Malware Analysis: Mystery USB File

Analysis Members:
James Baumhardt, Brendon Werner, Ben Saunders, Brendan Kinder

USB File Owners:
Alex Sobiesczyk, Clayton Wenzel

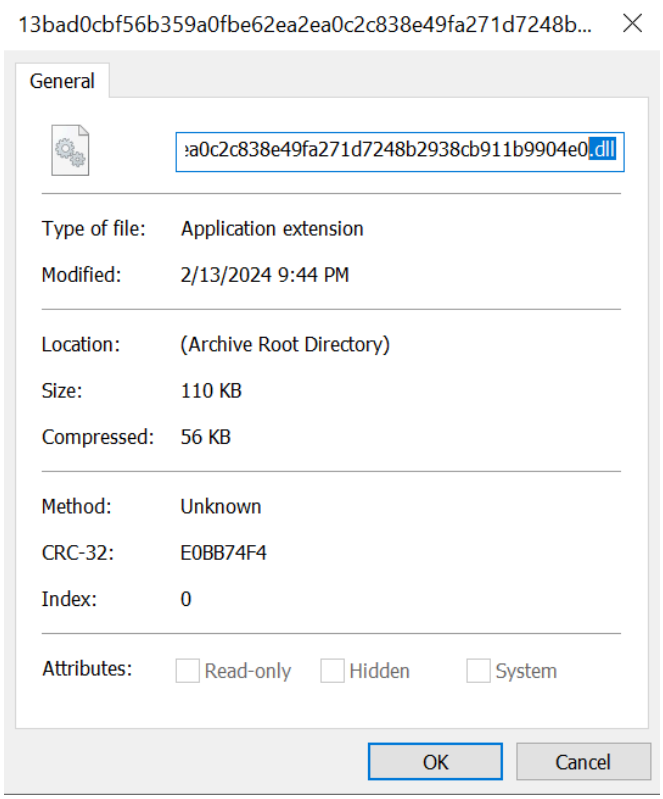
INTRO

Image of file in directory:

“Application extension?” 110kb size

Name	Type	Compressed size	Password pr...	Size	Ratio
13bad0cbf56b359a0fbe62ea2ea0c2c838...	Application extension	56 KB	Yes	110 KB	50%

^hash code? “name”



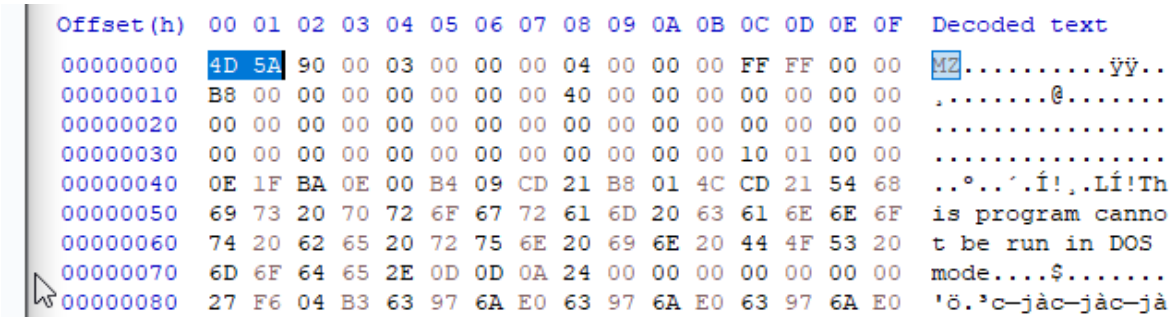
The properties Button tells us that the name of the file ends with a .dll.

This is interesting information.

IN-DEPTH BELOW

STATIC ANALYSIS

Determine the file type:
I was able to unzip the compressed zip file the file was located in with winrar. The entire time I was doing it, Windows Defender was blaring messages at me saying it was dangerous to copy. I had to turn off my defender in order to put the file into HxD. I put the file inside of HxD and I found that the file signature is 4D 5A with MZ so we know that this "application extension" is actually an executable file. → DLL extension



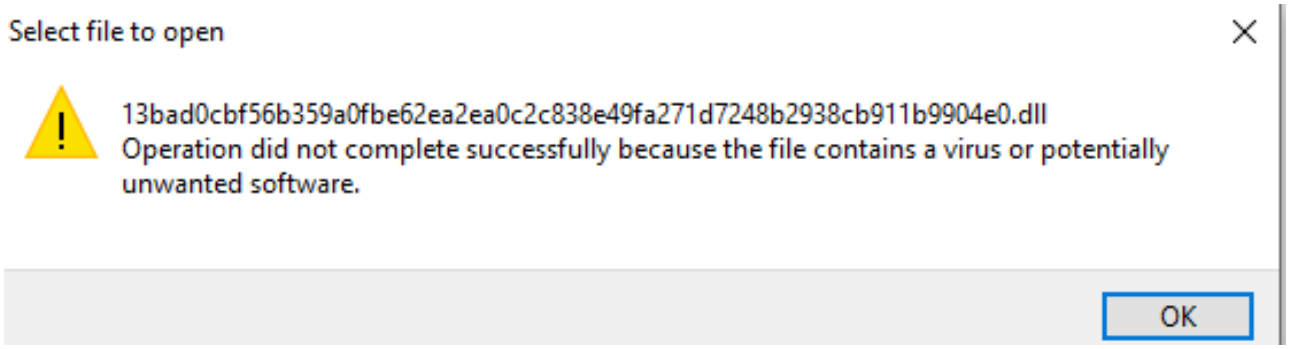
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.......@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	10	01	00	00	.....
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..º..´.Í!¸.LÍ!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	27	F6	04	B3	63	97	6A	E0	63	97	6A	E0	63	97	6A	E0	'ö.³c—jàc—jàc—jà

Fingerprinting the malware:
I was able to generate the hashes from the .dll file using the HashMyFiles Program.

Filename	MD5	SHA1	CRC32
13bad0cbf56b359a0fb...	dbd964c5bacfeccb4182e6c740f70916	e2d3b6d42fd41d890632636cca32d6cb6cdb3d5a	e0bb74f4

MD5 HASH: dbd964c5bacfeccb4182e6c740f70916

Anti-virus signatures:
Windows Defender says this is a bad/Malicious file.



Extracting Strings:

I used the Windows Strings64.exe analyzer to analyze the strings inside of the .dll file. I output the strings to a text file using Powershell.

These strings I found seem to be suspicious… Cripple.dll? I think it is doing something with memory here because it is talking about HEAP allocation. It also sends and opens requests via HTTP.

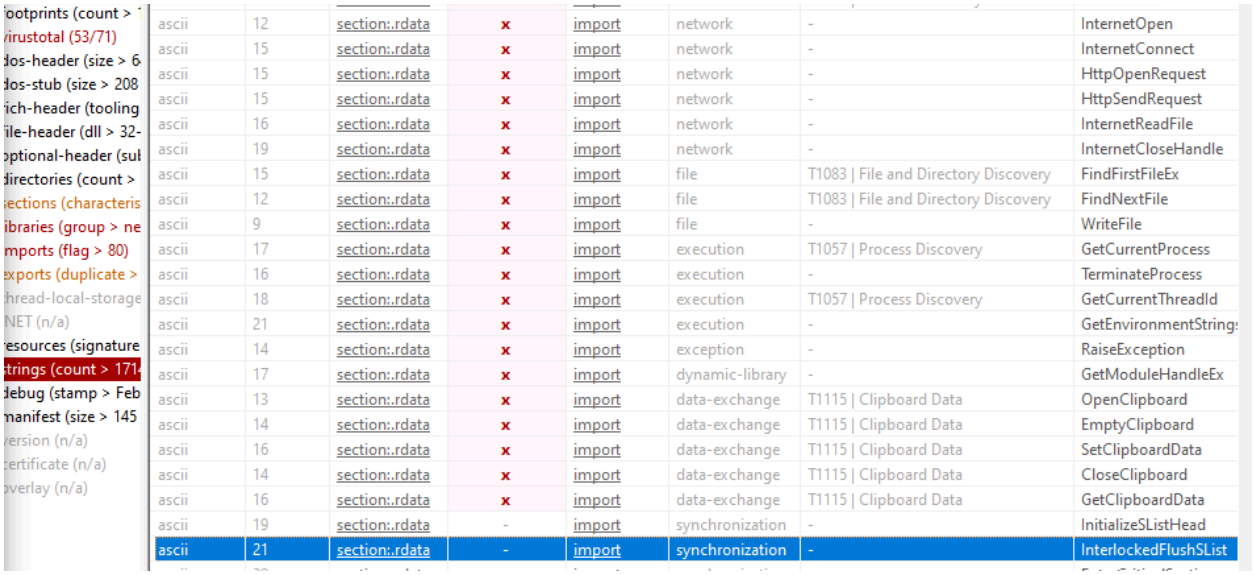
```
CLIPPERDLL.dll
??4CClipperDLL@@QAEAAV0@$$QAV0@@Z
??4CClipperDLL@@QAEAAV0@ABV0@@Z
Main
GlobalAlloc
GlobalLock
GlobalUnlock
WideCharToMultiByte
Sleep
KERNEL32.dll
OpenClipboard
EmptyClipboard
SetClipboardData
CloseClipboard
GetClipboardData
USER32.dll
InternetOpenW
InternetConnectA
HttpOpenRequestA
HttpSendRequestA
InternetReadFile
```

```
HeapAlloc
HeapFree
FindClose
FindFirstFileExW
FindNextFileW
IsValidCodePage
GetACP
GetOEMCP
GetCPInfo
GetCommandLineA
GetCommandLineW
MultiByteToWideChar
GetEnvironmentStringsW
FreeEnvironmentStringsW
LCMapStringW
GetProcessHeap
GetStdHandle
GetFileType
GetStringTypeW
HeapSize
HeapReAlloc
SetStdHandle
FlushFileBuffers
WriteFile
GetConsoleCP
GetConsoleMode
SetFilePointerEx
CreateFileW
CloseHandle
WriteConsoleW
DecodePointer
```

I also used PEStudio to analyze the strings differently to see what kind of functionality the .dll does.

indicator (30)	detail	level
virustotal > score	53/71	+++++
file > characteristics	cannot be executed	+++++
entry-point > invalid	0x0000664C	+++++
groups > API	memory | execution | console | file | exception | reconnaissance | dynamic...	+++++
libraries > flag	Internet Extensions for Win32 Library (WININET.dll)	+++++
mitre > technique	T1497 | T1057 | T1082 | T1124 | T1106 | T1083 | T1115	+++++
optional-header > size-of-code	0 bytes	++
size-of-headers > suspicious	0x010FE210	++
sections > virtualized	.reloc	++
imports > flag	21	++
exports > duplicates > count	2	++

PE Studio Strings Continued:



ascii	12	section:.rdata	x	import	network	-	InternetOpen
ascii	15	section:.rdata	x	import	network	-	InternetConnect
ascii	15	section:.rdata	x	import	network	-	HttpOpenRequest
ascii	15	section:.rdata	x	import	network	-	HttpSendRequest
ascii	16	section:.rdata	x	import	network	-	InternetReadFile
ascii	19	section:.rdata	x	import	network	-	InternetCloseHandle
ascii	15	section:.rdata	x	import	file	T1083 | File and Directory Discovery	FindFirstFileEx
ascii	12	section:.rdata	x	import	file	T1083 | File and Directory Discovery	FindNextFile
ascii	9	section:.rdata	x	import	file	-	WriteFile
ascii	17	section:.rdata	x	import	execution	T1057 | Process Discovery	GetCurrentProcess
ascii	16	section:.rdata	x	import	execution	-	TerminateProcess
ascii	18	section:.rdata	x	import	execution	T1057 | Process Discovery	GetCurrentThreadId
ascii	21	section:.rdata	x	import	execution	-	GetEnvironmentStrings
ascii	14	section:.rdata	x	import	exception	-	RaiseException
ascii	17	section:.rdata	x	import	dynamic-library	-	GetModuleHandleEx
ascii	13	section:.rdata	x	import	data-exchange	T1115 | Clipboard Data	OpenClipboard
ascii	14	section:.rdata	x	import	data-exchange	T1115 | Clipboard Data	EmptyClipboard
ascii	16	section:.rdata	x	import	data-exchange	T1115 | Clipboard Data	SetClipboardData
ascii	14	section:.rdata	x	import	data-exchange	T1115 | Clipboard Data	CloseClipboard
ascii	16	section:.rdata	x	import	data-exchange	T1115 | Clipboard Data	GetClipboardData
ascii	19	section:.rdata	-	import	synchronization	-	InitializeSListHead
ascii	21	section:.rdata	-	import	synchronization	-	InterlockedFlushSList

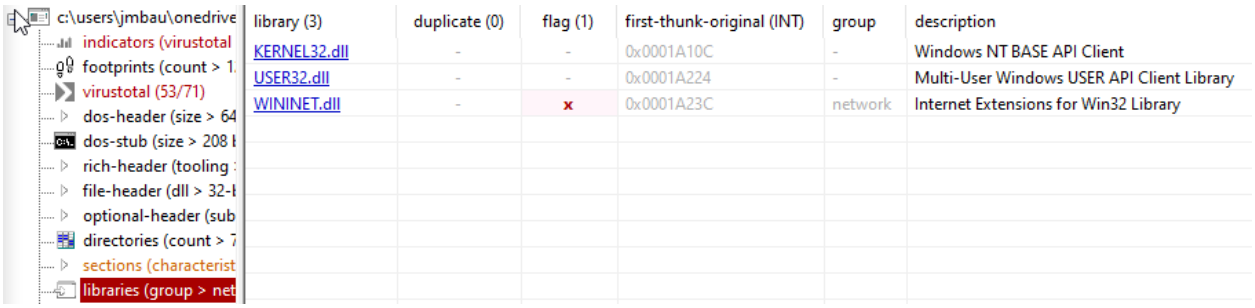
Determining File obfuscation:

We do not think it is obfuscated. The strings are clearly visible.

Inspecting PE header information:

I also used PEStudieo to look at PE head information.

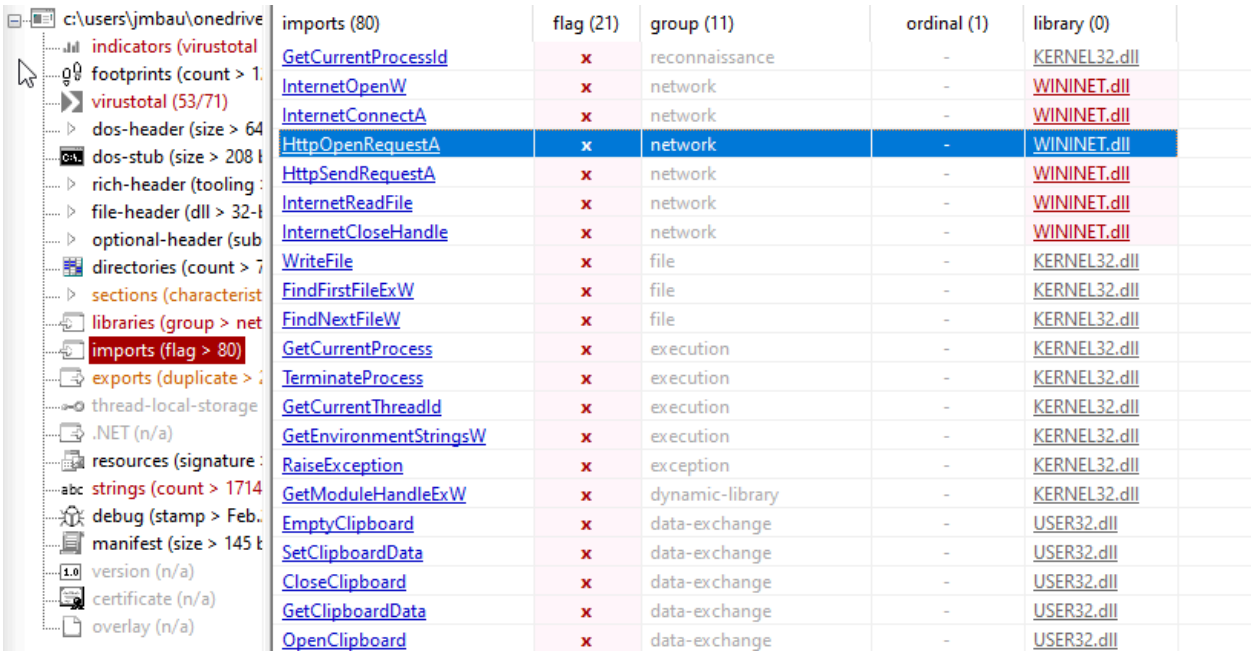
We looked at the imports and exports in which the executable was calling from the .dll files. These are the libraries the executable uses.



library (3)	duplicate (0)	flag (1)	first-thunk-original (INT)	group	description
KERNEL32.dll	-	-	0x0001A10C	-	Windows NT BASE API Client
USER32.dll	-	-	0x0001A224	-	Multi-User Windows USER API Client Library
WININET.dll	-	x	0x0001A23C	network	Internet Extensions for Win32 Library

DLL	Description
Kernel32.dll	This is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.
Advapi32.dll	This DLL provides access to advanced core Windows components such as the Service Manager and Registry.
User32.dll	This DLL contains all the user-interface components, such as buttons, scroll bars, and components for controlling and responding to user actions.
Gdi32.dll	This DLL contains functions for displaying and manipulating graphics.
Ntdll.dll	This DLL is the interface to the Windows kernel. Executables generally do not import this file directly, although it is always imported indirectly by *Kernel32.dll*. If an executable imports this file, it means that the author intended to use functionality not normally available to Windows programs. Some tasks, such as hiding functionality or manipulating processes, will use this interface.
WSock32.dll and *Ws2_32.dll*	These are networking DLLs. A program that accesses either of these most likely connects to a network or performs network-related tasks.
Wininet.dll	This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP, and NTP.

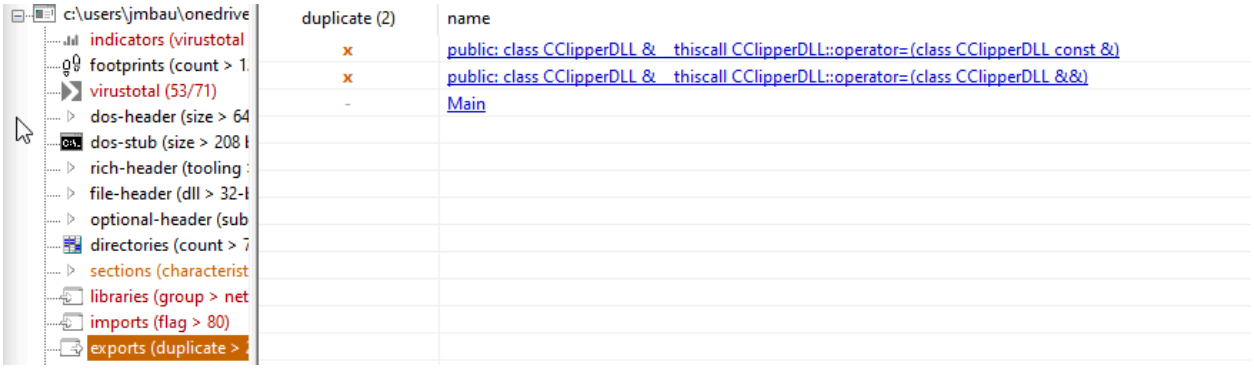
Here are all of the commands that the executable is importing to do its function.



c:\users\jmbau\onedrive
- indicators (virustotal
- footprints (count > 1
- virustotal (53/71)
- dos-header (size > 64
- dos-stub (size > 208 b
- rich-header (tooling
- file-header (dll > 32-b
- optional-header (sub
- directories (count > 7
- sections (characterist
- libraries (group > net
- imports (flag > 80)
- exports (duplicate > 2
- thread-local-storage
- .NET (n/a)
- resources (signature
- strings (count > 1714
- debug (stamp > Feb
- manifest (size > 145 b
- version (n/a)
- certificate (n/a)
- overlay (n/a)

imports (80)	flag (21)	group (11)	ordinal (1)	library (0)
GetCurrentProcessId	x	reconnaissance	-	KERNEL32.dll
InternetOpenW	x	network	-	WININET.dll
InternetConnectA	x	network	-	WININET.dll
HttpOpenRequestA	x	network	-	WININET.dll
HttpSendRequestA	x	network	-	WININET.dll
InternetReadFile	x	network	-	WININET.dll
InternetCloseHandle	x	network	-	WININET.dll
WriteFile	x	file	-	KERNEL32.dll
FindFirstFileExW	x	file	-	KERNEL32.dll
FindNextFileW	x	file	-	KERNEL32.dll
GetCurrentProcess	x	execution	-	KERNEL32.dll
TerminateProcess	x	execution	-	KERNEL32.dll
GetCurrentThreadId	x	execution	-	KERNEL32.dll
GetEnvironmentStringsW	x	execution	-	KERNEL32.dll
RaiseException	x	exception	-	KERNEL32.dll
GetModuleHandleExW	x	dynamic-library	-	KERNEL32.dll
EmptyClipboard	x	data-exchange	-	USER32.dll
SetClipboardData	x	data-exchange	-	USER32.dll
CloseClipboard	x	data-exchange	-	USER32.dll
GetClipboardData	x	data-exchange	-	USER32.dll
OpenClipboard	x	data-exchange	-	USER32.dll

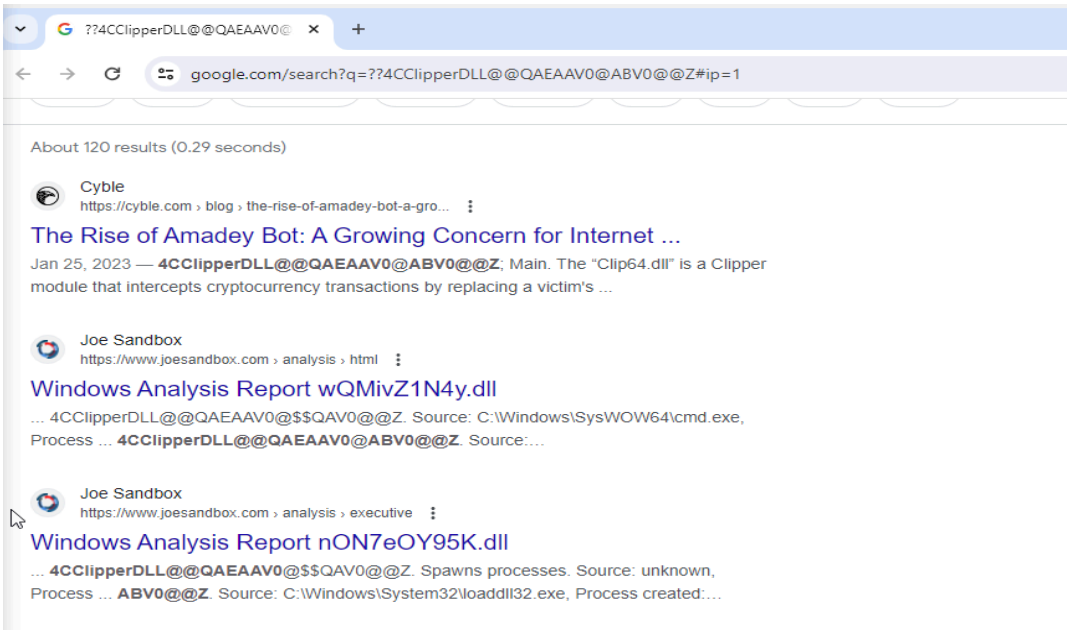
Here are the exports



c:\users\jmbau\onedrive
- indicators (virustotal
- footprints (count > 1
- virustotal (53/71)
- dos-header (size > 64
- dos-stub (size > 208 b
- rich-header (tooling
- file-header (dll > 32-b
- optional-header (sub
- directories (count > 7
- sections (characterist
- libraries (group > net
- imports (flag > 80)
- exports (duplicate > 2

duplicate (2)	name
x	public: class CClipperDLL & __thiscall CClipperDLL::operator=(class CClipperDLL const &)
x	public: class CClipperDLL & __thiscall CClipperDLL::operator=(class CClipperDLL &&)
-	Main

Googling that link ClipperDll:

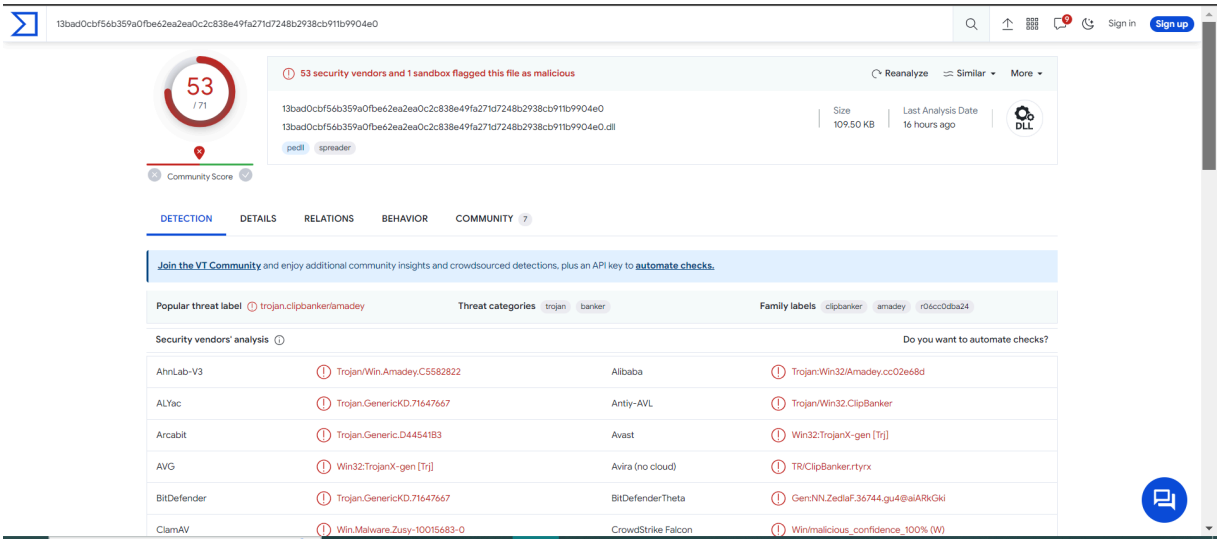


Compare and classify malware

We put the MD5 hash into virustotal.com and got this:

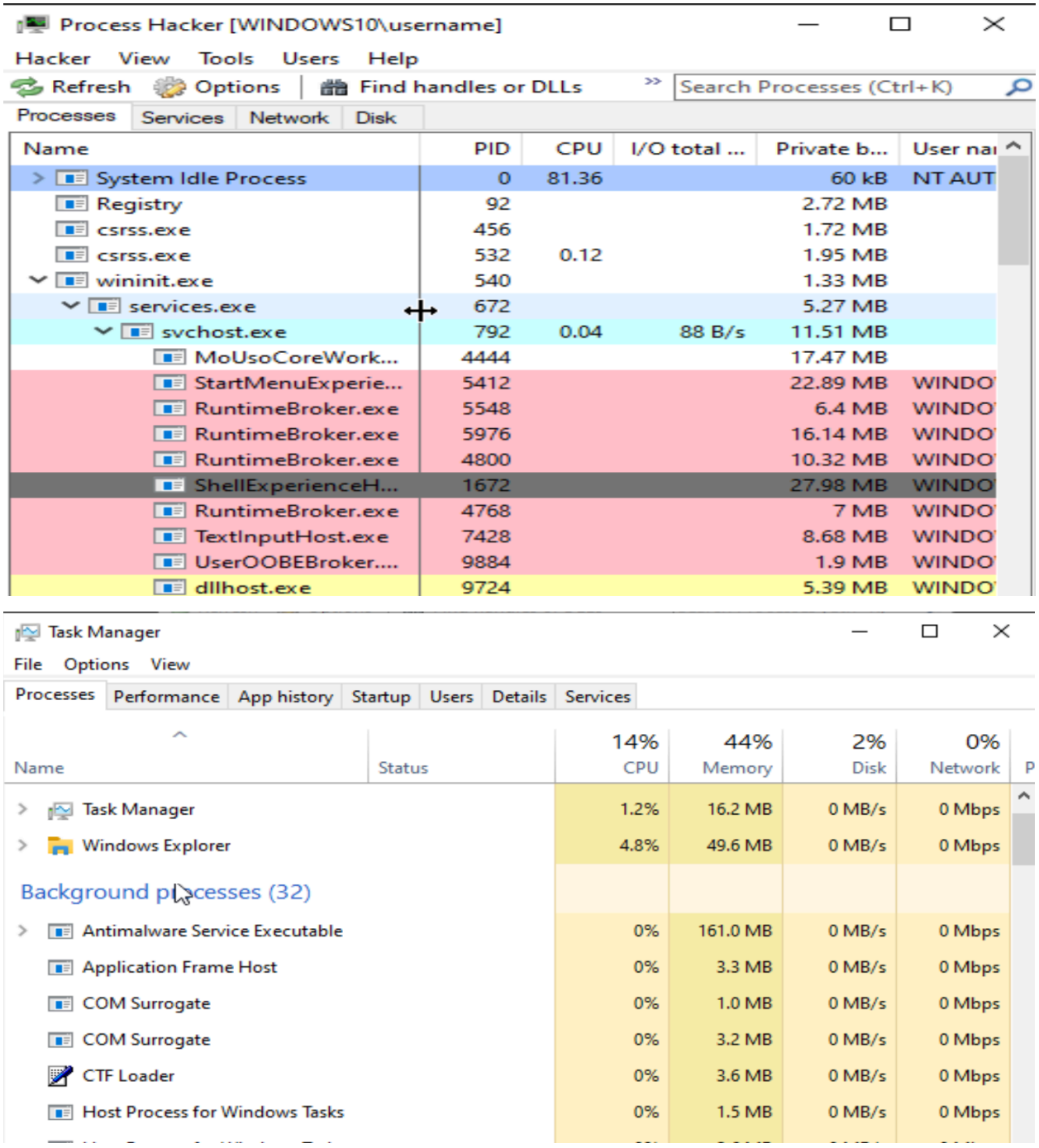
Link:

https://www.virustotal.com/gui/file/13bad0cbf56b359a0fbe62ea2ea0c2c838e49fa271d7248b2938cb911b9904e0/detection

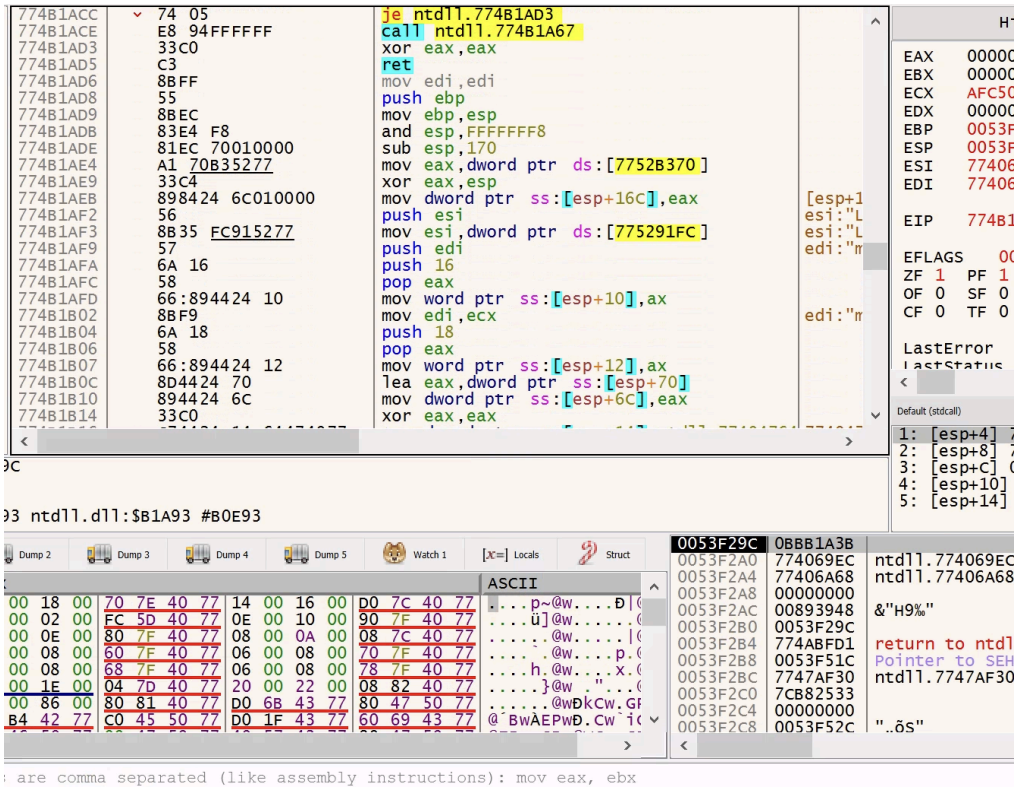


This malware can be classified as a trojan. It clearly is not an application extension even though in the directory it says it is. We think it might be some sort of crypto miner because of the ClipBanker names inside of the files.

DYNAMIC ANALYSIS



32dbg


774B1ACC 74 05 je ntdll.774B1AD3
774B1ACE E8 94FFFFFF call ntdll.774B1A67
774B1AD3 33C0 xor eax,eax
774B1AD5 C3 ret
774B1AD6 8BFF mov edi,edi
774B1AD8 55 push ebp
774B1AD9 8BEC mov ebp,esp
774B1ADB 83E4 F8 and esp,FFFFFFF8
774B1ADE 81EC 70010000 sub esp,170
774B1AE4 A1 70B35277 mov eax,dword ptr ds:[7752B370]
774B1AE9 33C4 xor eax,esp
774B1AEB 898424 6C010000 mov dword ptr ss:[esp+16C],eax
774B1AF2 56 push esi
774B1AF3 8B35 FC915277 mov esi,dword ptr ds:[775291FC]
774B1AF9 57 push edi
774B1AFA 6A 16 push 16
774B1AFC 58 pop eax
774B1AFD 66:894424 10 mov word ptr ss:[esp+10],ax
774B1B02 8BF9 mov edi,ecx
774B1B04 6A 18 push 18
774B1B06 58 pop eax
774B1B07 66:894424 12 mov word ptr ss:[esp+12],ax
774B1B0C 8D4424 70 lea eax,dword ptr ss:[esp+70]
774B1B10 894424 6C mov dword ptr ss:[esp+6C],eax
774B1B14 33C0 xor eax,eax
EAX
EBX
ECX
EDX
EBP 0053F
ESP 0053F
ESI 77406
EDI 77406
EIP 774B1
EFLAGS
ZF 1 PF 1
OF 0 SF 0
CF 0 TF 0
LastError
LastStatus
Default (stdcall)
1: [esp+4]
2: [esp+8]
3: [esp+C]
4: [esp+10]
5: [esp+14]
93 ntdll.dll:$B1A93 #B0E93
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
[x=] Locals
Struct
ASCII
0053F29C 0BBB1A3B
0053F2A0 774069EC ntdll.774069EC
0053F2A4 77406A68 ntdll.77406A68
0053F2A8 00000000
0053F2AC 00893948 &"H9‰"
0053F2B0 0053F29C
0053F2B4 774ABFD1 return to ntdl
0053F2B8 0053F51C Pointer to SEH
0053F2BC 7747AF30 ntdll.7747AF30
0053F2C0 7CB82533
0053F2C4 00000000
0053F2C8 0053F52C
are comma separated (like assembly instructions): mov eax, ebx