ContiUnpacker

github.com/cdong1012/ContiUnpacker

cdong1012

cdong1012/
ContiUnpacker

An automatic unpacker for a Conti sample

1 Contributor
1 Issue
9 Stars
4 Forks

An automatic unpacker for a Conti sample

Context

- This was inspired by James Bennett's blog post on how to programmatically unpack malware.

- This unpacker unpacks this specific Conti ransomware I found on MalwareBazaar.

Requirement

- Python 3
- Speakeasy

How it works

- The unpacker uses the Speakeasy Emulation Framework to run and unpack the sample.

- When I manually unpacked this, I noticed that the sample called **VirtualAlloc** to allocate memory, wrote the unpacked PE file to it, and called **VirtualProtect** on the **.text** region before executing it.

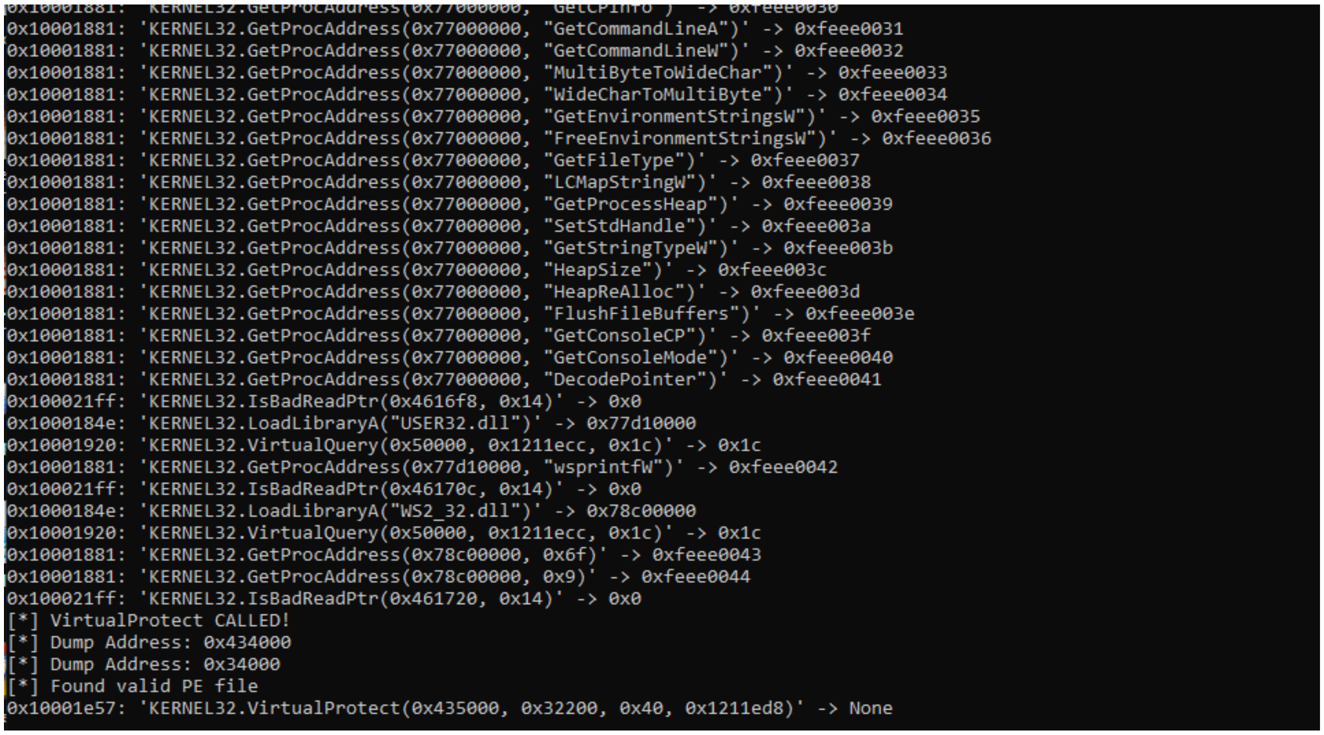
- From this, I halted the simulation at the first **VirtualProtect** call, dumped the PE file out, and mapped it accordingly to fix the IAT.

Usage

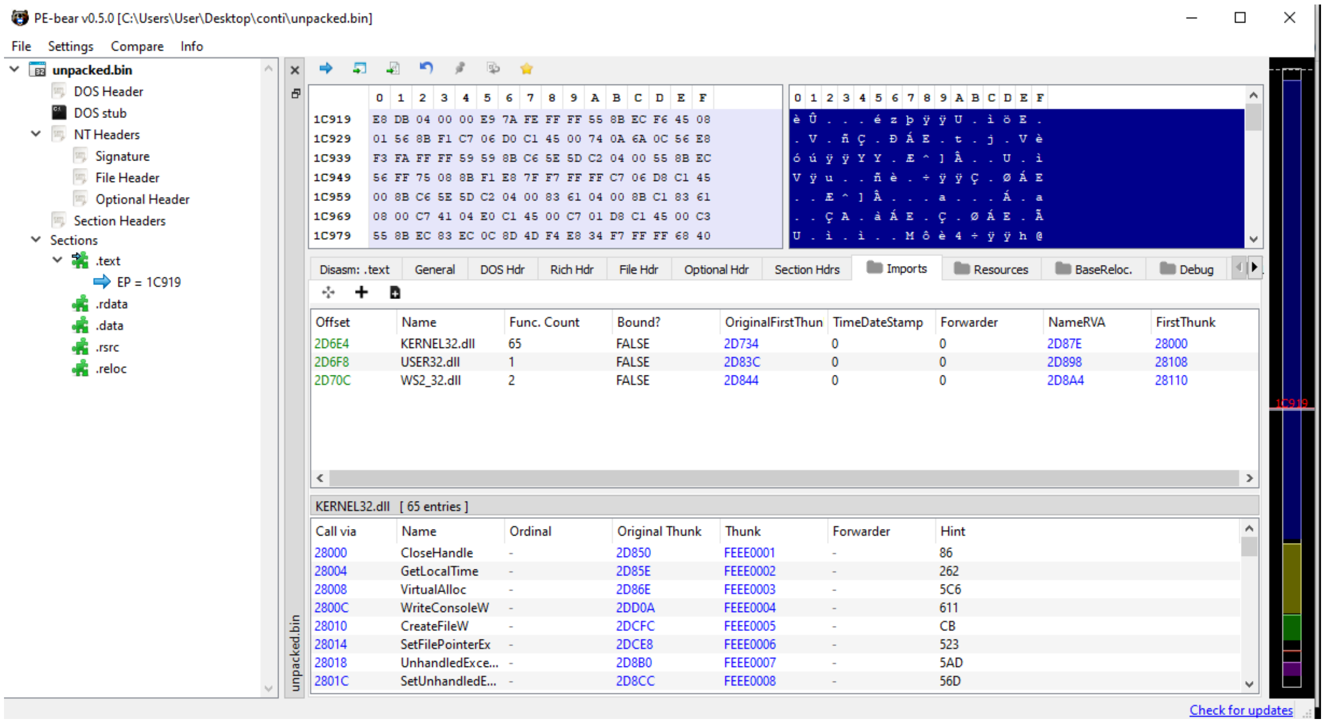
Running with Command Prompt

```
python ContiUnpacker.py -f conti.dll -o <output_file>
```

Image



```
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetCPInfo")' -> 0xfeee0030
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetCommandLineA")' -> 0xfeee0031
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetCommandLineW")' -> 0xfeee0032
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "MultiByteToWideChar")' -> 0xfeee0033
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "WideCharToMultiByte")' -> 0xfeee0034
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetEnvironmentStringsW")' -> 0xfeee0035
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "FreeEnvironmentStringsW")' -> 0xfeee0036
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetFileType")' -> 0xfeee0037
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "LCMapStringW")' -> 0xfeee0038
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetProcessHeap")' -> 0xfeee0039
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "SetStdHandle")' -> 0xfeee003a
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetStringTypeW")' -> 0xfeee003b
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "HeapSize")' -> 0xfeee003c
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "HeapReAlloc")' -> 0xfeee003d
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "FlushFileBuffers")' -> 0xfeee003e
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetConsoleCP")' -> 0xfeee003f
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "GetConsoleMode")' -> 0xfeee0040
0x10001881: 'KERNEL32.GetProcAddress(0x77000000, "DecodePointer")' -> 0xfeee0041
0x100021ff: 'KERNEL32.IsBadReadPtr(0x4616f8, 0x14)' -> 0x0
0x1000184e: 'KERNEL32.LoadLibraryA("USER32.dll")' -> 0x77d10000
0x10001920: 'KERNEL32.VirtualQuery(0x50000, 0x1211ecc, 0x1c)' -> 0x1c
0x10001881: 'KERNEL32.GetProcAddress(0x77d10000, "wsprintfW")' -> 0xfeee0042
0x100021ff: 'KERNEL32.IsBadReadPtr(0x46170c, 0x14)' -> 0x0
0x1000184e: 'KERNEL32.LoadLibraryA("WS2_32.dll")' -> 0x78c00000
0x10001920: 'KERNEL32.VirtualQuery(0x50000, 0x1211ecc, 0x1c)' -> 0x1c
0x10001881: 'KERNEL32.GetProcAddress(0x78c00000, 0x6f)' -> 0xfeee0043
0x10001881: 'KERNEL32.GetProcAddress(0x78c00000, 0x9)' -> 0xfeee0044
0x100021ff: 'KERNEL32.IsBadReadPtr(0x461720, 0x14)' -> 0x0
[*] VirtualProtect CALLED!
[*] Dump Address: 0x434000
[*] Dump Address: 0x34000
[*] Found valid PE file
0x10001e57: 'KERNEL32.VirtualProtect(0x435000, 0x32200, 0x40, 0x1211ed8)' -> None
```


PE-bear v0.5.0 [C:\Users\User\Desktop\conti\unpacked.bin]
File Settings Compare Info
unpacked.bin
DOS Header
DOS stub
NT Headers
Signature
File Header
Optional Header
Section Headers
Sections
.text
EP = 1C919
.rdata
.data
.rsrc
.reloc
Disasm: .text General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Imports Resources BaseReloc. Debug
unpacked.bin
Check for updates

```
         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F     0 1 2 3 4 5 6 7 8 9 A B C D E F
1C919    E8 DB 04 00 00 E9 7A FE FF FF 55 8B EC F6 45 08    è Û . . . é z þ ÿ ÿ U . ì ö E .
1C929    01 56 8B F1 C7 06 D0 C1 45 00 74 0A 6A 0C 56 E8    . V . ñ Ç . Ð Á E . t . j . V è
1C939    F3 FA FF FF 59 59 8B C6 5E 5D C2 04 00 55 8B EC    ó ú ÿ ÿ Y Y . Æ ^ ] Â . . U . ì
1C949    56 FF 75 08 8B F1 E8 7F F7 FF FF C7 06 D8 C1 45    V ÿ u . . ñ è . ÷ ÿ ÿ Ç . Ø Á E
1C959    00 8B C6 5E 5D C2 04 00 83 61 04 00 8B C1 83 61    . . Æ ^ ] Â . . . a . . . Á . a
1C969    08 00 C7 41 04 E0 C1 45 00 C7 01 D8 C1 45 00 C3    . . Ç A . à Á E . Ç . Ø Á E . Ã
1C979    55 8B EC 83 EC 0C 8D 4D F4 E8 34 F7 FF FF 68 40    U . ì . ì . . M ô è 4 ÷ ÿ ÿ h @
```

Offset	Name	Func. Count	Bound?	OriginalFirstThun	TimeDateStamp	Forwarder	NameRVA	FirstThunk
2D6E4	KERNEL32.dll	65	FALSE	2D734	0	0	2D87E	28000
2D6F8	USER32.dll	1	FALSE	2D83C	0	0	2D898	28108
2D70C	WS2_32.dll	2	FALSE	2D844	0	0	2D8A4	28110

KERNEL32.dll [65 entries]

Call via	Name	Ordinal	Original Thunk	Thunk	Forwarder	Hint
28000	CloseHandle	-	2D850	FEEE0001	-	86
28004	GetLocalTime	-	2D85E	FEEE0002	-	262
28008	VirtualAlloc	-	2D86E	FEEE0003	-	5C6
2800C	WriteConsoleW	-	2DD0A	FEEE0004	-	611
28010	CreateFileW	-	2DCFC	FEEE0005	-	CB
28014	SetFilePointerEx	-	2DCE8	FEEE0006	-	523
28018	UnhandledExce...	-	2D8B0	FEEE0007	-	5AD
2801C	SetUnhandledE...	-	2D8CC	FEEE0008	-	56D

Note

Please don't actually run this malware I included unless you know what you're doing. I'm not responsible if you end up encrypting your machine!

Also, I noticed that the function calls are a bit different on Speakeasy emulator compared to when running on x64dbg. During the VirtualProtect call, everything should technically be written into the allocated memory already, but that's not the case...

Apparently, only parts of the **.rdata** section is written, so the dumped executable won't be able to run.

I can't figure out why this is happening because Speakeasy is pretty weird, so this unpacker does not work 100%.

However, I'll still keep it here in case anyone wants to refer to this when writing their own unpacker using Speakeasy!

Acknowledgement

James T. Bennett - https://www.fireeye.com/blog/threat-research/2020/12/using-speakeasy-emulation-framework-programmatically-to-unpack-malware.html

FireEye's Speakeasy Emulation Framework - https://github.com/fireeye/speakeasy