Practical Assignment 2 Report

03/17/2021

Justin Miller: miller.j@ufl.edu

Assignment P0x02

CIS4930-108A(2506)

Table of Contents

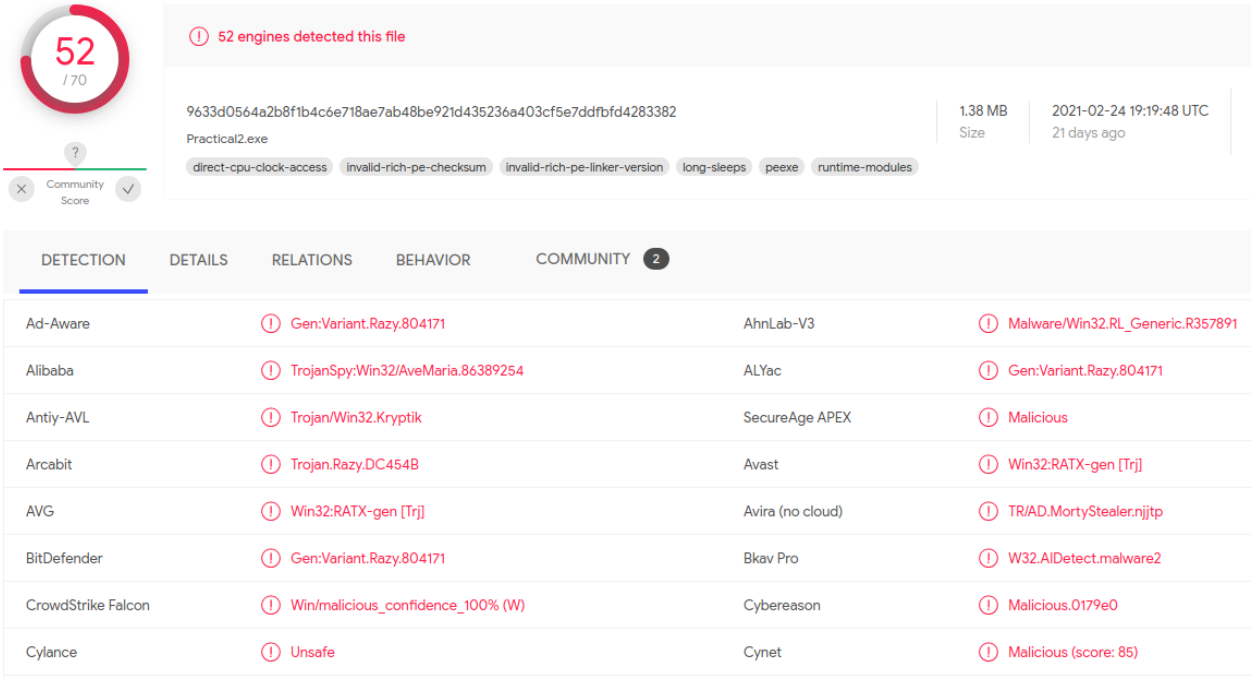
Executive Summary

The provided malware sample appears to a trojan, specifically it seems to be a launcher for the Ave Maria Trojan. On launch a terminal window appears and quickly closes and later the process creates a second thread. The malware repeatedly attempts to connect to a Command and Control(C&C) server over TCP using address 195.140.214.82. Based on its filesystem activity it appears to require being in the C:\Windows\System32 directory to be able to find required dlls. There is some interesting registry activity especially a modification to the maximum allowed internet connections, adding a value to the shellex registry, and seemingly unused key created in the CurrentVersion\Explorer registry. Additionally, there are a lot of registry values found in strings that suggest malicious registry activity. Using the imports VirtualAlloc and VirtualProtect, the malware uses memory allocation to create a memory location where it can write to using XOR decoding to de-obfuscate its contents and execute from (shellcode).

Static Analysis

Virus Total

After some basic static analysis was done, the malware sample was hashed and uploaded to VirusTotal. VirusTotal has a large database of malware samples with hashes, information about behavior, and a lot of other useful information. VirusTotal will run uploaded programs against many malware detection engines and show their results. For this sample, 52 of 70 engines detected this file as malicious, with quite a few of them identifying it as being either the Kryptic trojan or the AveMaria trojan.



VirusTotal provides a list of many common hashes which can be used as a unique identifier for this malware sample.

Basic Properties

MD5	971a3320179e0494fdb70b138ada2446
SHA-1	b04bba3b8be297b6178e73d10f0380897e76464c
SHA-256	9633d0564a2b8f1b4c6e718ae7ab48be921d435236a403cf5e7ddfbfd4283382
Vhash	016086551d551d1515156045z200677z90baz1bfz
Authentihash	70e203d43f38c128fed15c70ec8479eb5989ab63f535bf190d9c1d54847e9b45
Imphash	f396b39dbfa473ab2b7180d955fdc740
Rich PE header hash	94adf1b937a03026d0489a22843d03cc
SSDEEP	12288:hkhSL4pH7FYiliicuueTh9yeJWrpDz29Wa+QB1t6gMvlTpa6NYjHhtkaJN:h72Z/8VWrpn2ZF1Ea1jBH
TLSH	T1E3654A07A7628111FCFE12F774FFB2B8552CABA2279892C352C5A5ECA7055F06D30E52

Basic Static Analysis

An initial analysis of the program was done using PEStudio which is used to examine portable executable files. The malware has a compiler stamp showing that the malware was compiled on December 9th, 2020, this value may have been modified but considering how recent the date is it unlikely.

compiler-stamp	0x5FD1540F (Wed Dec 09 17:47:43 2020)

PEStudio identifies that this program is a console application, rather than being a GUI application.

subsystem	console

Also, worth noting is that this file is written in C++, was compiled in debug mode, and is 32-bit.

signature	Microsoft Visual C++ 8.0 Debug
entry-point	E9 17 B4 01 00 E9 D2 FC 01 00 E9 BD 0F 0A 00 E9 18 B7 08 00 E9 A3 2D 02 00 E9 0E 5B 07 00 E9 B9 E9
file-version	n/a
description	n/a
file-type	**executable**
cpu	**32-bit**
signature	Microsoft Visual C++ 8.0 Debug

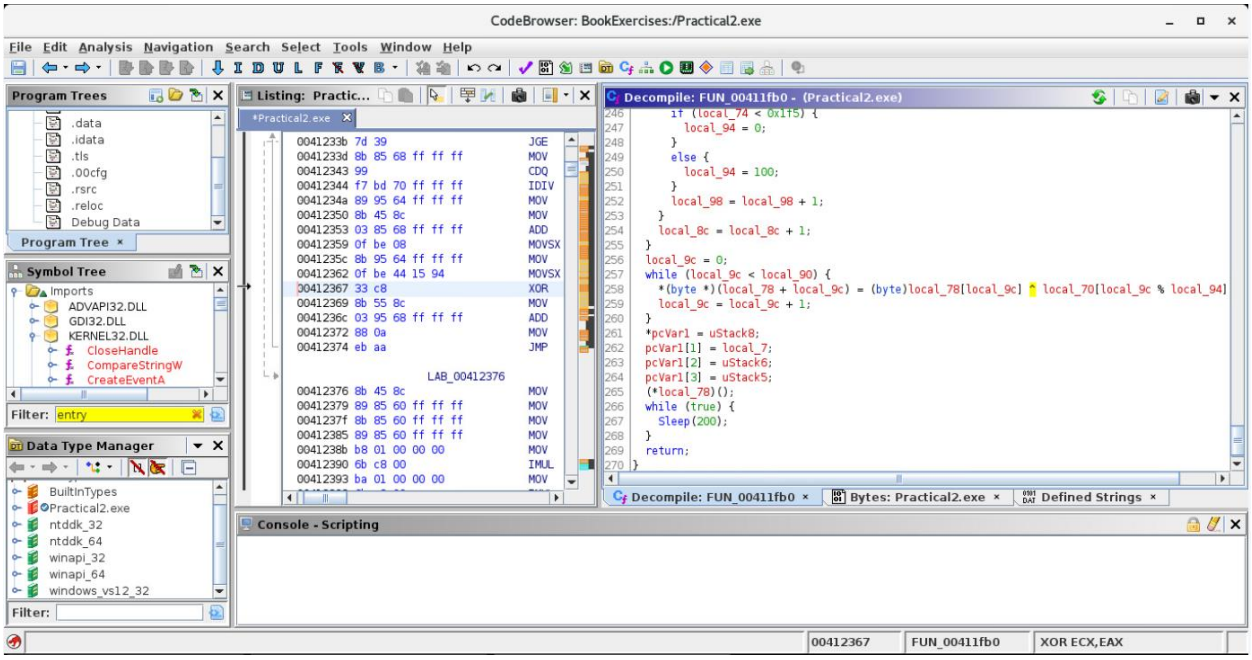
Packing and Obfuscation

Programs sections are likely to have been packed when their sections have an entropy between 7 and 8. The number of imports and strings is another indicator of packing, packed programs tend to have few imports and strings. None of the sections come close to the entropy range which identifies packing and there are many imports and strings, it is unlikely that this program has been packed.

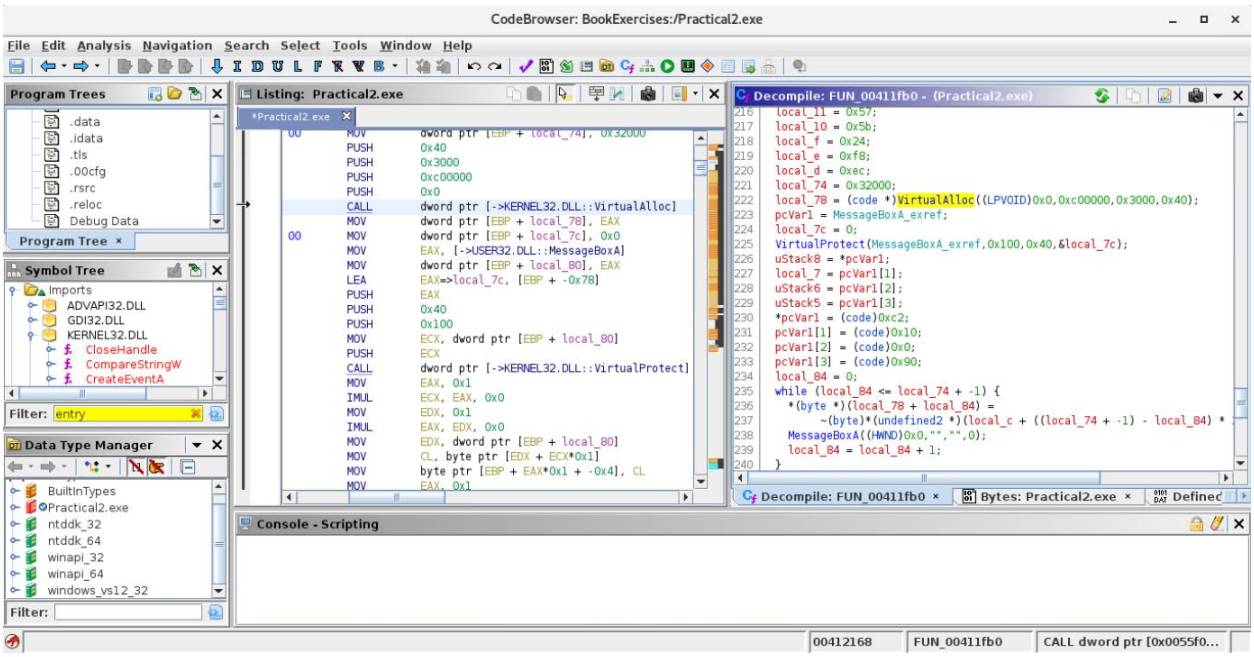
property	value	value	value	value
name	.text	.rdata	.data	.idata
md5	F4652CB5BEC6D3B287599D...	5BEE6DE30D069254DD3473D...	A164260C7E639AC06B537C2...	B93F40DB6CFB5866381BC50...
entropy	5.417	3.656	4.611	4.583
file-ratio (99.93%)	51.27 %	11.75 %	34.47 %	0.39 %
raw-address	0x00000400	0x000B5600	0x000DEE00	0x00158A00
raw-size (1445888 bytes)	0x000B5200 (741888 bytes)	0x00029800 (169984 bytes)	0x00079C00 (498688 bytes)	0x00001600 (5632 bytes)
virtual-address	0x00401000	0x004B7000	0x004E1000	0x0055F000
virtual-size (1460414 bytes)	0x000B5071 (741489 bytes)	0x000297DC (169948 bytes)	0x0007DB5C (514908 bytes)	0x0000150E (5390 bytes)
entry-point	0x000045D4	-	-	-
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040
writable	-	-	x	-
executable	x	-	-	-

value	value	value	value
.tls	.00cfg	.rsrc	.reloc
C573BD7CEA296A9C5D230C...	B113A4C4A7E762786CC1415...	EEA80AA728A3053C9C54C2...	4FDD24485D60B33CE44EB3...
0.011	0.061	2.141	6.094
0.07 %	0.04 %	0.11 %	1.84 %
0x0015A000	0x0015A400	0x0015A600	0x0015AC00
0x00000400 (1024 bytes)	0x00000200 (512 bytes)	0x00000600 (1536 bytes)	0x00006800 (26624 bytes)
0x00561000	0x00562000	0x00563000	0x00564000
0x00000309 (777 bytes)	0x00000104 (260 bytes)	0x0000043C (1084 bytes)	0x000067BE (26558 bytes)
-	-	-	-
0xC0000040	0x40000040	0x40000040	0x42000040
x	-	-	-
-	-	-	-

A closer look at the program using Ghidra to disassemble the executable reveals this while loop which appears to be performing XOR decoding and writing the result into a memory location.



The memory address being written to, local_78, is virtually allocated at the start of this function.



Program Sections, Strings, and Imports

PEStudio recognizes several program sections (see screenshots for packing), which provides some interesting information. Immediately noticeable is the .00cfg section, which seems to be a file for Control Flow Guard which is used to prevent memory corruption vulnerabilities. It seems strange for this to be included in malware but viewing the section in Ghidra confirms that is all it is there for.

The program contains a .tls section, however it does not have an entry point and is effectively empty meaning there is no TLS Callback being executed before the original entry point (OEP). The .rsrc section is also basically empty. There are three different data sections: .rdata, .data, and .idata. The .rdata section contains a lot of random constant strings, stuff like language indicators, some windows filesystem paths, and some function paths and names. The .data section has a lot of data but none of it is readable. The .idata section contains a lot of function imports and function pointers. The file has a large but otherwise uninteresting .reloc section.

The Strings section of PEStudio reveals a lot of visual studio paths which reveal that this program was created in Visual Studio Community Edition 2017.

type (2)	size (bytes)	file-offset	blac...	hint (171)	group...	value (4270)
ascii	6	0x000002B8	x	-	-	.00cfg
unicode	113	0x000B689A	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
unicode	112	0x000B69C9	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
unicode	113	0x000B6D52	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
ascii	114	0x000B6DF8	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
unicode	114	0x000B6F3B	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
unicode	126	0x000B70C7	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
unicode	112	0x000B7271	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...
ascii	112	0x000B7310	x	file	-	c:\program files (x86)\microsoft visual studio\2017\community\vc\tools\msvc\14.16.27023...

A particularly interesting and identifiable string is the local path to some debug information referenced by the program.

ascii	96	0x000DCF7C	x	file	-	C:\Users\W7H64\Desktop\VCSamples-master\VC2010Samples\ATL\General\AtlCon\bitcoin coinjoin op.pdb

The program has many imports which provide information about some of the potential behavior of the program. There are registry imports for close, delete, open, and add which suggest the program may modify registry keys.

RegCloseKey	registry	implicit	-	-	-	-	-	advapi32.dll
RegDeleteKeyA	registry	implicit	-	x	-	-	-	advapi32.dll
RegOpenKeyExA	registry	implicit	-	-	-	-	-	advapi32.dll
RegQueryInfoKeyA	registry	implicit	-	-	-	-	-	advapi32.dll

There are imports for QueryPerformanceCount and IsDebuggerPresent which can both be employed as anti-debugging techniques.

name (156)	group (11)	type (1)	ordinal (11)	blacklist (23)	anti-debug (0)	undocumented (0)	deprecated (3)	library (6)
ReadFile	file	implicit	-	-	-	-	-	kernel32.dll
QueryPerformanceCount...	system-information	implicit	-	-	-	-	-	kernel32.dll
MultiByteToWideChar	-	implicit	-	-	-	-	-	kernel32.dll
MoveWindow	windowing	implicit	-	-	-	-	-	user32.dll
MessageBoxA	-	implicit	-	-	-	-	-	user32.dll
LoadLibraryExW	dynamic-library	implicit	-	-	-	-	-	kernel32.dll
LoadLibraryExA	dynamic-library	implicit	-	-	-	-	-	kernel32.dll
LeaveCriticalSection	synchronization	implicit	-	-	-	-	-	kernel32.dll
LCMapStringW	-	implicit	-	-	-	-	x	kernel32.dll
IsWindow	windowing	implicit	-	-	-	-	-	user32.dll
IsValidLocale	-	implicit	-	-	-	-	-	kernel32.dll
IsValidCodePage	-	implicit	-	-	-	-	-	kernel32.dll
IsProcessorFeaturePresent	system-information	implicit	-	-	-	-	-	kernel32.dll
IsDebuggerPresent	system-information	implicit	-	-	-	-	-	kernel32.dll

Virtual memory allocation and protection imports are present which suggest it will be worth watching the memory map while debugging to see what happens in allocated sections.

VirtualAlloc	memory	implicit	-	-	-	-	-	kernel32.dll
VirtualFree	memory	implicit	-	-	-	-	-	kernel32.dll
VirtualProtect	memory	implicit	-	x	-	-	-	kernel32.dll
VirtualQuery	memory	implicit	-	-	-	-	-	kernel32.dll

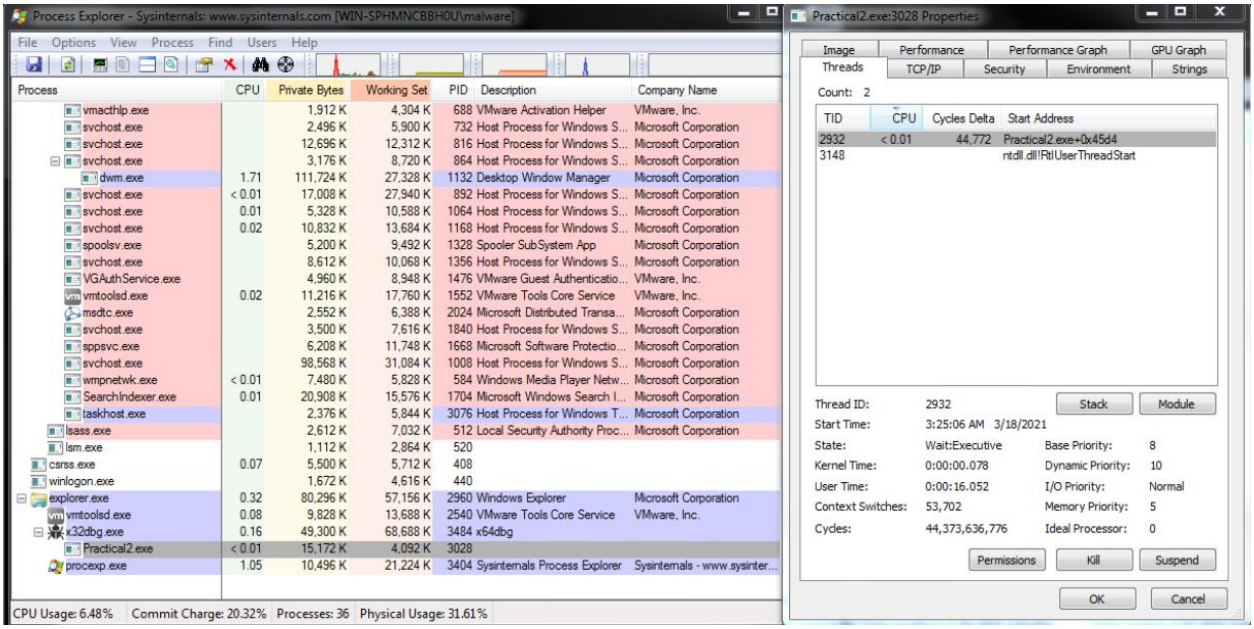
Anti-Disassembly

No anti-disassembly techniques were encountered when analyzing this program with Ghidra and Ida Pro.

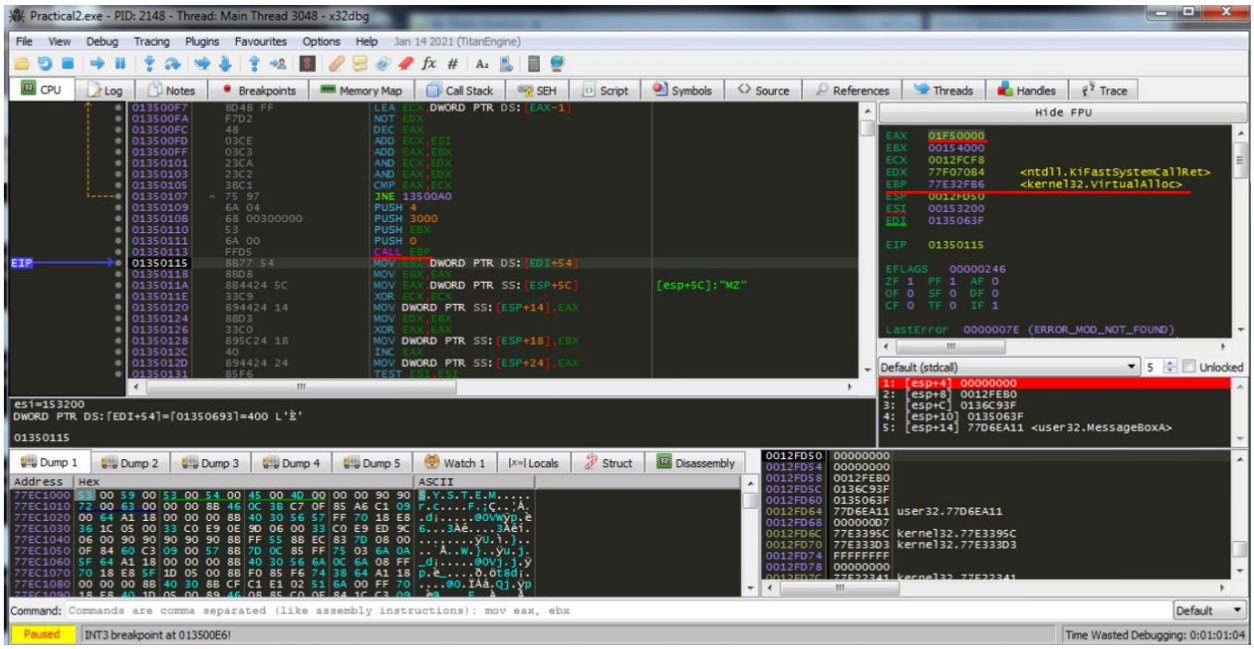
Dynamic Analysis

Process Behaviors

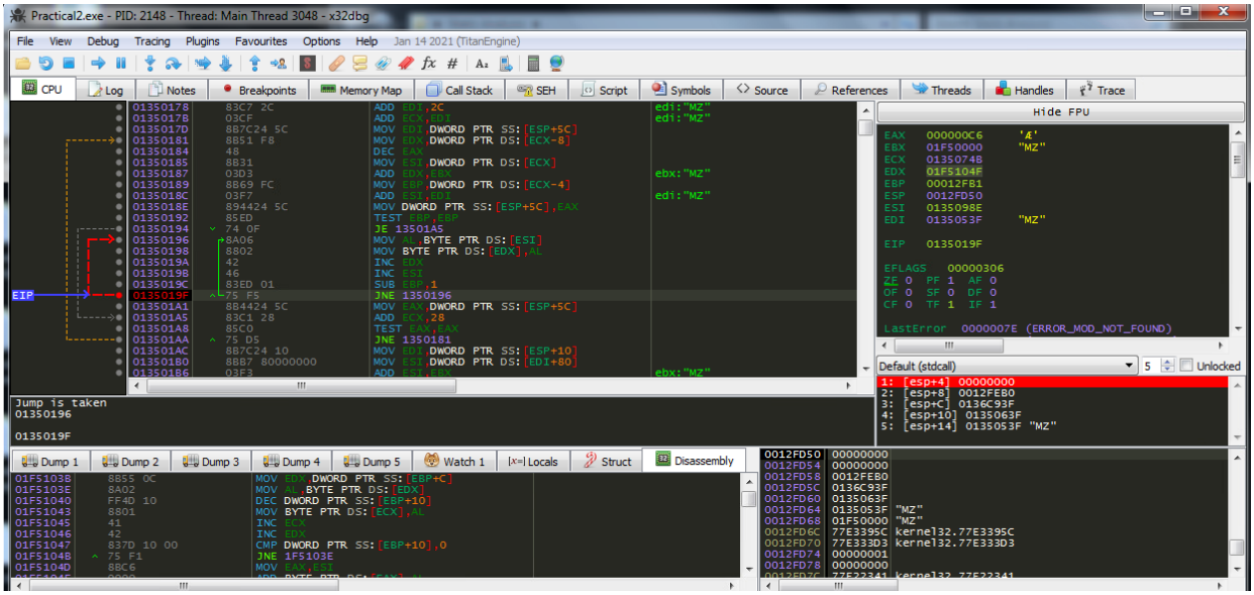
When the malware is run, a terminal window opens for a short second and closes soon thereafter. The process continues to exist until it is killed, or the computer is shutdown. The only interesting behavior seen in the properties of the process is that it eventually creates a second thread. No other processes were started by this malware.



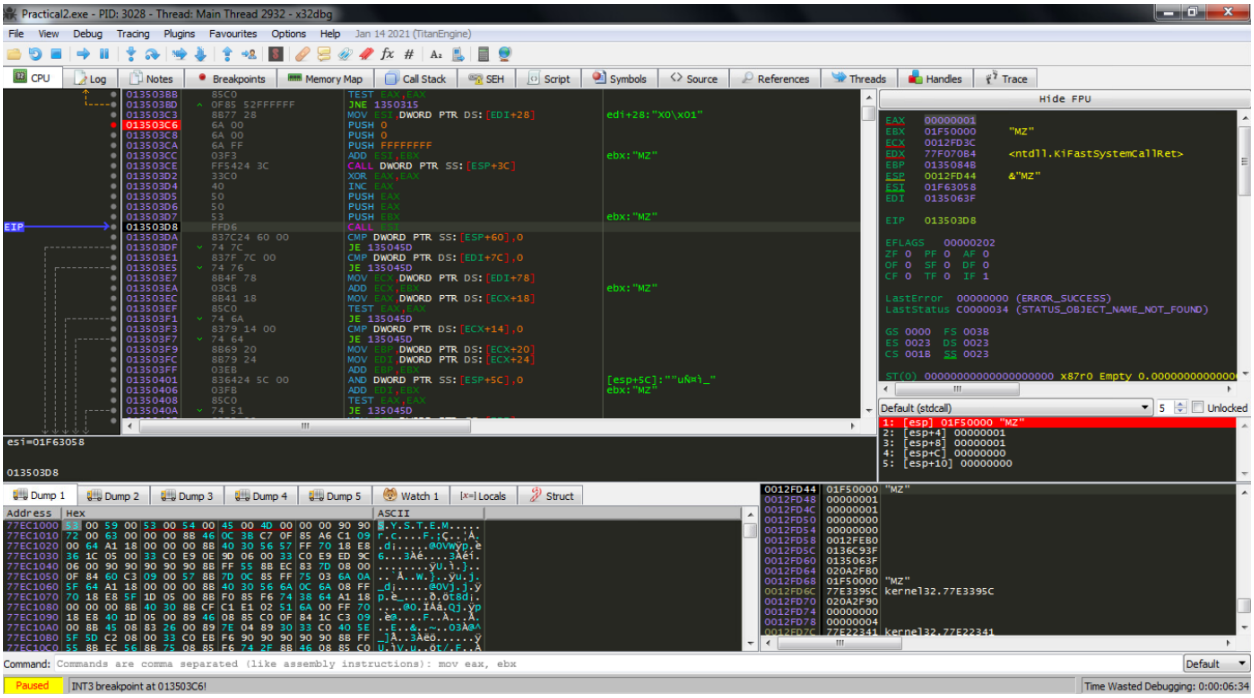
Using x32dbg it is possible to debug the malware and better observe its behavior. By stepping through the debugger until the call (004123F1) made before an infinitely looping sleep, then stepping into the next call it is possible to get to the code which allocates memory to the address in EAX which is then used to store the de-obfuscated code

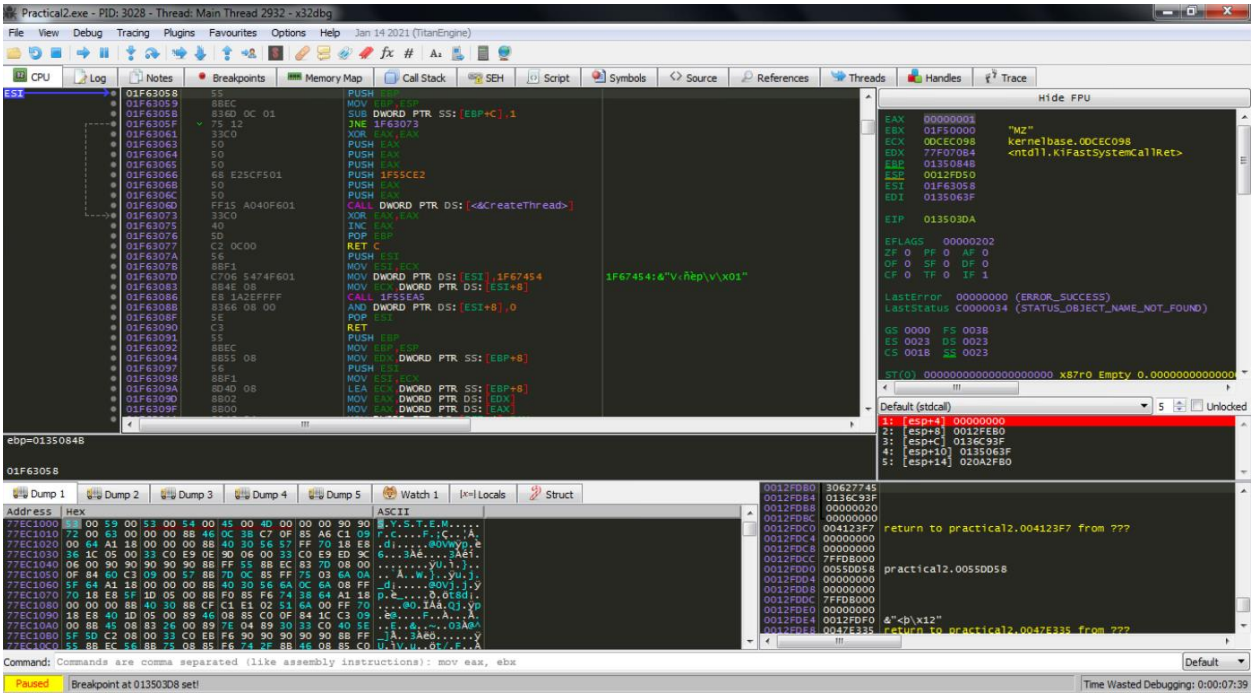


Move from ESI to AL and write from AL to EDX memory address.



After the malware is done decoding and writing to the memory, it calls to a memory location stored in ESI in that memory which creates the second thread.



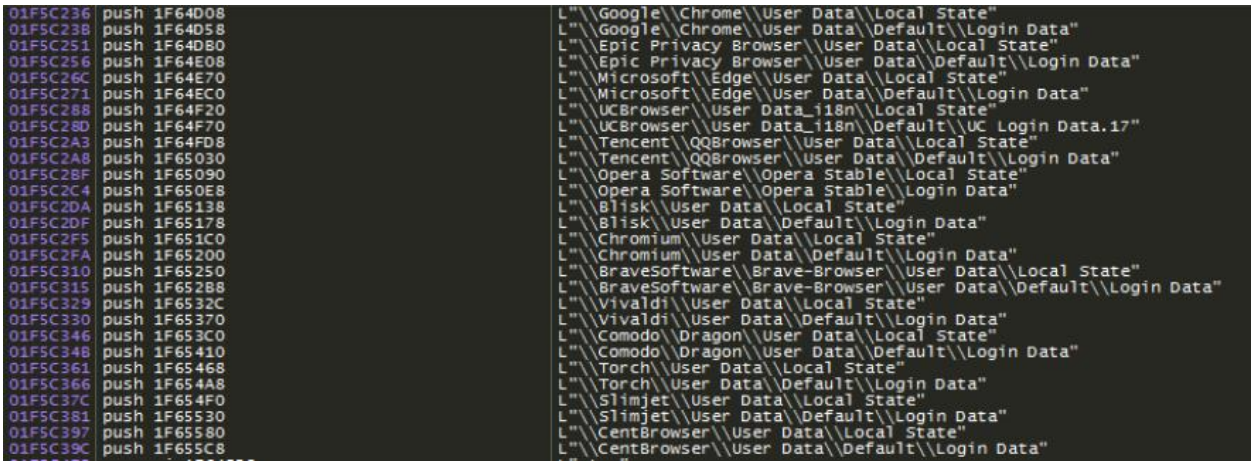


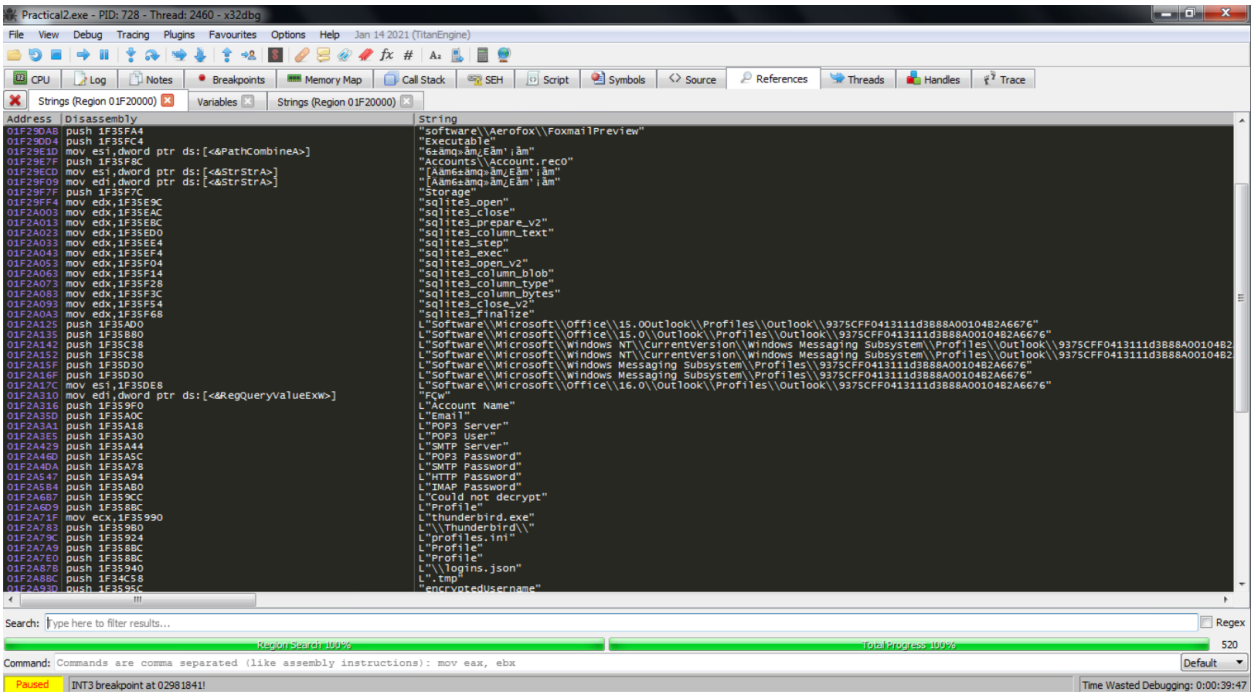
Using the Find Strings tool, it is possible to view strings in the de-obfuscated code. The strings that stand out the most being:

Address	Disassembly	String
01F60B7E	push 1F66EE0	L"Ave_Maria Stealer OpenSource github Link: https://github.com/syohex/java-simple-mine-sweeper"
01F60BAE	push 1F66FA0	L"C:\\Users\\Vitali Kremez\\Documents\\MidgetPorn\\workspace\\MsgBox.exe"

Which identify the malware as the AveMaria trojan (and gives the reward of java minesweeper), while taking a hit at a professional malware reverse engineer.

The rest of the strings identify the full capabilities of the trojan, stealing browser data, passwords, and likely connecting to a warzone C&C server.

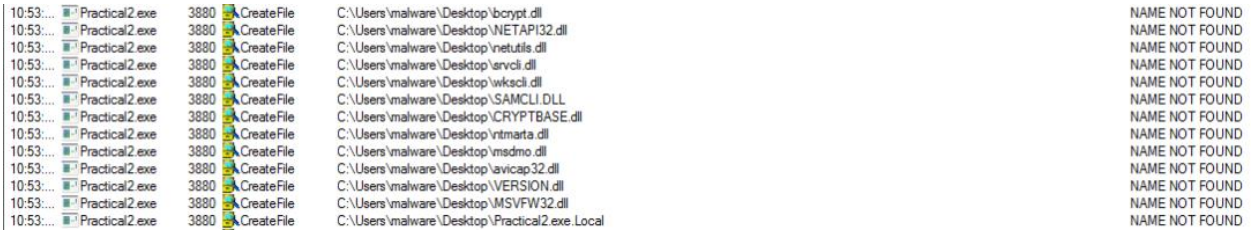




Filesystem Activity

ProcMon, which monitors and logs the behavior of programs, was used to monitor changes in the filesystem. The first time the program was run it was done from the C:/Users/{user}/Desktop directory. ProcMon logged several of CreateFile operations that resulted in “NAME NOT FOUND”, the CreateFile operation is used to access files on the filesystem. The paths of these logs suggested that the file was failing to access several dll files and was trying to find them by looking in the directory the program was run from. To resolve this, the executable was moved to

the C:/Windows/System32 folder so the executable could access the dlls. Without it being in the System32 directory, debugging failed much sooner.

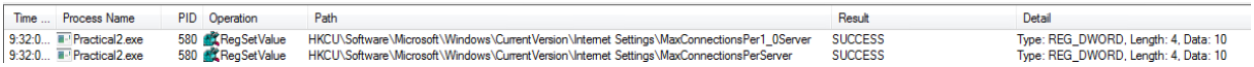


10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\bcrypt.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\NETAPI32.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\netutils.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\srvcli.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\wkscli.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\SAMCLI.DLL	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\CRYPTBASE.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\ntmarta.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\msdmo.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\avicap32.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\VERSION.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\MSVFW32.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\Practical2.exe.Local	NAME NOT FOUND

Registry Activity

RegShot was used to take a snapshot of the registry before and after running the malware. Comparing the snapshots should then show registry values that were modified by the malware. Based on the static analysis it was expected that there would be registry activity, however, RegShot was surprisingly useless in identifying any relevant activity.

ProcMon was able to pick up on a lot of registry activity that RegShot completely missed. The malware updates the value of a registry which controls the maximum number of connections to a web server.



Time ...	Process Name	PID	Operation	Path	Result	Detail
9:32:0...	Practical2.exe	580	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPer1_0Server	SUCCESS	Type: REG_DWORD, Length: 4, Data: 10
9:32:0...	Practical2.exe	580	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPerServer	SUCCESS	Type: REG_DWORD, Length: 4, Data: 10

The malware creates a registry key in HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer, the use of this registry is unclear, but the registry has a name which makes it very unique.

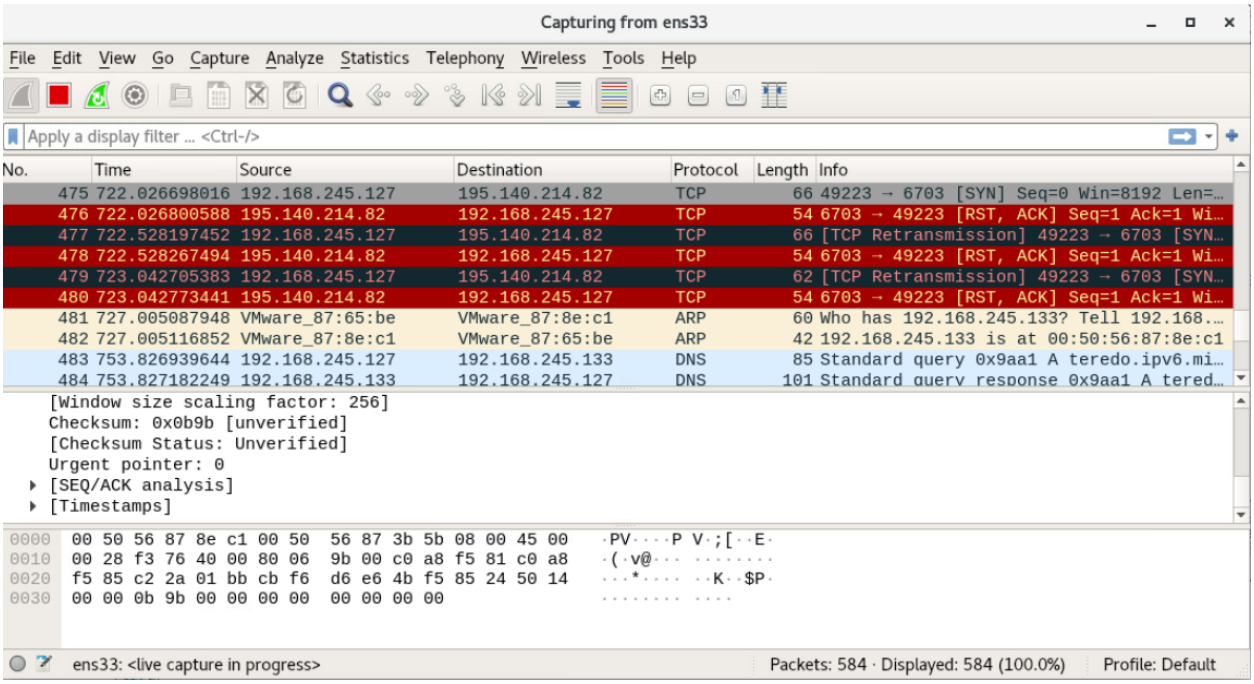
Time ...	Process Name	PID	Operation	Path	Result	Detail
9:32:0...	Practical2.exe	580	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OQWGJ0HVDJ	NAME NOT FOUND	Desired Access: All Access
9:32:0...	Practical2.exe	580	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OQWGJ0HVDJ	SUCCESS	Desired Access: Query Value, Disposition: REG_CREATED_NEW_KEY
9:32:0...	Practical2.exe	580	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OQWGJ0HVDJ	SUCCESS	

The malware also creates a key in HKCR\Drive\shellex\FolderExtensions which is for shell extension handlers.

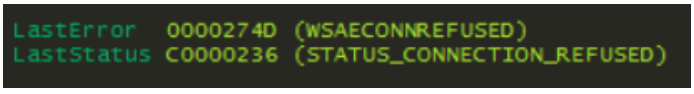
9:44:4...	Practical2.exe	1504	RegEnumKey	HKCR\Drive\shellex\FolderExtensions	SUCCESS	Index: 0, Name: {fbeb8a05-beee-4442-804e-409d6c4515e9}
9:44:4...	Practical2.exe	1504	RegEnumKey	HKCR\Drive\shellex\FolderExtensions	NO MORE ENTRIES	Index: 1, Length: 288

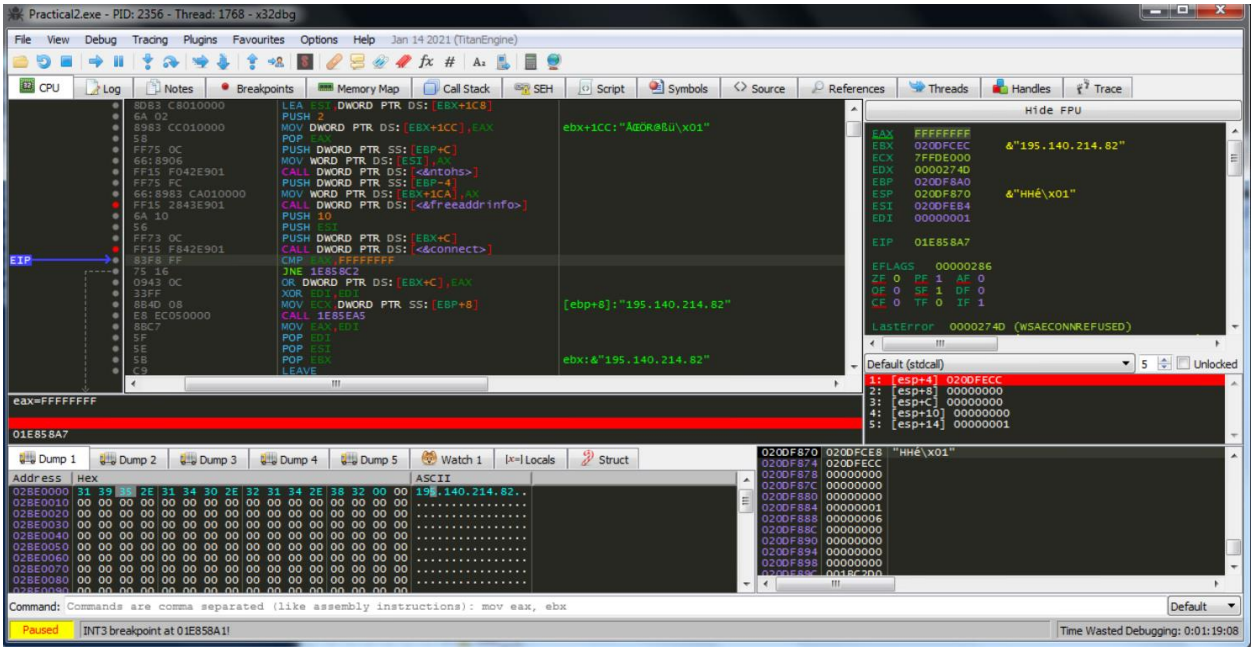
Network Activity

Network analysis was done using the accept-all-ips script included in REMnux to redirect outgoing traffic to a REMnux virtual machine on the same network as the windows VM that the malware was running on. WireShark was then used to capture and analyze the redirected network traffic. The malware attempts to establish a TCP connection with 195.140.214.82 three times before sleeping for a short amount of time and trying again.



Using x32dbg to debug the program it was possible was able to identify the connection that is seen by WireShark. Within the XOR decoded section of the program it uses the connect function from wsl_32.dll to attempt to establish a connection with 195.140.214.82. The connection errors with a STATUS_CONNECTION_REFUSED error then a jump is done to try to connect again. It is likely possible to make the program think it successfully connected and run the rest of the code; however, I was unable to do so.

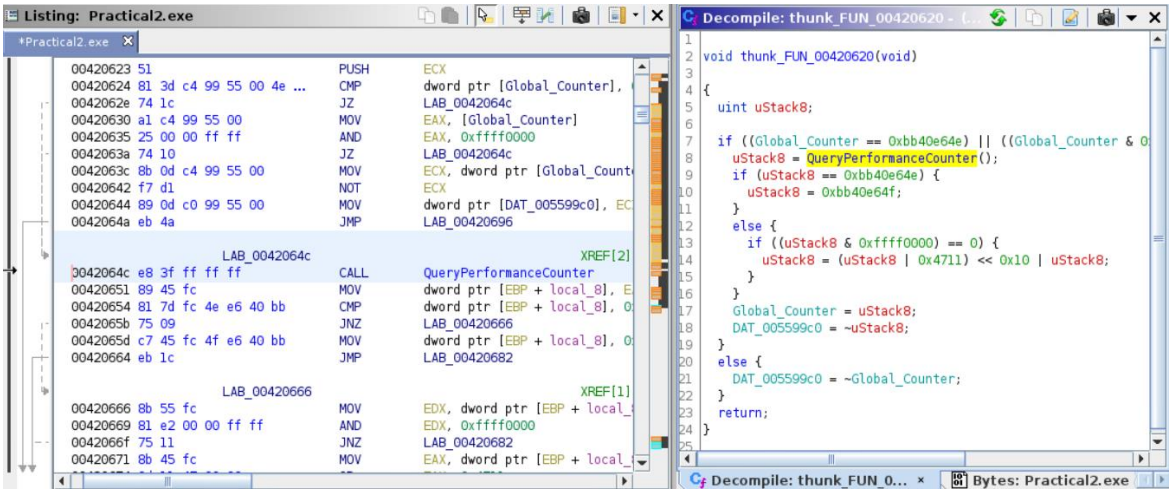


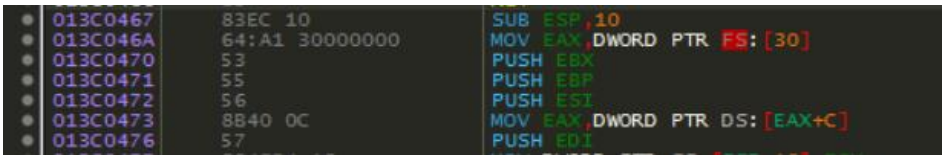


Anti-Debugging and Anti-Virtual-Machine

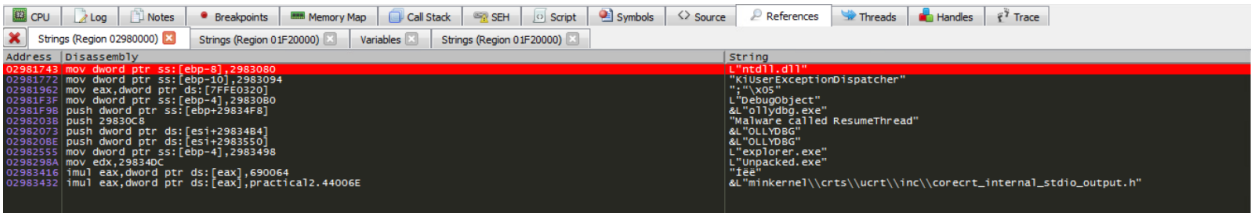
There are several anti-debugging techniques used in this malware, however, none of them were very effective. The main issues that I encountered while debugging was if I started the process in x32dbg and let it run the process would terminate without executing. This was avoidable by setting breakpoints at certain locations in the malware to get to the function call that happens right before an infinite sleep loop. By stepping into that function and the call made by it, it was possible to see the malware do the XOR decoding. Attempting to attach to the process while it was running would usually end up returning to the sleep loop. Letting it run after creating the thread to the decoded section and setting a breakpoint in the decoded section where the connection is attempted avoided the anti-debugging.

Ghidra found several places where functions and techniques related to anti-debugging are employed. None of them seemed to get in the way of debugging though. For example, QueryPerformanceCounter, and isDebuggerPresent get called and fs:[30] is checked in multiple locations but none of it prevented just stepping over the functions they get called in while debugging.





Additionally, there is a string “OLLYDBG” in some of the decoded memory, which is used to detect if ollydbg is being used.



I did not see anything that would suggest anti-VM techniques were being used.

Indicators of Compromise

Host Based:

Existence of the malware executable:

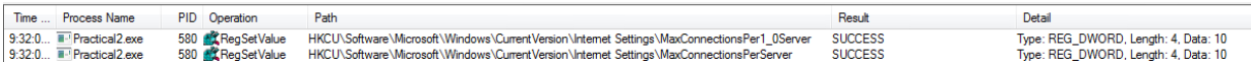
MD5 971a3320179e0494fdb70b138ada2446

SHA-1 b04bba3b8be297b6178e73d10f0380897e76464c

SHA-256 9633d0564a2b8f1b4c6e718ae7ab48be921d435236a403cf5e7ddfbfd4283382

The existence of these registry keys:

The HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPerServer has been set to 10.



Network Based:

Attempted TCP connection on 195.140.214.82

Yara Rule:

```
rule Practical2 {
        meta:
                file_md5 = “971a3320179e0494fdb70b138ada2446”
                file_sha1 = “b04bba3b8be297b6178e73d10f0380897e76464c”
                file_sha256 = “9633d0564a2b8f1b4c6e718ae7ab48be921d435236a403cf5e7ddfbfd4283382”
                author = “N\A”
                compilation date = “12/09/2020”
        strings:
                $debugString = “C:\\Users\\W7H64\\Desktop\\VCSamples-master\\VC2010\\ATL\\General\\AtlCon\\bitcoin coinjoin op.pdb” ascii wide nocase
        condition:
                $debugString
}
```