

# Practical Assignment 2 Report

03/17/2021

Justin Miller: miller.j@ufl.edu

Assignment P0x02

CIS4930-108A(2506)

## Table of Contents

Executive Summary.....	3
Static Analysis.....	3
Virus Total.....	3
Basic Static Analysis.....	4
Packing and Obfuscation.....	4
Program Sections, Strings, and Imports.....	6
Anti-Disassembly.....	7
Dynamic Analysis.....	8
Process Behaviors.....	8
Filesystem Activity.....	11
Network Activity.....	12
Registry Keys.....	12
Anti-Debugging and Anti-Virtual-Machine.....	13
Indicators of Compromise.....	15

## Executive Summary

The provided malware sample appears to be 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 Kryptik trojan or the AveMaria trojan.

52  
/ 70

Community Score

52 engines detected this file

9633d0564a2b8f1b4c6e718ae7ab48be921d435236a403cf5e7ddfbd4283382  
Practical2.exe  
direct-cpu-clock-access invalid-rich-pe-checksum invalid-rich-pe-linker-version long-sleeps peexe runtime-modules

1.38 MB  
Size

2021-02-24 19:19:48 UTC  
21 days ago

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Ad-Aware	Gen:Variant.Razy.804171	AhnLab-V3	Malware/Win32.RL_Generic.R357891	
Alibaba	TrojanSpy:Win32/AveMaria.86389254	ALYac	Gen:Variant.Razy.804171	
Antiy-AVL	Trojan/Win32.Kryptik	SecureAge APEX	Malicious	
Arcabit	Trojan.Razy.DC454B	Avast	Win32:RATX-gen [Trj]	
AVG	Win32:RATX-gen [Trj]	Avira (no cloud)	TR/AD.MortyStealer.njtp	
BitDefender	Gen:Variant.Razy.804171	Bkav Pro	W32.AIDetect.malware2	
CrowdStrike Falcon	Win/malicious_confidence_100% (W)	Cybereason	Malicious.0179e0	
Cylance	Unsafe	Cynet	Malicious (score: 85)	

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:hkhSL4pH7FYilicuuTh9yeJWrpDz29Wa+QB1t6gMvITpa6NYjHhtkaJN:h72Z/8VWrp2ZF1Ea1jBH
TLSH	T1E3654A07A762811FCFE12F774FFB2B8552CABA2279892C352C5A5ECA7055F06D30E52

## 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 9<sup>th</sup>, 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	<a href="#">Microsoft Visual C++ 8.0 Debug</a>
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	<b>executable</b>
cpu	<b>32-bit</b>
signature	<a href="#">Microsoft Visual C++ 8.0 Debug</a>

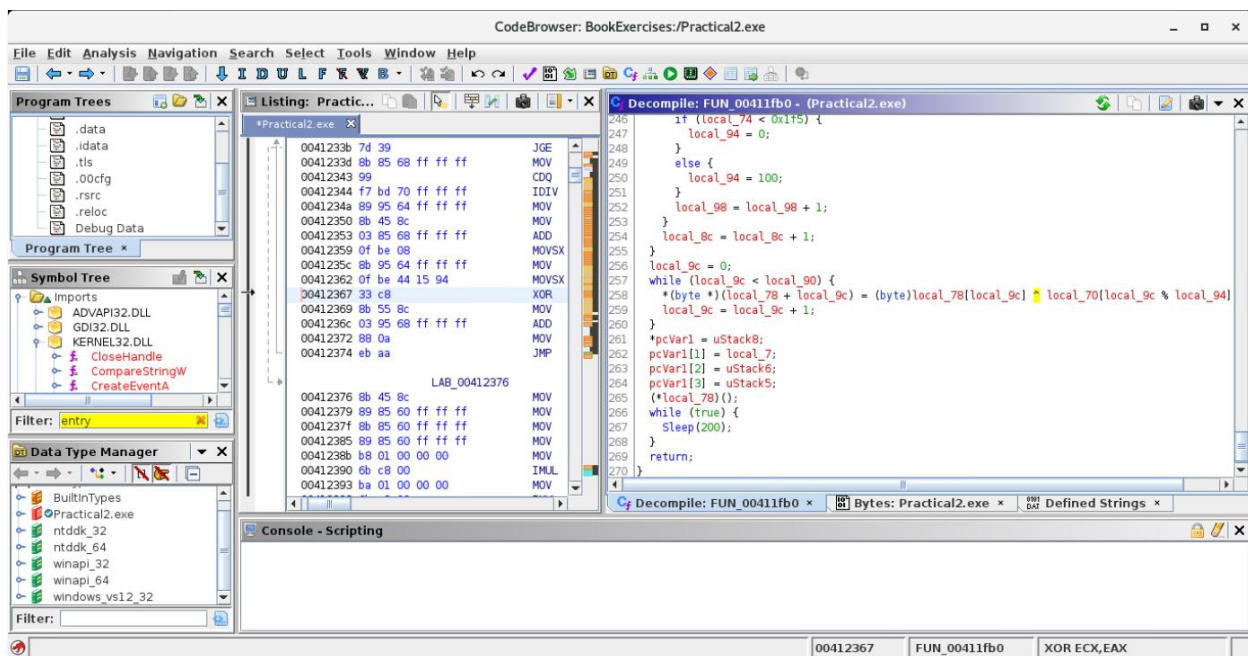
## Packing and Obfuscation

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.

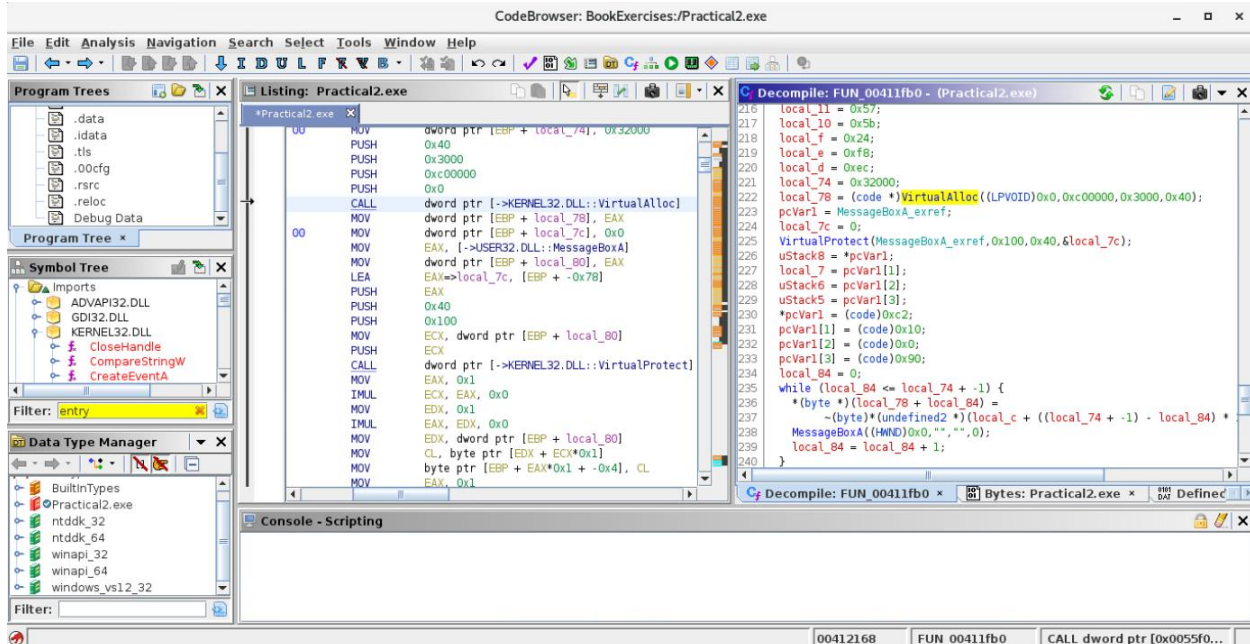
property	value	value	value	value
name	.text	.rdata	.data	.idata
md5	F4652CB58EC6D3B287599D...	58EE6DE30D069254DD3473D...	A164260C7E639AC06B537C2...	B93F40DB6CFB5866381BC50...
entropy	5.417	3.656	4.611	4.583
file-ratio (99.93%)	51.27 %	11.75 %	34.47 %	0.39 %
raw-address	0x0000400	0x000B5600	0x000DEE00	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
C573BD7CEFA296A9C5D230C...	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)	0x00000800 (26624 bytes)
0x00561000	0x00562000	0x00563000	0x00564000
0x00000309 (777 bytes)	0x00000104 (260 bytes)	0x0000043C (1084 bytes)	0x000007BE (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.

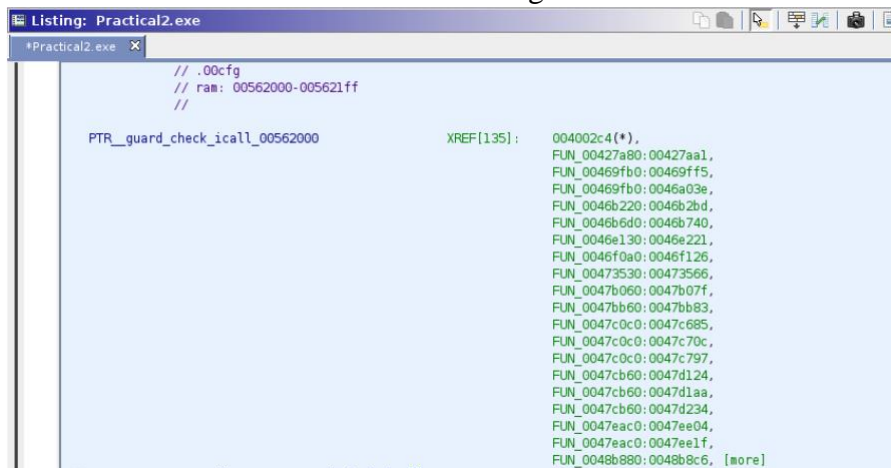


The memory address being written to, local\_78, is virtually allocated at the start of this function.



## 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	0x00002B8	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	0x000DC7C	x	file	-	C:\Users\W7H64\Desktop\VC Samples-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.

<a href="#">RegCloseKey</a>	registry	implicit	-	-	-	-	-	advapi32.dll
<a href="#">RegDeleteKeyA</a>	registry	implicit	-	x	-	-	-	advapi32.dll
<a href="#">RegOpenKeyExA</a>	registry	implicit	-	-	-	-	-	advapi32.dll
<a href="#">RegQueryInfoKeyA</a>	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)
<a href="#">ReadFile</a>	file	implicit	-	-	-	-	-	kernel32.dll
<a href="#">QueryPerformanceCount...</a>	system-information	implicit	-	-	-	-	-	kernel32.dll
<a href="#">MultiByteToWideChar</a>	-	implicit	-	-	-	-	-	kernel32.dll
<a href="#">MoveWindow</a>	windowing	implicit	-	-	-	-	-	user32.dll
<a href="#">MessageBoxA</a>	-	implicit	-	-	-	-	-	user32.dll
<a href="#">LoadLibraryExW</a>	dynamic-library	implicit	-	-	-	-	-	kernel32.dll
<a href="#">LoadLibraryExA</a>	dynamic-library	implicit	-	-	-	-	-	kernel32.dll
<a href="#">LeaveCriticalSection</a>	synchronization	implicit	-	-	-	-	-	kernel32.dll
<a href="#">LCMapStringW</a>	-	implicit	-	-	-	-	x	kernel32.dll
<a href="#">IsWindow</a>	windowing	implicit	-	-	-	-	-	user32.dll
<a href="#">IsValidLocale</a>	-	implicit	-	-	-	-	-	kernel32.dll
<a href="#">IsValidCodePage</a>	-	implicit	-	-	-	-	-	kernel32.dll
<a href="#">IsProcessorFeaturePresent</a>	system-information	implicit	-	-	-	-	-	kernel32.dll
<a href="#">IsDebuggerPresent</a>	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.

<a href="#">VirtualAlloc</a>	memory	implicit	-	-	-	-	-	kernel32.dll
<a href="#">VirtualFree</a>	memory	implicit	-	-	-	-	-	kernel32.dll
<a href="#">VirtualProtect</a>	memory	implicit	-	x	-	-	-	kernel32.dll
<a href="#">VirtualQuery</a>	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

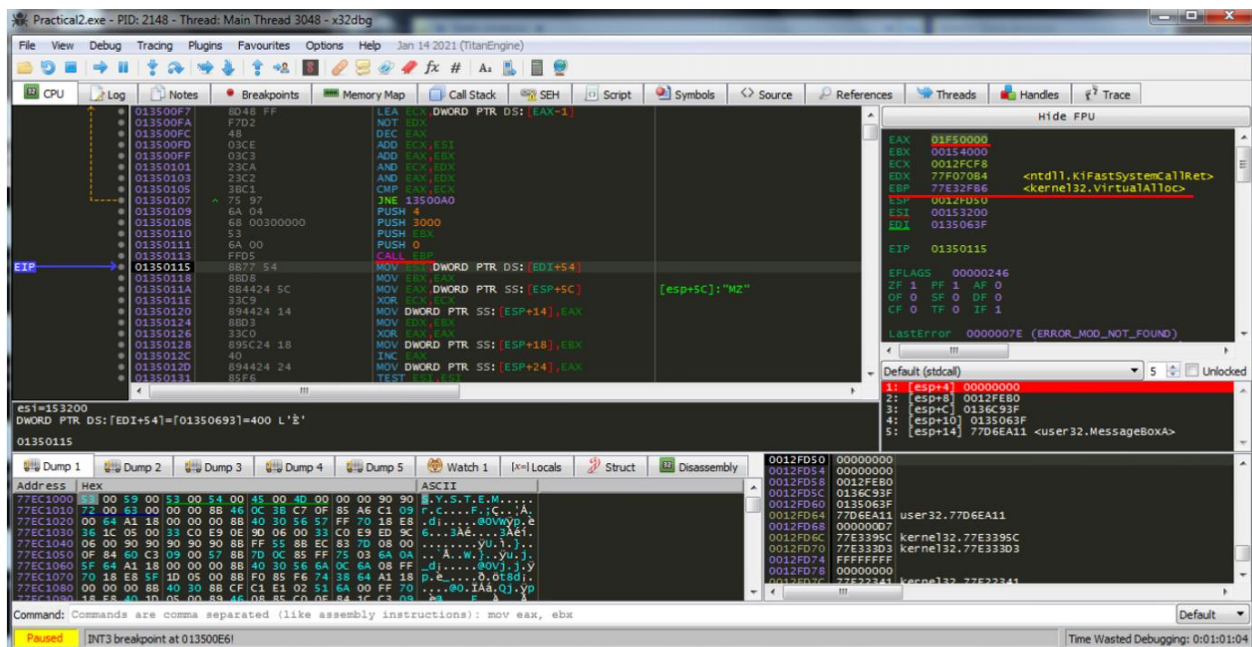
When the malware is run, a terminal window opens for a short second and closes soon thereafter. The process continues to exist until it is killed, or the computer is shutdown. The only interesting behavior seen in the properties of the process is that it eventually creates a second thread. No other processes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

The screenshot shows the Windows Task Manager (Process Explorer) and the Properties dialog box for Practical2.exe. The Process Explorer window displays a list of running processes, including vmacthlp.exe, svchost.exe, dwm.exe, vmtoolsd.exe, explorer.exe, and Practical2.exe. The Practical2.exe process is highlighted, showing its PID (3028) and Description (Sysinternals Process Explorer). The Properties dialog box for Practical2.exe is open, showing the Threads tab. It lists two threads: 2932 (CPU, Cycles Delta 44,772, Start Address Practical2.exe+0x45d4) and 3148 (ntdll.dll!RtlUserThreadStart). The dialog box also displays various system information, including Thread ID, Start Time, State, Kernel Time, User Time, Context Switches, Cycles, and Ideal Processor.

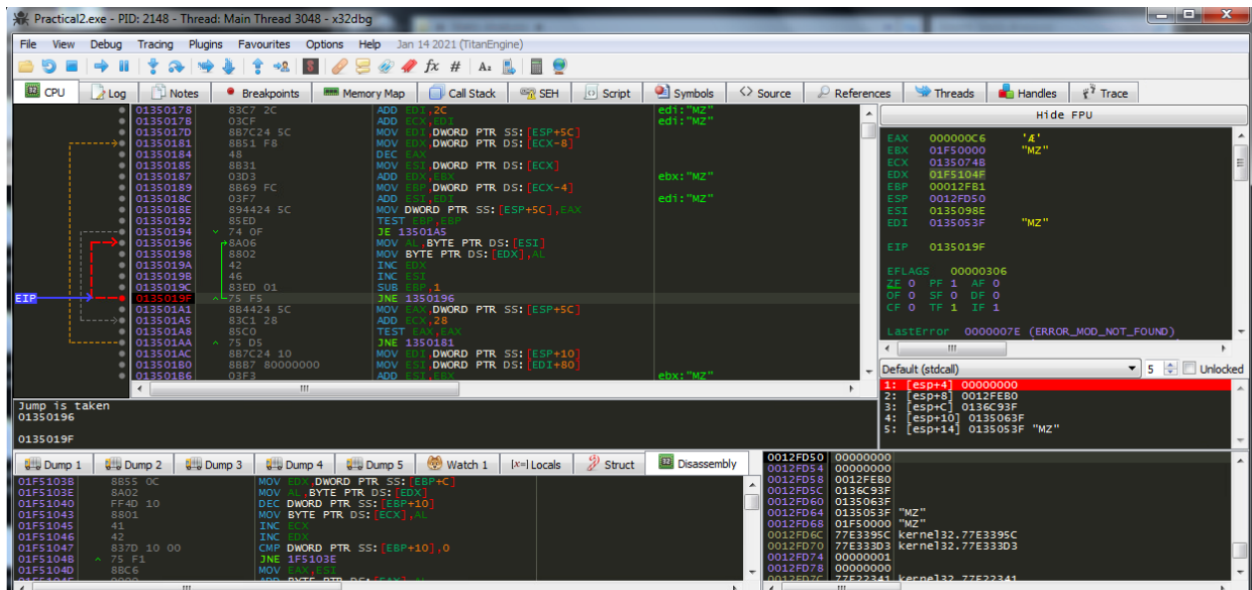
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
vmacthlp.exe		1,912 K	4,304 K	688	VMware Activation Helper	VMware, Inc.
svchost.exe		2,496 K	5,900 K	732	Host Process for Windows S...	Microsoft Corporation
svchost.exe		12,696 K	12,312 K	816	Host Process for Windows S...	Microsoft Corporation
svchost.exe		3,176 K	8,720 K	864	Host Process for Windows S...	Microsoft Corporation
dwm.exe	1.71	111,724 K	27,328 K	1132	Desktop Window Manager	Microsoft Corporation
svchost.exe	< 0.01	17,008 K	27,940 K	892	Host Process for Windows S...	Microsoft Corporation
svchost.exe	0.01	5,328 K	10,588 K	1064	Host Process for Windows S...	Microsoft Corporation
svchost.exe	0.02	10,832 K	13,684 K	1168	Host Process for Windows S...	Microsoft Corporation
spoolsv.exe		5,200 K	9,492 K	1328	Spooler SubSystem App	Microsoft Corporation
svchost.exe		8,612 K	10,068 K	1356	Host Process for Windows S...	Microsoft Corporation
VGAuthService.exe		4,960 K	8,948 K	1476	VMware Guest Authenticatio...	VMware, Inc.
vmtoolsd.exe	0.02	11,216 K	17,760 K	1552	VMware Tools Core Service	VMware, Inc.
msdtc.exe		2,552 K	6,388 K	2024	Microsoft Distributed Transa...	Microsoft Corporation
svchost.exe		3,500 K	7,616 K	1840	Host Process for Windows S...	Microsoft Corporation
sppsvc.exe		6,208 K	11,748 K	1668	Microsoft Software Protectio...	Microsoft Corporation
svchost.exe		98,568 K	31,084 K	1008	Host Process for Windows S...	Microsoft Corporation
wmpnetwk.exe	< 0.01	7,480 K	5,828 K	584	Windows Media Player Netw...	Microsoft Corporation
SearchIndexer.exe	0.01	20,908 K	15,576 K	1704	Microsoft Windows Search I...	Microsoft Corporation
taskhost.exe		2,376 K	5,844 K	3076	Host Process for Windows T...	Microsoft Corporation
lsass.exe		2,612 K	7,032 K	512	Local Security Authority Proc...	Microsoft Corporation
lsass.exe		1,112 K	2,864 K	520		
csrss.exe	0.07	5,500 K	5,712 K	408		
winlogon.exe		1,572 K	4,616 K	440		
explorer.exe	0.32	80,296 K	57,156 K	2960	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	0.08	9,828 K	13,688 K	2540	VMware Tools Core Service	VMware, Inc.
x32dbg.exe	0.16	49,300 K	68,688 K	3484	x64dbg	
Practical2.exe	< 0.01	15,172 K	4,092 K	3028		
procexp.exe	1.05	10,496 K	21,224 K	3404	Sysinternals Process Explorer	Sysinternals - www.sysinter...

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

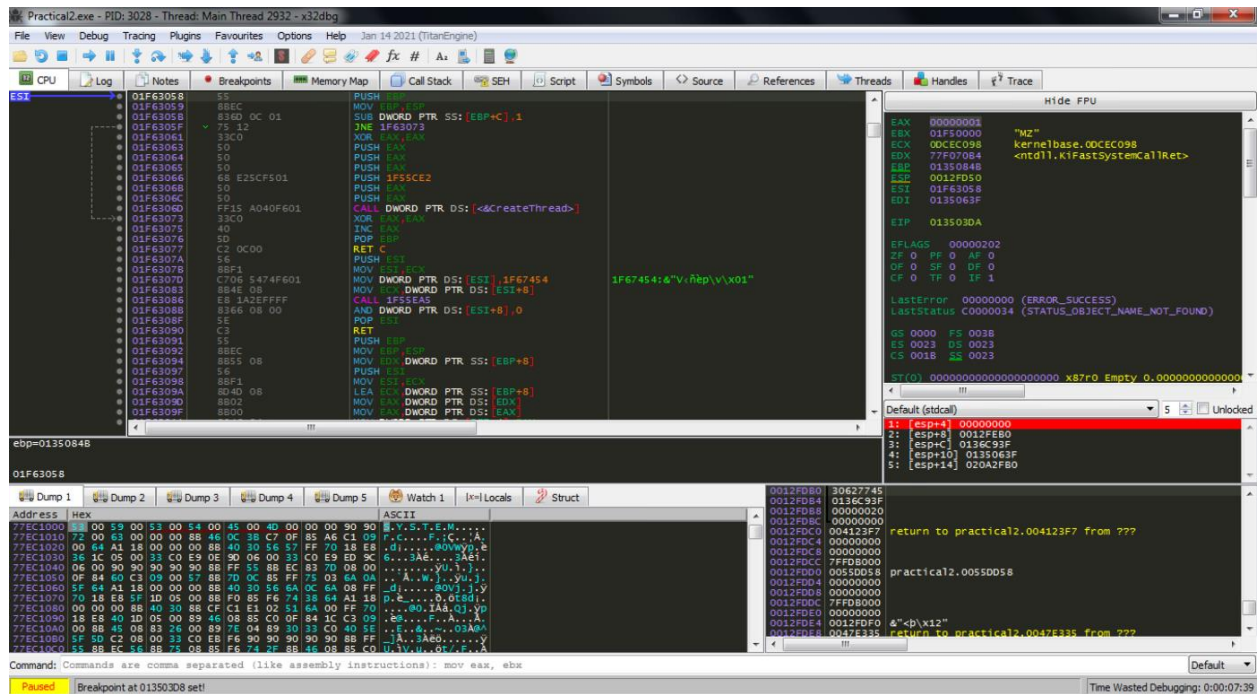




Move from ESI to AL and write from AL to EDX memory address.



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.



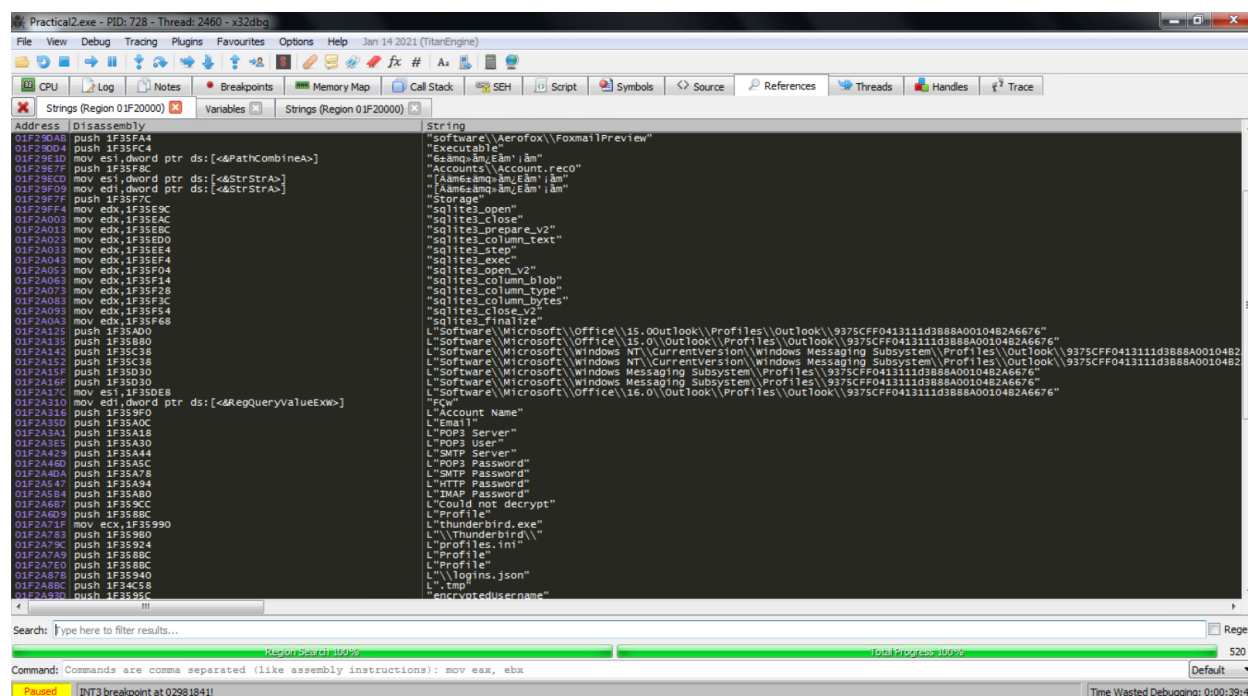
Address	Disassembly	String
01F60B7E	push 1F66EE0	L"Ave_Maria Stealer OpenSource github Link: <a href="https://github.com/syohex/java-simple-mine-sweeper">https://github.com/syohex/java-simple-mine-sweeper</a> "
01F60BAE	push 1F66FA0	L"C:\\Users\\Vital\\Kremez\\Documents\\MidgetPorn\\workspace\\MsBox.exe"

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.

```
01F5C236 push 1F64D08 L"Google\\Chrome\\User Data\\Local State"
01F5C238 push 1F64D58 L"Google\\Chrome\\User Data\\Default\\Login Data"
01F5C251 push 1F64D80 L"Epic Privacy Browser\\User Data\\Local State"
01F5C256 push 1F64E08 L"Epic Privacy Browser\\User Data\\Default\\Login Data"
01F5C26C push 1F64E70 L"Microsoft Edge\\User Data\\Local State"
01F5C271 push 1F64EC0 L"Microsoft Edge\\User Data\\Default\\Login Data"
01F5C288 push 1F64F20 L"UCBrowser\\User Data_118n\\Local State"
01F5C28D push 1F64F70 L"UCBrowser\\User Data_118n\\Default\\UC Login Data.17"
01F5C2A3 push 1F64FD8 L"Tencent QQBrowser\\User Data\\Local State"
01F5C2A8 push 1F65030 L"Tencent QQBrowser\\User Data\\Default\\Login Data"
01F5C28F push 1F65090 L"Opera Software\\Opera Stable\\Local State"
01F5C2C4 push 1F650E8 L"Opera Software\\Opera Stable\\Login Data"
01F5C2DA push 1F65138 L"Blink\\User Data\\Local State"
01F5C2DF push 1F65178 L"Blink\\User Data\\Default\\Login Data"
01F5C2F5 push 1F651C0 L"Chromium\\User Data\\Local State"
01F5C2FA push 1F65200 L"Chromium\\User Data\\Default\\Login Data"
01F5C310 push 1F65250 L"BraveSoftware\\Brave-Browser\\User Data\\Local State"
01F5C315 push 1F65288 L"BraveSoftware\\Brave-Browser\\User Data\\Default\\Login Data"
01F5C329 push 1F6532C L"Vivaldi\\User Data\\Local State"
01F5C330 push 1F65370 L"Vivaldi\\User Data\\Default\\Login Data"
01F5C346 push 1F653C0 L"Comodo Dragon\\User Data\\Local State"
01F5C348 push 1F65410 L"Comodo Dragon\\User Data\\Default\\Login Data"
01F5C361 push 1F65468 L"Torch\\User Data\\Local State"
01F5C366 push 1F654A8 L"Torch\\User Data\\Default\\Login Data"
01F5C37C push 1F654F0 L"Slimjet\\User Data\\Local State"
01F5C381 push 1F65530 L"Slimjet\\User Data\\Default\\Login Data"
01F5C397 push 1F65580 L"CentBrowser\\User Data\\Local State"
01F5C39C push 1F655C8 L"CentBrowser\\User Data\\Default\\Login Data"
```

```
01F555BA push 1F64850 "warzone160"
01F55688 push 1F64850 "warzone160"
```



## 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\invcli.dll	NAME NOT FOUND
10:53:...	Practical2.exe	3880	CreateFile	C:\Users\malware\Desktop\wksccli.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\MSFW32.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	RegSet Value	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\OQWVGJ0HVDJ	NAME NOT FOUND	Desired Access: All Access
9:32:0...	Practical2.exe	580	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OQWVGJ0HVDJ	SUCCESS	Desired Access: Query Value, Disposition: REG_CREATED_NEW_KEY
9:32:0...	Practical2.exe	580	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OQWVGJ0HVDJ	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: {bbe8a05beee-4442-804e-409d6c-4515e9}
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.

Capturing from ens33

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
475	722.026698016	192.168.245.127	195.140.214.82	TCP	66	49223 → 6703 [SYN] Seq=0 Win=8192 Len=...
476	722.026800588	195.140.214.82	192.168.245.127	TCP	54	6703 → 49223 [RST, ACK] Seq=1 Ack=1 Wi...
477	722.528197452	192.168.245.127	195.140.214.82	TCP	66	[TCP Retransmission] 49223 → 6703 [SYN...
478	722.528267494	195.140.214.82	192.168.245.127	TCP	54	6703 → 49223 [RST, ACK] Seq=1 Ack=1 Wi...
479	723.042705383	192.168.245.127	195.140.214.82	TCP	62	[TCP Retransmission] 49223 → 6703 [SYN...
480	723.042773441	195.140.214.82	192.168.245.127	TCP	54	6703 → 49223 [RST, ACK] Seq=1 Ack=1 Wi...
481	727.005087948	VMware_87:65:be	VMware_87:8e:c1	ARP	60	Who has 192.168.245.133? Tell 192.168...
482	727.005116852	VMware_87:8e:c1	VMware_87:65:be	ARP	42	192.168.245.133 is at 00:50:56:87:8e:c1
483	753.826939644	192.168.245.127	192.168.245.133	DNS	85	Standard query 0x9aa1 A teredo.ipv6.mi...
484	753.827182249	192.168.245.133	192.168.245.127	DNS	101	Standard query response 0x9aa1 A tered...

[Window size scaling factor: 256]  
Checksum: 0x0b9b [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
▶ [SEQ/ACK analysis]  
▶ [Timestamps]

```

0000  00 50 56 87 8e c1 00 50  56 87 3b 5b 08 00 45 00  .PV...P V.;[...E
0010  00 28 f3 76 40 00 00 06  9b 00 c0 a8 f5 81 c0 a8  .(.v@...
0020  f5 85 c2 2a 01 bb cb f6  d6 e6 4b f5 85 24 50 14  .*.~...K..$P
0030  00 00 0b 9b 00 00 00 00  00 00 00 00

```

ens33: <live capture in progress>      Packets: 584 · Displayed: 584 (100.0%)      Profile: Default

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.

```

LastError 00002740 (WSAECONNREFUSED)
LastStatus C0000236 (STATUS_CONNECTION_REFUSED)

```

PracticalExe - PID: 2356 - Thread: 1768 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Jan 14 2021 (TitanEngine)

Log Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace

8083 C0010000 LEA ECX, [DWORD PTR DS: [EBX+1C8]]  
 8A 02 PUSH 2  
 8983 C0010000 MOV DWORD PTR DS: [EBX+1CC], EAX  
 5B POP  
 FF75 0C PUSH DWORD PTR SS: [EBP+C]  
 66:8906 MOV WORD PTR DS: [ESI], AX  
 FF15 F042E901 CALL DWORD PTR DS: [<antohs>]  
 FF75 FC PUSH DWORD PTR SS: [EBP+4]  
 66:8983 C0010000 MOV WORD PTR DS: [EBX+1CA], AX  
 FF15 2843E901 CALL DWORD PTR DS: [<freeaddrinfo>]  
 5A 10 PUSH 10  
 FF73 0C PUSH DWORD PTR DS: [EBX+C]  
 FF15 F842E901 CALL DWORD PTR DS: [<connect>]  
 83FB FF CMP EAX, 0  
 75 16 JNE 1E858C2  
 0943 0C OR DWORD PTR DS: [EBX+C], EAX  
 XOR EAX, EAX  
 8B4D 08 MOV ECX, DWORD PTR DS: [EBP+8]  
 EB EC050000 CALL 1E85EAS  
 8BC7 MOV EAX, ECX  
 5F POP EDI  
 5E POP ESI  
 5B POP EDI  
 C9 LEAVE

ebx+1CC: "Az0ka8u\01"  
[ebp+8]: "195.140.214.82"  
ebx: &"195.140.214.82"

EAX FFFFFFFF  
 EBX 0200FCEC &"195.140.214.82"  
 ECX 7FF0E000  
 EDI 00002740  
 EBP 0200F8A0  
 ESP 0200F870 &"HHé\01"  
 ESI 0200FEB4  
 EDI 00000001  
 EIP 01E858A7  
 EFLAGS 00000286  
 ZF 0 PF 1 AF 0  
 OF 0 SF 1 OF 0  
 CF 0 TF 0 IF 1  
 LastError 00002740 (WSAECONNREFUSED)

Default (stdcal) 5 Unlocked

1: [esp+4] 0200FCEC  
 2: [esp+8] 00000000  
 3: [esp+C] 00000000  
 4: [esp+10] 00000000  
 5: [esp+14] 00000001

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 [x] Locals Struct

Address Hex ASCII  
 02BE0000 31 39 2E 31 34 30 2E 32 31 34 2E 38 32 00 00 195.140.214.82...  
 02BE0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 02BE0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

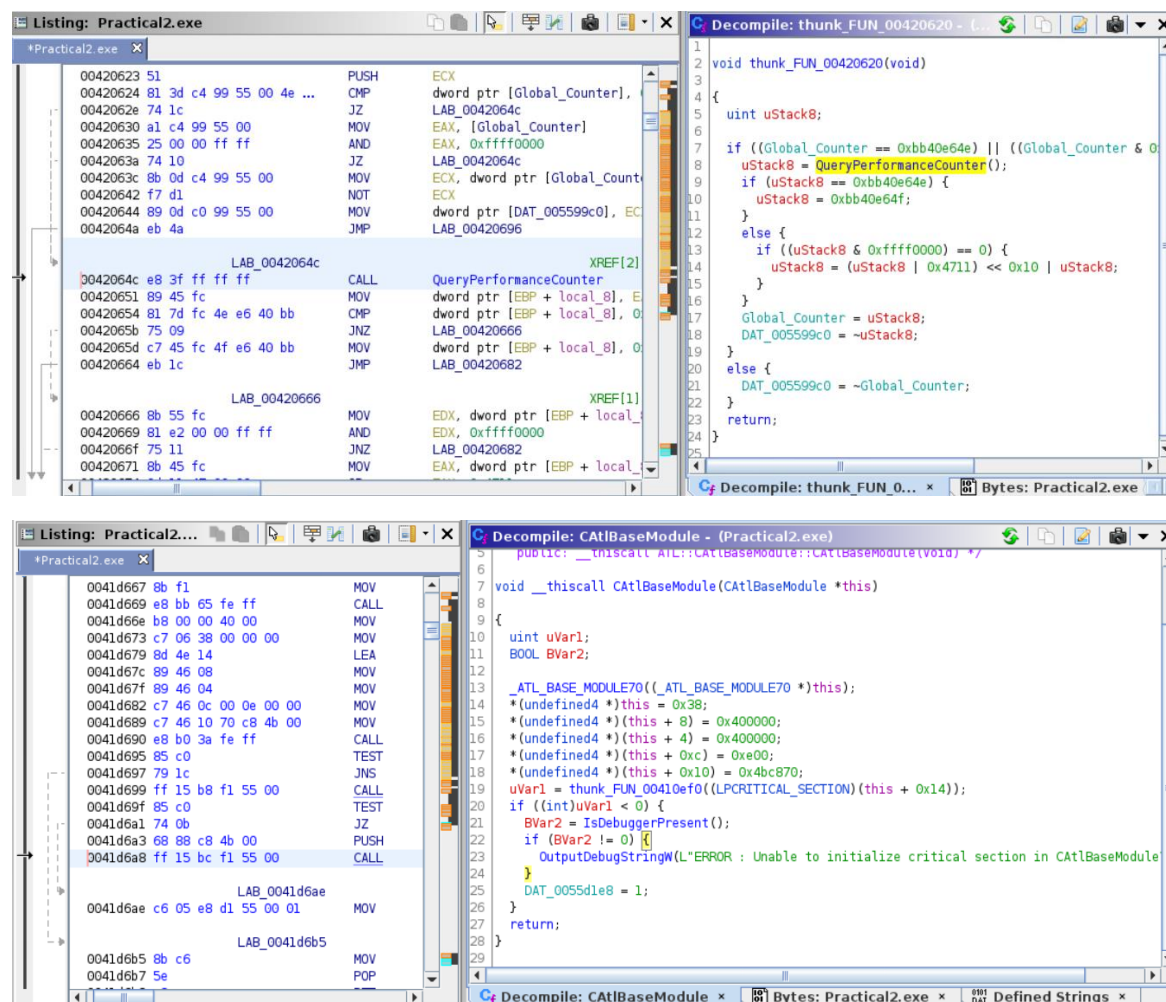
Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused [INT3 breakpoint at 01E858A1] Time Wasted Debugging: 0:01:19:08

## 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.



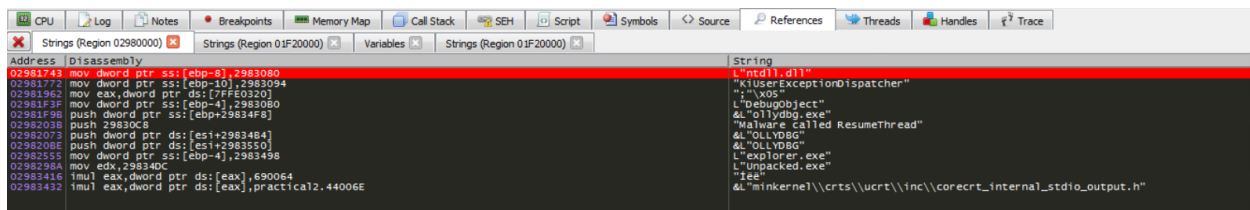


```

013C0467 83EC 10 SUB ESP,10
013C046A 64:A1 30000000 MOV EAX,DWORD PTR FS:[30]
013C0470 53 PUSH EBX
013C0471 55 PUSH EBP
013C0472 56 PUSH ESI
013C0473 8B40 0C MOV EAX,DWORD PTR DS:[EAX+C]
013C0476 57 PUSH EDI

```

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:

Time ...	Process Name	PID	Operation	Path	Result	Detail
9:32:0...	Practical2.exe	580	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OO\WGJHVDU	NAME NOT FOUND	Desired Access: All Access
9:32:0...	Practical2.exe	580	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OO\WGJHVDU	SUCCESS	Desired Access: Query Value, Disposition: REG_CREATED_NEW_KEY
9:32:0...	Practical2.exe	580	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\OO\WGJHVDU	SUCCESS	
9:44:4...	Practical2.exe	1504	RegEnumKey	HKCR\Drive\shellex\FolderExtensions	SUCCESS	Index: 0, Name: {beb8a05-beee-4442-804e-409d6c4515e9}
9:44:4...	Practical2.exe	1504	RegEnumKey	HKCR\Drive\shellex\FolderExtensions	NO MORE ENTRIES	Index: 1, Length: 288

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

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

### 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  
}
```