Manual Unpacking IcedID Write-up

Published: 2020-08-16 · Archived: 2026-04-05 20:18:18 UTC

Sample hash:

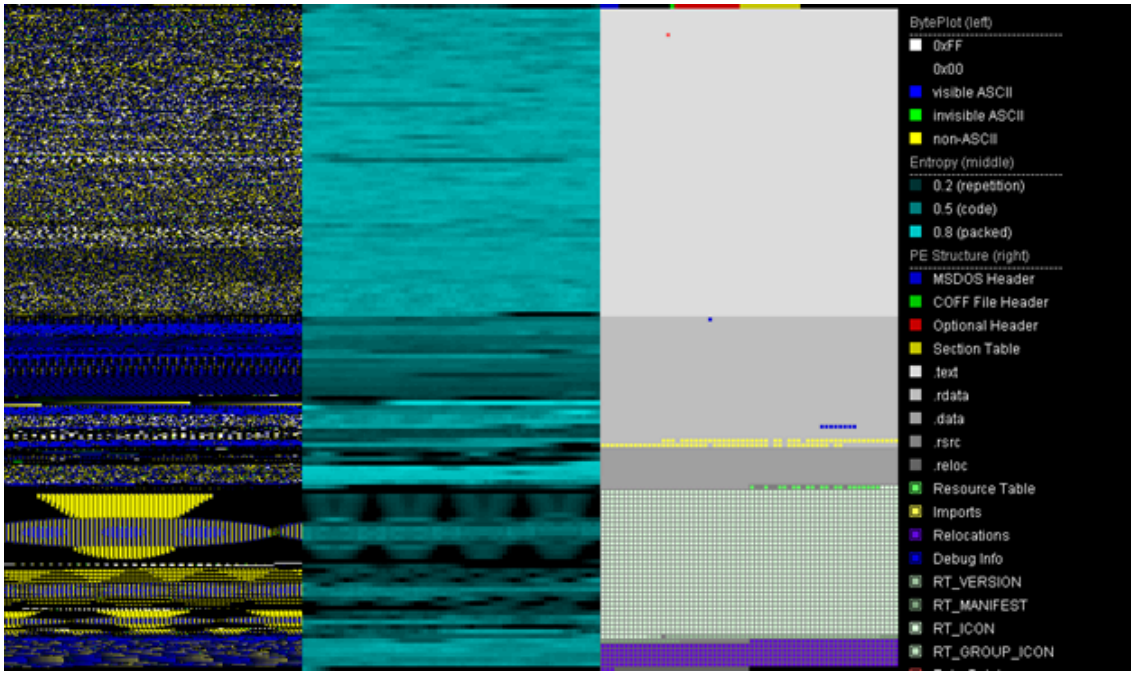
SHA256: 76cd290b236b11bd18d81e75e41682208e4c0a5701ce7834a9e289ea9e06eb7e

Tools:

- PE files static analysis: PortExAnalyzer; PE-bear
- Debugger & plugin: x64dbg + ScyllaHide Anti-Anti-Debug
- Aplib decompress: aplib-ripper

1. Static Analysis

Thow the sample to **PortEx Analyzer**, tool will analyse file with a special focus on malformation. We get the results:



The section **.text** has high entropy, so may be the sample is packed:

```
Section Table
*************
                        1. .text        2. .rdata       3. .data        4. .rsrc
-------------------------------------------------------------------------------
Entropy                 6.74            4.97            5.29            3.73
Pointer To Raw Data     0x400           0x10c00         0x17c00         0x19e00
Size Of Raw Data        0x10800         0x7000          0x2200          0x8400
Physical End            0x10c00         0x17c00         0x19e00         0x22200
Virtual Address         0x1000          0x12000         0x19000         0x34000
Virtual Size            0x106ab         0x6f4c          0x1a220         0x831c
-> actual virtual size  0x11000         0x7000          0x1b000         0x9000
Pointer To Relocations  0x0             0x0             0x0             0x0
Number Of Relocations   0x0             0x0             0x0             0x0
Pointer To Line Numbers 0x0             0x0             0x0             0x0
Number Of Line Numbers  0x0             0x0             0x0             0x0
Code                    x
Initialized Data                        x               x               x
Execute                 x
Read                    x               x               x               x
Write                                                   x
```

This sample is PE32 with **ASLR enabled** (can quickly disable this feature by using setdllcharacteristics):

```
Magic Number: PE32, normal executable file

Entry Point is in section 1 with name .text
DLL Characteristics  * DLL can be relocated at load time.
                     * Image is NX compatible.
                     * Terminal Server aware.
Subsystem:           The Windows graphical user interface (GUI) subsystem
```

This sample reveals information about the pdb path:

```
Debug Information
*****************

Time Date Stamp: Tue Feb 10 17:51:45 ICT 2015
Type: Visual C++ debug information

description         value           file offset
-----------------------------------------------------
Characteristics     0x0             0x10d70
Time/Date Stamp     0x54d9e2c1      0x10d74
Major Version       0x0             0x10d78
Minor Version       0x0             0x10d7a
Type                0x2             0x10d7c
Size of Data        0x7e            0x10d80
Address of Raw Data 0x17ef8         0x10d84
Pointer to Raw Data 0x16af8         0x10d88

Codeview
--------
Age:  1
GUID: 028afe6d-fecb-4ffb-862f-b9d8251b493f
File: c:\Sizeanger\CreatePick\mixpractice\Sciencescience\KeyContain\farterm\Tiresubtract\CenterSkinMass.pdb
```

Some anomalies were identified by **PortEx**:

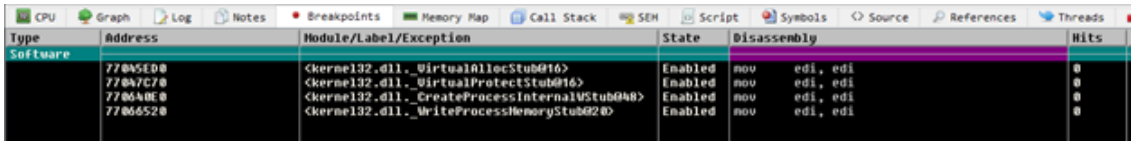
```
Anomalies
*********

* Import function typical for code injection: VirtualProtectEx may set PAGE_EXECUTE flag for memory region
* Import function typical for code injection: CreateThread is used to open and execute a thread in the victim process
```

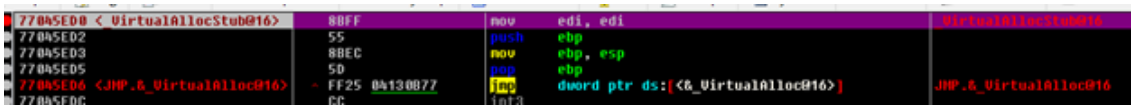
2. Dynamic Analysis

Load specimen to **x64dbg**, for unpacking process, we set breakpoints at some common APIs:

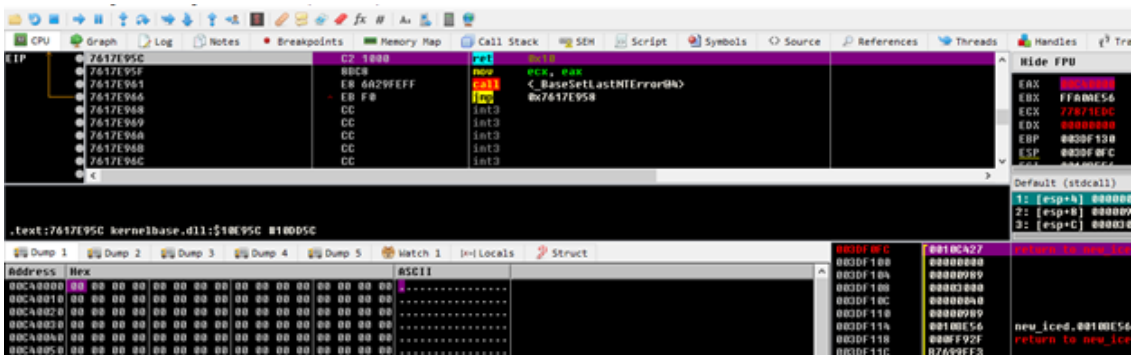
- `VirtualAlloc`
- `VirtualProtect`
- `CreateProcessInternalW`
- `WriteProcessMemory`



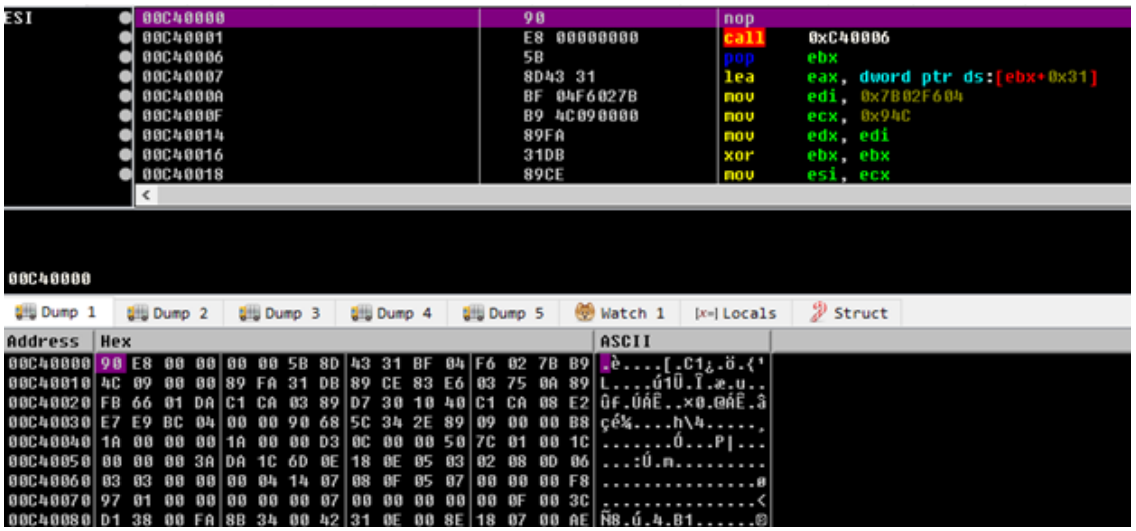
After placing the breakpoints like above picture, press `F9` to execute. First hit at `VirtualAlloc` :



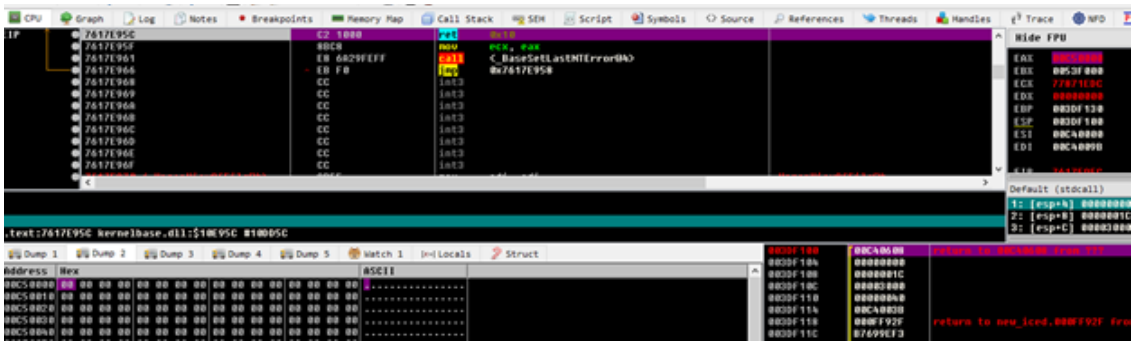
Execute till Return (`Ctrl+F9`) and Follow in dump the allocated memory (return in `EAX` register):



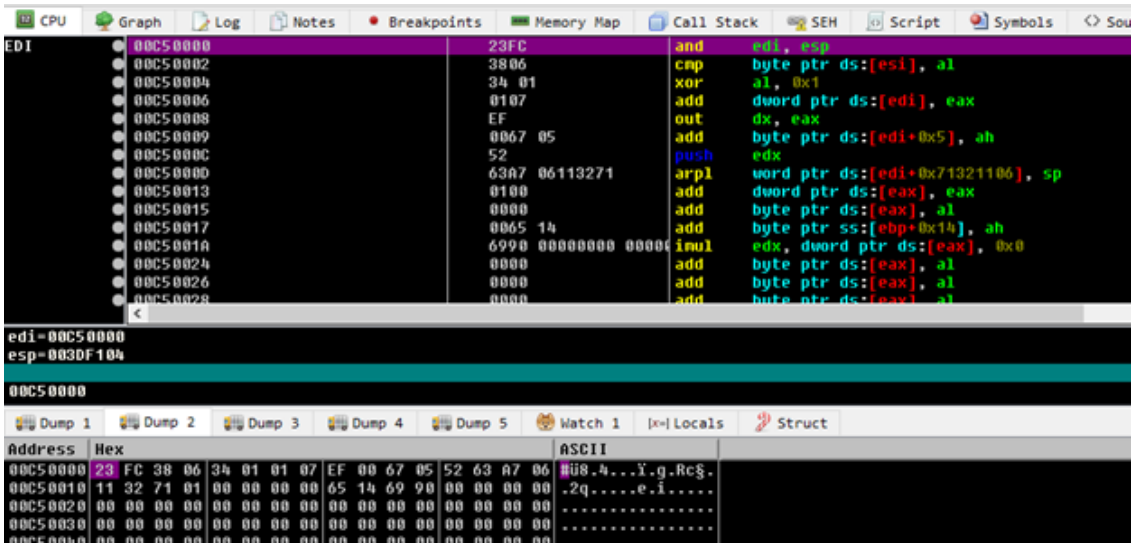
Continue run with `F9` , hit the second call to `VirtualAlloc` and observe changes in the allocated memory. We see new bytes value was written to this location and it is likely a shellcode:



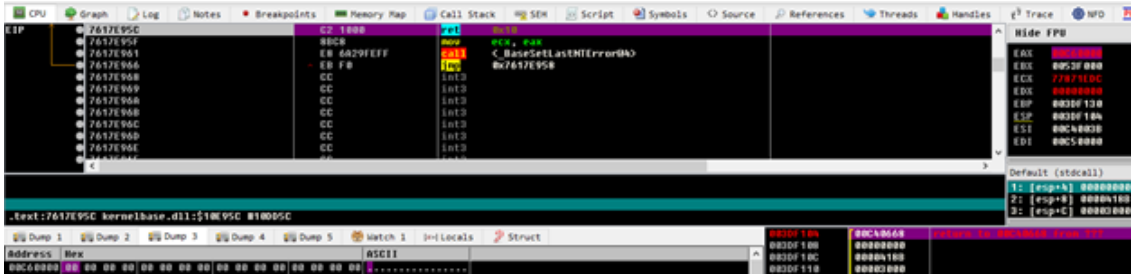
Once again, `Ctrl+F9` and Follow in dump the new allocated memory:



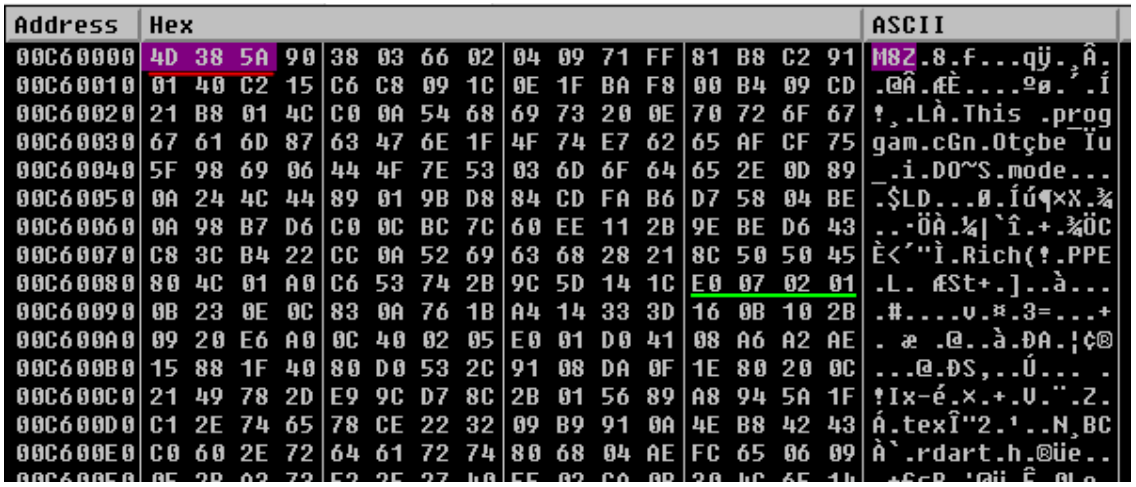
Let's continue execute and hit the third call to `VirtualAlloc`, some bytes were written to the new allocated memory. They do not look like shellcode but could be some data that malicious code uses:



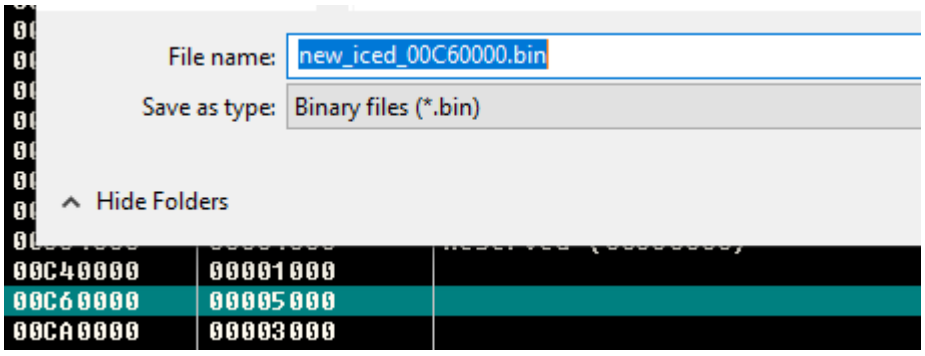
Continuing to execute the call to the `VirtuallAlloc` function, we have a newly allocated memory:



Press `F9`, we break at `VirtualProtect`. The newly allocated device has been filled with bytes. I spotted a PE file that has been compressed using *aPlib* because the PE magic bytes `MZ` become `M8Z`.

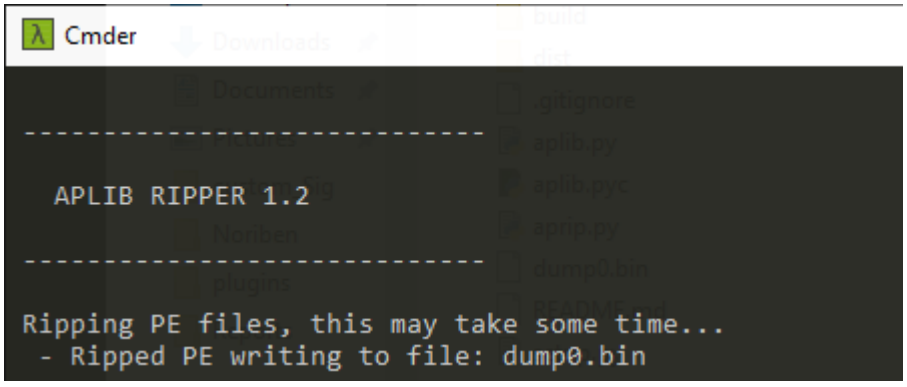


Follow this section in the **Memory Map** and dump it to file:

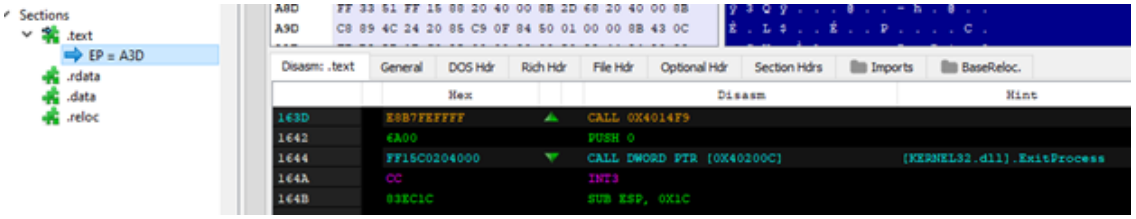


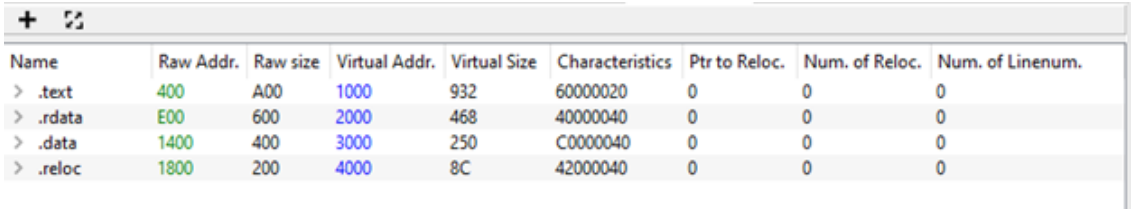
3. Decompress dumped file

From the command line, simply need to pass dumped file to `aprip.py`. The tool will do its job and each extracted file will be written to a file "**dump0.bin**", "**dump1.bin**", …



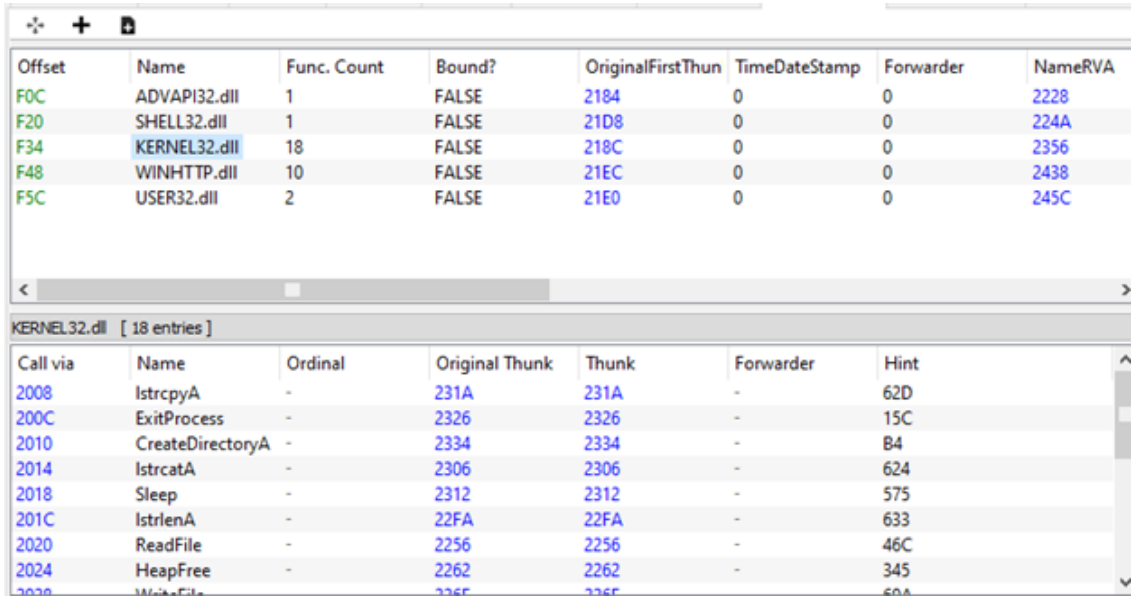
Check `dump0.bin (21dd005162c62af26f3f59e2ebcb345c)` with PE-bear: `AddressOfEntryPoint = 0x0000163D`





Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
.text	400	A00	1000	932	60000020	0	0	0
.rdata	E00	600	2000	468	40000040	0	0	0
.data	1400	400	3000	250	C0000040	0	0	0
.reloc	1800	200	4000	8C	42000040	0	0	0

Valid IATs:



Offset	Name	Func. Count	Bound?	OriginalFirstThun	TimeDateStamp	Forwarder	NameRVA
F0C	ADVAPI32.dll	1	FALSE	2184	0	0	2228
F20	SHELL32.dll	1	FALSE	21D8	0	0	224A
F34	KERNEL32.dll	18	FALSE	218C	0	0	2356
F48	WINHTTP.dll	10	FALSE	21EC	0	0	2438
F5C	USER32.dll	2	FALSE	21E0	0	0	245C

KERNEL32.dll [18 entries]

Call via	Name	Ordinal	Original Thunk	Thunk	Forwarder	Hint
2008	lstrcpyA	-	231A	231A	-	62D
200C	ExitProcess	-	2326	2326	-	15C
2010	CreateDirectoryA	-	2334	2334	-	B4
2014	lstrcatA	-	2306	2306	-	624
2018	Sleep	-	2312	2312	-	575
201C	lstrlenA	-	22FA	22FA	-	633
2020	ReadFile	-	2256	2256	-	46C
2024	HeapFree	-	2262	2262	-	345

End!

m4n0w4r

Source: https://kienmanowar.wordpress.com/2020/08/16/manual-unpacking-icedid-write-up/