2022

Malware Analysis Report

DIGITAL FORENSICS PROJECT

ABDULLAH IRFAN | AISHA IRFAN | MUHAMMAD HUZAIFA

SUBMITTED TO: | Sir Zeeshan Qaiser

Table of Contents

1- Case Summary

This case was assigned to us to combine all we learnt in our Digital Forensics course and apply as much of it as we can. We will be analyzing a few malware samples following the proper procedure and test how much information we can extract from it.

2- Examiners and their Qualifications

Abdullah Irfan	Digital Forensics Student
	Malware Analyst
	Top 7% on TryHackMe
Aisha Irfan	Digital Forensics Student
	Reverse Engineering Expert
	Top 5% on TryHackMe
Muhammad Huzaifa	Digital Forensics Student
	Vulnerability Assessment Engineer
	Top 6% on TryHackMe

3- Evidence

- 88c5be944437cc07361723a1745e9e643d54c8579b9e75d1e491a0746f689b01.zip
- 882a04265361d588801b3514a604182ce9b8271dd500728fa2897524a2f05a7e.zip
- dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000.zip

The samples listed above were taken from “dasmalwerk.eu”, a malware repository.

- pacman.exe
- uninstall.exe

These samples are self-made. Our team created a ransomware to prove our expertise and qualification in the field of malware analysis (by analyzing and reverse engineering it).

4- Objectives

We have to analysis the evidence files and list the indicators of compromise (IOCs) that flag these files as malicious. We also have to identify the general type of the malware found in each of these files. We have to perform static analysis to first get a general idea on how the samples work and what they supposedly do on the target system. For a good understanding of that, we have to perform advanced analysis by using reverse engineering techniques to get familiar with the layout of the code. After that, we have to perform a dynamic analysis to see how the malware actually behaves when it is run and analyze our findings.

5- Environment

5a- Windows-based FLARE Sandbox

- **peid:** a tool to detect any packing/obfuscation of a portable executable
- **pestudio:** a tool that gives you all metadata, libraries and strings in an executable
- **Detect It Easy!**: a more user-friendly pestudio
- **Ghidra:** a disassembler, decompiler and debugger for portable executables
- **Process Explorer:** a tool used to snoop on processes and application and record them
- **Process Hacker:** a tool used to identify the extremity of processes
- **ProcDot:** a tool used to generate a flow graph of a process

5b- Kali-based Sandbox

- **Pyinstxtractor:** a tool to unpack executables made by pyinstaller
- **Pydcd (Decompiler++):** a decompiler for unpacked python executables

6- Forensic Analysis

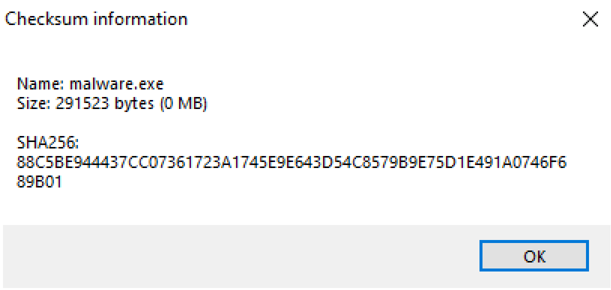
6a- Chain of Custody

The chain of custody does not apply here as such because it is an internal case and the evidence used is taken from an open-source malware repository. However, we have taken steps to ensure that our original samples are unaltered, in case the repository is down. These steps will be mentioned further along the report.

6b- Acquisition

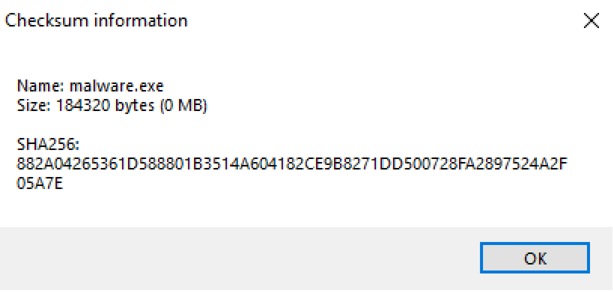
We acquired 3 of the samples from “dasmalwerk.eu” and made one sample ourselves. We have taken the hashes of the cloned samples to verify their integrity. These hashes are obtained from the copies of the original samples as we do not want our original data to lose its integrity.

Sample 1



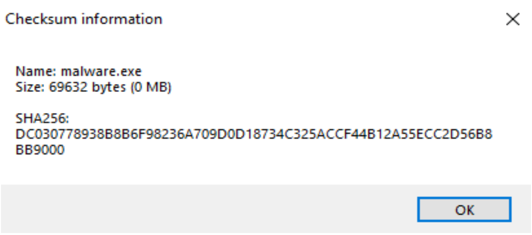
```
Name: malware.exe
Size: 291523 bytes (0 MB)

SHA256:
88C5BE944437CC07361723A1745E9E643D54C8579B9E75D1E491A0746F6
89B01
```

Sample 2



```
Name: malware.exe
Size: 184320 bytes (0 MB)

SHA256:
882A04265361D588801B3514A604182CE9B8271DD500728FA2897524A2F
05A7E
```

Sample 3

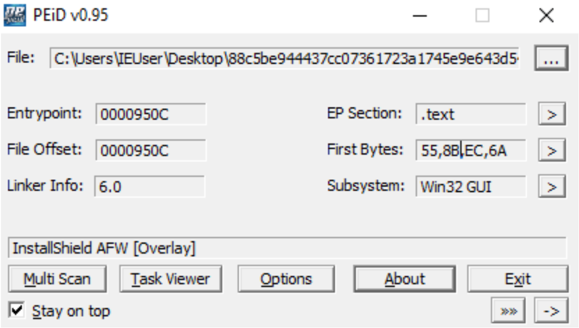


```
Name: malware.exe
Size: 69632 bytes (0 MB)

SHA256:
DC030778938B8B6F98236A709D0D18734C325ACCF44B12A55ECC2D56B8
BB9000
```

6c- Analysis

Static Analysis

We used peid to first detect if there was any obfuscation present in the samples.



File: C:\Users\IEUser\Desktop\88c5be944437cc07361723a1745e9e643d5

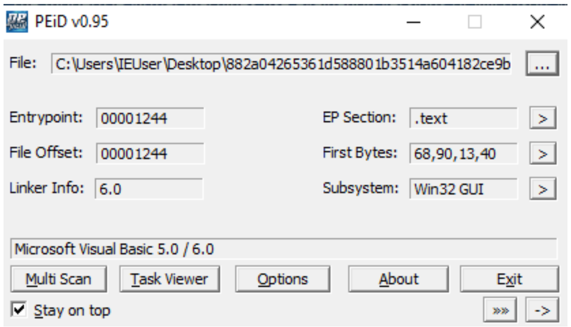
Entrypoint: 0000950C EP Section: .text

File Offset: 0000950C First Bytes: 55,8B,EC,6A

Linker Info: 6.0 Subsystem: Win32 GUI

InstallShield AFW [Overlay]

☑ Stay on top



File: C:\Users\IEUser\Desktop\882a04265361d588801b3514a604182ce9b

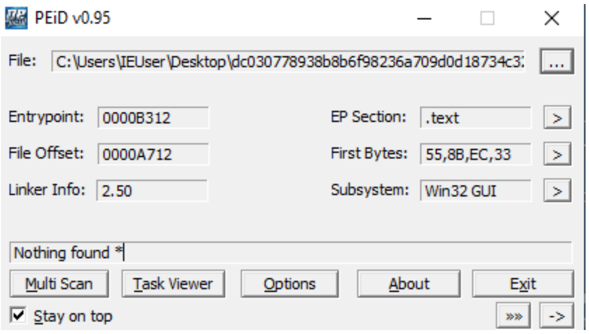
Entrypoint: 00001244 EP Section: .text

File Offset: 00001244 First Bytes: 68,90,13,40

Linker Info: 6.0 Subsystem: Win32 GUI

Microsoft Visual Basic 5.0 / 6.0

☑ Stay on top



File: C:\Users\IEUser\Desktop\dc030778938b8b6f98236a709d0d18734c3

Entrypoint: 0000B312 EP Section: .text

File Offset: 0000A712 First Bytes: 55,8B,EC,33

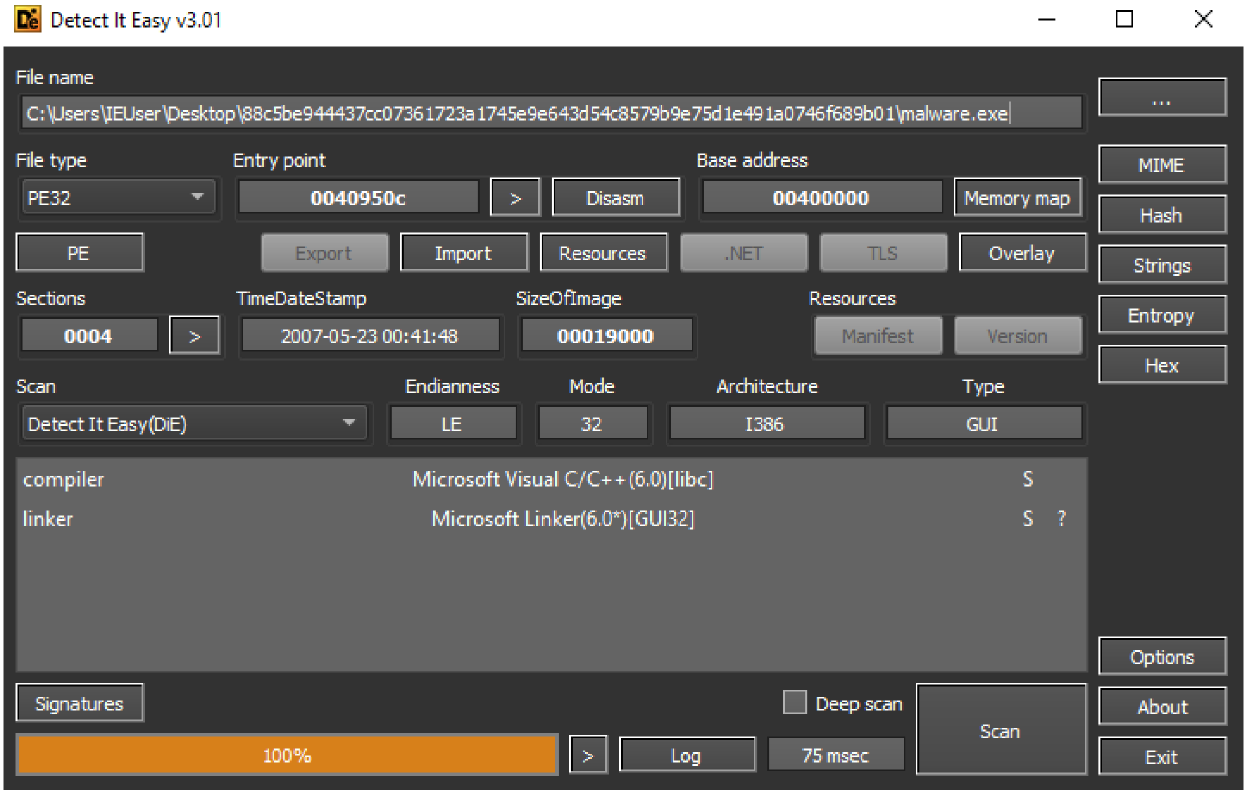
Linker Info: 2.50 Subsystem: Win32 GUI

Nothing found *

☑ Stay on top

Then we used Detect it Easy! And pestudio to check what system libraries and functions the samples used.

Sample 1



File name: C:\Users\IEUser\Desktop\88c5be944437cc07361723a1745e9e643d54c8579b9e75d1e491a0746f689b01\malware.exe

File type: PE32

Entry point: 0040950c

Base address: 00400000

Sections: 0004

TimeDateStamp: 2007-05-23 00:41:48

SizeOfImage: 00019000

Resources

Scan: Detect It Easy(DiE)

Endianness: LE

Mode: 32

Architecture: I386

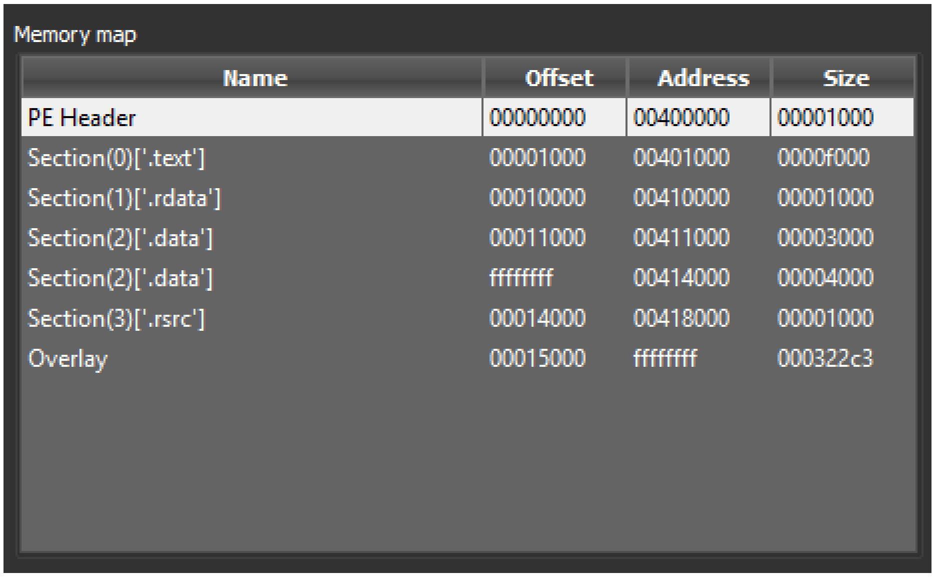
Type: GUI

```
compiler        Microsoft Visual C/C++(6.0)[libc]        S
linker          Microsoft Linker(6.0*)[GUI32]            S  ?
```

□ Deep scan

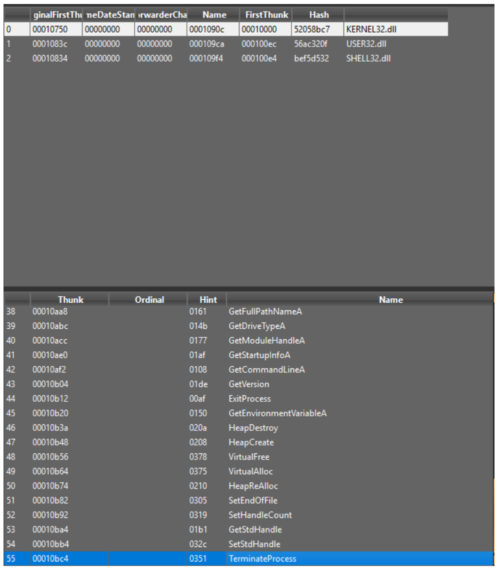
100%

75 msec



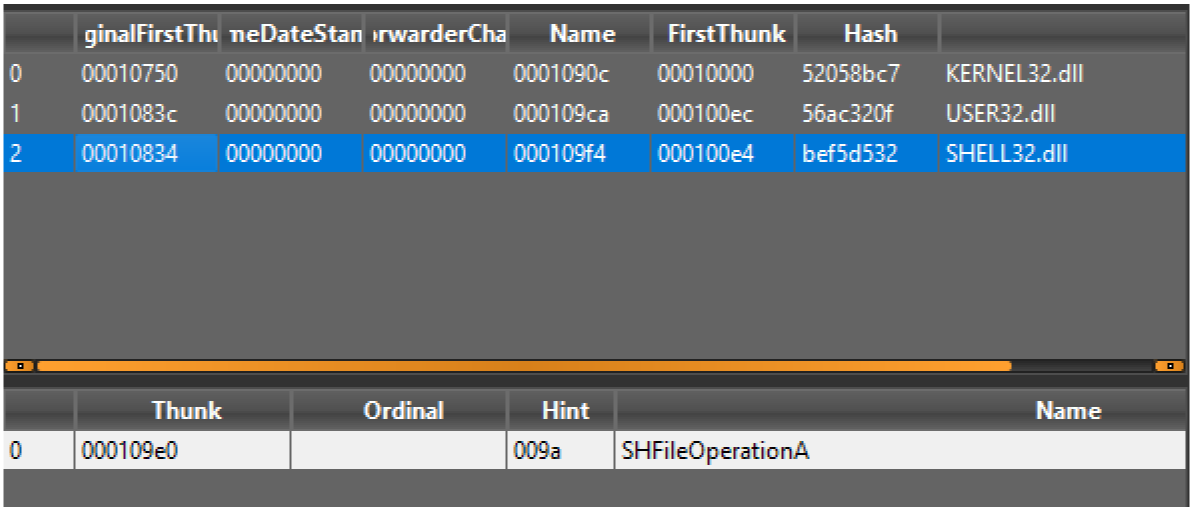
Memory map

Name	Offset	Address	Size
PE Header	00000000	00400000	00001000
Section(0)['.text']	00001000	00401000	0000f000
Section(1)['.rdata']	00010000	00410000	00001000
Section(2)['.data']	00011000	00411000	00003000
Section(2)['.data']	ffffffff	00414000	00004000
Section(3)['.rsrc']	00014000	00418000	00001000
Overlay	00015000	ffffffff	000322c3



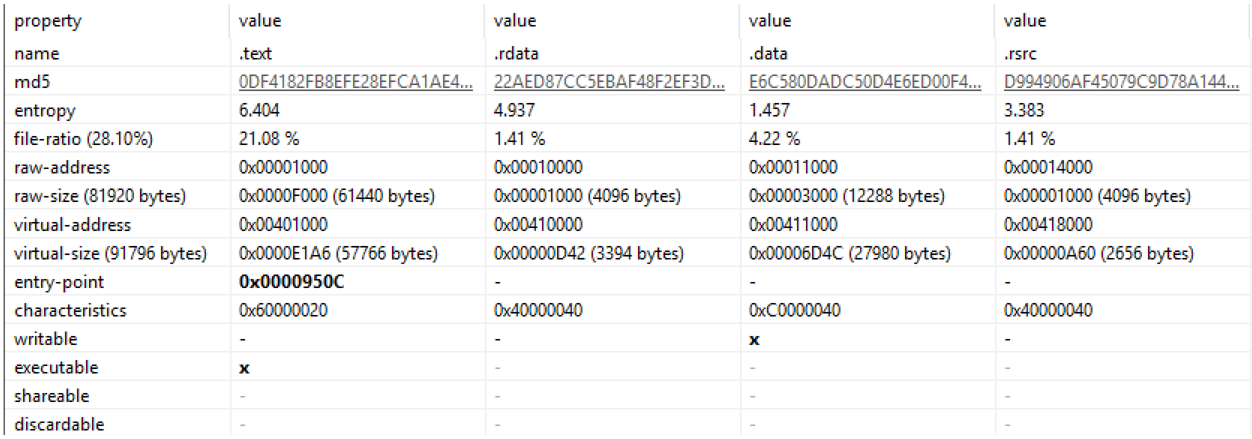
	ginalFirstThu	neDateStan	rwarderCha	Name	FirstThunk	Hash	
0	00010750	00000000	00000000	0001090c	00010000	52058bc7	KERNEL32.dll
1	0001083c	00000000	00000000	000109ca	000100ec	56ac320f	USER32.dll
2	00010834	00000000	00000000	000109f4	000100e4	bef5d532	SHELL32.dll

	Thunk	Ordinal	Hint	Name
38	00010aa8		0161	GetFullPathNameA
39	00010abc		014b	GetDriveTypeA
40	00010acc		0177	GetModuleHandleA
41	00010ae0		01af	GetStartupInfoA
42	00010af2		0108	GetCommandLineA
43	00010b04		01de	GetVersion
44	00010b12		00af	ExitProcess
45	00010b20		0150	GetEnvironmentVariableA
46	00010b3a		020a	HeapDestroy
47	00010b48		0208	HeapCreate
48	00010b56		0378	VirtualFree
49	00010b64		0375	VirtualAlloc
50	00010b74		0210	HeapReAlloc
51	00010b82		0305	SetEndOfFile
52	00010b92		0319	SetHandleCount
53	00010ba4		01b1	GetStdHandle
54	00010bb4		032c	SetStdHandle
55	00010bc4		0351	TerminateProcess



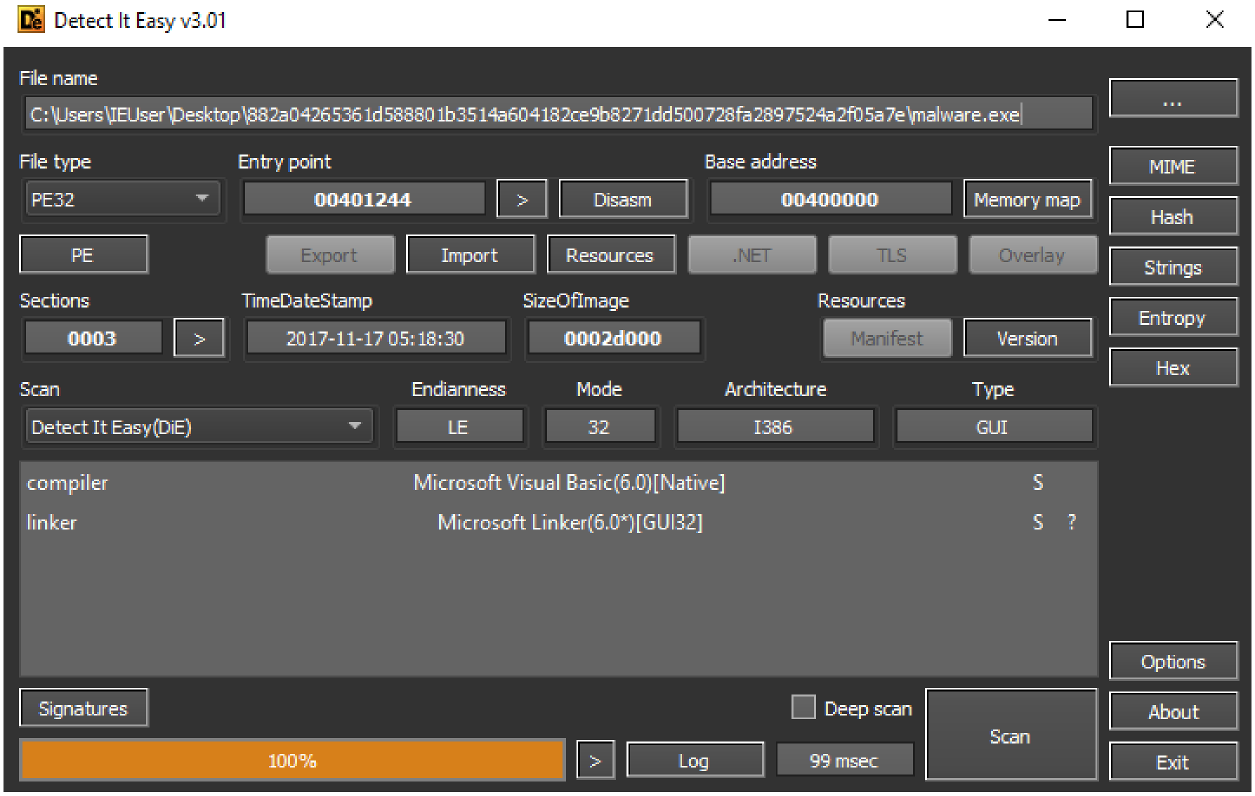
	ginalFirstThu	neDateStan	rwarderCha	Name	FirstThunk	Hash	
0	00010750	00000000	00000000	0001090c	00010000	52058bc7	KERNEL32.dll
1	0001083c	00000000	00000000	000109ca	000100ec	56ac320f	USER32.dll
2	00010834	00000000	00000000	000109f4	000100e4	bef5d532	SHELL32.dll

	Thunk	Ordinal	Hint	Name
0	000109e0		009a	SHFileOperationA



property	value	value	value	value
name	.text	.rdata	.data	.rsrc
md5	0DF4182FB8EFE28EFCA1AE4...	22AED87CC5EBAF48F2EF3D...	E6C580DADC50D4E6ED00F4...	D994906AF45079C9D78A144...
entropy	6.404	4.937	1.457	3.383
file-ratio (28.10%)	21.08 %	1.41 %	4.22 %	1.41 %
raw-address	0x00001000	0x00010000	0x00011000	0x00014000
raw-size (81920 bytes)	0x0000F000 (61440 bytes)	0x00001000 (4096 bytes)	0x00003000 (12288 bytes)	0x00001000 (4096 bytes)
virtual-address	0x00401000	0x00410000	0x00411000	0x00418000
virtual-size (91796 bytes)	0x0000E1A6 (57766 bytes)	0x00000D42 (3394 bytes)	0x00006D4C (27980 bytes)	0x00000A60 (2656 bytes)
entry-point	**0x0000950C**	-	-	-
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040
writable	-	-	**x**	-
executable	**x**	-	-	-
shareable	-	-	-	-
discardable	-	-	-	-

Sample 2

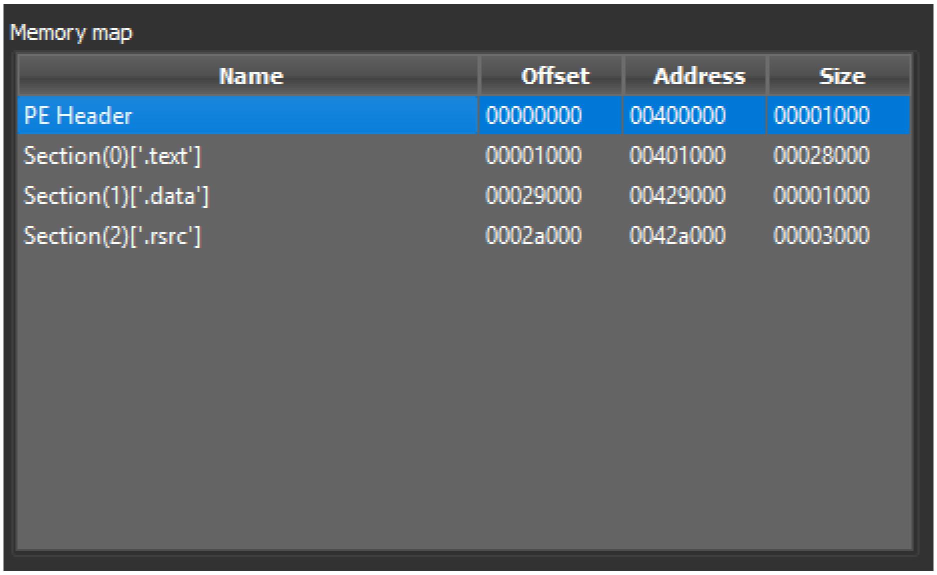


File name
C:\Users\IEUser\Desktop\882a04265361d588801b3514a604182ce9b8271dd500728fa2897524a2f05a7e\malware.exe

File type: PE32
Entry point: 00401244
Base address: 00400000
Sections: 0003
TimeDateStamp: 2017-11-17 05:18:30
SizeOfImage: 0002d000
Resources
Scan: Detect It Easy(DiE)
Endianness: LE
Mode: 32
Architecture: I386
Type: GUI

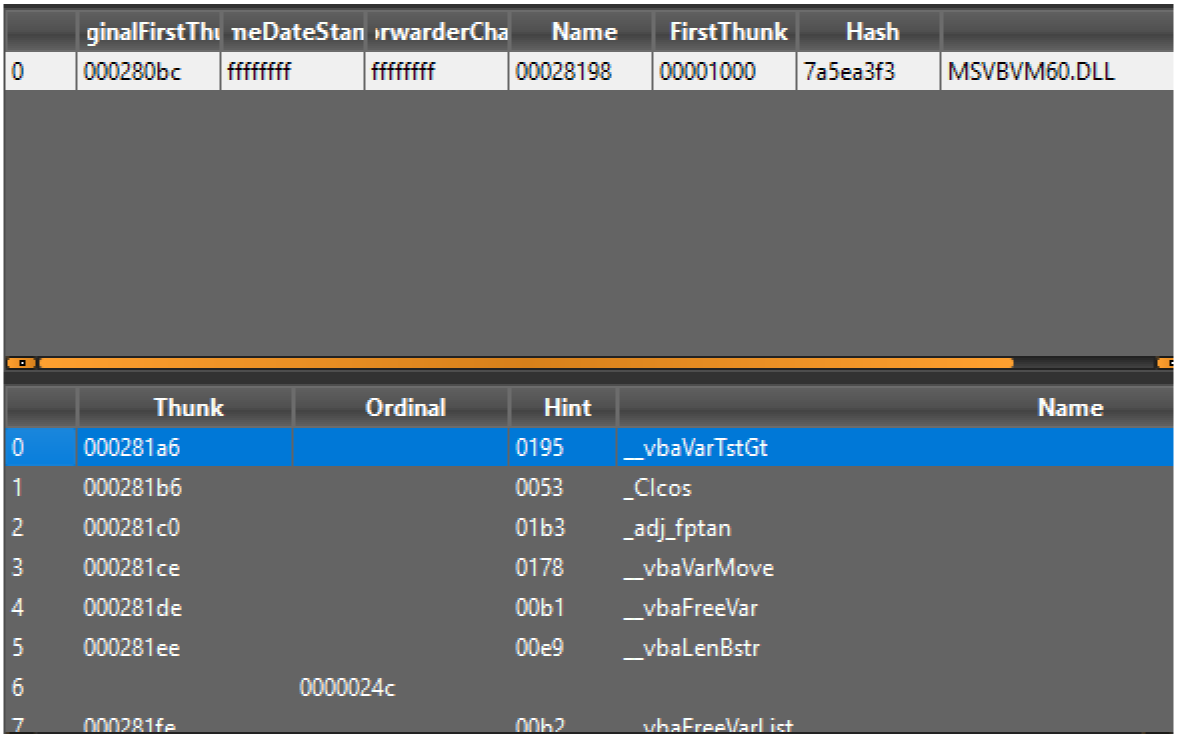
```
compiler    Microsoft Visual Basic(6.0)[Native]    S
linker      Microsoft Linker(6.0*)[GUI32]          S  ?
```

100%
99 msec



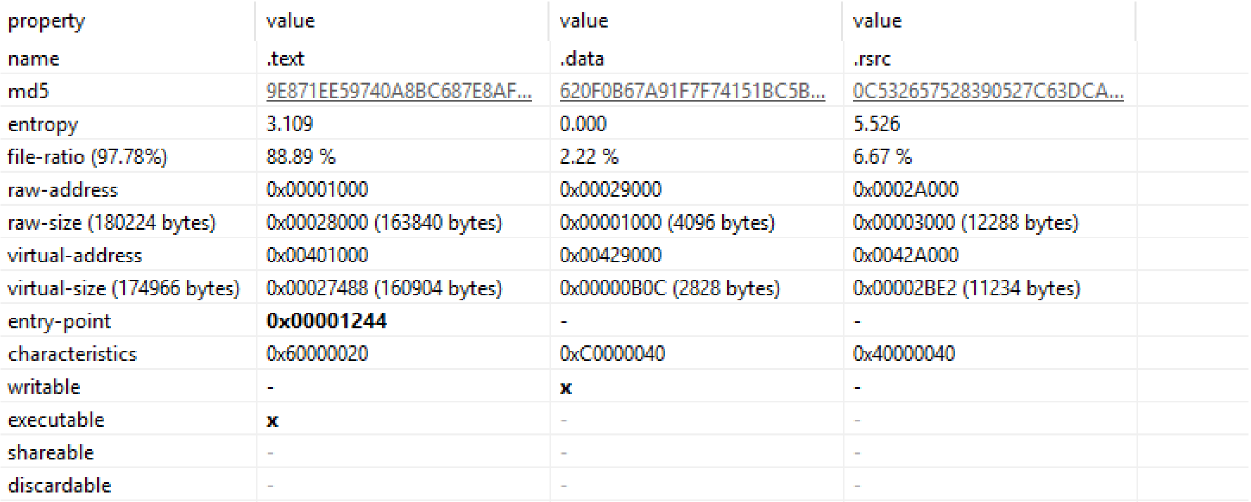
Memory map

Name	Offset	Address	Size
PE Header	00000000	00400000	00001000
Section(0)['.text']	00001000	00401000	00028000
Section(1)['.data']	00029000	00429000	00001000
Section(2)['.rsrc']	0002a000	0042a000	00003000



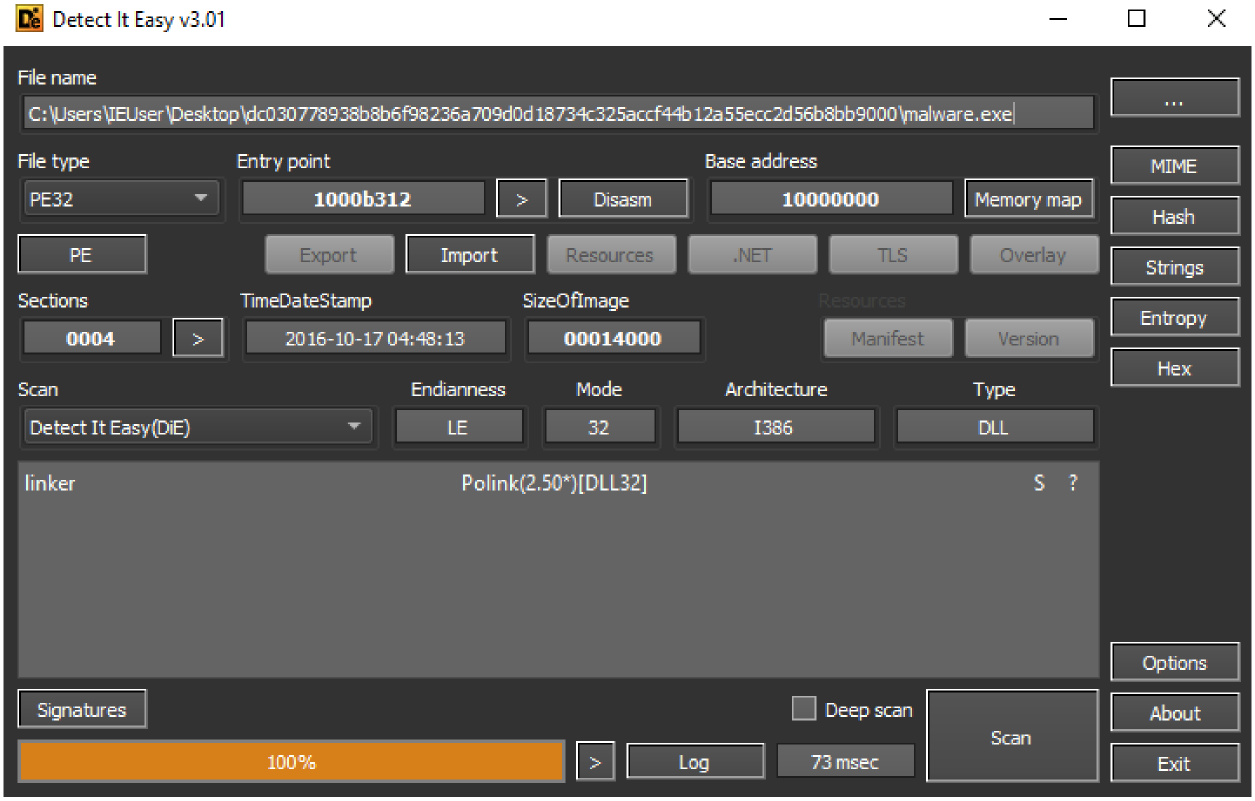
	ginalFirstThu	neDateStan	rwarderCha	Name	FirstThunk	Hash	
0	000280bc	ffffffff	ffffffff	00028198	00001000	7a5ea3f3	MSVBVM60.DLL

	Thunk	Ordinal	Hint	Name
0	000281a6		0195	__vbaVarTstGt
1	000281b6		0053	_CIcos
2	000281c0		01b3	_adj_fptan
3	000281ce		0178	__vbaVarMove
4	000281de		00b1	__vbaFreeVar
5	000281ee		00e9	__vbaLenBstr
6		0000024c		
7	000281fe		00b2	__vbaFreeVarList



property	value	value	value
name	.text	.data	.rsrc
md5	9E871EE59740A8BC687E8AF...	620F0B67A91F7F74151BC5B...	0C532657528390527C63DCA...
entropy	3.109	0.000	5.526
file-ratio (97.78%)	88.89 %	2.22 %	6.67 %
raw-address	0x00001000	0x00029000	0x0002A000
raw-size (180224 bytes)	0x00028000 (163840 bytes)	0x00001000 (4096 bytes)	0x00003000 (12288 bytes)
virtual-address	0x00401000	0x00429000	0x0042A000
virtual-size (174966 bytes)	0x00027488 (160904 bytes)	0x00000B0C (2828 bytes)	0x00002BE2 (11234 bytes)
entry-point	**0x00001244**	-	-
characteristics	0x60000020	0xC0000040	0x40000040
writable	-	**x**	-
executable	**x**	-	-
shareable	-	-	-
discardable	-	-	-

Sample 3



File name: C:\Users\IEUser\Desktop\dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000\malware.exe

File type: PE32

Entry point: 1000b312

Base address: 10000000

Sections: 0004

TimeDateStamp: 2016-10-17 04:48:13

SizeOfImage: 00014000

Scan: Detect It Easy(DiE)

Endianness: LE

Mode: 32

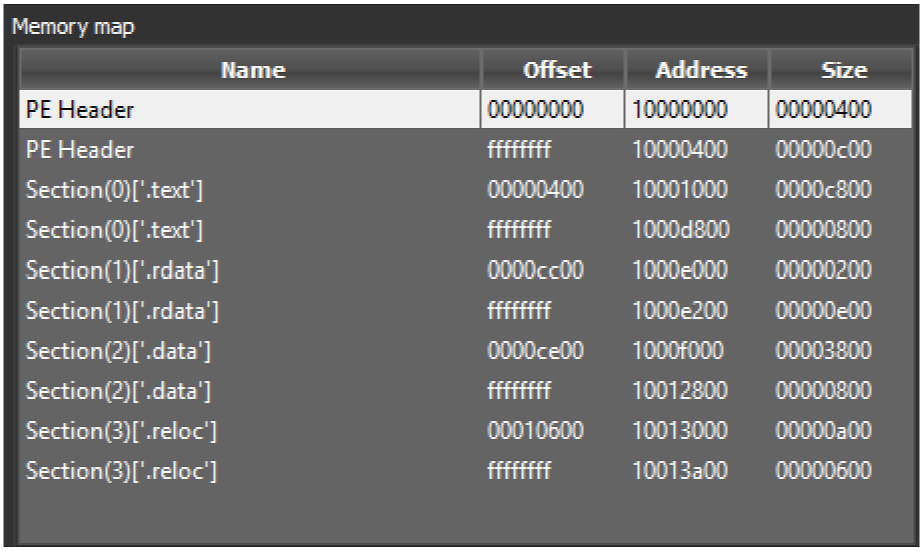
Architecture: I386

Type: DLL

linker Polink(2.50*)[DLL32]

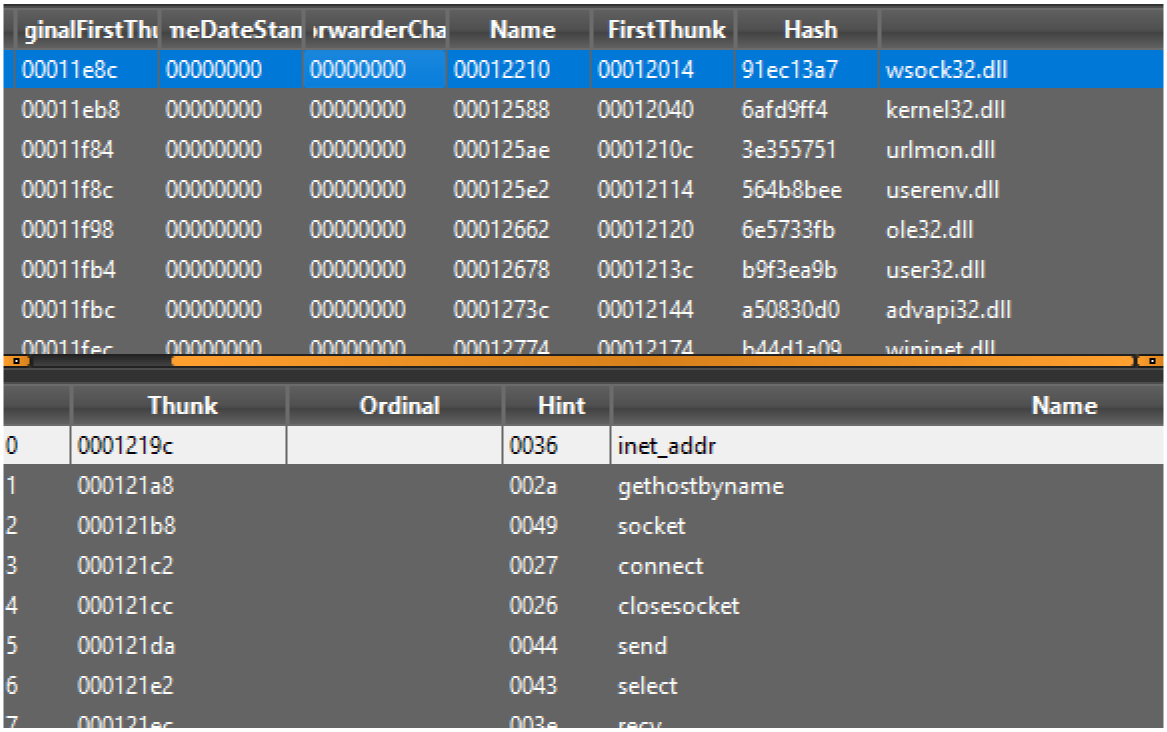
100%

73 msec



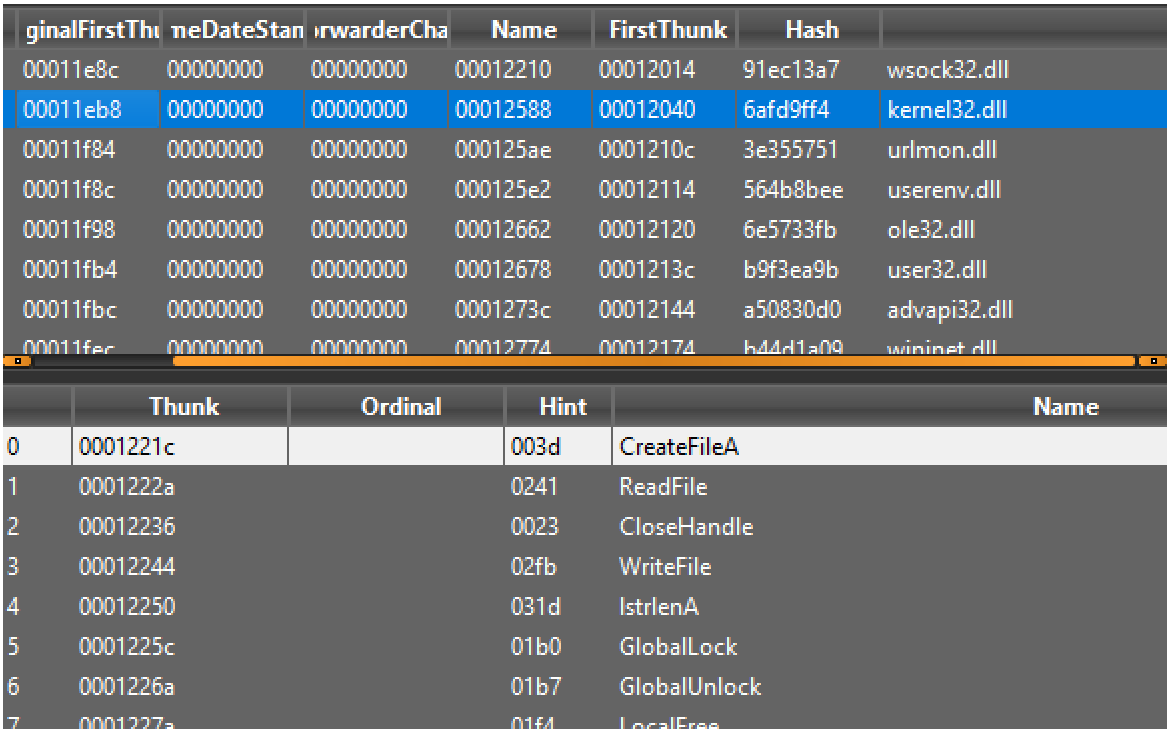
Memory map

Name	Offset	Address	Size
PE Header	00000000	10000000	00000400
PE Header	ffffffff	10000400	00000c00
Section(0)['.text']	00000400	10001000	0000c800
Section(0)['.text']	ffffffff	1000d800	00000800
Section(1)['.rdata']	0000cc00	1000e000	00000200
Section(1)['.rdata']	ffffffff	1000e200	00000e00
Section(2)['.data']	0000ce00	1000f000	00003800
Section(2)['.data']	ffffffff	10012800	00000800
Section(3)['.reloc']	00010600	10013000	00000a00
Section(3)['.reloc']	ffffffff	10013a00	00000600



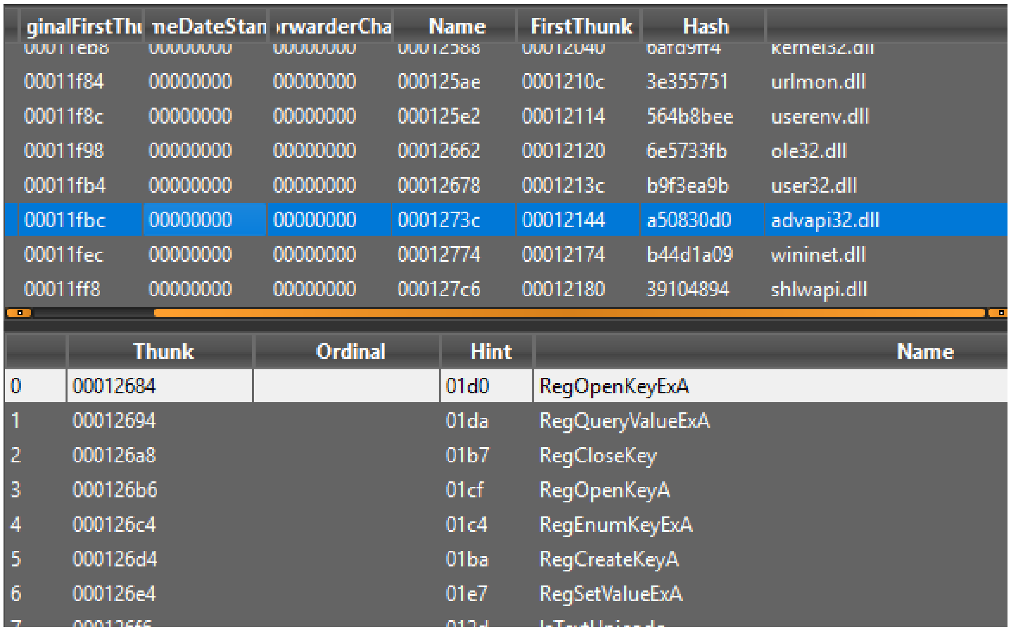
ginalFirstThu	neDateStan	rwarderCha	Name	FirstThunk	Hash	
00011e8c	00000000	00000000	00012210	00012014	91ec13a7	wsock32.dll
00011eb8	00000000	00000000	00012588	00012040	6afd9ff4	kernel32.dll
00011f84	00000000	00000000	000125ae	0001210c	3e355751	urlmon.dll
00011f8c	00000000	00000000	000125e2	00012114	564b8bee	userenv.dll
00011f98	00000000	00000000	00012662	00012120	6e5733fb	ole32.dll
00011fb4	00000000	00000000	00012678	0001213c	b9f3ea9b	user32.dll
00011fbc	00000000	00000000	0001273c	00012144	a50830d0	advapi32.dll
00011fec	00000000	00000000	00012774	00012174	b44d1a09	wininet.dll

	Thunk	Ordinal	Hint	Name
0	0001219c		0036	inet_addr
1	000121a8		002a	gethostbyname
2	000121b8		0049	socket
3	000121c2		0027	connect
4	000121cc		0026	closesocket
5	000121da		0044	send
6	000121e2		0043	select
7	000121ec		003e	recv



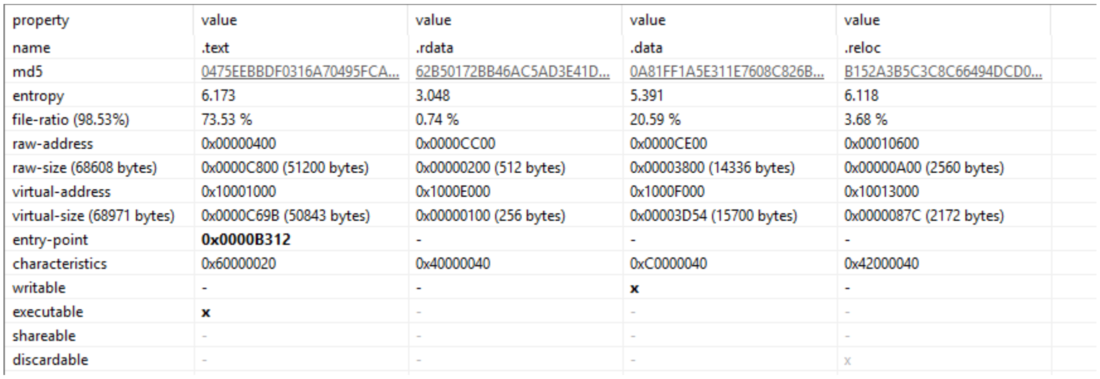
ginalFirstThu	neDateStan	rwarderCha	Name	FirstThunk	Hash	
00011e8c	00000000	00000000	00012210	00012014	91ec13a7	wsock32.dll
00011eb8	00000000	00000000	00012588	00012040	6afd9ff4	kernel32.dll
00011f84	00000000	00000000	000125ae	0001210c	3e355751	urlmon.dll
00011f8c	00000000	00000000	000125e2	00012114	564b8bee	userenv.dll
00011f98	00000000	00000000	00012662	00012120	6e5733fb	ole32.dll
00011fb4	00000000	00000000	00012678	0001213c	b9f3ea9b	user32.dll
00011fbc	00000000	00000000	0001273c	00012144	a50830d0	advapi32.dll
00011fec	00000000	00000000	00012774	00012174	b44d1a09	wininet.dll

	Thunk	Ordinal	Hint	Name
0	0001221c		003d	CreateFileA
1	0001222a		0241	ReadFile
2	00012236		0023	CloseHandle
3	00012244		02fb	WriteFile
4	00012250		031d	lstrlenA
5	0001225c		01b0	GlobalLock
6	0001226a		01b7	GlobalUnlock
7	0001227a		01f4	LocalFree



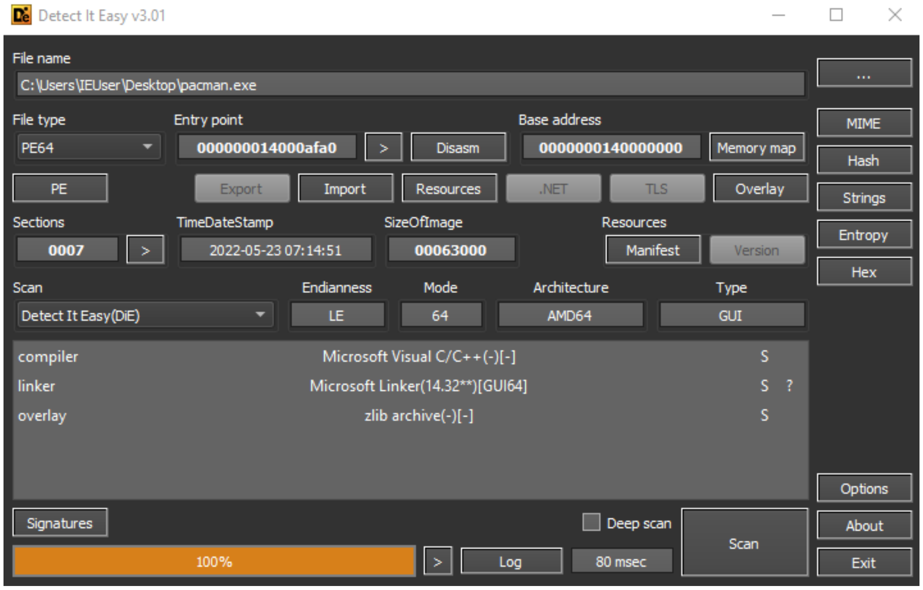
ginalFirstThu	meDateStan	rwarderCha	Name	FirstThunk	Hash	
00011eb8	00000000	00000000	00012588	00012040	6afd9ff4	kernel32.dll
00011f84	00000000	00000000	000125ae	0001210c	3e355751	urlmon.dll
00011f8c	00000000	00000000	000125e2	00012114	564b8bee	userenv.dll
00011f98	00000000	00000000	00012662	00012120	6e5733fb	ole32.dll
00011fb4	00000000	00000000	00012678	0001213c	b9f3ea9b	user32.dll
00011fbc	00000000	00000000	0001273c	00012144	a50830d0	advapi32.dll
00011fec	00000000	00000000	00012774	00012174	b44d1a09	wininet.dll
00011ff8	00000000	00000000	000127c6	00012180	39104894	shlwapi.dll

	Thunk	Ordinal	Hint	Name
0	00012684		01d0	RegOpenKeyExA
1	00012694		01da	RegQueryValueExA
2	000126a8		01b7	RegCloseKey
3	000126b6		01cf	RegOpenKeyA
4	000126c4		01c4	RegEnumKeyExA
5	000126d4		01ba	RegCreateKeyA
6	000126e4		01e7	RegSetValueExA
7	000126f6		012d	IsTextUnicode



property	value	value	value	value
name	.text	.rdata	.data	.reloc
md5	0475EEBBDF0316A70495FCA...	62B50172BB46AC5AD3E41D...	0A81FF1A5E311E7608C826B...	B152A3B5C3C8C66494DCD0...
entropy	6.173	3.048	5.391	6.118
file-ratio (98.53%)	73.53 %	0.74 %	20.59 %	3.68 %
raw-address	0x00000400	0x0000CC00	0x0000CE00	0x00010600
raw-size (68608 bytes)	0x0000C800 (51200 bytes)	0x00000200 (512 bytes)	0x00003800 (14336 bytes)	0x00000A00 (2560 bytes)
virtual-address	0x10001000	0x1000E000	0x1000F000	0x10013000
virtual-size (68971 bytes)	0x0000C69B (50843 bytes)	0x00000100 (256 bytes)	0x00003D54 (15700 bytes)	0x0000087C (2172 bytes)
entry-point	**0x0000B312**	-	-	-
characteristics	0x60000020	0x40000040	0xC0000040	0x42000040
writable	-	-	**x**	-
executable	**x**	-	-	-
shareable	-	-	-	-
discardable	-	-	-	x

Sample 4



File name
C:\Users\IEUser\Desktop\pacman.exe

File type	Entry point	Base address
PE64	000000014000afa0	0000000140000000

Sections	TimeDateStamp	SizeOfImage
0007	2022-05-23 07:14:51	00063000

Scan	Endianness	Mode	Architecture	Type
Detect It Easy(DiE)	LE	64	AMD64	GUI

compiler	Microsoft Visual C/C++(-)[-]	S	
linker	Microsoft Linker(14.32**)[GUI64]	S	?
overlay	zlib archive(-)[-]	S	

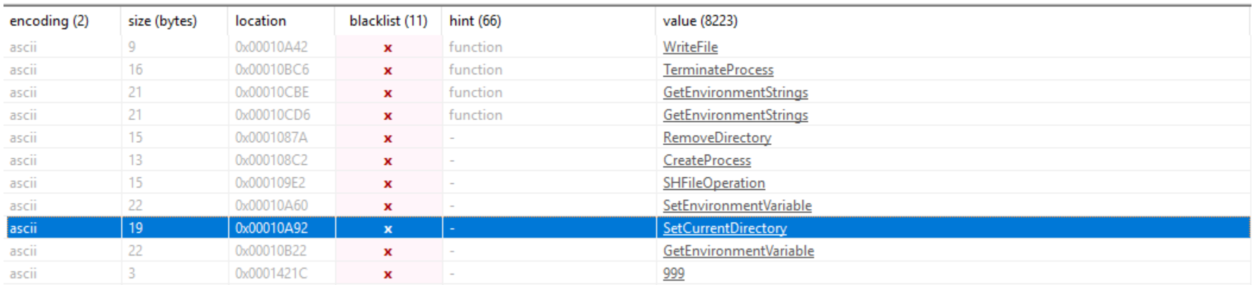
☐ Deep scan

100%

80 msec

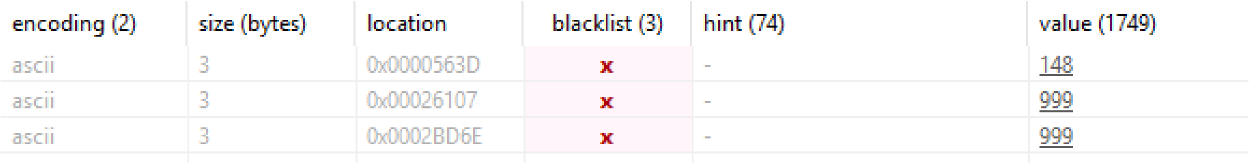
We then did a string analysis on all the samples using pestudio.

Sample 1



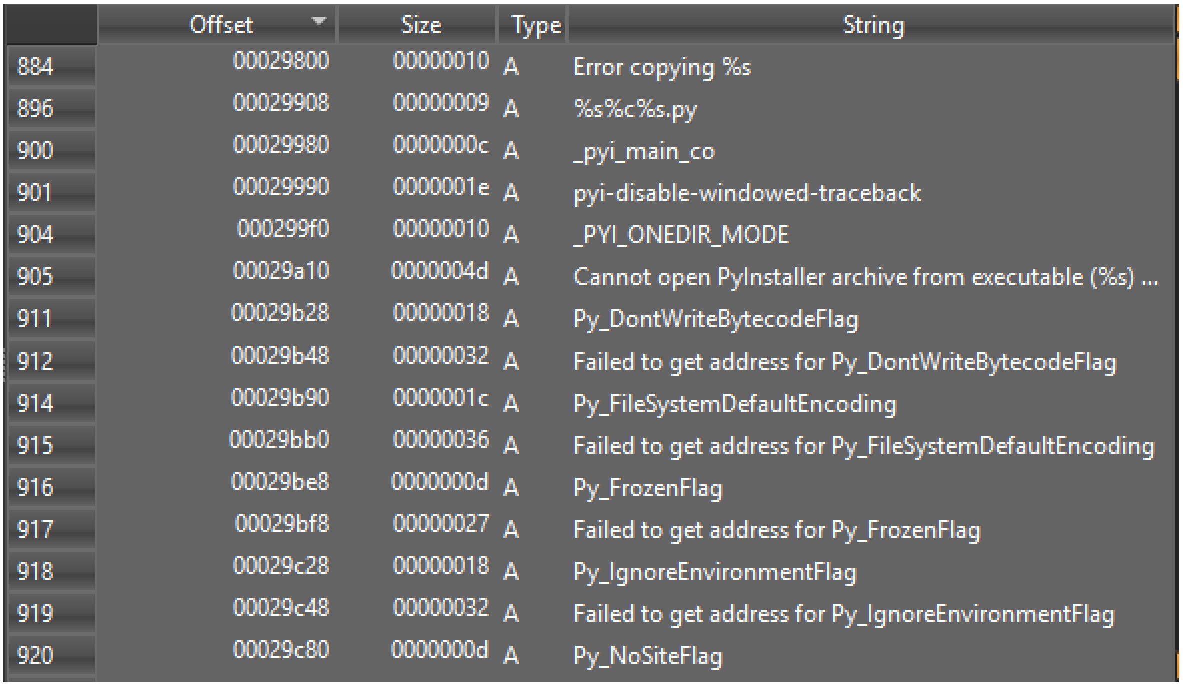
encoding (2)	size (bytes)	location	blacklist (11)	hint (66)	value (8223)
ascii	9	0x00010A42	x	function	WriteFile
ascii	16	0x00010BC6	x	function	TerminateProcess
ascii	21	0x00010CBE	x	function	GetEnvironmentStrings
ascii	21	0x00010CD6	x	function	GetEnvironmentStrings
ascii	15	0x0001087A	x	-	RemoveDirectory
ascii	13	0x000108C2	x	-	CreateProcess
ascii	15	0x000109E2	x	-	SHFileOperation
ascii	22	0x00010A60	x	-	SetEnvironmentVariable
ascii	19	0x00010A92	x	-	SetCurrentDirectory
ascii	22	0x00010B22	x	-	GetEnvironmentVariable
ascii	3	0x0001421C	x	-	999

Sample 2



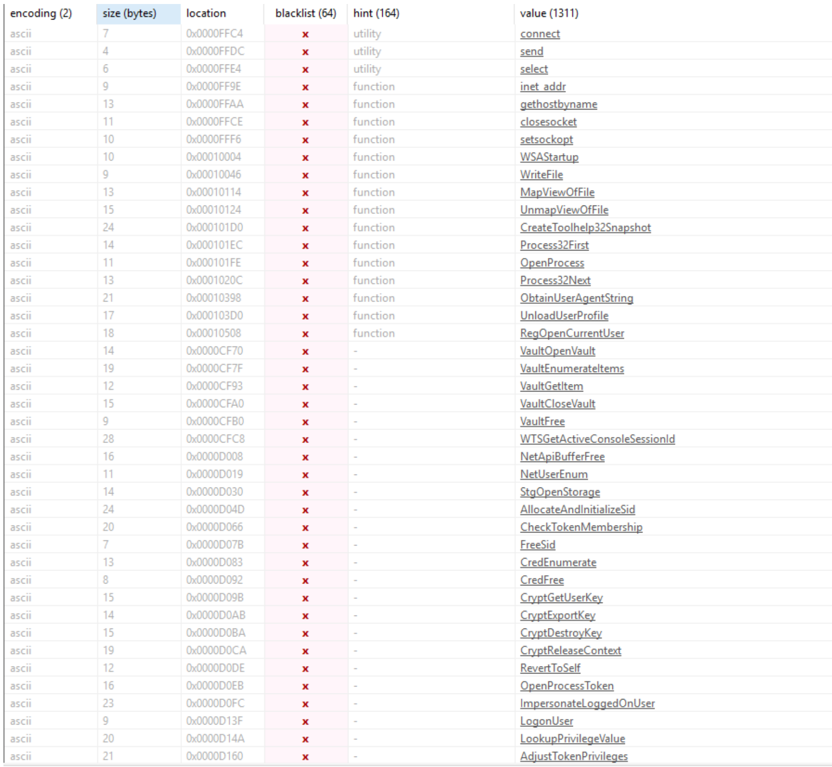
encoding (2)	size (bytes)	location	blacklist (3)	hint (74)	value (1749)
ascii	3	0x0000563D	x	-	148
ascii	3	0x00026107	x	-	999
ascii	3	0x0002BD6E	x	-	999

Sample 4



	Offset	Size	Type	String
884	00029800	00000010	A	Error copying %s
896	00029908	00000009	A	%s%c%s.py
900	00029980	0000000c	A	_pyi_main_co
901	00029990	0000001e	A	pyi-disable-windowed-traceback
904	000299f0	00000010	A	_PYI_ONEDIR_MODE
905	00029a10	0000004d	A	Cannot open PyInstaller archive from executable (%s) ...
911	00029b28	00000018	A	Py_DontWriteBytecodeFlag
912	00029b48	00000032	A	Failed to get address for Py_DontWriteBytecodeFlag
914	00029b90	0000001c	A	Py_FileSystemDefaultEncoding
915	00029bb0	00000036	A	Failed to get address for Py_FileSystemDefaultEncoding
916	00029be8	0000000d	A	Py_FrozenFlag
917	00029bf8	00000027	A	Failed to get address for Py_FrozenFlag
918	00029c28	00000018	A	Py_IgnoreEnvironmentFlag
919	00029c48	00000032	A	Failed to get address for Py_IgnoreEnvironmentFlag
920	00029c80	0000000d	A	Py_NoSiteFlag

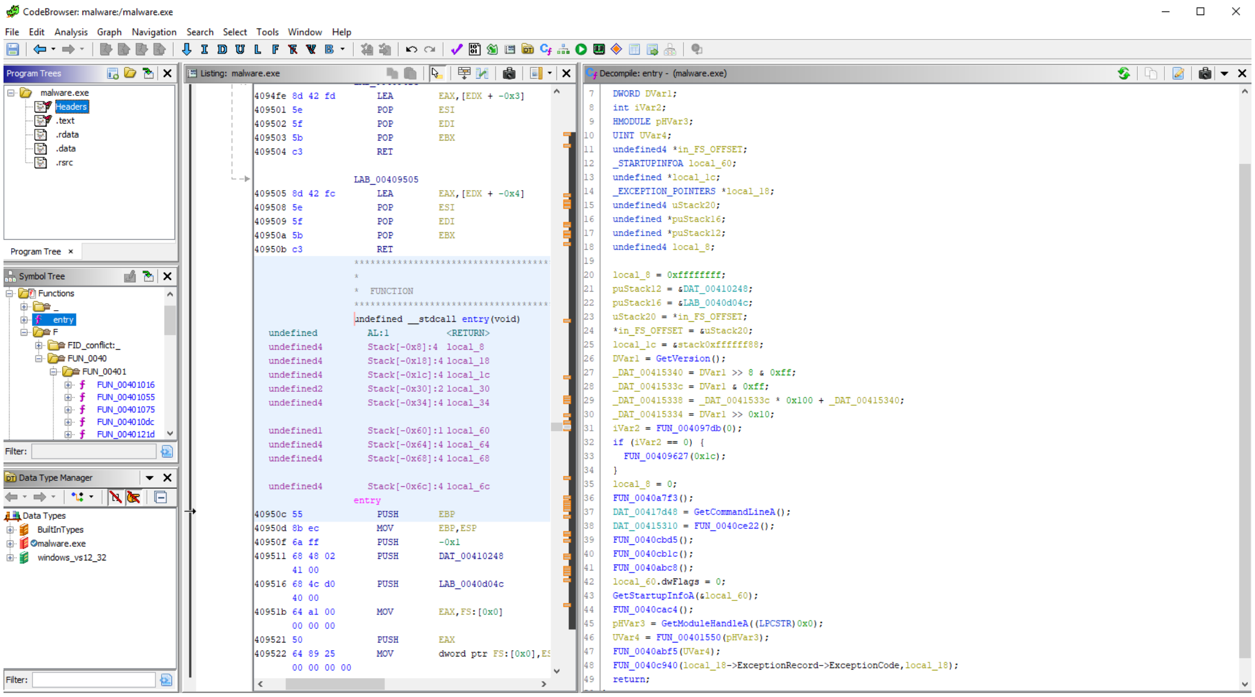
Sample 3



encoding (2)	size (bytes)	location	blacklist (64)	hint (164)	value (1311)
ascii	7	0x0000FFC4	x	utility	connect
ascii	4	0x0000FFDC	x	utility	send
ascii	6	0x0000FFE4	x	utility	select
ascii	9	0x0000FF9E	x	function	inet_addr
ascii	13	0x0000FFAA	x	function	gethostbyname
ascii	11	0x0000FFCE	x	function	closesocket
ascii	10	0x0000FFF6	x	function	setsockopt
ascii	10	0x00010004	x	function	WSAStartup
ascii	9	0x00010046	x	function	WriteFile
ascii	13	0x00010114	x	function	MapViewOfFile
ascii	15	0x00010124	x	function	UnmapViewOfFile
ascii	24	0x000101D0	x	function	CreateToolhelp32Snapshot
ascii	14	0x000101EC	x	function	Process32First
ascii	11	0x000101FE	x	function	OpenProcess
ascii	13	0x0001020C	x	function	Process32Next
ascii	21	0x00010398	x	function	ObtainUserAgentString
ascii	17	0x000103D0	x	function	UnloadUserProfile
ascii	18	0x00010508	x	function	RegOpenCurrentUser
ascii	14	0x0000CF70	x	-	VaultOpenVault
ascii	19	0x0000CF7F	x	-	VaultEnumerateItems
ascii	12	0x0000CF93	x	-	VaultGetItem
ascii	15	0x0000CFA0	x	-	VaultCloseVault
ascii	9	0x0000CFB0	x	-	VaultFree
ascii	28	0x0000CFC8	x	-	WTSGetActiveConsoleSessionId
ascii	16	0x0000D008	x	-	NetApiBufferFree
ascii	11	0x0000D019	x	-	NetUserEnum
ascii	14	0x0000D030	x	-	StgOpenStorage
ascii	24	0x0000D04D	x	-	AllocateAndInitializeSid
ascii	20	0x0000D066	x	-	CheckTokenMembership
ascii	7	0x0000D07B	x	-	FreeSid
ascii	13	0x0000D083	x	-	CredEnumerate
ascii	8	0x0000D092	x	-	CredFree
ascii	15	0x0000D09B	x	-	CryptGetUserKey
ascii	14	0x0000D0AB	x	-	CryptExportKey
ascii	15	0x0000D0BA	x	-	CryptDestroyKey
ascii	19	0x0000D0CA	x	-	CryptReleaseContext
ascii	12	0x0000D0DE	x	-	RevertToSelf
ascii	16	0x0000D0EB	x	-	OpenProcessToken
ascii	23	0x0000D0FC	x	-	ImpersonateLoggedOnUser
ascii	9	0x0000D13F	x	-	LogonUser
ascii	20	0x0000D14A	x	-	LookupPrivilegeValue
ascii	21	0x0000D160	x	-	AdjustTokenPrivileges

For our final step in static analysis, we used reverse engineering to examine the code.

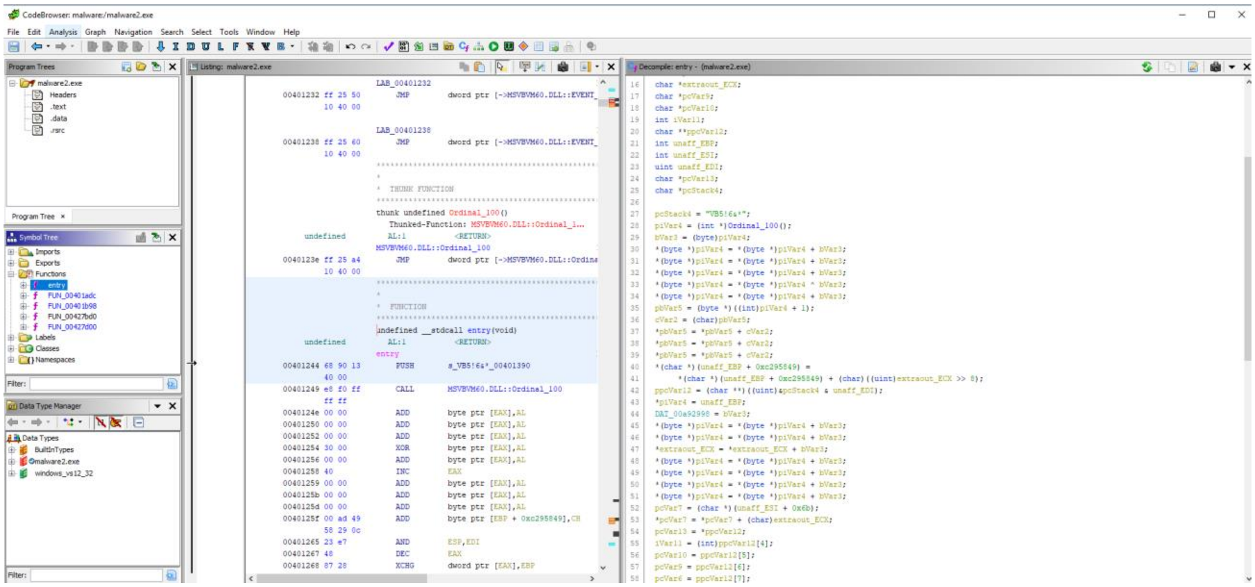
Sample 1



```
4094fe 8d 42 fd        LEA        EAX,[EDX + -0x3]
409501 5e              POP        ESI
409502 5f              POP        EDI
409503 5b              POP        EBX
409504 c3              RET

                   LAB_00409505
409505 8d 42 fc        LEA        EAX,[EDX + -0x4]
409508 5e              POP        ESI
409509 5f              POP        EDI
40950a 5b              POP        EBX
40950b c3              RET
                   ************************************
                   *
                   *  FUNCTION
                   ************************************
                   undefined __stdcall entry(void)
  undefined          AL:1           <RETURN>
  undefined4         Stack[-0x8]:4  local_8
  undefined4         Stack[-0x18]:4 local_18
  undefined4         Stack[-0x1c]:4 local_1c
  undefined2         Stack[-0x30]:2 local_30
  undefined4         Stack[-0x34]:4 local_34

  undefined1         Stack[-0x60]:1 local_60
  undefined4         Stack[-0x64]:4 local_64
  undefined4         Stack[-0x68]:4 local_68

  undefined4         Stack[-0x6c]:4 local_6c
                   entry
40950c 55              PUSH       EBP
40950d 8b ec           MOV        EBP,ESP
40950f 6a ff           PUSH       -0x1
409511 68 48 02        PUSH       DAT_00410248
       41 00
409516 68 4c d0        PUSH       LAB_0040d04c
       40 00
40951b 64 a1 00        MOV        EAX,FS:[0x0]
       00 00 00
409521 50              PUSH       EAX
409522 64 89 25        MOV        dword ptr FS:[0x0],ES
       00 00 00 00
```

```
  DWORD DVar1;
  int iVar2;
  HMODULE pHVar3;
  UINT UVar4;
  undefined4 *in_FS_OFFSET;
  _STARTUPINFOA local_60;
  undefined *local_1c;
  _EXCEPTION_POINTERS *local_18;
  undefined4 uStack20;
  undefined *puStack16;
  undefined *puStack12;
  undefined4 local_8;

  local_8 = 0xffffffff;
  puStack12 = &DAT_00410248;
  puStack16 = &LAB_0040d04c;
  uStack20 = *in_FS_OFFSET;
  *in_FS_OFFSET = &uStack20;
  local_1c = &stack0xffffff88;
  DVar1 = GetVersion();
  _DAT_00415340 = DVar1 >> 8 & 0xff;
  _DAT_0041533c = DVar1 & 0xff;
  _DAT_00415338 = _DAT_0041533c * 0x100 + _DAT_00415340;
  _DAT_00415334 = DVar1 >> 0x10;
  iVar2 = FUN_004097db(0);
  if (iVar2 == 0) {
    FUN_00409627(0x1c);
  }
  local_8 = 0;
  FUN_0040a7f3();
  DAT_00417d48 = GetCommandLineA();
  DAT_00415310 = FUN_0040ce22();
  FUN_0040cbd5();
  FUN_0040cb1c();
  FUN_0040abc8();
  local_60.dwFlags = 0;
  GetStartupInfoA(&local_60);
  FUN_0040cac4();
  pHVar3 = GetModuleHandleA((LPCSTR)0x0);
  UVar4 = FUN_00401550(pHVar3);
  FUN_0040abf5(UVar4);
  FUN_0040c940(local_18->ExceptionRecord->ExceptionCode,local_18);
  return;
```

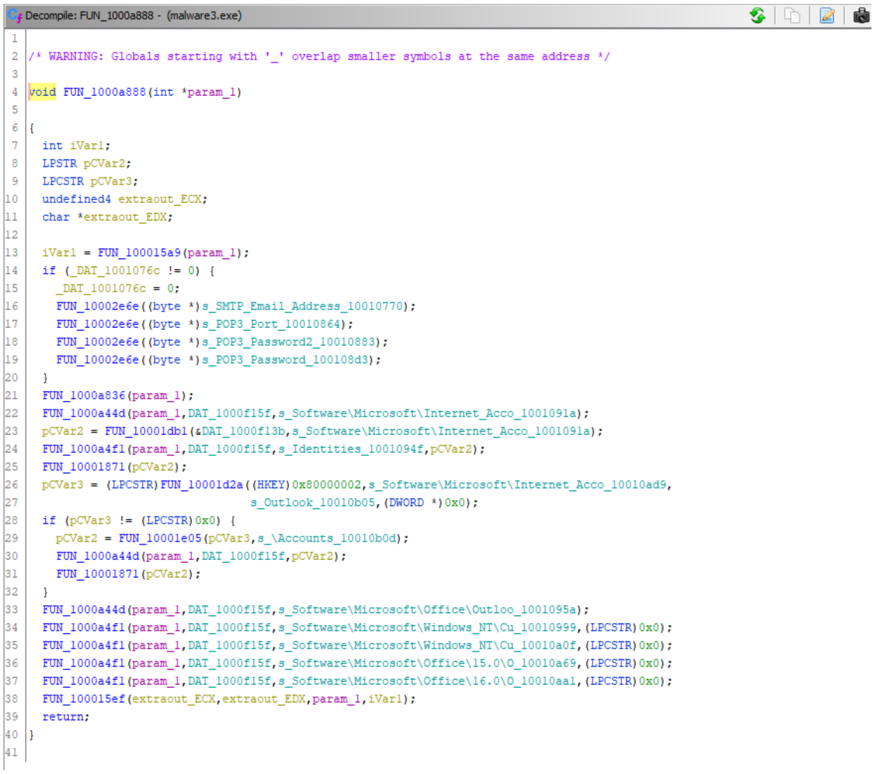
Sample 2



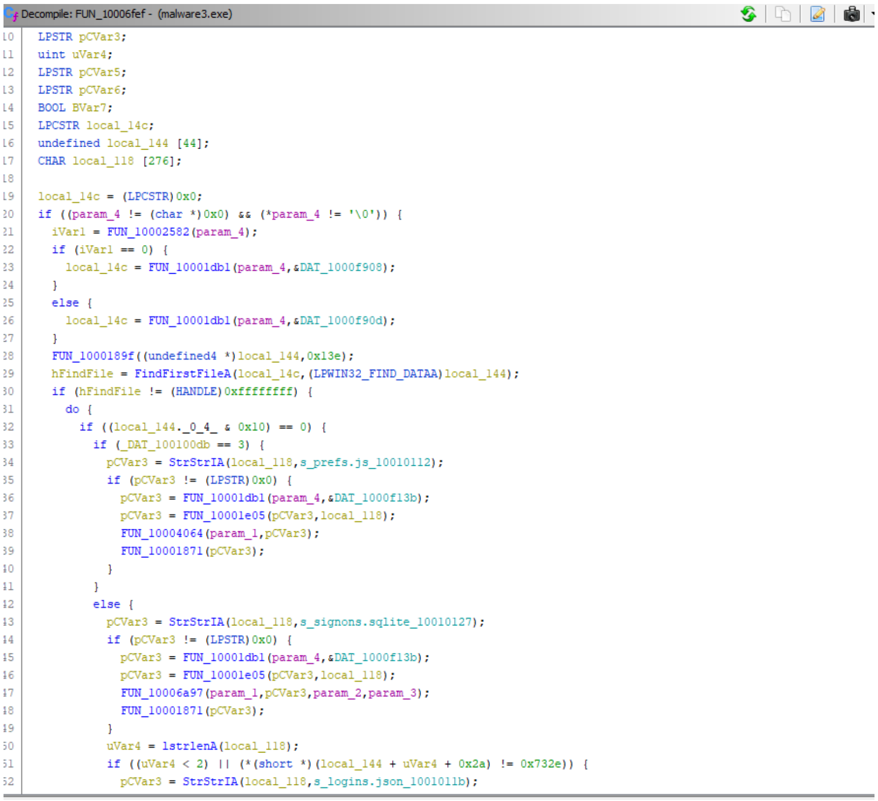
```
                         LAB_00401232
00401232 ff 25 50           JMP        dword ptr [->MSVBVM60.DLL::EVENT_
         10 40 00

                         LAB_00401238
00401238 ff 25 60           JMP        dword ptr [->MSVBVM60.DLL::EVENT_
         10 40 00
                         ****************************************
                         *
                         *  THUNK FUNCTION
                         ****************************************
                         thunk undefined Ordinal_100()
                           Thunked-Function: MSVBVM60.DLL::Ordinal_1...
      undefined            AL:1           <RETURN>
                         MSVBVM60.DLL::Ordinal_100
0040123e ff 25 a4           JMP        dword ptr [->MSVBVM60.DLL::Ordina
         10 40 00
                         ****************************************
                         *
                         *  FUNCTION
                         ****************************************
                         undefined __stdcall entry(void)
      undefined            AL:1           <RETURN>
                         entry
00401244 68 90 13           PUSH       s_VB5!6&*_00401390
         40 00
00401249 e8 f0 ff           CALL       MSVBVM60.DLL::Ordinal_100
         ff ff
0040124e 00 00              ADD        byte ptr [EAX],AL
00401250 00 00              ADD        byte ptr [EAX],AL
00401252 00 00              ADD        byte ptr [EAX],AL
00401254 30 00              XOR        byte ptr [EAX],AL
00401256 00 00              ADD        byte ptr [EAX],AL
00401258 40                 INC        EAX
00401259 00 00              ADD        byte ptr [EAX],AL
0040125b 00 00              ADD        byte ptr [EAX],AL
0040125d 00 00              ADD        byte ptr [EAX],AL
0040125f 00 ad 49           ADD        byte ptr [EBP + 0xc295849],CH
         58 29 0c
00401265 23 e7              AND        ESP,EDI
00401267 48                 DEC        EAX
00401268 87 28              XCHG       dword ptr [EAX],EBP
```

```
  char *extraout_ECX;
  char *pcVar9;
  char *pcVar10;
  int iVar11;
  char **ppcVar12;
  int unaff_EBP;
  int unaff_ESI;
  uint unaff_EDI;
  char *pcVar13;
  char *pcStack4;

  pcStack4 = "VB5!6&*";
  piVar4 = (int *)Ordinal_100();
  bVar3 = (byte)piVar4;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 ^ bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  pbVar5 = (byte *)((int)piVar4 + 1);
  cVar2 = (char)pbVar5;
  *pbVar5 = *pbVar5 + cVar2;
  *pbVar5 = *pbVar5 + cVar2;
  *pbVar5 = *pbVar5 + cVar2;
  *(char *)(unaff_EBP + 0xc295849) =
       *(char *)(unaff_EBP + 0xc295849) + (char)((uint)extraout_ECX >> 8);
  ppcVar12 = (char **)((uint)&pcStack4 & unaff_EDI);
  *piVar4 = unaff_EBP;
  DAT_00a92998 = bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *extraout_ECX = *extraout_ECX + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  *(byte *)piVar4 = *(byte *)piVar4 + bVar3;
  pcVar7 = (char *)(unaff_ESI + 0x6b);
  *pcVar7 = *pcVar7 + (char)extraout_ECX;
  pcVar13 = *ppcVar12;
  iVar11 = (int)ppcVar12[4];
  pcVar10 = ppcVar12[5];
  pcVar9 = ppcVar12[6];
  pcVar6 = ppcVar12[7];
```

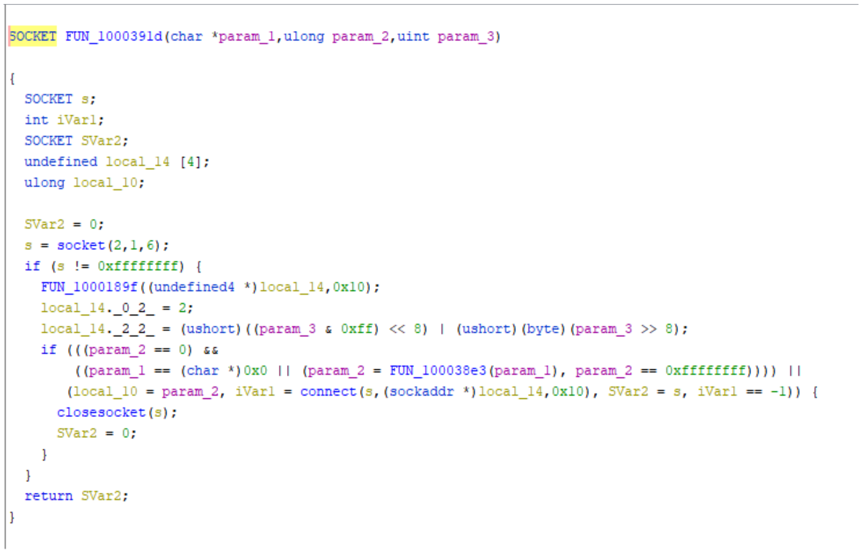
Sample 3



```

/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void FUN_1000a888(int *param_1)

{
  int iVar1;
  LPSTR pCVar2;
  LPCSTR pCVar3;
  undefined4 extraout_ECX;
  char *extraout_EDX;

  iVar1 = FUN_100015a9(param_1);
  if (_DAT_1001076c != 0) {
    _DAT_1001076c = 0;
    FUN_10002e6e((byte *)s_SMTP_Email_Address_10010770);
    FUN_10002e6e((byte *)s_POP3_Port_10010864);
    FUN_10002e6e((byte *)s_POP3_Password2_10010883);
    FUN_10002e6e((byte *)s_POP3_Password_100108d3);
  }
  FUN_1000a836(param_1);
  FUN_1000a44d(param_1,DAT_1000f15f,s_Software\Microsoft\Internet_Acco_1001091a);
  pCVar2 = FUN_10001dbl(&DAT_1000f13b,s_Software\Microsoft\Internet_Acco_1001091a);
  FUN_1000a4f1(param_1,DAT_1000f15f,s_Identities_1001094f,pCVar2);
  FUN_10001871(pCVar2);
  pCVar3 = (LPCSTR)FUN_10001d2a((HKEY)0x80000002,s_Software\Microsoft\Internet_Acco_10010ad9,
                                s_Outlook_10010b05,(DWORD *)0x0);
  if (pCVar3 != (LPCSTR)0x0) {
    pCVar2 = FUN_10001e05(pCVar3,s_\Accounts_10010b0d);
    FUN_1000a44d(param_1,DAT_1000f15f,pCVar2);
    FUN_10001871(pCVar2);
  }
  FUN_1000a44d(param_1,DAT_1000f15f,s_Software\Microsoft\Office\Outloo_1001095a);
  FUN_1000a4f1(param_1,DAT_1000f15f,s_Software\Microsoft\Windows_NT\Cu_10010999,(LPCSTR)0x0);
  FUN_1000a4f1(param_1,DAT_1000f15f,s_Software\Microsoft\Windows_NT\Cu_10010a0f,(LPCSTR)0x0);
  FUN_1000a4f1(param_1,DAT_1000f15f,s_Software\Microsoft\Office\15.0\O_10010a69,(LPCSTR)0x0);
  FUN_1000a4f1(param_1,DAT_1000f15f,s_Software\Microsoft\Office\16.0\O_10010aa1,(LPCSTR)0x0);
  FUN_100015ef(extraout_ECX,extraout_EDX,param_1,iVar1);
  return;
}

```



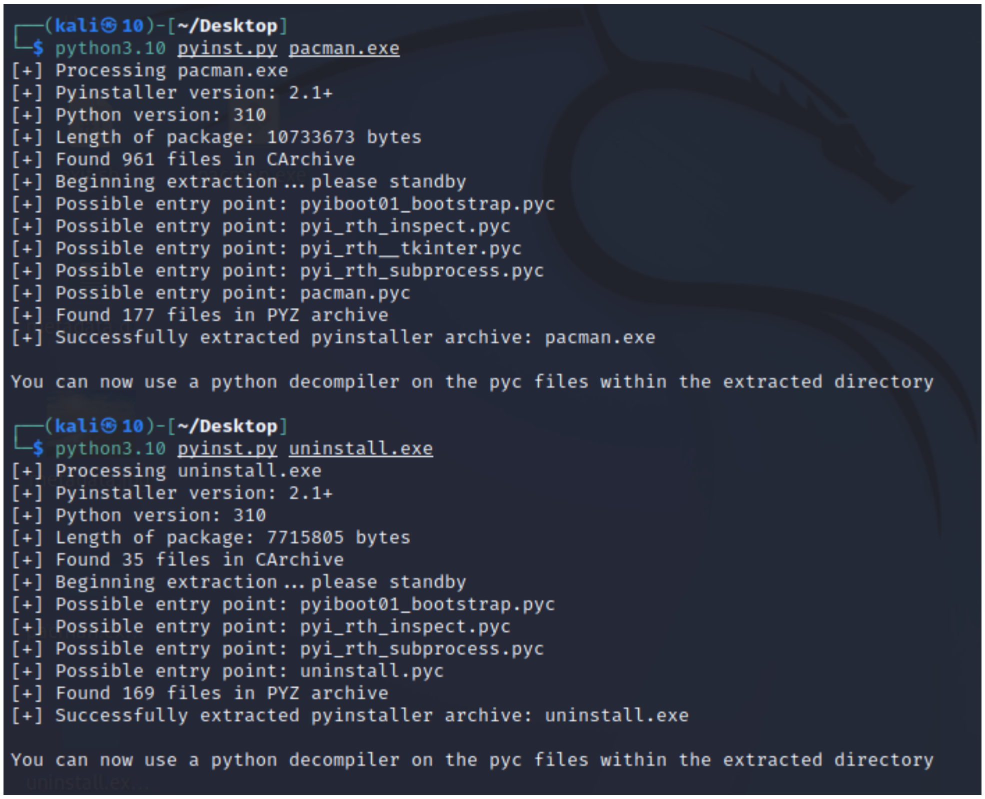
```
  LPSTR pCVar3;
  uint uVar4;
  LPSTR pCVar5;
  LPSTR pCVar6;
  BOOL BVar7;
  LPCSTR local_14c;
  undefined local_144 [44];
  CHAR local_118 [276];

  local_14c = (LPCSTR)0x0;
  if ((param_4 != (char *)0x0) && (*param_4 != '\0')) {
    iVar1 = FUN_10002582(param_4);
    if (iVar1 == 0) {
      local_14c = FUN_10001dbl(param_4,&DAT_1000f908);
    }
    else {
      local_14c = FUN_10001dbl(param_4,&DAT_1000f90d);
    }
    FUN_1000189f((undefined4 *)local_144,0x13e);
    hFindFile = FindFirstFileA(local_14c,(LPWIN32_FIND_DATAA)local_144);
    if (hFindFile != (HANDLE)0xffffffff) {
      do {
        if ((local_144._0_4_ & 0x10) == 0) {
          if (_DAT_100100db == 3) {
            pCVar3 = StrStrIA(local_118,s_prefs.js_10010112);
            if (pCVar3 != (LPSTR)0x0) {
              pCVar3 = FUN_10001dbl(param_4,&DAT_1000f13b);
              pCVar3 = FUN_10001e05(pCVar3,local_118);
              FUN_10004064(param_1,pCVar3);
              FUN_10001871(pCVar3);
            }
          }
          else {
            pCVar3 = StrStrIA(local_118,s_signons.sqlite_10010127);
            if (pCVar3 != (LPSTR)0x0) {
              pCVar3 = FUN_10001dbl(param_4,&DAT_1000f13b);
              pCVar3 = FUN_10001e05(pCVar3,local_118);
              FUN_10006a97(param_1,pCVar3,param_2,param_3);
              FUN_10001871(pCVar3);
            }
            uVar4 = lstrlenA(local_118);
            if ((uVar4 < 2) || (*(short *)(local_144 + uVar4 + 0x2a) != 0x732e)) {
              pCVar3 = StrStrIA(local_118,s_logins.json_1001011b);
```



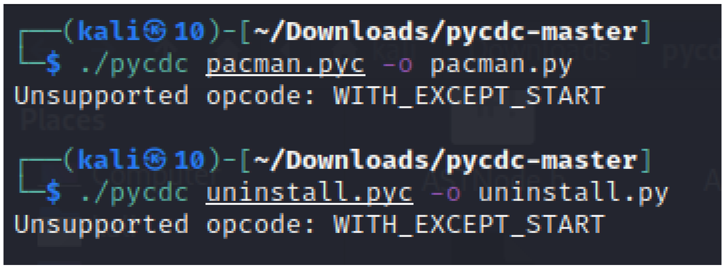
```
SOCKET FUN_1000391d(char *param_1,ulong param_2,uint param_3)

{
  SOCKET s;
  int iVar1;
  SOCKET SVar2;
  undefined local_14 [4];
  ulong local_10;

  SVar2 = 0;
  s = socket(2,1,6);
  if (s != 0xffffffff) {
    FUN_1000189f((undefined4 *)local_14,0x10);
    local_14._0_2_ = 2;
    local_14._2_2_ = (ushort)((param_3 & 0xff) << 8) | (ushort)(byte)(param_3 >> 8);
    if (((param_2 == 0) &&
        ((param_1 == (char *)0x0 || (param_2 = FUN_100038e3(param_1), param_2 == 0xffffffff)))) ||
       (local_10 = param_2, iVar1 = connect(s,(sockaddr *)local_14,0x10), SVar2 = s, iVar1 == -1)) {
      closesocket(s);
      SVar2 = 0;
    }
  }
  return SVar2;
}
```

Sample 4

Since sample 4 was confirmed to be a python file, we used pyinstxtractor to unpack our files and pycdc to decompile it.

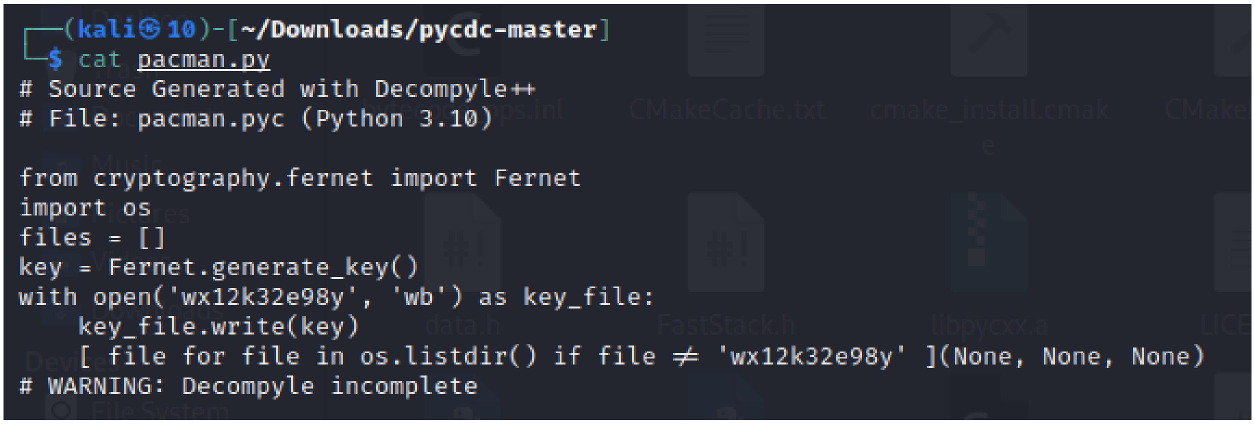


```
┌──(kali㉿10)-[~/Desktop]
└─$ python3.10 pyinst.py pacman.exe
[+] Processing pacman.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 310
[+] Length of package: 10733673 bytes
[+] Found 961 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth__tkinter.pyc
[+] Possible entry point: pyi_rth_subprocess.pyc
[+] Possible entry point: pacman.pyc
[+] Found 177 files in PYZ archive
[+] Successfully extracted pyinstaller archive: pacman.exe

You can now use a python decompiler on the pyc files within the extracted directory

┌──(kali㉿10)-[~/Desktop]
└─$ python3.10 pyinst.py uninstall.exe
[+] Processing uninstall.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 310
[+] Length of package: 7715805 bytes
[+] Found 35 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth_subprocess.pyc
[+] Possible entry point: uninstall.pyc
[+] Found 169 files in PYZ archive
[+] Successfully extracted pyinstaller archive: uninstall.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

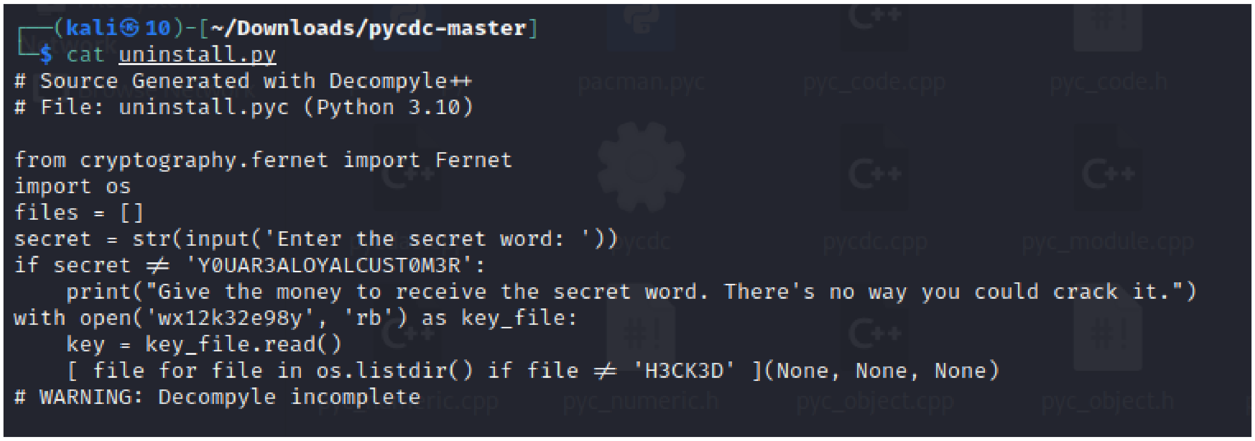


```
┌──(kali㉿10)-[~/Downloads/pycdc-master]
└─$ ./pycdc pacman.pyc -o pacman.py
Unsupported opcode: WITH_EXCEPT_START

┌──(kali㉿10)-[~/Downloads/pycdc-master]
└─$ ./pycdc uninstall.pyc -o uninstall.py
Unsupported opcode: WITH_EXCEPT_START
```

We were unable to decompile to full python code as this program was written in python 3.10 (latest version of python) and there is no fully compatible unpacker developed yet (for this particular version). However, we were able to extract important data.



```
┌──(kali㉿10)-[~/Downloads/pycdc-master]
└─$ cat pacman.py
# Source Generated with Decompyle++
# File: pacman.pyc (Python 3.10)

from cryptography.fernet import Fernet
import os
files = []
key = Fernet.generate_key()
with open('wx12k32e98y', 'wb') as key_file:
    key_file.write(key)
    [ file for file in os.listdir() if file ≠ 'wx12k32e98y' ](None, None, None)
# WARNING: Decompyle incomplete
```

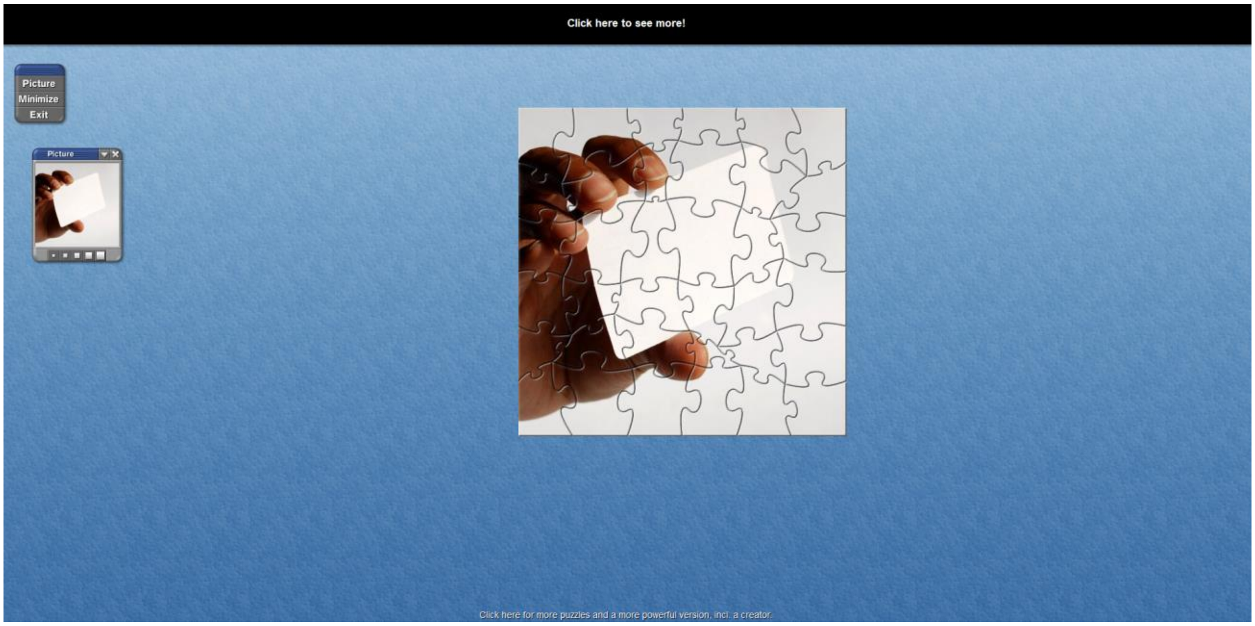


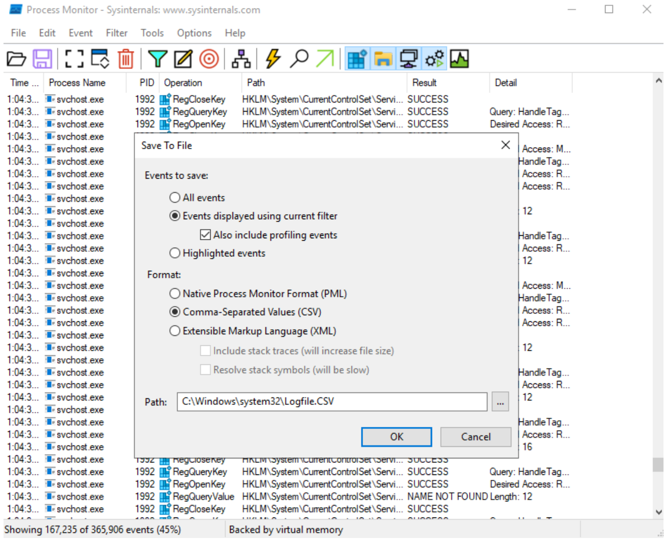
```
┌──(kali㉿10)-[~/Downloads/pycdc-master]
└─$ cat uninstall.py
# Source Generated with Decompyle++
# File: uninstall.pyc (Python 3.10)

from cryptography.fernet import Fernet
import os
files = []
secret = str(input('Enter the secret word: '))
if secret ≠ 'Y0UAR3AL0YALCUST0M3R':
    print("Give the money to receive the secret word. There's no way you could crack it.")
with open('wx12k32e98y', 'rb') as key_file:
    key = key_file.read()
    [ file for file in os.listdir() if file ≠ 'H3CK3D' ](None, None, None)
# WARNING: Decompyle incomplete
```

Dynamic Analysis

For dynamic analysis, we turned on Process Explorer. After starting the capture of processes, we ran each malware sample and waited for 1-2 minutes. Then, we stopped the capture and saved our captures in a procmon-CSV file. We opened that CSV file in ProcDot and selected the sample file we ran as the launcher. Then we analyzed the flow of each malware in a flow graph created by ProcDot.

Sample 1


Click here to see more!
Picture
Minimize
Exit
Picture
Click here for more puzzles and a more powerful version, incl. a creator.



```
Time ...   Process Name   PID   Operation      Path                                    Result            Detail
1:04:3...  svchost.exe    1992  RegCloseKey    HKLM\System\CurrentControlSet\Servi...  SUCCESS
1:04:3...  svchost.exe    1992  RegQueryKey    HKLM\System\CurrentControlSet\Servi...  SUCCESS           Query: HandleTag...
1:04:3...  svchost.exe    1992  RegOpenKey     HKLM\System\CurrentControlSet\Servi...  SUCCESS           Desired Access: R...
...
1:04:3...  svchost.exe    1992  RegQueryKey    HKLM\System\CurrentControlSet\Servi...  SUCCESS           Query: HandleTag...
1:04:3...  svchost.exe    1992  RegOpenKey     HKLM\System\CurrentControlSet\Servi...  SUCCESS           Desired Access: R...
1:04:3...  svchost.exe    1992  RegQueryValue  HKLM\System\CurrentControlSet\Servi...  NAME NOT FOUND    Length: 12
1:04:3...  svchost.exe    1992  RegCloseKey    HKLM\System\CurrentControlSet\Servi...  SUCCESS
```

Save To File

Events to save:

- ○ All events
- ● Events displayed using current filter
  - ☑ Also include profiling events
- ○ Highlighted events

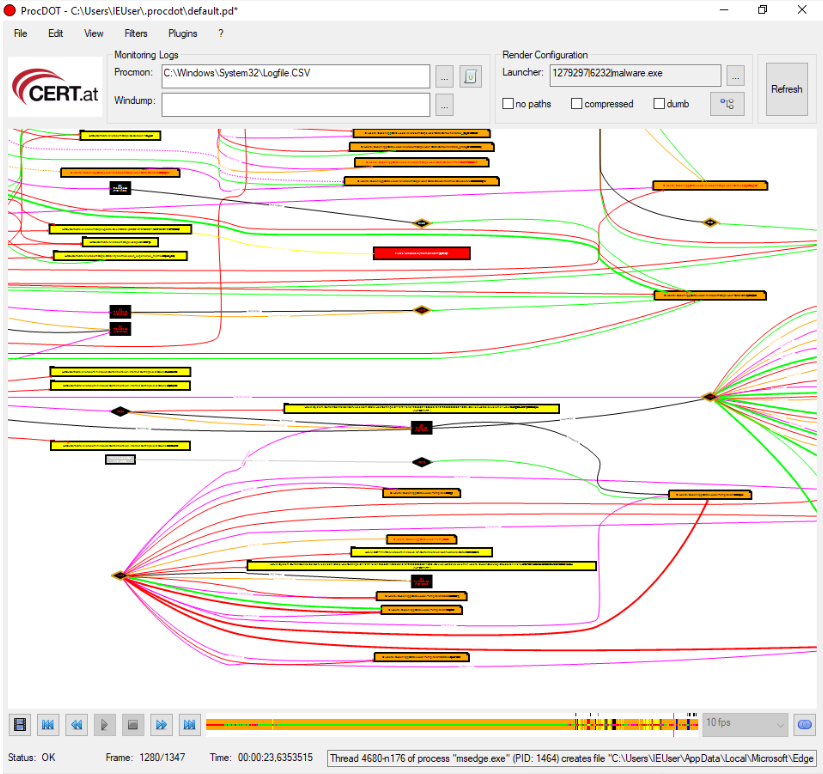
Format:

- ○ Native Process Monitor Format (PML)
- ● Comma-Separated Values (CSV)
- ○ Extensible Markup Language (XML)
  - □ Include stack traces (will increase file size)
  - □ Resolve stack symbols (will be slow)

Path: C:\Windows\system32\Logfile.CSV

OK Cancel

Showing 167,235 of 365,906 events (45%)   Backed by virtual memory



Procmon: C:\Windows\System32\Logfile.CSV

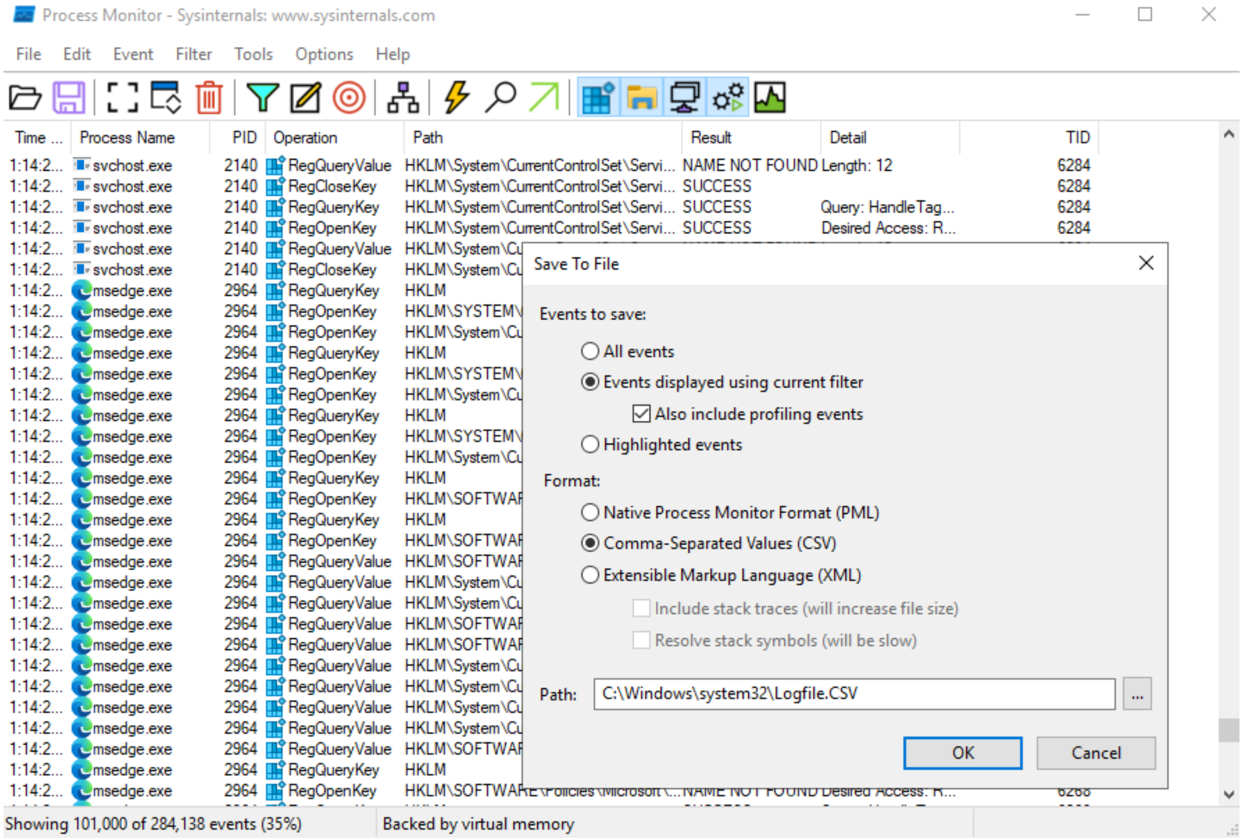
Windump:

Launcher: 1279297|6232|malware.exe

□ no paths □ compressed □ dumb

Status: OK Frame: 1280/1347 Time: 00:00:23,6353515 Thread 4680-n176 of process "msedge.exe" (PID: 1464) creates file "C:\Users\IEUser\AppData\Local\Microsoft\Edge

Sample 2



Time ...	Process Name	PID	Operation	Path	Result	Detail	TID
1:14:2...	svchost.exe	2140	RegQueryValue	HKLM\System\CurrentControlSet\Servi...	NAME NOT FOUND	Length: 12	6284
1:14:2...	svchost.exe	2140	RegCloseKey	HKLM\System\CurrentControlSet\Servi...	SUCCESS		6284
1:14:2...	svchost.exe	2140	RegQueryKey	HKLM\System\CurrentControlSet\Servi...	SUCCESS	Query: HandleTag...	6284
1:14:2...	svchost.exe	2140	RegOpenKey	HKLM\System\CurrentControlSet\Servi...	SUCCESS	Desired Access: R...	6284
1:14:2...	svchost.exe	2140	RegQueryValue	HKLM\System\Cu			
1:14:2...	svchost.exe	2140	RegCloseKey	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryKey	HKLM			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\SYSTEM\			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryKey	HKLM			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\SYSTEM\			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryKey	HKLM			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\SYSTEM\			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryKey	HKLM			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\SOFTWAR			
1:14:2...	msedge.exe	2964	RegQueryKey	HKLM			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\SOFTWAR			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\SOFTWAR			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\SOFTWAR			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\SOFTWAR			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\System\Cu			
1:14:2...	msedge.exe	2964	RegQueryValue	HKLM\SOFTWAR			
1:14:2...	msedge.exe	2964	RegQueryKey	HKLM			
1:14:2...	msedge.exe	2964	RegOpenKey	HKLM\SOFTWARE\Policies\Microsoft\...	NAME NOT FOUND	Desired Access: R...	6268

Save To File

Events to save:

○ All events
● Events displayed using current filter
☑ Also include profiling events
○ Highlighted events

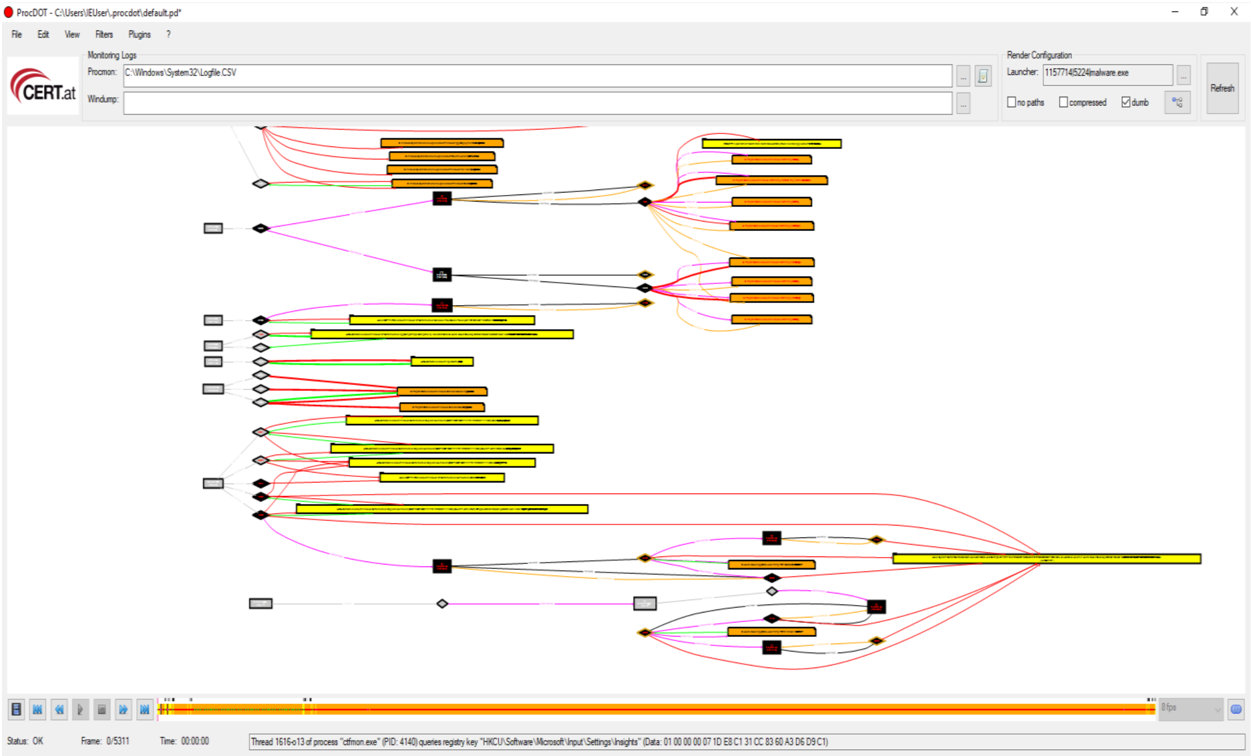
Format:

○ Native Process Monitor Format (PML)
● Comma-Separated Values (CSV)
○ Extensible Markup Language (XML)
□ Include stack traces (will increase file size)
□ Resolve stack symbols (will be slow)

Path: C:\Windows\system32\Logfile.CSV

OK Cancel

Showing 101,000 of 284,138 events (35%) Backed by virtual memory



ProcDOT - C:\Users\IEUser\.procdot\default.pd*

Monitoring Logs

Procmon: C:\Windows\System32\Logfile.CSV

Windump:

Render Configuration

Launcher: 1157714|5224|malware.exe

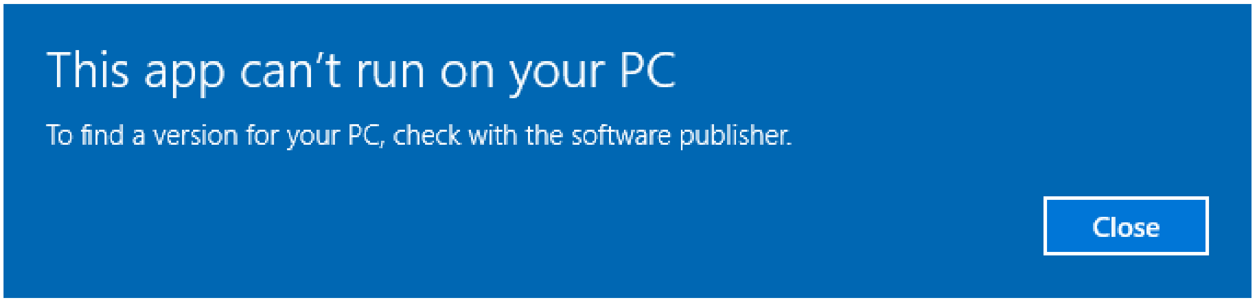
□ no paths □ compressed ☑ dumb

Status: OK Frame: 0/5311 Time: 00:00:00

```
Thread 1616-o13 of process "ctfmon.exe" (PID: 4140) queries registry key "HKCU\Software\Microsoft\Input\Settings\Insights" (Data: 01 00 00 00 07 1D E8 C1 31 CC 83 60 A3 D6 D9 C1)
```

Sample 3

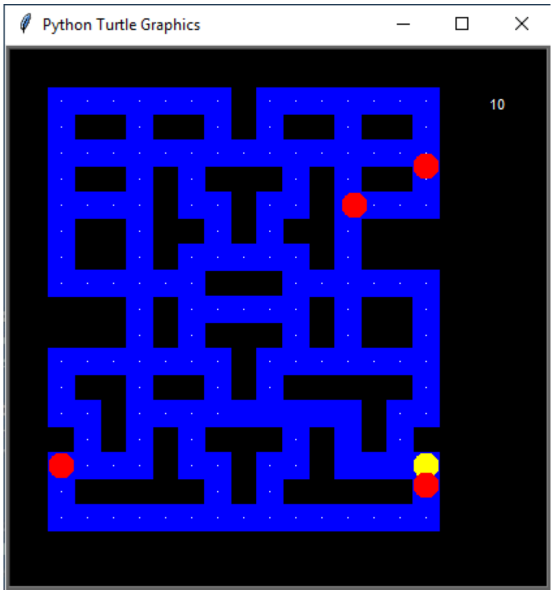
We were unable to run the 3rd sample as it was a dynamically linked library instead of an executable. However, we got more than enough about this malware through static analysis.

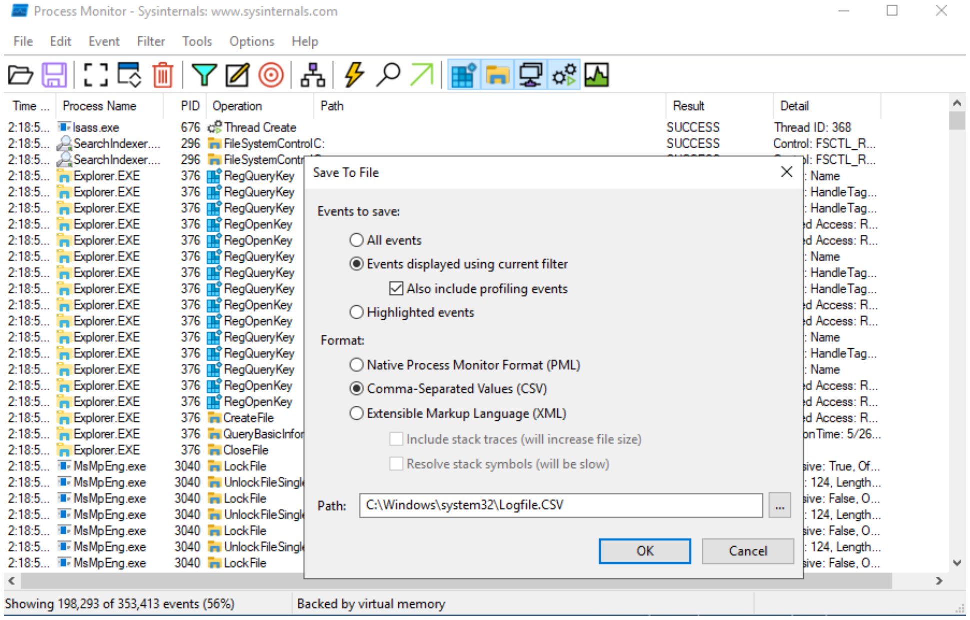


This app can't run on your PC

To find a version for your PC, check with the software publisher.

Sample 4 (pacman.exe)




Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
2:18:5...	lsass.exe	676	Thread Create		SUCCESS	Thread ID: 368
2:18:5...	SearchIndexer....	296	FileSystemControl	C:	SUCCESS	Control: FSCTL_R...
2:18:5...	SearchIndexer....	296	FileSystemContr			l: FSCTL_R...
2:18:5...	Explorer.EXE	376	RegQueryKey			: Name
2:18:5...	Explorer.EXE	376	RegQueryKey			: HandleTag...
2:18:5...	Explorer.EXE	376	RegQueryKey			: HandleTag...
2:18:5...	Explorer.EXE	376	RegOpenKey			d Access: R...
2:18:5...	Explorer.EXE	376	RegOpenKey			d Access: R...
2:18:5...	Explorer.EXE	376	RegQueryKey			: Name
2:18:5...	Explorer.EXE	376	RegQueryKey			: HandleTag...
2:18:5...	Explorer.EXE	376	RegQueryKey			: HandleTag...
2:18:5...	Explorer.EXE	376	RegOpenKey			d Access: R...
2:18:5...	Explorer.EXE	376	RegOpenKey			d Access: R...
2:18:5...	Explorer.EXE	376	RegQueryKey			: Name
2:18:5...	Explorer.EXE	376	RegQueryKey			: HandleTag...
2:18:5...	Explorer.EXE	376	RegQueryKey			: Name
2:18:5...	Explorer.EXE	376	RegOpenKey			d Access: R...
2:18:5...	Explorer.EXE	376	RegOpenKey			d Access: R...
2:18:5...	Explorer.EXE	376	CreateFile			d Access: R...
2:18:5...	Explorer.EXE	376	QueryBasicInfor			onTime: 5/26...
2:18:5...	Explorer.EXE	376	CloseFile			
2:18:5...	MsMpEng.exe	3040	LockFile			ive: True, Of...
2:18:5...	MsMpEng.exe	3040	UnlockFileSingl			: 124, Length...
2:18:5...	MsMpEng.exe	3040	LockFile			ive: False, O...
2:18:5...	MsMpEng.exe	3040	UnlockFileSingl			: 124, Length...
2:18:5...	MsMpEng.exe	3040	LockFile			ive: False, O...
2:18:5...	MsMpEng.exe	3040	UnlockFileSingl			: 124, Length...
2:18:5...	MsMpEng.exe	3040	LockFile			ive: False, O...

Save To File

Events to save:

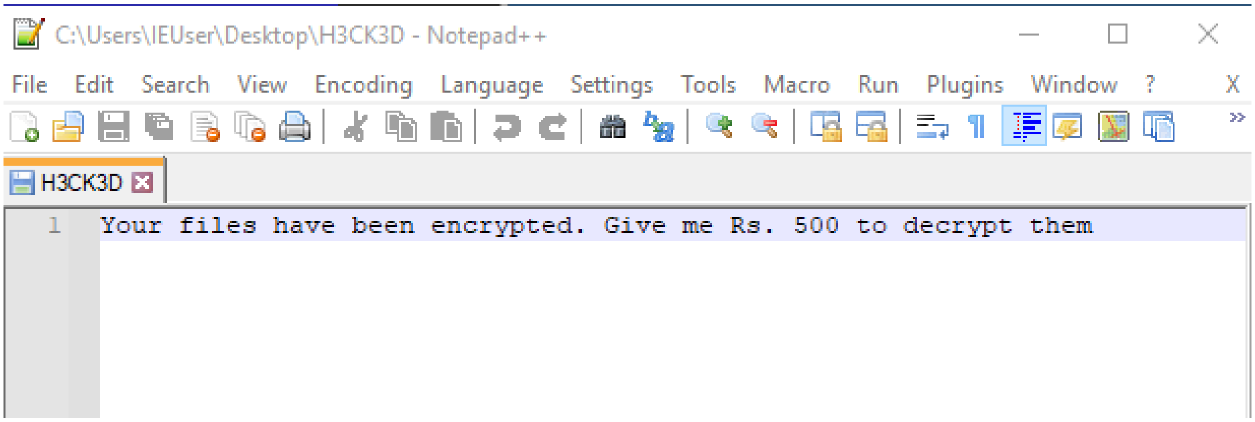
○ All events
● Events displayed using current filter
☑ Also include profiling events
○ Highlighted events

Format:

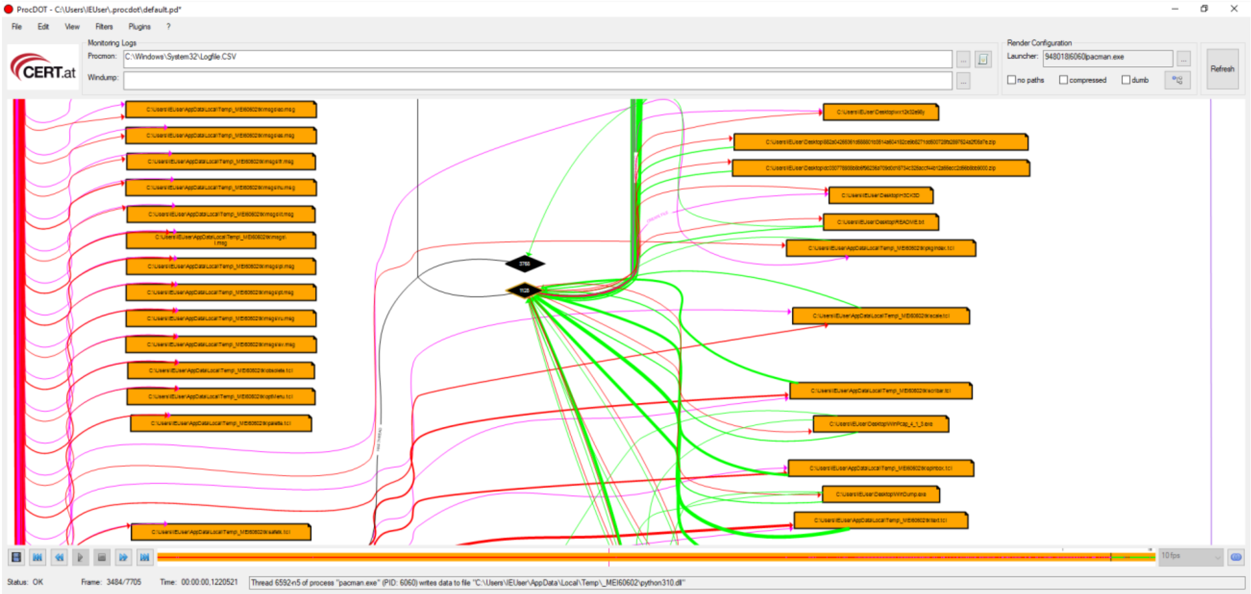
○ Native Process Monitor Format (PML)
● Comma-Separated Values (CSV)
○ Extensible Markup Language (XML)
□ Include stack traces (will increase file size)
□ Resolve stack symbols (will be slow)

Path: C:\Windows\system32\Logfile.CSV

Showing 198,293 of 353,413 events (56%) | Backed by virtual memory


C:\Users\IEUser\Desktop\H3CK3D - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
H3CK3D

```
Your files have been encrypted. Give me Rs. 500 to decrypt them
```



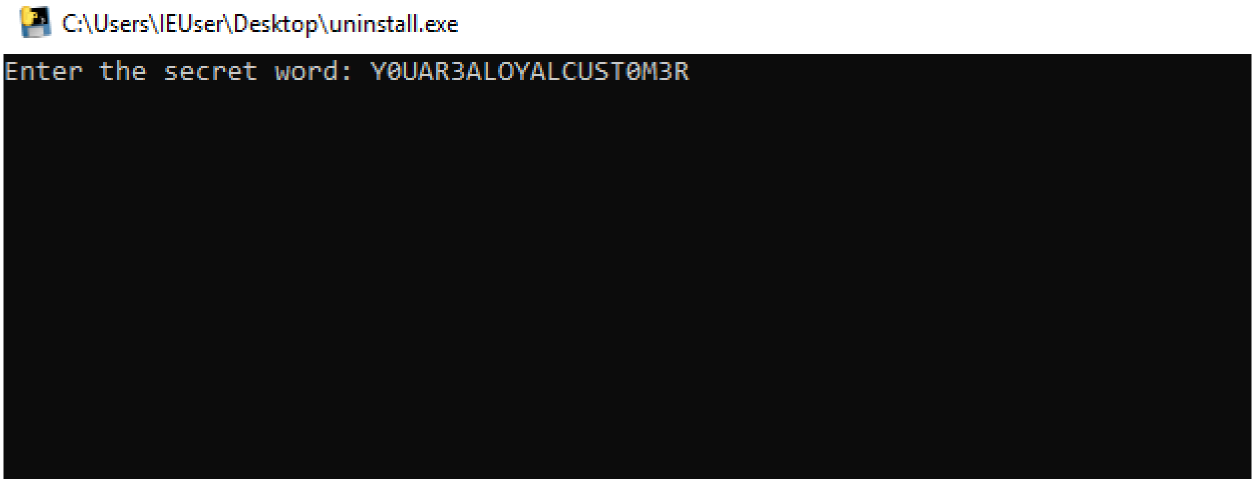
Procmon: C:\Windows\System32\Logfile.CSV

Windump:

Launcher: 948018|6060|pacman.exe

Status: OK Frame: 3484/7705 Time: 00:00:00,1220521 Thread 6592-n5 of process "pacman.exe" (PID: 6060) writes data to file "C:\Users\IEUser\AppData\Local\Temp_MEI60602\python310.dll"

Sample 4 (uninstall.exe)



C:\Users\IEUser\Desktop\uninstall.exe

```
Enter the secret word: Y0UAR3ALOYALCUST0M3R
```

7- Relevant Findings

7a- Static Analysis

Malware 1:

Hash (SHA256)	88c5be944437cc07361723a1745e9e643d54c8579b9e75d1e491a0746f689b01		
File Type	Portable Executable (PE)		
Target System	Windows		
Target CPU	32 bit		
Compiler Stamp	2007-05-23 00:41:48		
Subsystem	GUI		
Permissions	**.text**	**.rdata**	**.data**
	-	-	writeable
	executable	-	-
	-	-	-
Potentially abused Libraries	Kernel32.dll	User32.dll	Shell32.dll
Packing	unpacked		
String Analysis	String Analysis confirms the use of malicious libraries.		
Ghidra Analysis	• Lots of unnecessary functions • Duplicate functions that perform similar functions. Called on either conditions of comparison-based jumps which implies that a function is being called forcefully. • Creation of processes and deletion of system files and directories (through system calls)		

Malware 2:

<table>
<tr><td>Hash (SHA256)</td><td colspan="3">882a04265361d588801b3514a604182ce9b8271dd500728fa2897524a2f05a7e</td></tr>
<tr><td>File Type</td><td colspan="3">Portable Executable (PE)</td></tr>
<tr><td>Target System</td><td colspan="3">Windows</td></tr>
<tr><td>Target CPU</td><td colspan="3">32 bit</td></tr>
<tr><td>Compiler Stamp</td><td colspan="3">2017-11-17 05:18:30</td></tr>
<tr><td>Subsystem</td><td colspan="3">GUI</td></tr>
<tr><td rowspan="4">Permissions</td><td><b>.text</b></td><td><b>.rsrc</b></td><td><b>.data</b></td></tr>
<tr><td>-</td><td>-</td><td>writeable</td></tr>
<tr><td>executable</td><td>-</td><td>-</td></tr>
<tr><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Potentially abused Libraries</td><td colspan="3">Msbvm60.dll</td></tr>
<tr><td>Packing</td><td colspan="3">unpacked</td></tr>
<tr><td>String Analysis</td><td colspan="3">String analysis hints at similar patterns as other malwares eg. 999.</td></tr>
<tr><td>Ghidra Analysis</td><td colspan="3">• Confirmation of the use of KERNEL32.dll which could not be detected by basic static analysis. (i.e. attacker has taken additional steps to hide its use)
• Functions:
○ FUN_00401adc
○ FUN_00401b98
○ FUN_00427bd0
○ FUN_00427d00</td></tr>
</table>

Malware 3:

<table>
<tr><td>Hash (SHA256)</td><td>Dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000</td></tr>
<tr><td>File Type</td><td>dll</td></tr>
<tr><td>Target System</td><td>Windows</td></tr>
<tr><td>Target CPU</td><td>32 bit</td></tr>
<tr><td>Compiler Stamp</td><td>2016-10-17 04:48:13</td></tr>
<tr><td>Subsystem</td><td>GUI</td></tr>
<tr><td>Permissions</td><td><table>
<tr><th>.text</th><th>.data</th><th>.rdata</th><th>.reloc</th></tr>
<tr><td>-</td><td>writeable</td><td>-</td><td>-</td></tr>
<tr><td>executable</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table></td></tr>
<tr><td>Potentially abused Libraries</td><td><table>
<tr><td>Kernel32.dll</td><td>Wsock32.dll</td><td>advapi32.dll</td><td>urlmon.dll</td><td>winnet.dll</td><td>shlwapi.dll</td><td>Crypt32.dll</td><td>ole32.dll</td><td>More</td></tr>
</table></td></tr>
<tr><td>Packing</td><td>unpacked</td></tr>
<tr><td>String Analysis</td><td>String analysis confirms the use of cryptography, usage of socket programming and usage of administrative privileges without detection. It also shows some suspicious URLs that are most probably used for remote communication (for a spyware).</td></tr>
<tr><td>Ghidra Analysis</td><td>• Creates and destroys files.
• Creates a socket for an SMTP connection.
• Interacts with sqlite server.
• Connects to IP Addresses using sockets.
• Connects to an FTP server and sends files.</td></tr>
</table>

Malware 4:

<table>
<tr><td>Hash (SHA256)</td><td colspan="5">F9909c40c27c08095fc081c39eabcf59205d78b2d640dd1879dbc74f06c608cf</td></tr>
<tr><td>File Type</td><td colspan="5">Executable</td></tr>
<tr><td>Target System</td><td colspan="5">Windows</td></tr>
<tr><td>Target CPU</td><td colspan="5">64 bit</td></tr>
<tr><td>Compiler Stamp</td><td colspan="5">2022-5-23 14:14:51</td></tr>
<tr><td>Subsystem</td><td colspan="5">GUI</td></tr>
<tr><td rowspan="4">Permissions</td><td><b>.text</b></td><td><b>.rsrc</b></td><td><b>.reloc</b></td><td><b>.rdata</b></td><td><b>.pdata</b></td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Potentially abused Libraries</td><td colspan="5">Kernel32.dll</td></tr>
<tr><td>Packing</td><td colspan="5">Pyinstaller</td></tr>
<tr><td>String Analysis</td><td colspan="5">Because of obfuscation, string analysis did not yield any important giveaways. However, it helped us identify that the executable was packed used pyinstaller.</td></tr>
<tr><td>Python Code Analysis</td><td colspan="5">• File uses cryptography.fernet library which is used for asymmetric encryption.
• Possibility of being a ransomware.
• Pacman.exe encrypts files while uninstall.exe decrypts them using a random key and a secret word.
• Random key is stored in a text file on the target system.
• Secret word is possibly “Y0UAR3ALOYALCUST0M3R”</td></tr>
</table>

7b- Dynamic Analysis

Malware 1:

- Type: Trojan
- Aliases
  - Svchost.exe
  - Malware.exe
  - Setup.exe
  - Msedge.exe
- Malicious activity
  - Replicates itself through multithreading and file creation
  - Adds autostart to registry
  - Steals web browser data
  - Kills main process from time to time to hide itself
  - Adds registry keys and manipulates internet settings in registry
  - Connects to IP Address: 224.0.0.251

Malware 2:

- Type: Password Stealer
- Aliases
  - Malware.exe
  - Wmiprvse.exe
  - Svchost.exe
  - Taskhostw.exe
  - Minibus-cpp.exe
- Malicious activity
  - Hides internet explorer logs
  - Adds autostart to registry
  - Constantly creates processes and threads to kill the parent and hide itself.
  - Changing input settings in registry
  - Writes to security logs
  - Changing write permissions of system file
  - Creates and deletes files in system and temp directories

Malware 3:

- Type: Password Stealer
- Aliases: N/A
- Malicious Activity: N/A
- Note: We were unable to run this malware as it was a dynamically linked library. However, there was no real need for this because advanced static analysis revealed enough of what we needed to know about this malware.

Malware 4:

- Type: Ransomware + Trojan
- Aliases
    - Pacman.exe
    - Uninstall.exe
- Malicious Activity
    - Pacman.exe
        - Opens a pacman game in GUI mode
        - Loads python cryptography library into %APPDATA%
        - Reads all files from /Desktop
        - Encrypts all read files
        - Creates a file "H3CK3D" with a message.
        - Creates a file "wx12k32e98y" with what looks look an AES key.
    - Uninstall.exe
        - Opens a terminal asking for the secret passcode
        - Decrypts all files on the desktop if we use the secret passcode we found in static analysis ("Y0UAR3ALOYALCUST0M3R").

8- Exhibit

Sample 1

Sample 1 is a trojan and possibly a spyware that sends logs to an IP Address 224.0.0.251. However, the spyware may be dormant as it is old and the IP Address leads to a dead end. It is also a virus i.e. it replicates itself, mostly in system directories.

Sample 2

Sample 2 is a Keylogger/Password stealer which adds itself to the root registry and possibly sends logs through internet explorer, as it manipulates IE logs. It is a very smart malware which disguises itself as other processes to manipulate operating system security and can not be detected by the normal user.

Sample 3

Sample 3 is a password stealer which uses ftp and smtp servers to transfer logs/passwords collected from the host system. It also uses cryptography possibly for encrypted transfer or to unencrypt found passwords.

Sample 4

Sample 4 is a basic level ransomware that encrypts all files in the current directory (the directory in which it is run) using python cryptography library- “Fernet”. It can not cause any harm to system files as it does not alter the registry and does not have administrative privileges. However, if it is in an important user directory, the user could lose their important data. If the malware is run twice in the same directory, it is possible that the key will be overwritten and there will be permanent data loss.

9- References

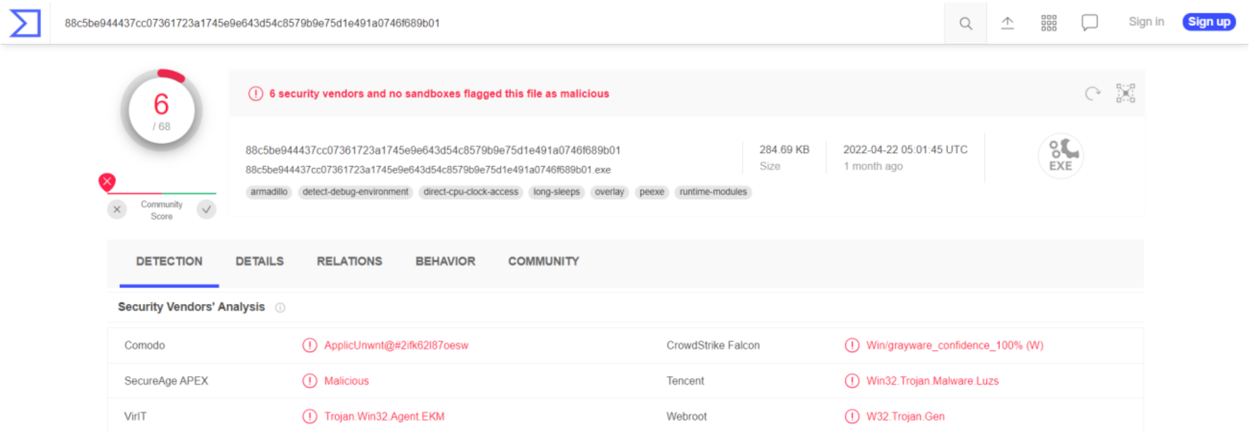
zrax/pycdc: C++ python bytecode disassembler and decompiler (github.com)

extremecoders-re/pyinstxtractor: PyInstaller Extractor (github.com)

Ghidra (ghidra-sre.org)

10- Virus Total Results

Sample 1



6 security vendors and no sandboxes flagged this file as malicious

88c5be944437cc07361723a1745e9e643d54c8579b9e75d1e491a0746f689b01
88c5be944437cc07361723a1745e9e643d54c8579b9e75d1e491a0746f689b01.exe

284.69 KB Size

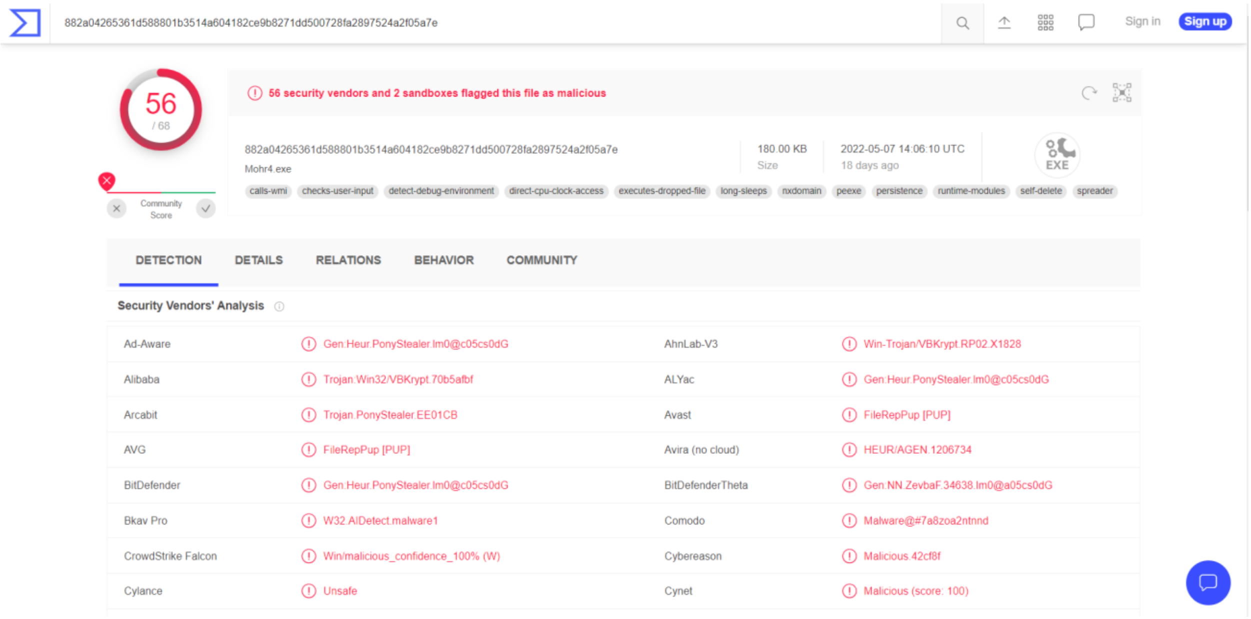
2022-04-22 05:01:45 UTC 1 month ago

armadillo, detect-debug-environment, direct-cpu-clock-access, long-sleeps, overlay, peexe, runtime-modules

Security Vendors' Analysis

Comodo	ApplicUnwnt@#2ifk62l87oesw	CrowdStrike Falcon	Win/grayware_confidence_100% (W)
SecureAge APEX	Malicious	Tencent	Win32.Trojan.Malware.Luzs
VirIT	Trojan.Win32.Agent.EKM	Webroot	W32.Trojan.Gen

Sample 2



56 security vendors and 2 sandboxes flagged this file as malicious

882a04265361d588801b3514a604182ce9b8271dd500728fa2897524a2f05a7e
Mohr4.exe

180.00 KB Size

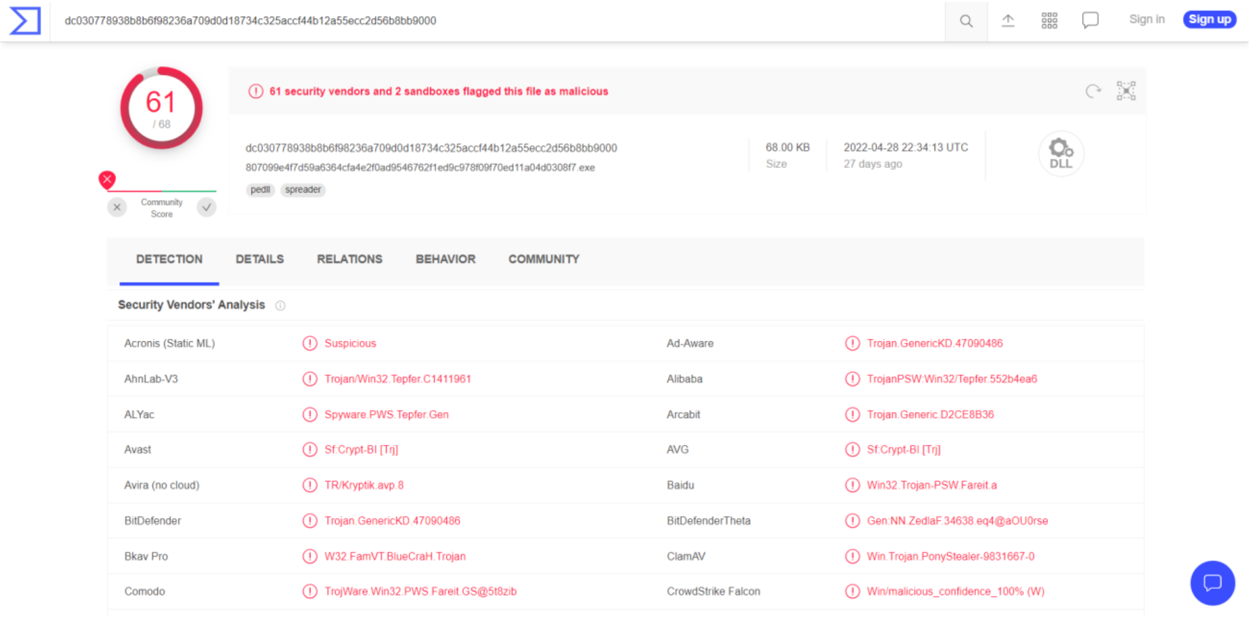
2022-05-07 14:06:10 UTC 18 days ago

calls-wmi, checks-user-input, detect-debug-environment, direct-cpu-clock-access, executes-dropped-file, long-sleeps, nxdomain, peexe, persistence, runtime-modules, self-delete, spreader

Security Vendors' Analysis

Ad-Aware	Gen:Heur.PonyStealer.lm0@c05cs0dG	AhnLab-V3	Win-Trojan/VBKrypt.RP02.X1828
Alibaba	Trojan:Win32/VBKrypt.70b5afbf	ALYac	Gen:Heur.PonyStealer.lm0@c05cs0dG
Arcabit	Trojan.PonyStealer.EE01CB	Avast	FileRepPup [PUP]
AVG	FileRepPup [PUP]	Avira (no cloud)	HEUR/AGEN.1206734
BitDefender	Gen:Heur.PonyStealer.lm0@c05cs0dG	BitDefenderTheta	Gen:NN.ZevbaF.34638.lm0@a05cs0dG
Bkav Pro	W32.AIDetect.malware1	Comodo	Malware@#7a8zoa2ntnnd
CrowdStrike Falcon	Win/malicious_confidence_100% (W)	Cybereason	Malicious.42cf8f
Cylance	Unsafe	Cynet	Malicious (score: 100)

Sample 3



dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000

61 security vendors and 2 sandboxes flagged this file as malicious

dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000
807099e4f7d59a6364cfa4e2f0ad9546762f1ed9c978f09f70ed11a04d0308f7.exe
pedll spreader

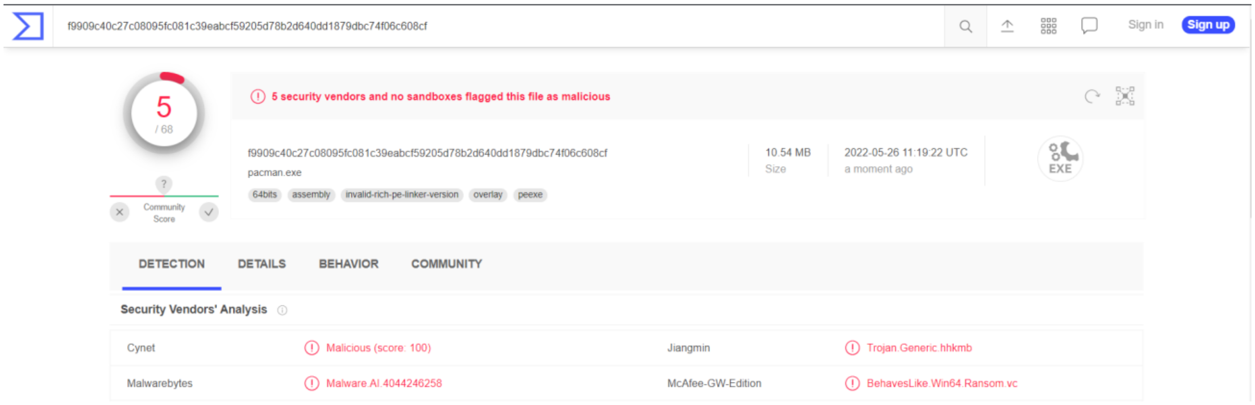
68.00 KB
Size

2022-04-28 22:34:13 UTC
27 days ago

Security Vendors' Analysis

Acronis (Static ML)	Suspicious	Ad-Aware	Trojan.GenericKD.47090486
AhnLab-V3	Trojan/Win32.Tepfer.C1411961	Alibaba	TrojanPSW:Win32/Tepfer.552b4ea6
ALYac	Spyware.PWS.Tepfer.Gen	Arcabit	Trojan.Generic.D2CE8B36
Avast	Sf:Crypt-BI [Trj]	AVG	Sf:Crypt-BI [Trj]
Avira (no cloud)	TR/Kryptik.avp.8	Baidu	Win32.Trojan-PSW.Fareit.a
BitDefender	Trojan.GenericKD.47090486	BitDefenderTheta	Gen:NN.ZedlaF.34638.eq4@aOU0rse
Bkav Pro	W32.FamVT.BlueCraH.Trojan	ClamAV	Win.Trojan.PonyStealer-9831667-0
Comodo	TrojWare.Win32.PWS.Fareit.GS@5t8zib	CrowdStrike Falcon	Win/malicious_confidence_100% (W)

Sample 4



f9909c40c27c08095fc081c39eabcf59205d78b2d640dd1879dbc74f06c608cf

5 security vendors and no sandboxes flagged this file as malicious

f9909c40c27c08095fc081c39eabcf59205d78b2d640dd1879dbc74f06c608cf
pacman.exe
64bits assembly invalid-rich-pe-linker-version overlay peexe

10.54 MB
Size

2022-05-26 11:19:22 UTC
a moment ago

Security Vendors' Analysis

Cynet	Malicious (score: 100)	Jiangmin	Trojan.Generic.hhkmb
Malwarebytes	Malware.AI.4044246258	McAfee-GW-Edition	BehavesLike.Win64.Ransom.vc