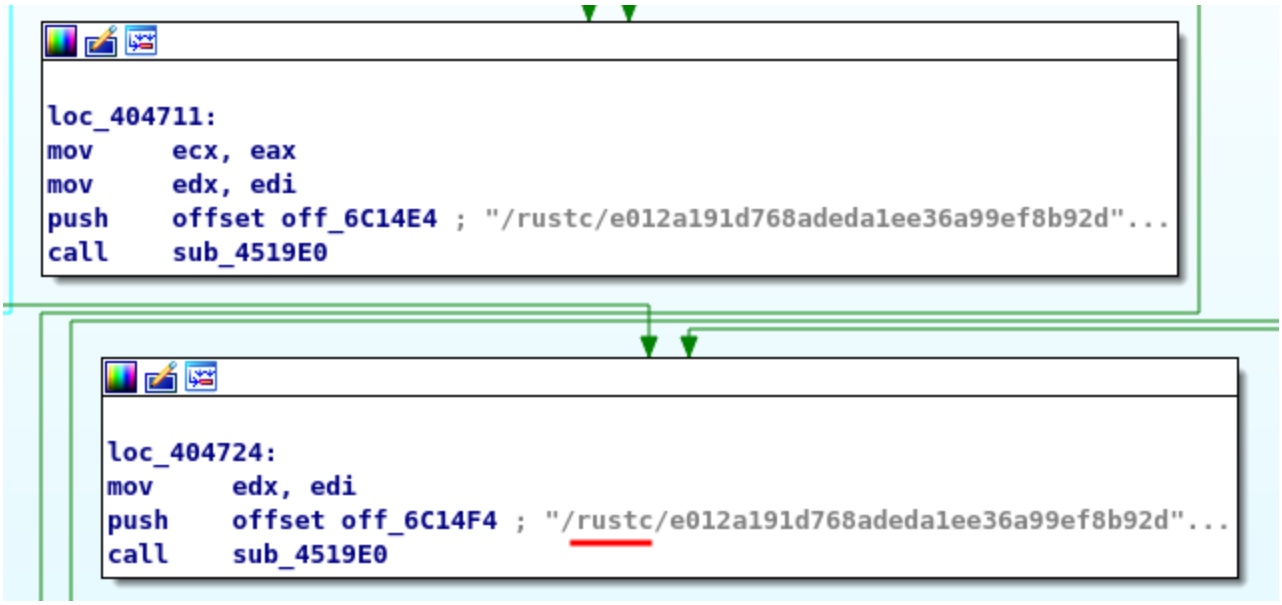
Can be distributed via hacking via an insecure RDP configuration, email spam and malicious attachments, inaccurate downloads, botnets, exploits, malicious advertisements, web injections, fake updates, repackaged and infected installers.

Ransom Note:

```
» Introduction
Important files on your system was ENCRYPTED and now they have have "sykffle"
extension.
In order to recover your files you need to follow instructions below.
» Sensitive Data
Sensitive data on your system was downloaded and it will be published if you refuse
to cooperate.
Data includes:
- Employees personal data, CVs, DL , SSN.
- Complete network map including credentials for local and remote services.
- Financial information including clients data, bills, budgets, annual reports, bank
statements.
- Complete datagrams/schemas/drawings for manufacturing in solidworks format
- And more...
Private preview is published here:
hxxx://zujgzbu5y64xbmvc42addp4lxkoosb4tslf5mehnh7pvqjpwxn5gokyd.onion/***
» CAUTION
DO NOT MODIFY FILES YOURSELF.
DO NOT USE THIRD PARTY SOFTWARE TO RESTORE YOUR DATA.
YOU MAY DAMAGE YOUR FILES, IT WILL RESULT IN PERMANENT DATA LOSS.
YOUR DATA IS STRONGLY ENCRYPTED, YOU CAN NOT DECRYPT IT WITHOUT CIPHER KEY.
» Recovery procedure
Follow these simple steps to get in touch and recover your data:
1) Download and install Tor Browser from: https://torprojoject.org
2) Navigate to:
hxxx://mu75ltv3lxd24dbyu6gtvmnwybecigs5auki7fces437xvvflzva2nqd.onion/?access-key=***
```

Second sample is written in Rust:





Initialisation and propagation - BlackCat samples that we analyzed could be launched with any string provided as the access token:

```
.\sample.exe -v --access-token 1234567
```

```
USAGE:
    [FLAGS] [OPTIONS] --access-token <ACCESS_TOKEN> [SUBCOMMAND]

FLAGS:
    -c, --child          Run as child process
    -h, --help           Print help information
    -n, --no-net         Do not process network shares
    -p, --propagated     Run as propagated process
    -u, --ui             Show user interface
    -v, --verbose        Log to console
    -V, --version        Print version information

OPTIONS:
    -a, --access-token <ACCESS_TOKEN>            Access Token
    -l, --log-file <LOG_FILE>                    Enable logging to specified file
    -n, --no-prop-servers <NO_PROP_SERVERS>...   Do not propagate to defined servers
    -p, --paths <PATHS>...                       Only process files inside defined paths
```

The malware will immediately attempt to validate the existence of the aforementioned `access-token`, followed by a query for the system UUID:

```
cmd.exe /c wmic csproduct get UUID
```

Also, it employs the `GetCommandLineW` API to determine whether the supplied access token is valid:

```
push    ebp
mov     ebp, esp
sub     esp, 14h
lea     eax, [ebp+pNumArgs]
push    esi
push    edi
push    eax             ; pNumArgs
call    ds:GetCommandLineW
push    eax             ; lpCmdLine
call    ds:CommandLineToArgvW
test    eax, eax
jz      loc_4011B8
```

BlackCat spawns a number of its own processes, with the following syntax (for Windows):

```
wmic.exe Shadowcopy Delete"
"iisreset.exe /stop"
bcdedit.exe /set {default} recoveryenabled No
```

or

```
cmd /c vssadmin.exe delete shadows /all /quiet
```

As you can see, in order to prevent the organization from restoring encrypted files, the ransomware first deletes any available shadow copies, as is characteristic of ransomware attacks.

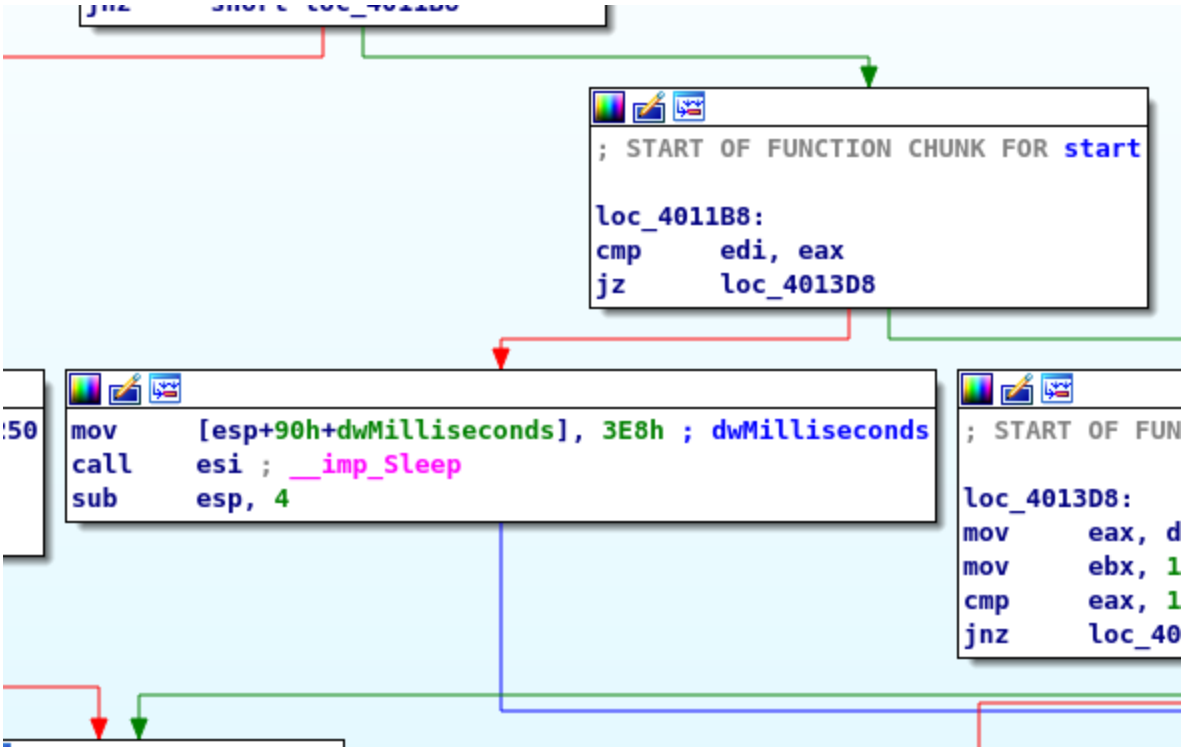
BlackCat also attempts to propagate via `PsExec`:

```
psexec.exepsexec_adv=
locker::core::windows::psexecsrc/core/windows/psexec.rs -accepteula X
remnux@remnux:~/malware/2023-07-13-malware-analysis$
```

Privilege Escalation - Using `CoGetObject`, the ransomware registers itself with the CLSID `3E5FC7F9-9A51-4367-9063-A120244FBEC7`, which is legitimately used to execute applications with elevated privileges. This technique enables the malware to circumvent the UAC prompt and execute its malicious actions without being detected or blocked by the system's security measures.

```
if ( CoGetObject_(
       L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}",
       (BIND_OPTS *)&CLSID_ICMLuaUtil,
       (const IID *const)4,
       (void **)ppv) )
```

Anti-disassembly - `Sleep` function make stepping through code in a debugger more time-consuming and thus complicate the process of reverse engineering:



Terminating all active services and processes - BlackCat will now attempt to terminate any processes or services specified in the configuration, such as processes that may inhibit the encryption procedure.

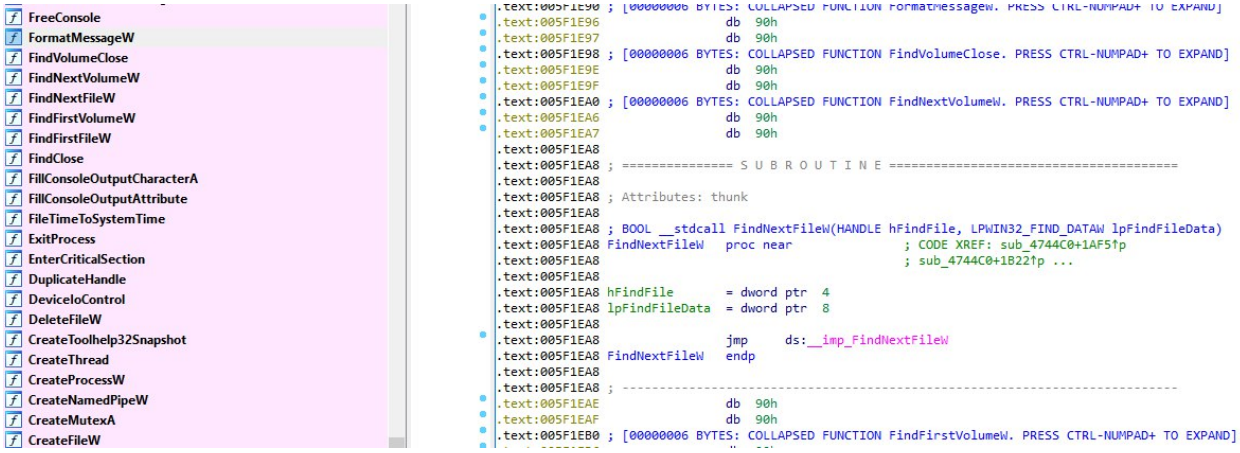
Kill services:

```
backup memtas mepocs msexchange sql svc$ veeam vss
```

and processes:

```
"kill_processes": [
  "encsvc",
  "thebat",
  "mydesktopqos",
  "xfssvccon",
  "firefox",
  "infopath",
  "winword",
  "steam",
  "synctime",
  "notepad",
  "ocomm",
  "onenote",
  "mspub",
  "thunderbird",
  "agntsvc",
  "sql",
  "excel",
  "powerpnt",
  "outlook",
  "wordpad",
  "dbeng50",
  "isqlplussvc",
  "sqbcoreservice",
  "oracle",
  "ocautoupds",
  "dbsnmp",
  "msaccess",
  "tbirdconfig",
  "ocssd",
  "mydesktopservice",
  "visio"
]
```

Encryption process - BlackCat initially traverses the system using a cycle of `FindFirstFile` and `FindNextFile` to locate all system files:



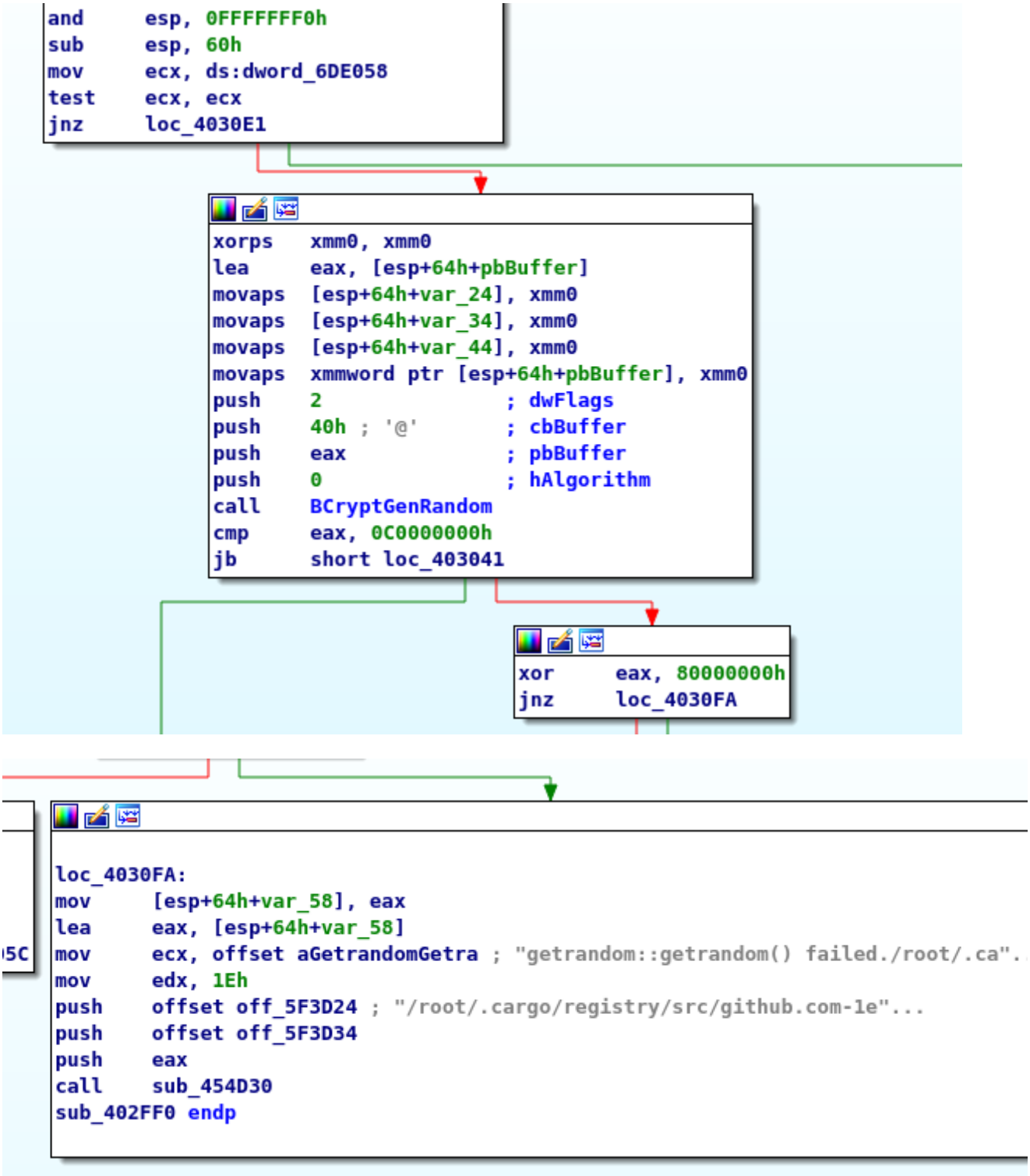
```
; Attributes: thunk

; HANDLE __stdcall FindFirstFileW(LPCWSTR lpFileName, LPWIN32_FIND_DATAW lpFindFileData)
FindFirstFileW proc near

lpFileName= dword ptr  4
lpFindFileData= dword ptr  8

jmp     ds:__imp_FindFirstFileW
FindFirstFileW endp
```

Then ransom note is written using `WriteFile` to each directory:

```
mov     eax, [ebp+hFile]
cmp     eax, 0FFFFFFFFh
jz      short loc_401EFE

push    0                   ; lpOverlapped
lea     ecx, [ebp+NumberOfBytesWritten]
mov     [ebp+NumberOfBytesWritten], 0
push    ecx                 ; lpNumberOfBytesWritten
push    esi                 ; nNumberOfBytesToWrite
push    edi                 ; lpBuffer
push    eax                 ; hFile
call    ds:WriteFile
test    eax, eax
jz      short loc_401EFE
```

Using `BCryptGenRandom`, the ransomware calculates a random `AES` key:

```
loc_4018F0:
xor     byte ptr [ebx+eax*4+1], 0C0h
movdqa  xmm4, ds:xmmword_5F2060
movdqa  xmm6, ds:xmmword_5F2080
movdqa
movdqa
inc
add
inc     edx
movdqu  xmm0, xmmword ptr [ebx]

loc_401DCA:
mov     edx, 8
push    offset off_5F3CA0 ; "/root/.cargo/registry/src/github.com-1e"...
call    sub_4519E0

aRootCargoRegis_1 db '/root/.cargo/registry/src/github.com-1ecc6299db9ec823/aes-0.7.5/s'
                                        ; DATA XREF: .rdata:off_5F3C70↓o
                                        ; .rdata:off_5F3C80↓o ...
                  db 'rc/soft/fixslice32.rs',0
```



The file's contents are written to the file using `ReadFile` and `WriteFile` after it has been encrypted with `AES`. The new file extension is listed in the BlackCat configuration.

```
hFile= dword ptr  4
lpBuffer= dword ptr  8
nNumberOfBytesToWrite= dword ptr  0Ch
lpNumberOfBytesWritten= dword ptr  10h
lpOverlapped= dword ptr  14h

jmp     ds:__imp_WriteFile
WriteFile endp
```

We created a simple BlackCat Ransomware configuration extractor:

```
import hashlib
import os
import json
import binascii
import argparse
import sys
from typing import Union

class BlackCatConfig:
    def __init__(self, config: dict):
        self.config = config

    def __str__(self):
        output = ""
        for key in self.config.keys():
            output += f"{key}: {self.config[key]}\n"
        return output

def calc_md5(data: bytes) -> str:
    hasher = hashlib.md5()
    hasher.update(data)
    return hasher.hexdigest()

def calc_sha256(data: bytes) -> str:
    hasher = hashlib.sha256()
    hasher.update(data)
    return hasher.hexdigest()

def get_file_info(file: str) -> int:
    return os.stat(file).st_size

def scan_file(data: bytes, search: bytes) -> Union[int, None]:
    return data.find(search)

def main():
    parser = argparse.ArgumentParser(description='BlackCat Ransomware conf
extractor')
    parser.add_argument('-j', '--json', action='store_true', help='dump extracted
config to a json file')
    parser.add_argument('file', type=str, help='path to sample')
    args = parser.parse_args()

    try:
        with open(args.file, 'rb') as f:
            data = f.read()

        print(f"file size (bytes): {get_file_info(args.file)}")
        print(f"MD5: {calc_md5(data)}")
        print(f"SHA-256: {calc_sha256(data)}")

        off = scan_file(data, binascii.unhexlify("7B22636F6E6669675F696422"))  # =
{"config_id"
```

```
        if off == -1:
            print("\nunable to find config offset :(\n\n")
            sys.exit(1)

        cfg = data[off: off+8000].strip()

        if args.json:
            filename = f"blackCat_config-{calc_md5(data)}.json"
            with open(filename, 'w') as jsonOutput:
                jsonOutput.write(cfg.decode('utf-8'))
                print(f"\nwrote {len(cfg)} bytes to {filename}\n\n")

        config = BlackCatConfig(json.loads(cfg))
        print(config)

    except Exception as e:
        print(f"an error occurred: {e}")
        sys.exit(1)

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

After BlackCat has finished encrypting all files on the system, the desktop wallpaper is altered to direct the user to the ransom note.

IOC

Another samples (SHA-256):

- 0c6f444c6940a3688ffc6f8b9d5774c032e3551ebbccb64e4280ae7fc1fac479
- 13828b390d5f58b002e808c2c4f02fdd920e236cc8015480fa33b6c1a9300e31
- 15b57c1b68cd6ce3c161042e0f3be9f32d78151fe95461eedc59a79fc222c7ed
- 1af1ca666e48afc933e2eda0ae1d6e88ebd23d27c54fd1d882161fd8c70b678e
- 2587001d6599f0ec03534ea823aab0febb75e83f657fadc3a662338cc08646b0
- 28d7e6fe31dc00f82cb032ba29aad6429837ba5efb83c2ce4d31d565896e1169
- 2cf54942e8cf0ef6296deaa7975618dadff0c32535295d3f0d5f577552229ffc
- 38834b796ed025563774167716a477e9217d45e47def20facb027325f2a790d1
- 3d7cf20ca6476e14e0a026f9bdd8ff1f26995cdc5854c3adb41a6135ef11ba83
- 40f57275721bd74cc59c0c59c9f98c8e0d1742b7ae86a46e83e985cc4039c3a5
- 4e18f9293a6a72d5d42dad179b532407f45663098f959ea552ae43dbb9725cbf
- 59868f4b346bd401e067380cac69080709c86e06fae219bfb5bc17605a71ab3f
- 5bdc0fb5cfbd42de726aacc40eddca034b5fa4afcc88ddfb40a3d9ae18672898

- 658e07739ad0137bceb910a351ce3fe4913f6fcc3f63e6ff2eb726e45f29e582
- 731adcf2d7fb61a8335e23dbee2436249e5d5753977ec465754c6b699e9bf161
- 7b2449bb8be1b37a9d580c2592a67a759a3116fe640041d0f36dc93ca3db4487
- 7e363b5f1ba373782261713fa99e8bbc35ddda97e48799c4eb28f17989da8d8e
- bd337d4e83ab1c2cacb43e4569f977d188f1bb7c7a077026304bf186d49d4117
- be8c5d07ab6e39db28c40db20a32f47a97b7ec9f26c9003f9101a154a5a98486
- c3e5d4e62ae4eca2bfca22f8f3c8cbec12757f78107e91e85404611548e06e40
- c5ad3534e1c939661b71f56144d19ff36e9ea365fdb47e4f8e2d267c39376486
- c8b3b67ea4d7625f8b37ba59eed5c9406b3ef04b7a19b97e5dd5dab1bd59f283
- cefea76dfdbb48cfe1a3db2c8df34e898e29bec9b2c13e79ef40655c637833ae
- f815f5d6c85bcbc1ec071dd39532a20f5ce910989552d980d1d4346f57b75f89
- f837f1cd60e9941aa60f7be50a8f2aaaac380f560db8ee001408f35c1b7a97cb

Bitcoin: `1H3JFbyiwv6YeVW7K2mVjxHgNvJdXqJxiP`
Monero:
`46JqTG57Pv6GBRzjM9kHyCF8XHrAo9sr8dLuvqwcGbxT92dUAW12QpgZJnu32KrTfL1BzLp2sBi9G49JyXuRaKmT6JrJL9r`

Yara rules

Yara rule for BlackCat Ransomware threat hunting:

```
rule win_blackcat_auto {

    meta:
        author = "Felix Bilstein - yara-signator at cocacoding dot com"
        date = "2023-03-28"
        version = "1"
        description = "Detects win.blackcat."
        info = "autogenerated rule brought to you by yara-signator"
        tool = "yara-signator v0.6.0"
        signator_config = "callsandjumps;datarefs;binvalue"
        malpedia_reference =
"https://malpedia.caad.fkie.fraunhofer.de/details/win.blackcat"
        malpedia_rule_date = "20230328"
        malpedia_hash = "9d2d75cef573c1c2d861f5197df8f563b05a305d"
        malpedia_version = "20230407"
        malpedia_license = "CC BY-SA 4.0"
        malpedia_sharing = "TLP:WHITE"

    /* DISCLAIMER
     * The strings used in this rule have been automatically selected from the
     * disassembly of memory dumps and unpacked files, using YARA-Signator.
     * The code and documentation is published here:
     * https://github.com/fxb-cocacoding/yara-signator
     * As Malpedia is used as data source, please note that for a given
     * number of families, only single samples are documented.
     * This likely impacts the degree of generalization these rules will offer.
     * Take the described generation method also into consideration when you
     * apply the rules in your use cases and assign them confidence levels.
     */


    strings:
        $sequence_0 = { c3 81f90a010000 7e6a 81f9e2030000 0f8fcc000000 81f90b010000 }
            // n = 6, score = 600
            //   c3                  | ret
            //   81f90a010000        | cmp                 ecx, 0x10a
            //   7e6a                | jle                 0x6c
            //   81f9e2030000        | cmp                 ecx, 0x3e2
            //   0f8fcc000000        | jg                  0xd2
            //   81f90b010000        | cmp                 ecx, 0x10b

        $sequence_1 = { 85f6 0f8482000000 bb03000000 8d0437 }
            // n = 4, score = 600
            //   85f6                | test                esi, esi
            //   0f8482000000        | je                  0x88
            //   bb03000000          | mov                 ebx, 3
            //   8d0437              | lea                 eax, [edi + esi]

        $sequence_2 = { 885405cc 48 eb19 89ca 83fa63 7fbe }
            // n = 6, score = 600
            //   885405cc            | mov                 byte ptr [ebp + eax -
0x34], dl
```

```
        //   48                  | dec                 eax
        //   eb19                | jmp                 0x1b
        //   89ca                | mov                 edx, ecx
        //   83fa63              | cmp                 edx, 0x63
        //   7fbe                | jg                  0xffffffc0

    $sequence_3 = { f20f104808 8d45d4 894dec c645f004 8d4dec }
        // n = 5, score = 600
        //   f20f104808          | movsd               xmm1, qword ptr [eax + 8]
        //   8d45d4              | lea                 eax, [ebp - 0x2c]
        //   894dec              | mov                 dword ptr [ebp - 0x14],
ecx
        //   c645f004            | mov                 byte ptr [ebp - 0x10], 4
        //   8d4dec              | lea                 ecx, [ebp - 0x14]

    $sequence_4 = { 3d32210000 747b 3d33210000 0f8571050000 8b07 }
        // n = 5, score = 600
        //   3d32210000          | cmp                 eax, 0x2132
        //   747b                | je                  0x7d
        //   3d33210000          | cmp                 eax, 0x2133
        //   0f8571050000        | jne                 0x577
        //   8b07                | mov                 eax, dword ptr [edi]

    $sequence_5 = { b005 5e 5d c3 81f90a010000 7e6a 81f9e2030000 }
        // n = 7, score = 600
        //   b005                | mov                 al, 5
        //   5e                  | pop                 esi
        //   5d                  | pop                 ebp
        //   c3                  | ret
        //   81f90a010000        | cmp                 ecx, 0x10a
        //   7e6a                | jle                 0x6c
        //   81f9e2030000        | cmp                 ecx, 0x3e2

    $sequence_6 = { 747b 3d33210000 0f8571050000 8b07 83f00a }
        // n = 5, score = 600
        //   747b                | je                  0x7d
        //   3d33210000          | cmp                 eax, 0x2133
        //   0f8571050000        | jne                 0x577
        //   8b07                | mov                 eax, dword ptr [edi]
        //   83f00a              | xor                 eax, 0xa

    $sequence_7 = { b806000000 c7460400000000 894608 c70601000000 83c430 }
        // n = 5, score = 600
        //   b806000000          | mov                 eax, 6
        //   c7460400000000      | mov                 dword ptr [esi + 4], 0
        //   894608              | mov                 dword ptr [esi + 8], eax
        //   c70601000000        | mov                 dword ptr [esi], 1
        //   83c430              | add                 esp, 0x30

    $sequence_8 = { 89d0 ba3e000000 897e0c f7e2 }
        // n = 4, score = 600
        //   89d0                | mov                 eax, edx
```

```
            //   ba3e000000           | mov                 edx, 0x3e
            //   897e0c               | mov                 dword ptr [esi + 0xc],
edi
            //   f7e2                 | mul                 edx

        $sequence_9 = { c6410b00 66c741090000 8b45ec 894110 c7411400000000 b801000000
8901 }
            // n = 7, score = 600
            //   c6410b00             | mov                 byte ptr [ecx + 0xb], 0
            //   66c741090000         | mov                 word ptr [ecx + 9], 0
            //   8b45ec               | mov                 eax, dword ptr [ebp -
0x14]
            //   894110               | mov                 dword ptr [ecx + 0x10],
eax
            //   c7411400000000       | mov                 dword ptr [ecx + 0x14], 0
            //   b801000000           | mov                 eax, 1
            //   8901                 | mov                 dword ptr [ecx], eax

    condition:
        7 of them and filesize < 29981696
}
```

MITRE ATT&CK

T1027.002 - Obfuscated Files or Information: Software Packing
T1027 - Obfuscated Files or Information
T1007 - System Service Discovery
T1059 - Command and Scripting Interpreter
TA0010 - Exfiltration
T1082 - System Information Discovery
T1490 - Inhibit System Recovery
T1485 - Data Destruction
T1078 - Valid Accounts
T1486 - Data Encrypted For Impact
T1140 - Encode/Decode Files or Information
T1202 - Indirect Command Execution
T1543.003 - Create or Modify System Process: Windows Service
T1550.002 - Use Alternate Authentication Material: Pass The Hash

By Cyber Threat Hunters from MSSPLab:

- @cocomelonc
- @wqkasper
- @mgmadr

References

MITRE ATT&CK: BlackCat
Salsa20 wikipedia
An ALPHV (BlackCat) representative discusses the group's plans for a ransomware 'meta-universe'
malpedia: BlackCat

Thanks for your time happy hacking and good bye!
All drawings and screenshots are MSSPLab's