Surreaction - StarHackademINT 2023

Introduction

This post is about an interesting forensic challenge that I solved during the StarHackademINT 2023 CTF. We were given a `Quarantine` folder coming from Windows Defender, and we had to find back the original filename on the internet.

The structure of the given folder looks like this:

```
Quarantine
├── Entries
│   └── {80059732-0000-0000-6667-EF3396D235E7}
├── ResourceData
│   └── 9F
│       └── 9F598F562DDCFB69FA21A077BAD87F01A3F6258E
└── Resources
    └── 9F
        └── 9F598F562DDCFB69FA21A077BAD87F01A3F6258E
```

If we look on the internet, we can find some blog post that gives a lot of useful information on how to get the original files back. We can find the `Quarantine` directory structure:

- The `Entries` folder contains information about the quarantined files, such as the reason of the quarantine, the original full path, timestamp, etc. The files are encrypted in a particular way that we will see later.
- The `ResourceData` folder contains the encrypted original binary.
- The `Resources` folder won't be used here, but it contains information such as a GUID and a hash used to link the `Entries` and the `ResourceData` folder.

We can also find the hardcoded RC4 key used to encrypt the files (it is common for AV vendors to have a single RC4/XOR hardcoded key) used for the quarantine process. So we can use this small python script in order to decrypt the files:

```
from Crypto.Cipher import ARC4
import sys

with open(sys.argv[1], "rb") as encrypted_file:
    print(f"Reading {sys.argv[1]}")
    data = encrypted_file.read()

key = b"\x1E\x87\x78\x1B\x8D\xBA\xA8\x44\xCE\x69\x70\x2C\x0C\x78\xB7\x86\xA3\xF6\x23\xB7\x38\xF5\xED\xF9\xAF\x83\x53\x0F\xB
3\xFC\x54\xFA\xA2\x1E\xB9\xCF\x13\x31\xFD\x0F\x0D\xA9\x54\xF6\x87\xCB\x9E\x18\x27\x96\x97\x90\x0E\x53\xFB\x31\x7C\x9C\xBC\x
E4\x8E\x23\xD0\x53\x71\xEC\xC1\x59\x51\xB8\xF3\x64\x9D\x7C\xA3\x3E\xD6\x8D\xC9\x04\x7E\x82\xC9\xBA\xAD\x97\x99\xD0\xD4\x58
\xCB\x84\x7C\xA9\xFF\xBE\x3C\x8A\x77\x52\x33\x55\x7D\xDE\x13\xA8\xB1\x40\x87\xCC\x1B\xC8\xF1\x0F\x6E\xCD\xD0\x83\xA9\x59\xC
F\xF8\x4A\x9D\x1D\x50\x75\x5E\x3E\x19\x18\x18\xAF\x23\xE2\x29\x35\x58\x76\x6D\x2C\x07\xE2\x57\x12\xB2\xCA\x0B\x53\x5E\xD8\x
F6\xC5\x6C\xE7\x3D\x24\xBD\xD0\x29\x17\x71\x86\x1A\x54\xB4\xC2\x85\xA9\xA3\xDB\x7A\xCA\x6D\x22\x4A\xEA\xCD\x62\x1D\xB9\xF2
\xA2\x2E\xD1\xE9\xE1\x1D\x75\xBE\xD7\xDC\x0E\xCB\x0A\x8E\x68\xA2\xFF\x12\x63\x40\x8D\xC8\x08\xDF\xFD\x16\x4B\x11\x67\x74\xC
D\x0B\x9B\x8D\x05\x41\x1E\xD6\x26\x2E\x42\x9B\xA4\x95\x67\x6B\x83\x98\xDB\x2F\x35\xD3\xC1\xB9\xCE\xD5\x26\x36\xF2\x76\x5E\x
1A\x95\xCB\x7C\xA4\xC3\xDD\xAB\xDD\xBF\xF3\x82\x53"

cipher = ARC4.new(key)
decrypted = cipher.decrypt(data)

with open(sys.argv[1]+".decrypted", "wb") as dec:
    print(f"Writing decrypted file to {sys.argv[1]}.decrypted")
    dec.write(decrypted)
```

Many scripts are avaibles on the internet, but they were not working properly for me.

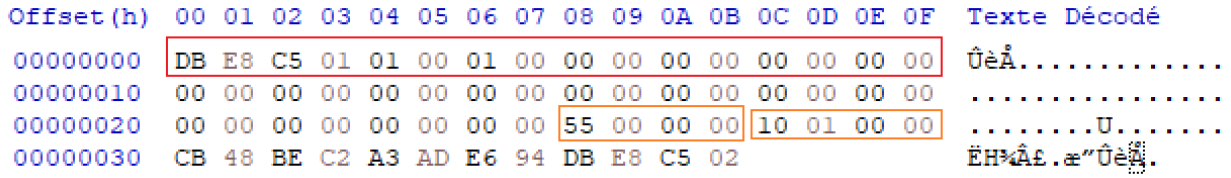
Entries folder

The first thing that I wanted to do was to decrypt the file in the `Entries` folder. The main issue is that the file is separated in 3 chunks, encrypted separately using the RC4 key.

If we take a look at this document, we can see that the first chunk has a fixed size:

```
seq:
  - id: header
    type: rc4encrypted_header
    size: 0x3C
    process: util.custom_arc4.custom_arc4(<RC4 key>)
  - id: data1
    size: header.len1
    type: encrypted_data1
    process: util.custom_arc4.custom_arc4(<RC4 key>)
  - id: data2
    size: header.len2
    type: encrypted_data2
    process: util.custom_arc4.custom_arc4(<RC4 key>)
```

So we can just truncate the first chunk (I used HxD in order to display the data efficiently, but you can do it with python), and decrypt it using our script. We get the following data:



We have our magic bytes, `0x18` null bytes, 2 sizes (4 bytes each), and then `0x0C` bytes:

```
  size: 0x10
  contents: [0xdb, 0xe8, 0xc5, 0x01, 0x01, 0, 0x01, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- id: unknown1
  size: 0x18
- id: len1
  type: u4
- id: len2
  type: u4
- id: unknown2
  size: 0x0C
```

We got the two chunks size (big endian, then we have `0x55` and `0x101`), so let's truncate (**[0x3C:0x3C+0x55]** and [**0x3C+0x56:END]**), and decrypt them:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	32	97	05	80	00	00	00	00	66	67	EF	33	96	D2	35	E7	2—.€....fgï3–Ò5ç
00000010	FB	CB	56	03	D6	B5	3C	45	AE	22	59	58	EE	C6	04	AE	ûËV.Öµ<E®"YXîÆ.®
00000020	D6	2B	DA	40	43	DF	D9	01	32	97	05	80	00	00	00	00	Ö+Ú@CßÙ.2—.€....
00000030	01	00	00	00	54	72	6F	6A	61	6E	3A	57	69	6E	36	34	....Trojan:Win64
00000040	2F	4D	65	74	61	73	70	6C	6F	69	74	2E	43	52	54	44	/Metasploit.CRTD
00000050	21	4D	54	42	00												!MTB.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	01	00	00	00	08	00	00	00	5C	00	5C	00	3F	00	5C	00	........\.\.?.\.
00000010	43	00	3A	00	5C	00	55	00	73	00	65	00	72	00	73	00	C.:.\.U.s.e.r.s.
00000020	5C	00	6F	00	77	00	6C	00	79	00	64	00	5C	00	44	00	\.o.w.l.y.d.\.D.
00000030	6F	00	77	00	6E	00	6C	00	6F	00	61	00	64	00	73	00	o.w.n.l.o.a.d.s.
00000040	5C	00	76	00	69	00	72	00	75	00	73	00	2E	00	65	00	\.v.i.r.u.s...e.
00000050	78	00	65	00	00	00	09	00	66	69	6C	65	00	00	00	00	x.e.....file....
00000060	14	00	02	40	9F	59	8F	56	2D	DC	FB	69	FA	21	A0	77	...@ŸY.V-ÜûiúÍ w
00000070	BA	D8	7F	01	A3	F6	25	8E	08	00	12	60	00	1A	00	00	°Ø..£ö%Ž...`....
00000080	00	00	00	00	08	00	11	60	FC	1A	C4	B9	42	DF	D9	01	.......`ü.Ä¹BßÙ.
00000090	08	00	10	60	13	91	D9	40	43	DF	D9	01	08	00	0F	60	...`.'Ù@CßÙ....`
000000A0	6E	D7	48	B5	42	DF	D9	01	04	00	0A	30	20	00	00	00	n×HµBßÙ....0 ...
000000B0	46	00	0C	20	43	00	3A	00	5C	00	55	00	73	00	65	00	F.. C.:.\.U.s.e.
000000C0	72	00	73	00	5C	00	6F	00	77	00	6C	00	79	00	64	00	r.s.\.o.w.l.y.d.
000000D0	5C	00	44	00	6F	00	77	00	6E	00	6C	00	6F	00	61	00	\.D.o.w.n.l.o.a.
000000E0	64	00	73	00	5C	00	76	00	69	00	72	00	75	00	73	00	d.s.\.v.i.r.u.s.
000000F0	2E	00	65	00	78	00	65	00	00	00	00	00	04	00	0E	30	..e.x.e........0
00000100	80	00	00	00	08	00	0D	50	11	EC	02	68	67	26	00	00	€.....P.ì.hg&..

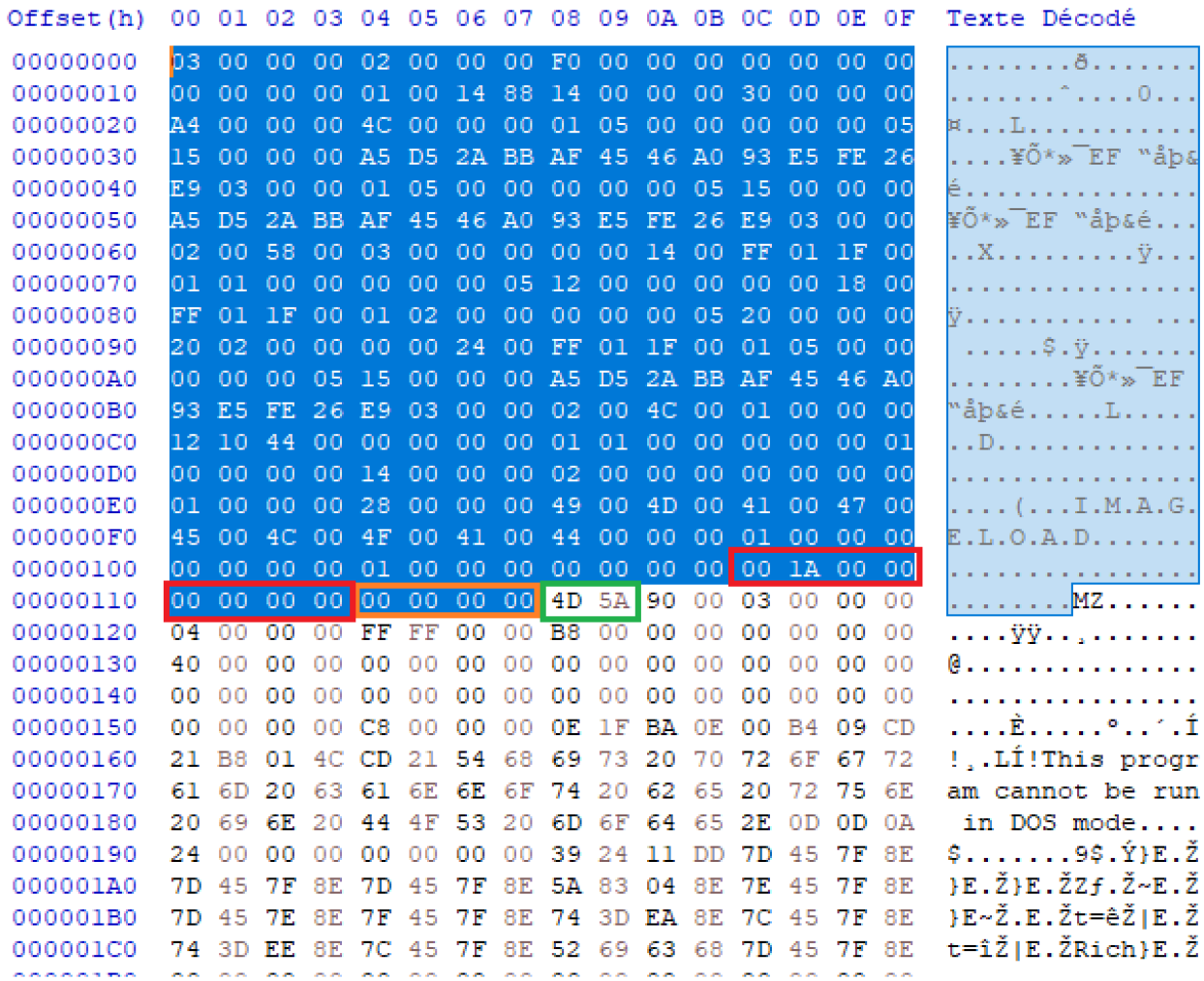
Well, this is interesting but `Star{virus.exe}` is not the flag… So we will have to decrypt the original malware file.

ResourceData folder

Luckily for us, the malware file is encrypted in one chunk. Some metadata are added at the beginning, and then we have our malware file:

```
rc4encrypted:
  seq:
  - id: fixed
  contents: [0x03, 0, 0, 0, 0x02, 0, 0, 0]
  size: 8
  - id: length
  type: u4
  - id: padding
  size: 0x08
  - id: binarysd
  size: length
  - id: unknown1
  size: 0x08
  - id: len_malfile
  type: u8
  - id: unknown2
  size: 0x04
  - id: mal_file
  size: len_malfile
```

If we decrypt it, we can see the PE magic bytes (in green) as well as the *This program cannot be run in DOS mode.* Everything before is our metadatas. If we look at the metadata structure, we can see that the malfile size is 8 bytes long (in red). So we have a malfile size of `1A00` (6656) bytes:



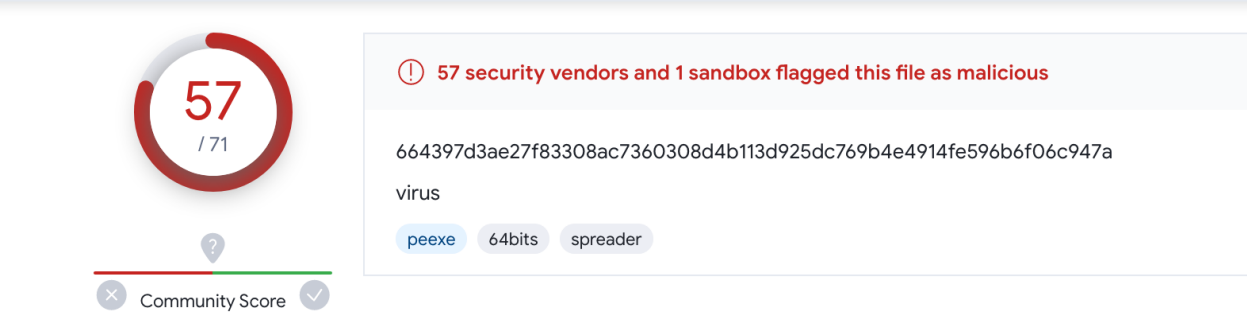
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	03	00	00	00	02	00	00	00	F0	00	00	00	00	00	00	00	........ð.......
00000010	00	00	00	00	01	00	14	88	14	00	00	00	30	00	00	00	........ˆ....0...
00000020	A4	00	00	00	4C	00	00	00	01	05	00	00	00	00	00	05	¤...L...........
00000030	15	00	00	00	A5	D5	2A	BB	AF	45	46	A0	93	E5	FE	26	....¥Õ*»¯EF "åþ&
00000040	E9	03	00	00	01	05	00	00	00	00	00	05	15	00	00	00	é...............
00000050	A5	D5	2A	BB	AF	45	46	A0	93	E5	FE	26	E9	03	00	00	¥Õ*»¯EF "åþ&é...
00000060	02	00	58	00	03	00	00	00	00	00	14	00	FF	01	1F	00	..X.........ÿ...
00000070	01	01	00	00	00	00	00	05	12	00	00	00	00	00	18	00	....
00000080	FF	01	1F	00	01	02	00	00	00	00	00	05	20	00	00	00	ÿ........... ...
00000090	20	02	00	00	00	00	24	00	FF	01	1F	00	01	05	00	00	.....$.ÿ.......
000000A0	00	00	00	05	15	00	00	00	A5	D5	2A	BB	AF	45	46	A0	.........¥Õ*»¯EF
000000B0	93	E5	FE	26	E9	03	00	00	02	00	4C	00	01	00	00	00	"åþ&é.....L.....
000000C0	12	10	44	00	00	00	00	00	01	01	00	00	00	00	00	01	..D.............
000000D0	00	00	00	00	14	00	00	00	02	00	00	00	00	00	00	00	....
000000E0	01	00	00	00	28	00	00	00	49	00	4D	00	41	00	47	00	....(...I.M.A.G.
000000F0	45	00	4C	00	4F	00	41	00	44	00	00	00	01	00	00	00	E.L.O.A.D.......
00000100	00	00	00	00	01	00	00	00	00	00	00	00	00	1A	00	00	....
00000110	00	00	00	00	00	00	00	00	4D	5A	90	00	03	00	00	00	........MZ......
00000120	04	00	00	00	FF	FF	00	00	B8	00	00	00	00	00	00	00	....ÿÿ..¸.......
00000130	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	@...............
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000150	00	00	00	00	C8	00	00	00	0E	1F	BA	0E	00	B4	09	CD	....È.....º..´.Í
00000160	21	B8	01	4C	CD	21	54	68	69	73	20	70	72	6F	67	72	!¸.LÍ!This progr
00000170	61	6D	20	63	61	6E	6E	6F	74	20	62	65	20	72	75	6E	am cannot be run
00000180	20	69	6E	20	44	4F	53	20	6D	6F	64	65	2E	0D	0D	0A	in DOS mode....
00000190	24	00	00	00	00	00	00	00	39	24	11	DD	7D	45	7F	8E	$.......9$.Ý}E.Ž
000001A0	7D	45	7F	8E	7D	45	7F	8E	5A	83	04	8E	7E	45	7F	8E	}E.Ž}E.ŽZƒ.Ž~E.Ž
000001B0	7D	45	7E	8E	7F	45	7F	8E	74	3D	EA	8E	7C	45	7F	8E	}E~Ž.E.Žt=êŽ|E.Ž
000001C0	74	3D	EE	8E	7C	45	7F	8E	52	69	63	68	7D	45	7F	8E	t=îŽ|E.ŽRich}E.Ž

At first, I truncated from the MZ bytes to the end and sent it to Virustotal. The file was flagged as malicious by ~50 AV vendors, but I couldn't find any original name for the file.

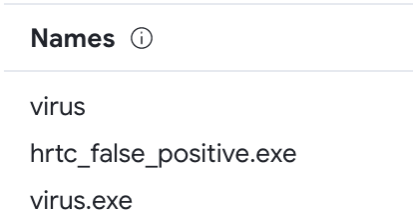
So I ended up truncating 6656 bytes from the MZ bytes. If we take a look at the end of the file, there seems to have appended data (after the highlighted part). Thoses data looks like a Mark of the web, but I don't really know why it is here:

```
000019F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001A00  02 00 00 00 00 00 00 00 72 00 00 00 00 00 00 00  ........r.......
00001A10  00 00 00 00 00 00 00 00 00 16 53 00 24 4B 45 52  ..........S.$KER
00001A20  4E 45 4C 2E 50 55 52 47 45 2E 45 53 42 43 41 43  NEL.PURGE.ESBCAC
00001A30  48 45 00 53 00 00 00 03 00 02 02 30 C9 4F B2 80  HE.S.......0ÉO²€
00001A40  DD D9 01 80 A7 04 01 FF 49 D8 01 02 80 00 00 35  ÝÙ.€§..ÿIØ..€..5
00001A50  00 0E 06 7C A8 D8 93 53 DF D9 01 FF FF FF FF 27  ...|¨Ø"SßÙ.ÿÿÿÿ'
00001A60  04 0C 80 00 00 20 1D 0E 7F 20 DD 44 F2 D8 89 84  ..€.. ... ÝDòØ‰„
00001A70  01 E0 6E 7D 4C EA E3 A8 BD 41 F0 AC 99 36 2C FC  .àn}LêãÂ½Að¬™6,ü
00001A80  8D B4 32 7D A5 E3 04 00 00 00 00 00 00 07 00  .´2}¥ã..........
00001A90  00 00 00 00 00 00 24 00 00 00 3A 00 53 00 6D 00  ......$...:.S.m.
00001AA0  61 00 72 00 74 00 53 00 63 00 72 00 65 00 65 00  a.r.t.S.c.r.e.e.
00001AB0  6E 00 3A 00 24 00 44 00 41 00 54 00 41 00 41 6E  n.:.$.D.A.T.A.An
00001AC0  61 68 65 69 6D                                   aheim
```

If we submit our new file to VT, it is still widely detected as malicious:



In the details section, we can see the name with which this files has been submitted or seen in the wild:



So the flag is `Star{hrtc_false_positive.exe}` !

Sources

https://static.ernw.de/whitepaper/ERNW-Whitepaper-71_AV_Quarantine_signed.pdf

https://reversingfun.com/posts/how-to-extract-quarantine-files-from-windows-defender/

https://www.virustotal.com/gui/file/664397d3ae27f83308ac7360308d4b113d925dc769b4e4914fe596b6f06c947a/details