Practical3 Report

Name: Rachana Nitinkumar Gugale
UFID: 6353-2454
Email: rgugale@ufl.edu
Date: 12th April 2023
Class: CAP6137
Assignment: Practical 3

Executive Summary

The sample is a **trojan** that **steals credentials and information** from FTP clients, browsers, bitcoin wallets, email clients and other applications like Adobe. The program belongs to the **Fareit/Pony** malware family. Windows Defender flags the program as suspicious and identifies it as Fareit trojan.

The malware is very good at hiding itself. It doesn't create any files on the system. It recursively loops over all the folders on the Desktop and in Documents directory. It seems to be looking for some specific files and folders to steal and exfiltrate the data found. The sample adds the key *HKU\<SID>\Software\WinRAR\HWID* to the registry. It adds the path of sample3.exe as the value to *HKU\<SID>\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store*.

The program sends encrypted data to the following URLs at port **10015** using WinSock functions socket(), connect() and send(). The sample imports functions from libraries like bcrypt.dll and crypt32.dll for encrypting data.

- http://85.192.165.229:10015/gate.php
- http://176.96.187.114:10015/gate.php
- http://176.96.187.116:10015/gate.php
- http://80.254.98.212:10015/gate.php
- http://5.144.66.227:10015/gate.php

The malware's CnC server could be located at these IP address (some of these IPs are registered with Russian organizations) and it might be sending the data it steals to the CnC servers.

Dumping the program's memory with volatility and analyzing its strings reveals a lot of information. There are strings related to various FTP applications like Putty, FileZilla, CuteFTP, FTPGetter, SmartFTP, FTPExplorer which the malware might be stealing data from. There were also strings related to browsers like Firefox, Chrome, Opera, strings for email clients like Outlook, Yandex and BatMail and strings related to other applications like Directory Opus and Windows Commander. The malware sniffs data from all these applications. It also tries to crack the passwords of these apps by comparing them against a set of common passwords.
The sample does anti-debugging using **GetTickCount()** and goes into an infinite loop if a debugger is detected. After executing, the sample deletes itself.

Host based indicators: The malware does not create any files. It only adds one registry key *HKU\<SID>\Software\WinRAR\HWID* which can be used as a host based indicator.
Network based indicators: Any traffic to 85.192.165.229, 176.96.187.114, 176.96.187.116, 80.254.98.212, 5.144.66.227 these IPs can act as a network-based indicator.

Figure 1 Windows defender flags the program as Fareit trojan

Static Analysis

PE Studio

The following information was gathered with PEStudio:

1. **Compilation Date:** 23 March 2015 at 19:18:36 UTC
2. The program is a **32-bit GUI** program as can be seen from the “subsystem” value.
3. The file has a description of “ManyBytes program” which is suspicious.

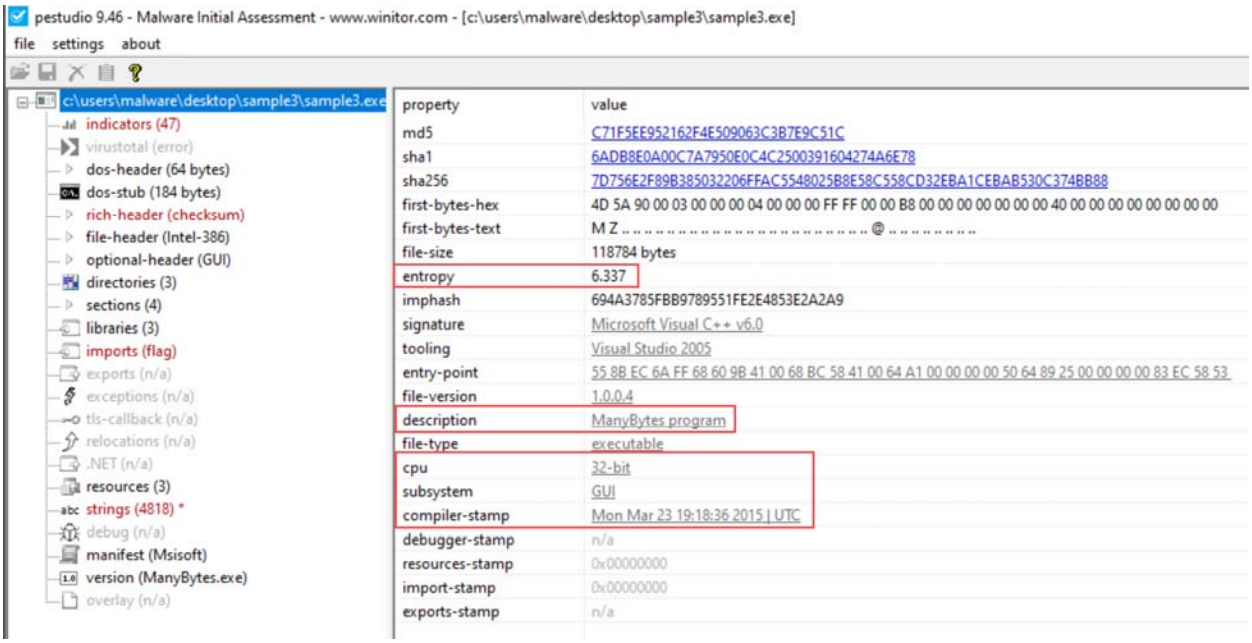


Figure 2 Basic PEStudio Analysis

4. The following basic indicators suggest that the program is not packed:
    a. **Entropy:** The diversity within a program or file is measured by its entropy, which is typically scored on a scale of 0 to 8. A value of 0 indicates no diversity, meaning that all bytes in the file are the same, while a value of 8 indicates maximum diversity with all bytes being different. For normal executable programs, an entropy value of around 6 is typical. In the case of our sample, its entropy value of 6.337 suggests the sample isn't packed.
    b. **Names of PE sections:** The program contains the standard PE sections - *.text*, *.rdata*, *.rsrc* and *.data*. These sections are commonly found in all PE files. Since the program does not contain any non-standard sections and the section names are non-obfuscated, the malware might not be packed.
    c. **Raw and virtual sizes of PE sections:** A section's raw size being significantly smaller than its virtual size suggests that the malware may be unpacking or extracting new data. However, the .text, .rdata and .rsrc sections in the program have comparable raw and virtual sizes, indicating that no significant extraction or unpacking is taking place. The .data section has a larger virtual size, but this is typical for Windows executables and does not suggest packing of malware. These observations support the conclusion that the sample isn't packed.
    d. **Strings:** PEStudio found 4818 strings for the sample out of which many are readable. The existence of numerous readable strings within a program can act as an indicator of its non-obfuscatedness. In the case of our malware sample,

since PE Studio detected a substantial number of readable strings, it suggests that the program is not obfuscated.

e. **Imports:** The program statically imports functions from 3 DLLs -USER32.dll, KERNEL32.dll, SHELL32.dll. While the presence of these imported functions alone does not confirm that the malware is not packed, when combined with other indicators, it offers substantial evidence to suggest that the malware is obfuscated. Less number of imported libraries also suggest that the sample might be importing other DLLs dynamically.

property	value	value	value	value
general				
name	.text	.rdata	.data	.rsrc
md5	14E7F71B1F431B23606CD2F...	BA424D1F80E113350C05A4A...	7597430628ACB3A8EC870E7...	C8560FAC07D5C07C3FE85D...
entropy	6.643	6.033	1.691	3.959
file-ratio (99.14%)	77.59 %	9.05 %	10.78 %	1.72 %
raw-address	0x00000400	0x00016C00	0x00019600	0x0001C800
raw-size (117760 bytes)	0x00016800 (92160 bytes)	0x00002A00 (10752 bytes)	0x00003200 (12800 bytes)	0x00000800 (2048 bytes)
virtual-address	0x00001000	0x00018000	0x0001B000	0x00023000
virtual-size (135518 bytes)	0x00016764 (92004 bytes)	0x0000297E (10622 bytes)	0x0000795C (31068 bytes)	0x00000720 (1824 bytes)

Figure 3 Readable PE section names and comparable raw and virtual PE section sizes

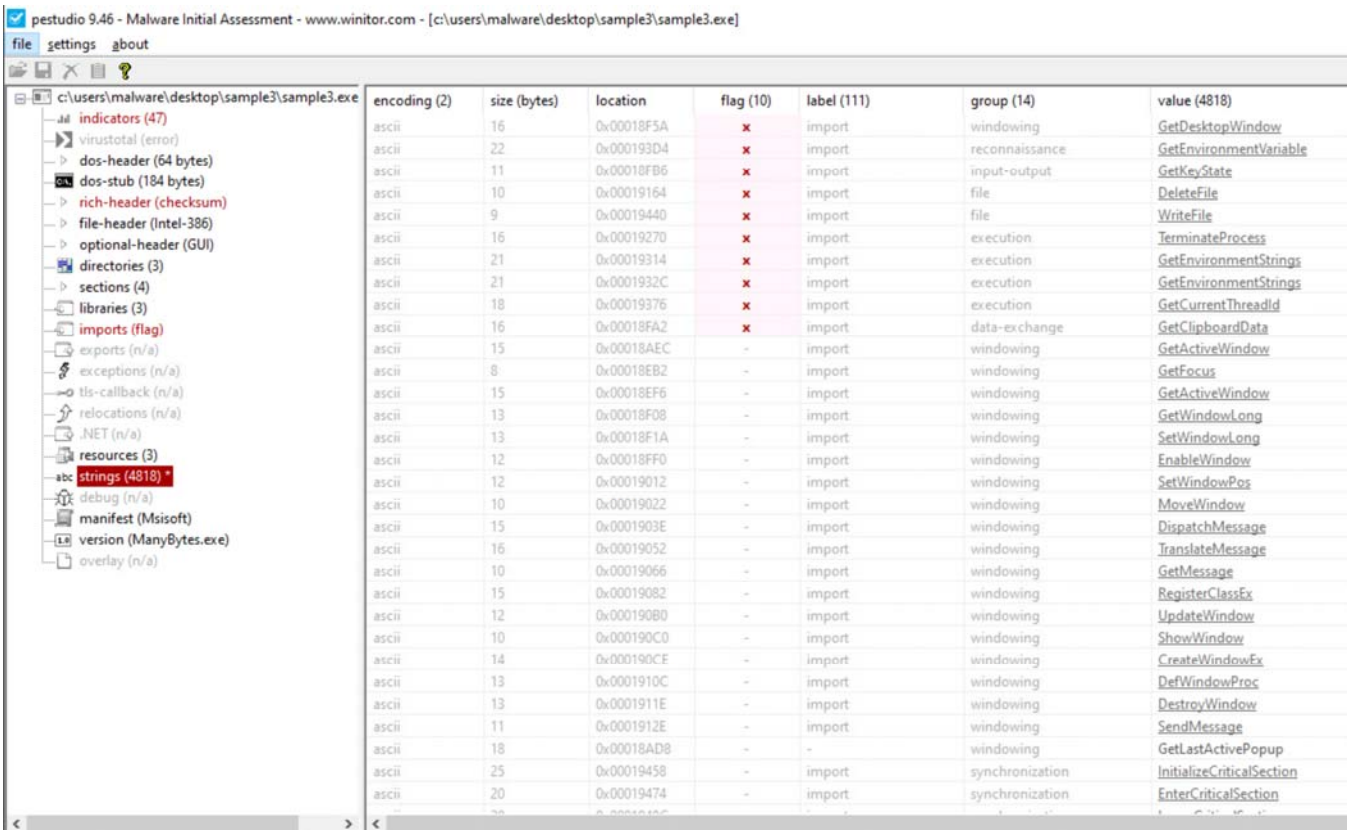
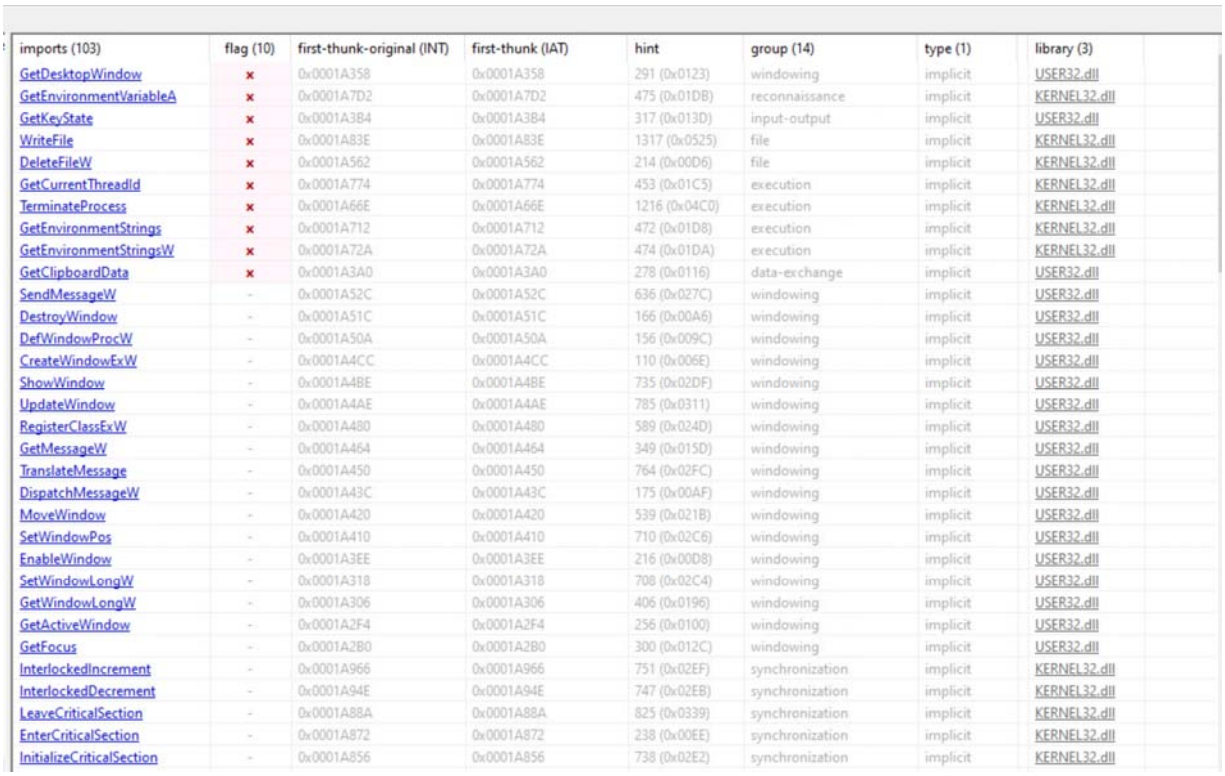
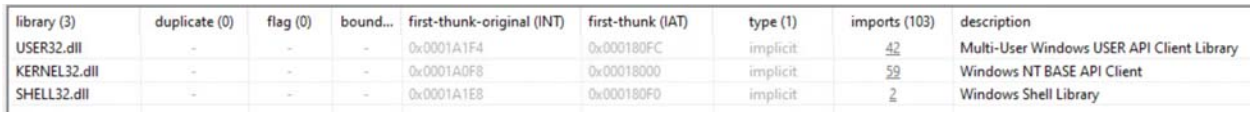


Figure 4 Lots of readable strings found by PEStudio



imports (103)	flag (10)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (14)	type (1)	library (3)
GetDesktopWindow	x	0x0001A358	0x0001A358	291 (0x0123)	windowing	implicit	USER32.dll
GetEnvironmentVariableA	x	0x0001A7D2	0x0001A7D2	475 (0x01DB)	reconnaissance	implicit	KERNEL32.dll
GetKeyState	x	0x0001A3B4	0x0001A3B4	317 (0x013D)	input-output	implicit	USER32.dll
WriteFile	x	0x0001A83E	0x0001A83E	1317 (0x0525)	file	implicit	KERNEL32.dll
DeleteFileW	x	0x0001A562	0x0001A562	214 (0x00D6)	file	implicit	KERNEL32.dll
GetCurrentThreadId	x	0x0001A774	0x0001A774	453 (0x01C5)	execution	implicit	KERNEL32.dll
TerminateProcess	x	0x0001A66E	0x0001A66E	1216 (0x04C0)	execution	implicit	KERNEL32.dll
GetEnvironmentStrings	x	0x0001A712	0x0001A712	472 (0x01D8)	execution	implicit	KERNEL32.dll
GetEnvironmentStringsW	x	0x0001A72A	0x0001A72A	474 (0x01DA)	execution	implicit	KERNEL32.dll
GetClipboardData	x	0x0001A3A0	0x0001A3A0	278 (0x0116)	data-exchange	implicit	USER32.dll
SendMessageW	-	0x0001A52C	0x0001A52C	636 (0x027C)	windowing	implicit	USER32.dll
DestroyWindow	-	0x0001A51C	0x0001A51C	166 (0x00A6)	windowing	implicit	USER32.dll
DefWindowProcW	-	0x0001A50A	0x0001A50A	156 (0x009C)	windowing	implicit	USER32.dll
CreateWindowExW	-	0x0001A4CC	0x0001A4CC	110 (0x006E)	windowing	implicit	USER32.dll
ShowWindow	-	0x0001A4BE	0x0001A4BE	735 (0x02DF)	windowing	implicit	USER32.dll
UpdateWindow	-	0x0001A4AE	0x0001A4AE	785 (0x0311)	windowing	implicit	USER32.dll
RegisterClassExW	-	0x0001A480	0x0001A480	589 (0x024D)	windowing	implicit	USER32.dll
GetMessageW	-	0x0001A464	0x0001A464	349 (0x015D)	windowing	implicit	USER32.dll
TranslateMessage	-	0x0001A450	0x0001A450	764 (0x02FC)	windowing	implicit	USER32.dll
DispatchMessageW	-	0x0001A43C	0x0001A43C	175 (0x00AF)	windowing	implicit	USER32.dll
MoveWindow	-	0x0001A420	0x0001A420	539 (0x021B)	windowing	implicit	USER32.dll
SetWindowPos	-	0x0001A410	0x0001A410	710 (0x02C6)	windowing	implicit	USER32.dll
EnableWindow	-	0x0001A3EE	0x0001A3EE	216 (0x00D8)	windowing	implicit	USER32.dll
SetWindowLongW	-	0x0001A318	0x0001A318	708 (0x02C4)	windowing	implicit	USER32.dll
GetWindowLongW	-	0x0001A306	0x0001A306	406 (0x0196)	windowing	implicit	USER32.dll
GetActiveWindow	-	0x0001A2F4	0x0001A2F4	256 (0x0100)	windowing	implicit	USER32.dll
GetFocus	-	0x0001A2B0	0x0001A2B0	300 (0x012C)	windowing	implicit	USER32.dll
InterlockedIncrement	-	0x0001A966	0x0001A966	751 (0x02EF)	synchronization	implicit	KERNEL32.dll
InterlockedDecrement	-	0x0001A94E	0x0001A94E	747 (0x02EB)	synchronization	implicit	KERNEL32.dll
LeaveCriticalSection	-	0x0001A88A	0x0001A88A	825 (0x0339)	synchronization	implicit	KERNEL32.dll
EnterCriticalSection	-	0x0001A872	0x0001A872	238 (0x00EE)	synchronization	implicit	KERNEL32.dll
InitializeCriticalSection	-	0x0001A856	0x0001A856	738 (0x02E2)	synchronization	implicit	KERNEL32.dll

Figure 5 Program statically imports functions from USER32.dll, KERNEL32.dll and SHELL32.dll



library (3)	duplicate (0)	flag (0)	bound...	first-thunk-original (INT)	first-thunk (IAT)	type (1)	imports (103)	description
USER32.dll	-	-	-	0x0001A1F4	0x000180FC	implicit	42	Multi-User Windows USER API Client Library
KERNEL32.dll	-	-	-	0x0001A0F8	0x00018000	implicit	59	Windows NT BASE API Client
SHELL32.dll	-	-	-	0x0001A1E8	0x000180F0	implicit	2	Windows Shell Library

Figure 6 Program statically imports only 3 libraries

PE Sections

The program has 4 PE sections:

1. **.text:** The .text section is responsible for storing the **executable** instructions of the program and is the only section with executable permissions. The entry point of the .text section serves as the entry point for the program code, and program execution begins at this point.
2. **.rdata:** Since the program does not contain a separate .idata and .edata sections, information about imported and exported functions and imported libraries is stored in .rdata. The information regarding imported libraries and their functions is stored in tables known as the Import Address Table (IAT) and Import Directory Table (IDT). The IDT is loaded first and contains information and addresses related to the imported libraries, while the IAT is loaded subsequently and includes details and addresses of

imported functions. The data stored in .rdata is globally accessible and **read-only**. As this section doesn't contain executable instructions, it does not have executable permission.

3. **.data:** The .data section stores globally accessible data, such as arrays, structs, or strings, that can be read and modified from any part of the program. That is why this section has **writable** permission. Unlike the .text section, this section does not contain executable instructions and does not have executable permission.
4. **.rsrc:** This section stores the resources needed by the program like strings, images, text and manifest files. Since this section doesn't contain any code to execute, it does not have executable permission.

property	value	value	value	value
general				
name	.text	.rdata	.data	.rsrc
md5	14E7F71B1F431B23606CD2F...	BA424D1F80E113350C05A4A...	7597430628ACB3A8EC870E7...	C8560FAC07D5C07C3FE85D...
entropy	6.643	6.033	1.691	3.959
file-ratio (99.14%)	77.59 %	9.05 %	10.78 %	1.72 %
raw-address	0x00000400	0x00016C00	0x00019600	0x0001C800
raw-size (117760 bytes)	0x00016800 (92160 bytes)	0x00002A00 (10752 bytes)	0x00003200 (12800 bytes)	0x00000800 (2048 bytes)
virtual-address	0x00001000	0x00018000	0x0001B000	0x00023000
virtual-size (135518 bytes)	0x00016764 (92004 bytes)	0x0000297E (10622 bytes)	0x0000795C (31068 bytes)	0x00000720 (1824 bytes)
characteristics				
value	0x60000020	0x40000040	0xC0000040	0x40000040
writable	-	-	x	-
executable	x	-	-	-
shareable	-	-	-	-
discardable	-	-	-	-
initialized-data	-	x	x	x
uninitialized-data	-	-	-	-
self-modifying	-	-	-	-
virtualized	-	-	-	-

Figure 7 PE Sections and the permissions given to each section

Resources

The program contains 3 resources:

1. Version
2. Manifest
3. Dialog

name	instance	signature	location	size (1575 bytes)	file-ratio (1.33%)	hash	entropy	language	first-by
version	1	version	.rsrc:0x0001CA38	700	0.59 %	F79D0EE62B18B6CF675BE81DD0E3514E	3.372	English-US	BC 02 3
manifest	1	manifest	.rsrc:0x0001CCF8	547	0.46 %	669D75D4D7C0F96A699F2C8A90F70CB0	5.125	English-US	3C 3F 7
dialog	28	dialog	.rsrc:0x0001C8F0	328	0.28 %	1215CF59D4FF737080C602087F325663	3.198	English-US	01 00 FI

Figure 8 Resources in .rsrc section

Dumping the resources into their own files gives the following:

a) **Version**: Dumping this resource gives a file that contains some version information. The malware seems to be related to an organization called “ManyBytes”.

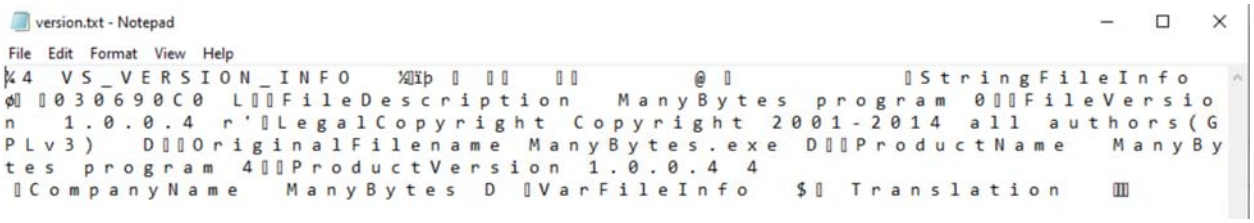


Figure 9 Version dump

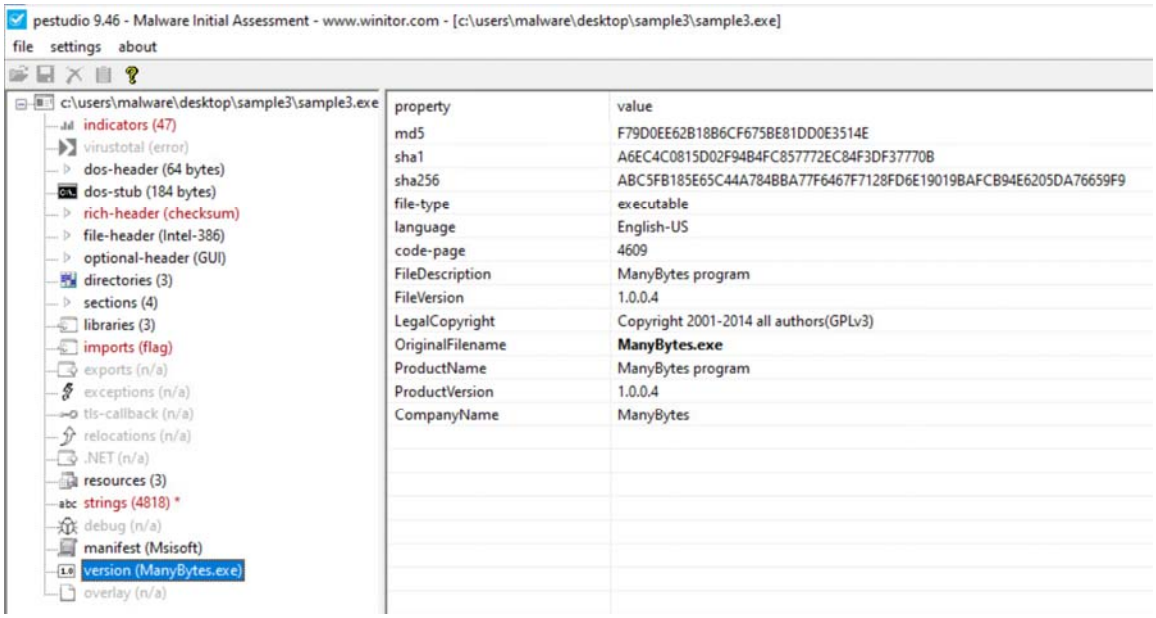


Figure 10 Information about version file in PEStudio

b) **Manifest**: Dumping this gives a **Common-Controls** XML manifest file. Common-Controls is a DLL provided by Microsoft that contains functions related to common GUI elements like buttons, scroll bars, progress bars and list boxes. When an application uses the Common Controls library, it includes a reference to the manifest file in its resources. The manifest contains information about the specific version of the Common Controls library that the application requires.

Figure 11 Manifest file contains XML data

c) **Dialog**: This could be the text on some dialog displayed by the malware.

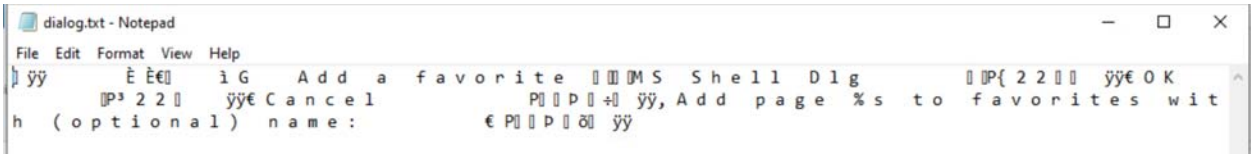


Figure 12 Dialog dump

Suspicious Imports

Imports related to following activities were found from PEStudio:

1. **Keylogging related:**
    a. **GetKeyState** – This function is used to check if a key is pressed or not. The malware could be looping over all the keys to check which one is pressed at the given moment. It could capture the user's keystrokes this way.

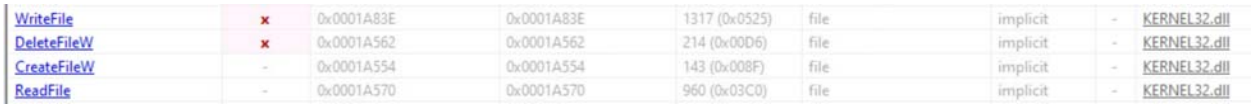
Figure 13 Keylogging related import

    b. **GetClipboardData** – The malware might be stealing the data the user copies to clipboard.

2. **Process related:** TerminateProcess, Sleep and ExitProcess imports indicate that the malware might be killing its process or own process or going into sleep mode if some criteria is met. This could possibly be an anti-analysis technique.

TerminateProcess	x	0x0001A66E	0x0001A66E	1216 (0x04C0)	execution	implicit	-	KERNEL32.dll
Sleep	-	0x0001A5D8	0x0001A5D8	1202 (0x04B2)	execution	implicit	-	KERNEL32.dll
ExitProcess	-	0x0001A660	0x0001A660	281 (0x0119)	execution	implicit	-	KERNEL32.dll

3. **File related:** The program does some file manipulation. Though file manipulation by itself is not suspicious, from dynamic analysis we can see that the sample iterates over all the files on Desktop and probably steals their data.



WriteFile	x	0x0001A83E	0x0001A83E	1317 (0x0525)	file	implicit	-	KERNEL32.dll
DeleteFileW	x	0x0001A562	0x0001A562	214 (0x00D6)	file	implicit	-	KERNEL32.dll
CreateFileW	-	0x0001A554	0x0001A554	143 (0x008F)	file	implicit	-	KERNEL32.dll
ReadFile	-	0x0001A570	0x0001A570	960 (0x03C0)	file	implicit	-	KERNEL32.dll

Figure 14 File manipulation related imports

4. **Graphics related:** The malware has a lot of imports related to GUI elements like windows, cursor, scrollbars and icons. But after running the sample, no GUI elements are displayed. This is highly suspicious and indicates that the program might be trying to hide itself from the user.

imports (103)	flag (10)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (14)	type (1)	or...	library (3)
GetDesktopWindow	x	0x0001A358	0x0001A358	291 (0x0123)	windowing	implicit	-	USER32.dll
SendMessageW	-	0x0001A52C	0x0001A52C	636 (0x027C)	windowing	implicit	-	USER32.dll
DestroyWindow	-	0x0001A51C	0x0001A51C	166 (0x00A6)	windowing	implicit	-	USER32.dll
DefWindowProcW	-	0x0001A50A	0x0001A50A	156 (0x009C)	windowing	implicit	-	USER32.dll
CreateWindowExW	-	0x0001A4CC	0x0001A4CC	110 (0x006E)	windowing	implicit	-	USER32.dll
ShowWindow	-	0x0001A4BE	0x0001A4BE	735 (0x02DF)	windowing	implicit	-	USER32.dll
UpdateWindow	-	0x0001A4AE	0x0001A4AE	785 (0x0311)	windowing	implicit	-	USER32.dll
RegisterClassExW	-	0x0001A480	0x0001A480	589 (0x024D)	windowing	implicit	-	USER32.dll
GetMessageW	-	0x0001A464	0x0001A464	349 (0x015D)	windowing	implicit	-	USER32.dll
TranslateMessage	-	0x0001A450	0x0001A450	764 (0x02FC)	windowing	implicit	-	USER32.dll
DispatchMessageW	-	0x0001A43C	0x0001A43C	175 (0x00AF)	windowing	implicit	-	USER32.dll
MoveWindow	-	0x0001A420	0x0001A420	539 (0x021B)	windowing	implicit	-	USER32.dll
SetWindowPos	-	0x0001A410	0x0001A410	710 (0x02C6)	windowing	implicit	-	USER32.dll
EnableWindow	-	0x0001A3EE	0x0001A3EE	216 (0x00D8)	windowing	implicit	-	USER32.dll
SetWindowLongW	-	0x0001A318	0x0001A318	708 (0x02C4)	windowing	implicit	-	USER32.dll
GetWindowLongW	-	0x0001A306	0x0001A306	406 (0x0196)	windowing	implicit	-	USER32.dll
GetActiveWindow	-	0x0001A2F4	0x0001A2F4	256 (0x0100)	windowing	implicit	-	USER32.dll
GetFocus	-	0x0001A2B0	0x0001A2B0	300 (0x012C)	windowing	implicit	-	USER32.dll

Figure 15 Graphics related imports

5. **Environment variable related:** The malware might be accessing environment variables through GetEnvironmentVariable and GetEnvironmentString functions.

GetEnvironmentVariableA	x	0x0001A7D2	0x0001A7D2	475 (0x01DB)	reconnaissance	implicit	-	KERNEL32.dll
GetEnvironmentStrings	x	0x0001A712	0x0001A712	472 (0x01D8)	execution	implicit	-	KERNEL32.dll
GetEnvironmentStringsW	x	0x0001A72A	0x0001A72A	474 (0x01DA)	execution	implicit	-	KERNEL32.dll

Figure 16 Environment variable related strings

6. **TLS related:** A program using thread-related functionality isn't suspicious by itself. But since this sample is a malware, TLS functions could be related to an attack where the malware stores malicious code in the thread's local storage.

TlsGetValue	-	0x0001A7B4	0x0001A7B4	1223 (0x04C7)	execution	implicit	-	KERNEL32.dll
TlsAlloc	-	0x0001A798	0x0001A798	1221 (0x04C5)	execution	implicit	-	KERNEL32.dll
TlsSetValue	-	0x0001A78A	0x0001A78A	1224 (0x04C8)	execution	implicit	-	KERNEL32.dll

Figure 17 TLS related imports

Suspicious or Significant Strings

The program has very few interesting strings other than the function names, manifest and version files. The following are the interesting strings from the program:

- **Favorite related strings** – These strings might be related to a browser. This shows that the malware might be manipulating the browser's data.

unicode	46	0x0287C9B8	-	format-string	-	Add page %s to favorites with (optional) name:
unicode	14	0x0287C90E	-	utility	-	Add a favorite

- **ManyBytes.exe:** The presence of this string in the program could act as an indicator of compromise.

unicode	13	0x0287CBE8	-	file	-	ManyBytes.exe

- **screenssanges:** The purpose of this weird looking string is not clear from static analysis.

unicode	13	0x02879BC8	-	-	-	screenssanges

- **guikas.txt:** This seems to be the name of a file the malware is tampering with.

unicode	10	0x02878704	-	file	-	guikas.txt

- **zolupalim:** Another weird string which could act as an indicator of compromise.

unicode	9	0x02879C14	-	-	-	zolupalim

- **C runtime error strings:**

R6027\r\n- not enough space for lowio initialization\r\n
R6026\r\n- not enough space for stdio initialization\r\n
R6024\r\n- not enough space for _onexit/atexit table\r\n
</dependentAssembly>\r\n</dependency>\r\n</assembly>
R6017\r\n- unexpected multithread lock error\r\n
R6016\r\n- not enough space for thread data\r\n
R6009\r\n- not enough space for environment\r\n
Add page %s to favorites with (optional) name:
R6008\r\n- not enough space for arguments\r\n
R6019\r\n- unable to open console device\r\n
name="Microsoft.Windows.Common-Controls"\r\n
R6025\r\n- pure virtual function call\r\n
!This program cannot be run in DOS mode.
R6028\r\n- unable to initialize heap\r\n
R6002\r\n- floating point not loaded\r\n
Copyright 2001-2014 all authors(GPLv3)
publicKeyToken="6595b64144ccf1df"\r\n
R6018\r\n- unexpected heap error\r\n

Figure 18 C runtime error strings

Dynamic Analysis

ProcMon

1. ProcMon logs show that the malware tries to read data from the following programs. The malware probably steals the credentials and data from these programs and sends it to a C&C server.
   a) **Browsers:** The malware tries to access data from these browsers – Firefox, Chrome, Chromium, ChromePlus, Bromium, Nichrome, Comodo, RockMelt, Opera

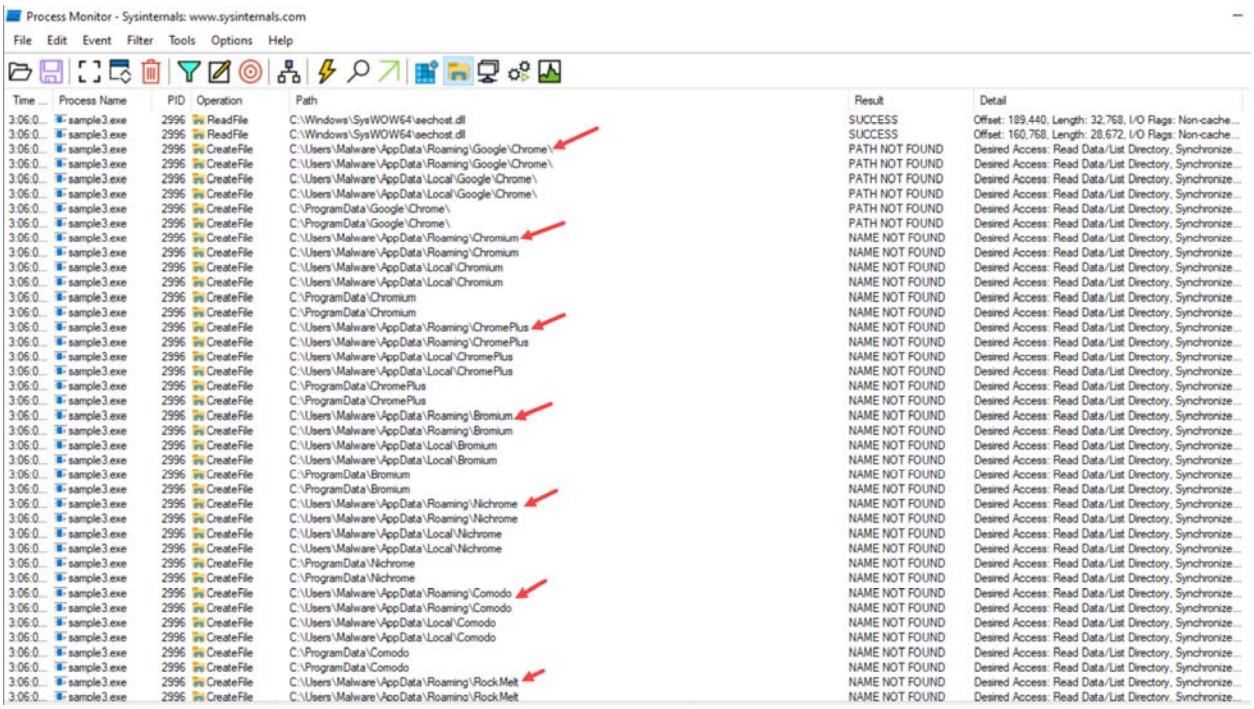


Figure 19 Browser data stolen by malware

   b) **FTP clients:** The malware tries to access data from these FTP clients – FileZilla, LeapFTP, 32BitFTP, BulletProof, SmartFTP, TurboFTP, FTP Explorer, FTPRush, BitKnex, ExpanDrive, FTPGetter, ALFTP, NetSarang, etc.

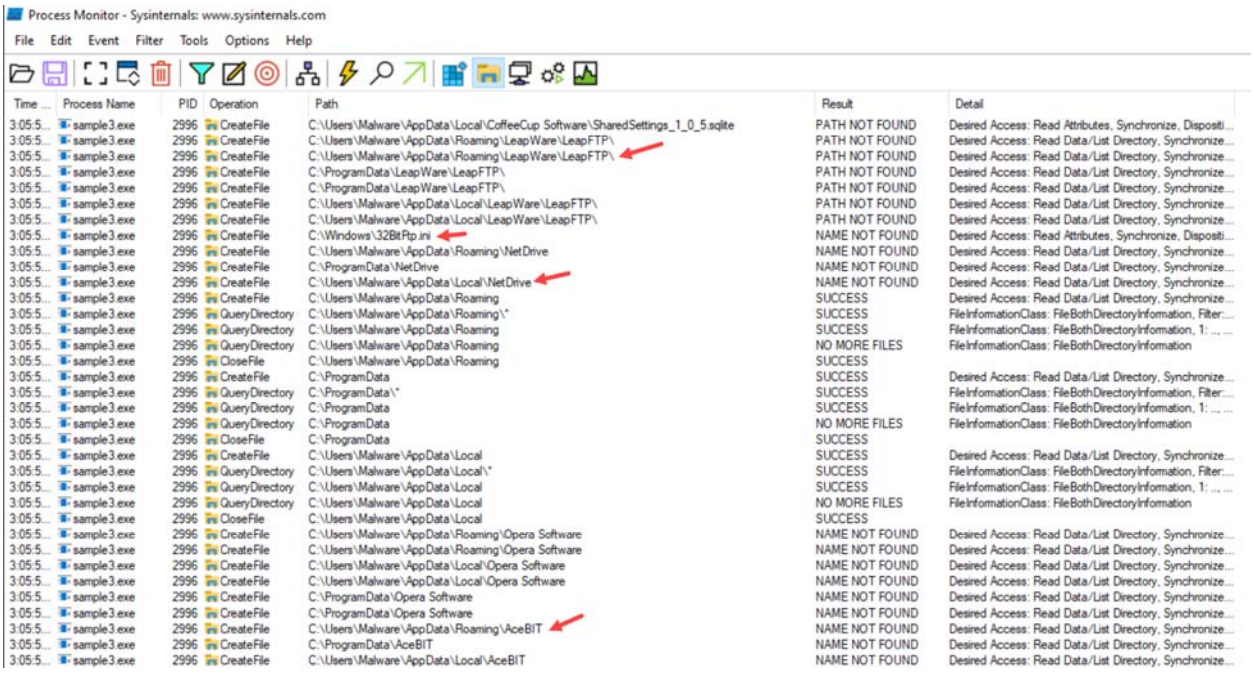
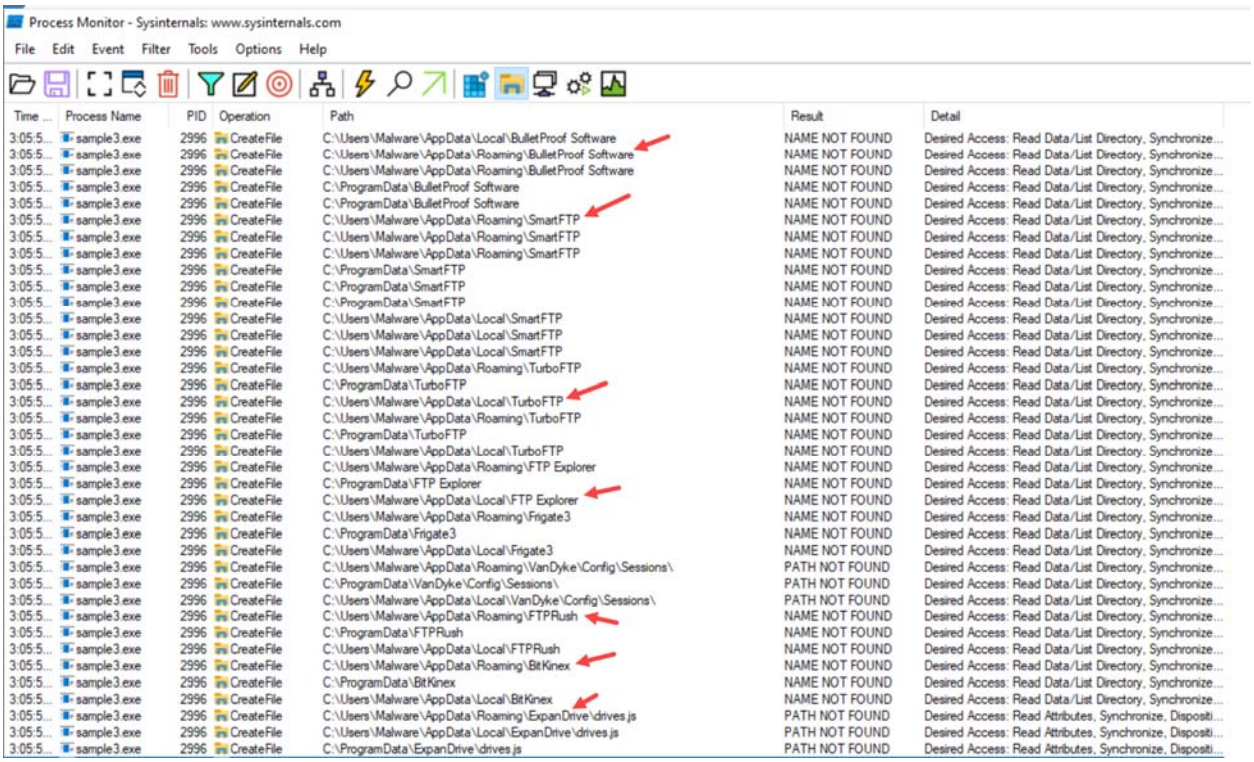


Figure 20 Some of the FTP clients targeted by the malware



c) **Email clients:** The malware tries to read data from the following email clients: Yandex, BatMail, The Bat!, Outlook, Pocomail, etc.

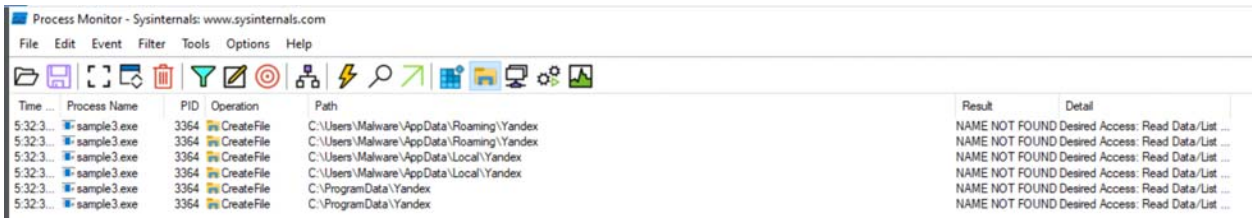


Figure 21 Yandex

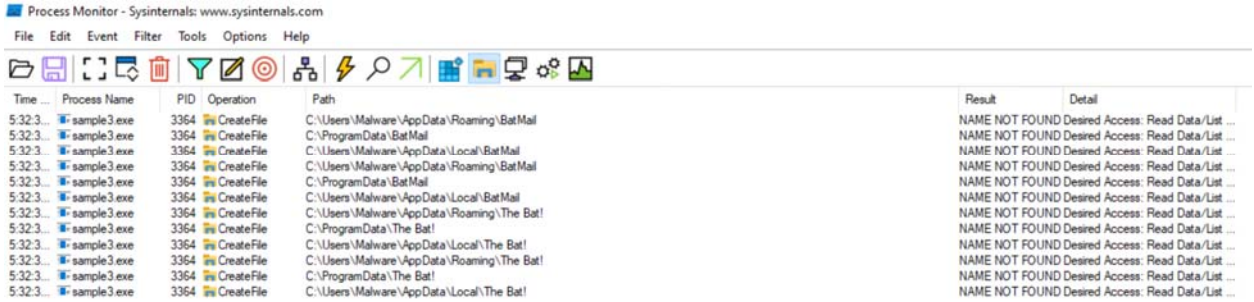
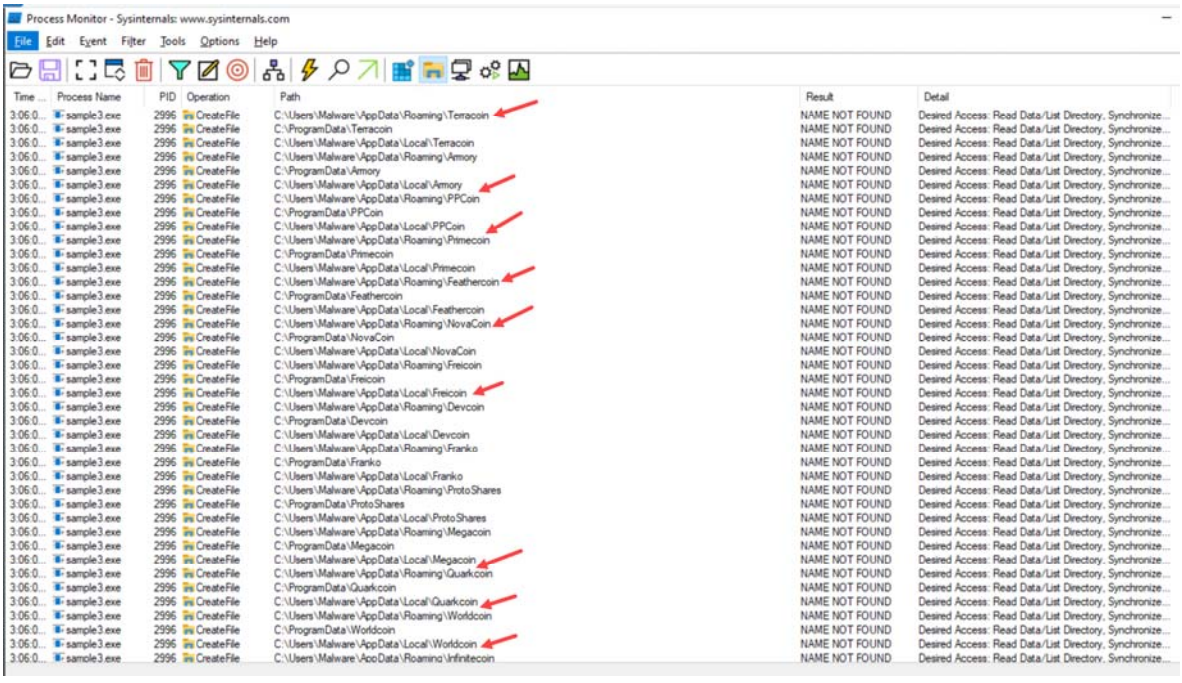


Figure 22 The Bat! and BatMail

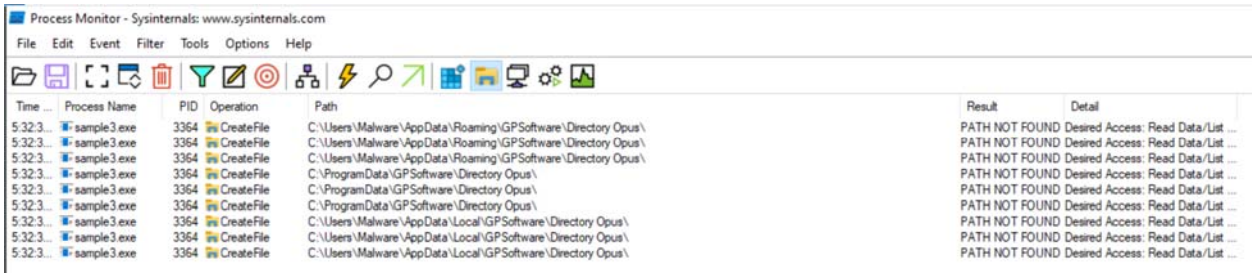
d) **Bitcoin wallets:** The malware might be trying to steal bitcoin wallet information from multiple wallets like Terracoin, Novacoin, Megacoin, etc.



e) **Other software:** Data from other software like Adobe, Directory Opus, Windows Commander and Total Commander is also accessed by the program.

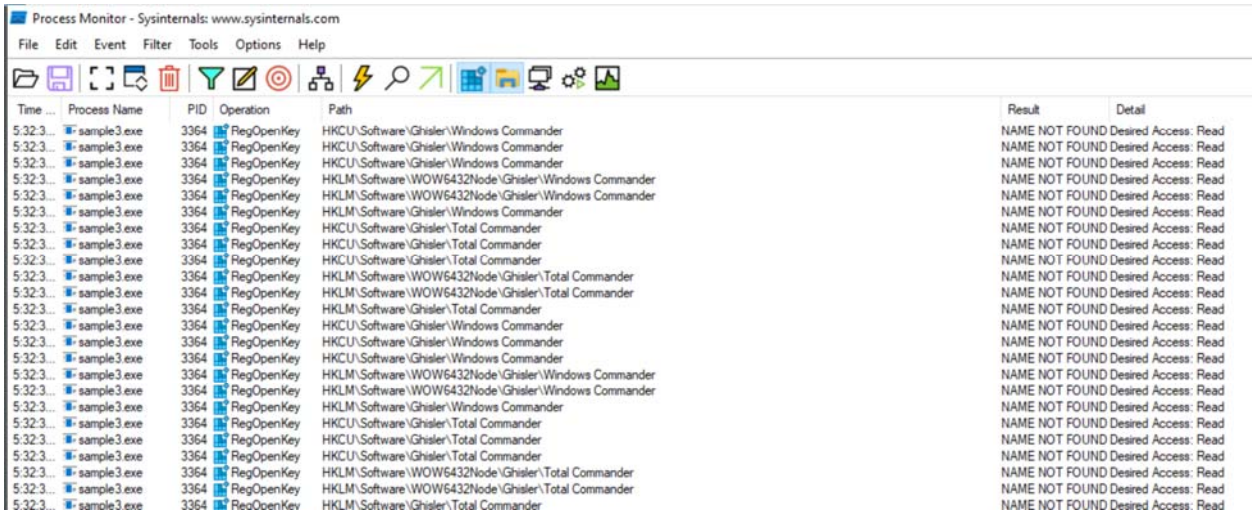
Time ...	Process Name	PID	Operation	Path	Result	Detail
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Adobe\Common	NAME NOT FOUND	Desired Access: Maximum Allowed

Figure 23 Adobe



Time ...	Process Name	PID	Operation	Path	Result	Detail
5:32:3...	sample3.exe	3364	CreateFile	C:\Users\Malware\AppData\Roaming\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\Users\Malware\AppData\Roaming\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\Users\Malware\AppData\Roaming\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\ProgramData\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\ProgramData\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\ProgramData\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\Users\Malware\AppData\Local\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\Users\Malware\AppData\Local\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...
5:32:3...	sample3.exe	3364	CreateFile	C:\Users\Malware\AppData\Local\GPSoftware\Directory Opus\	PATH NOT FOUND	Desired Access: Read Data/List ...

Figure 24 Directory Opus



Time ...	Process Name	PID	Operation	Path	Result	Detail
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\Ghisler\Windows Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKCU\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\WOW6432Node\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read
5:32:3...	sample3.exe	3364	RegOpenKey	HKLM\Software\Ghisler\Total Commander	NAME NOT FOUND	Desired Access: Read

Figure 25 Windows Commander and Total Commander

2. **Thread activity:** The sample creates and terminates a lot of threads.

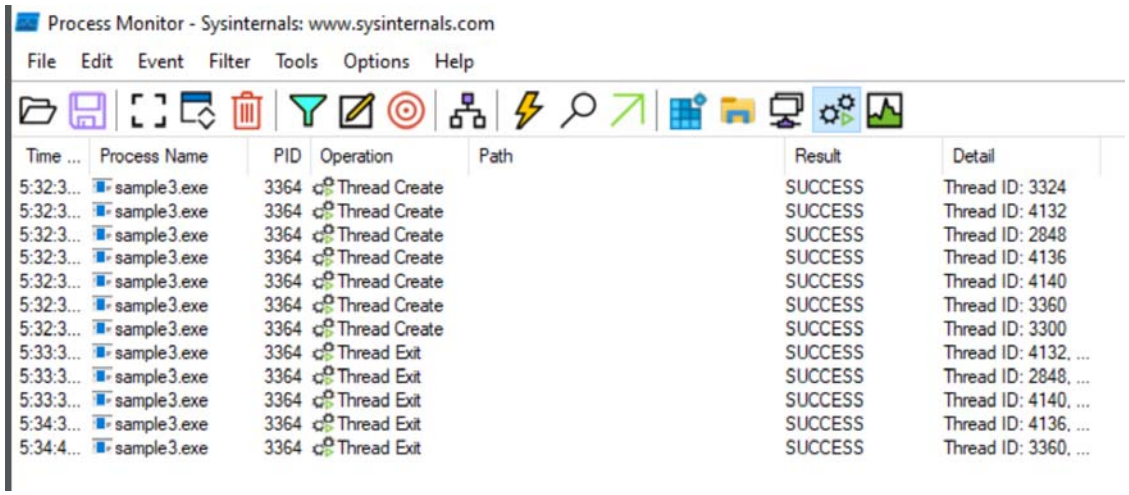


Figure 26 Program's thread activity

3. **Network activity:** The malware does some TCP communication with IP **229.165.192.5** on port **10015** (a non-standard port).

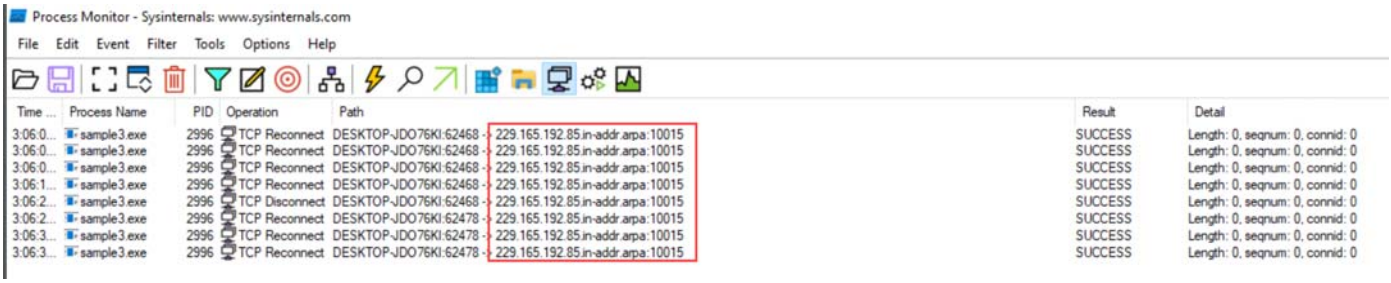


Figure 27 Malware connects to 229.165.192.5 on port 10015

- Capturing the packets sent by the malware using **accept-all-ips** and **netcat** shows that the malware posts some encrypted data to **/gate.php** at IP **85.192.165.229**. This data could be the credentials stolen by the malware that it sends to its C&C server.

```
remnux@remnux:~$ accept-all-ips start
OK, iptables will accept and redirect connections to all IPs on ens33.
Remember to set the client system's default gateway to IP of this REMnux host.
remnux@remnux:~$
```

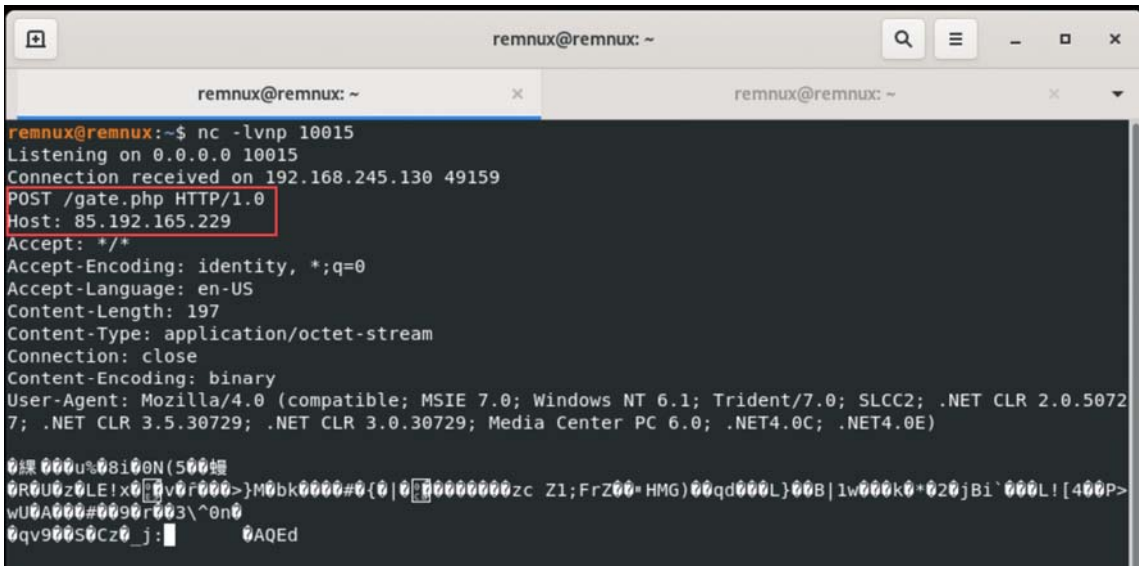


Figure 28 Malware posts encrypted data to /gate.php

- 85.192.165.229 is probably the C&C server of the malware. The **whois** details of 85.192.165.229 show a **Russian** location which hints that the malware might be Russian in origin.

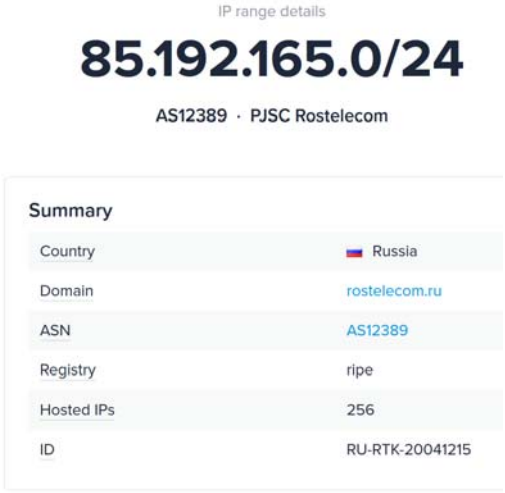


Figure 29 whois details of IP 85.192.165.229

4. **Registry activity:**
   - The malware reads and modifies registry keys related to Internet Explorer. It also manipulates the keys related Internet Explorer's security.

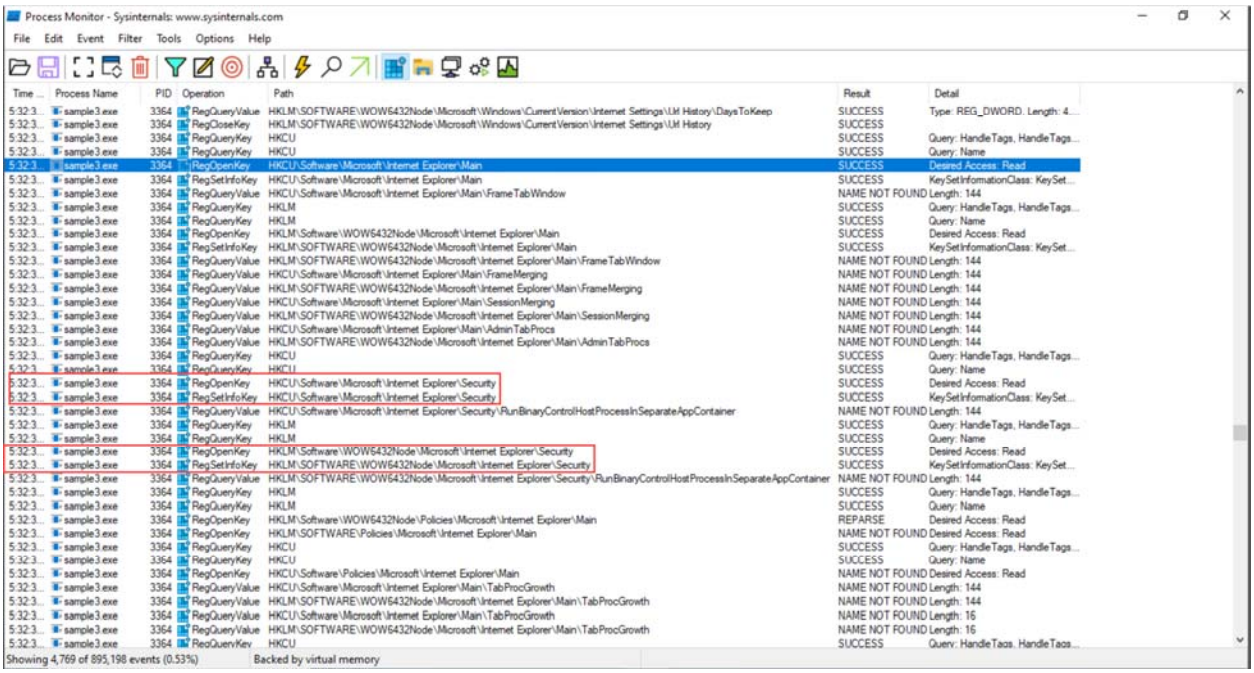


Figure 30 Malware modifies Internet Explorer related registry keys

- The malware also tries to access registry keys related to various other browsers, FTP clients and other utility software but it doesn't succeed as such software isn't installed on the VM we are using for analysis.

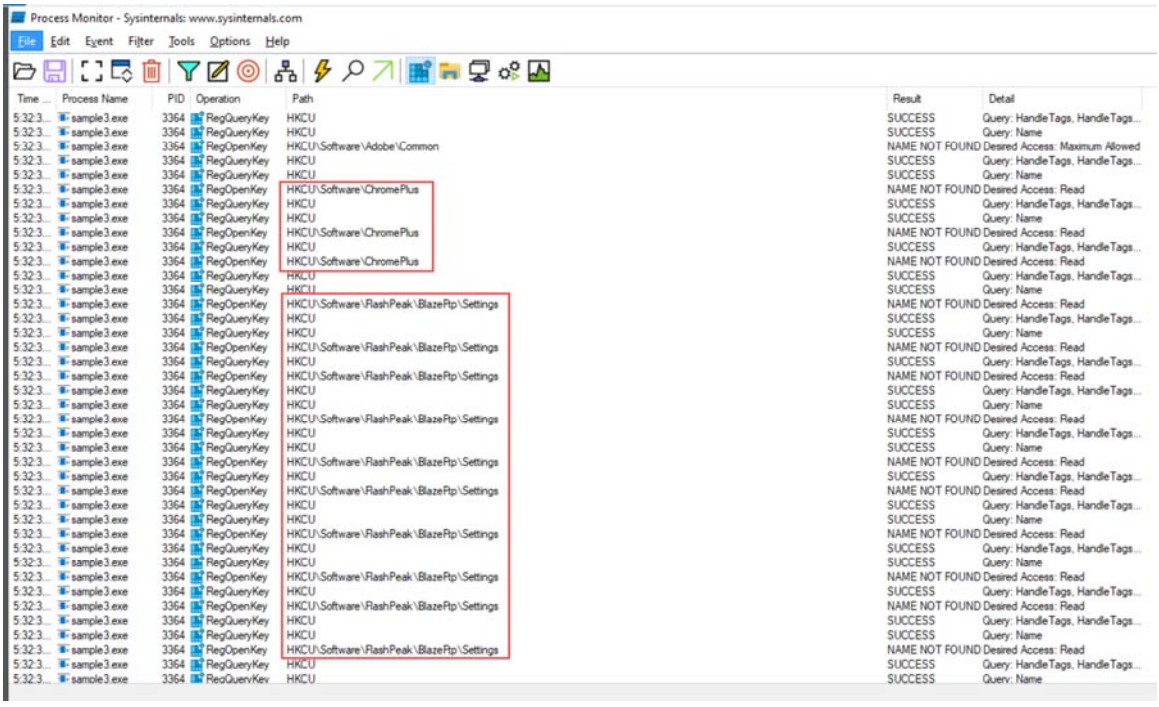


Figure 31 Malware tries to access registry keys for other browsers and FTP clients

- **Opening folders and reading files:** The malware seems to be iterating over all the directories and subdirectories and reading all the available files. It might be doing so in an attempt to exfiltrate data from the files.

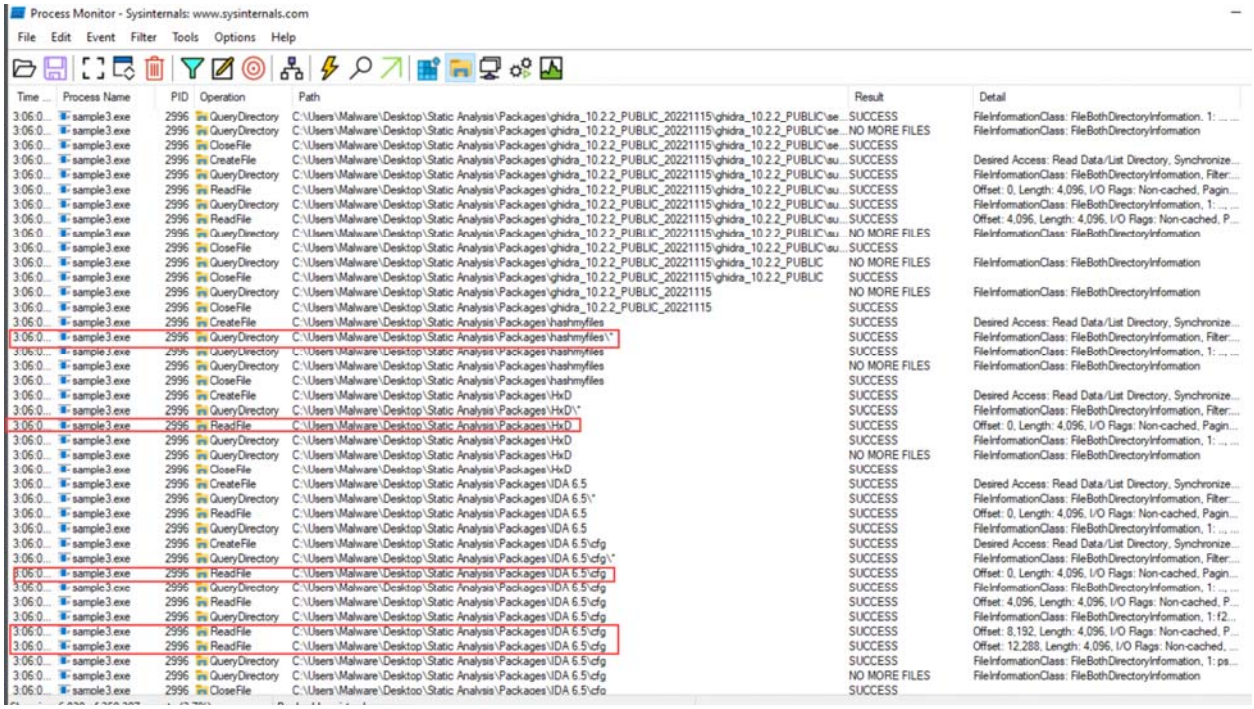


Figure 32 Sample tries to read all files on the Desktop

I added folders and files at multiple locations to check if the sample was opening them. It looks like the sample only looks through the directories on Desktop and Documents folder. It doesn't iterate over Program Files, Downloads, Windows or Users folders.

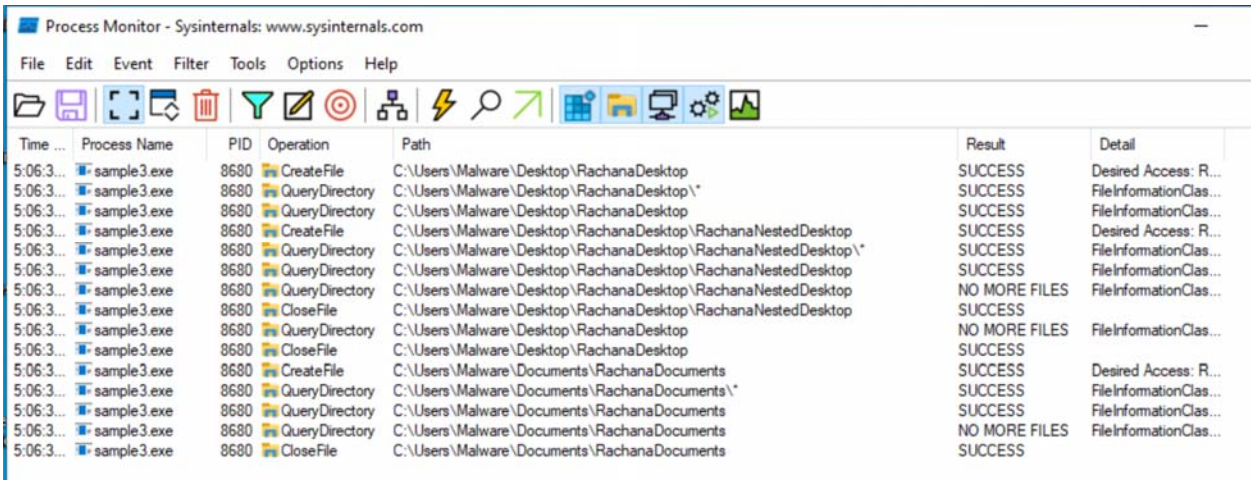


Figure 33 Added some folders to test which folders the malware accesses

Process Explorer

- **Mutants:** The malware process has 2 mutants. But these mutants are present in almost every Windows process so they don't make the process suspicious.

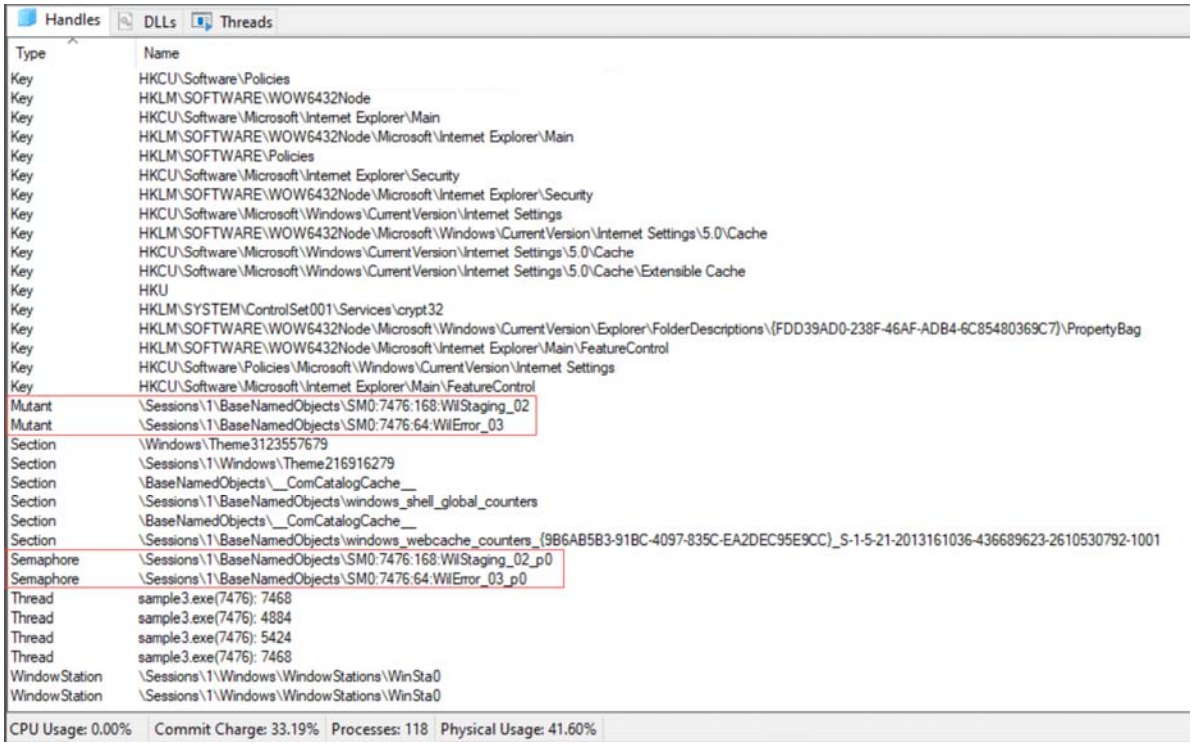


Figure 34 Mutants and Semaphores

- The malware doesn't seem to start any services or sub-processes.
- **DLLs:** The program loads the following DLLs. All of them except User32.dll, Shell32.dll and Kernel32.dll are loaded dynamically at runtime. Following are some of the suspicious DLLs loaded by the malware.
    - **Cryptography related DLLs:** The malware loads bcrypt.dll, bcryptprimitives.dll and crypt32.dll which contain cryptography related exports. This indicates that the program might be encrypting the data it steals before it sends it to its C&C server.
    - **Graphics related DLLs:** The sample loads user32.dll, shell32.dll, SHCore.dll, shlwapi.dll all of which are used to manipulate GUI elements like Windows, buttons and icons. Since no windows appear after the malware is run, it indicates that the malware might be trying to hide itself.
    - **Network related DLLs:** urlmon.dll, netapi32.dll, netutils.dll, winhttp.dll, wininet.dll, ws2_32.dll, wsock32.dll are used by the malware. They indicate that the malware does some network communication probably using HTTP. The malware might be trying to send some data to its command server.
    - **File and storage related DLLs:** The malware uses windows.storage.dll. This DLL provides functionality related to storage and file management. It contains functions to access and manipulate files and folders, including reading and writing data to

storage devices, and managing file permissions and attributes. The malware might be using the functions provided by this DLL to iterate over directories and steal information from files.

- **Environment variable related DLLs:** userenv.dll is loaded by the malware. This seems to be a very suspicious DLL as it exports functions to add and delete user profiles and change environment variables.
- **Rich text application DLLs:** Malware loads riched20.dll and riched32.dll. These DLLs provide functionality related to Rich Text Format (RTF) documents, such as editing and displaying formatted text in applications such as Microsoft Word, WordPad, and Outlook.

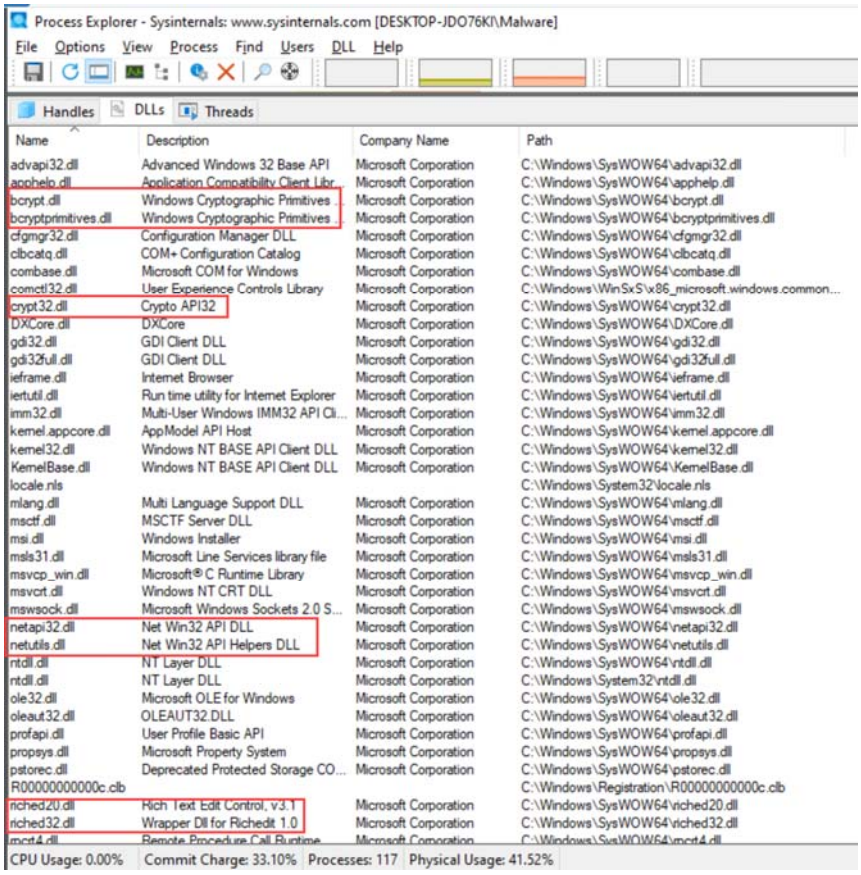


Figure 35 DLLs loaded by the sample

rpcrt4.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\Windows\SysWOW64\rpcrt4.dll
samcli.dll	Security Accounts Manager Client ...	Microsoft Corporation	C:\Windows\SysWOW64\samcli.dll
sample3.exe			C:\Users\Malware\Desktop\sample3\sample3.exe
sechost.dll	Host for SCM/SDDL/LSA Lookup ...	Microsoft Corporation	C:\Windows\SysWOW64\sechost.dll
secur32.dll	Security Support Provider Interface	Microsoft Corporation	C:\Windows\SysWOW64\secur32.dll
SHCore.dll	SHCORE	Microsoft Corporation	C:\Windows\SysWOW64\SHCore.dll
shell32.dll	Windows Shell Common Dll	Microsoft Corporation	C:\Windows\SysWOW64\shell32.dll
shlwapi.dll	Shell Light-weight Utility Library	Microsoft Corporation	C:\Windows\SysWOW64\shlwapi.dll
SortDefault.nls			C:\Windows\Globalization\Sorting\SortDefault.nls
srvcli.dll	Server Service Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\srvcli.dll
sspicli.dll	Security Support Provider Interface	Microsoft Corporation	C:\Windows\SysWOW64\sspicli.dll
ucrtbase.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\SysWOW64\ucrtbase.dll
urlmon.dll	OLE32 Extensions for Win32	Microsoft Corporation	C:\Windows\SysWOW64\urlmon.dll
user32.dll	Multi-User Windows USER API Cli...	Microsoft Corporation	C:\Windows\SysWOW64\user32.dll
userenv.dll	Userenv	Microsoft Corporation	C:\Windows\SysWOW64\userenv.dll
usp10.dll	Uniscribe Unicode script processor	Microsoft Corporation	C:\Windows\SysWOW64\usp10.dll
uxtheme.dll	Microsoft UxTheme Library	Microsoft Corporation	C:\Windows\SysWOW64\uxtheme.dll
version.dll	Version Checking and File Installati...	Microsoft Corporation	C:\Windows\SysWOW64\version.dll
win32u.dll	Win32u	Microsoft Corporation	C:\Windows\SysWOW64\win32u.dll
windows.storage.dll	Microsoft WinRT Storage API	Microsoft Corporation	C:\Windows\SysWOW64\windows.storage.dll
winhttp.dll	Windows HTTP Services	Microsoft Corporation	C:\Windows\SysWOW64\winhttp.dll
wininet.dll	Internet Extensions for Win32	Microsoft Corporation	C:\Windows\SysWOW64\wininet.dll
wkscli.dll	Workstation Service Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\wkscli.dll
wldp.dll	Windows Lockdown Policy	Microsoft Corporation	C:\Windows\SysWOW64\wldp.dll
wow64.dll	Win32 Emulation on NT64	Microsoft Corporation	C:\Windows\System32\wow64.dll
wow64cpu.dll	AMD64 Wow64 CPU	Microsoft Corporation	C:\Windows\System32\wow64cpu.dll
wow64win.dll	Wow64 Console and Win32 API L...	Microsoft Corporation	C:\Windows\System32\wow64win.dll
ws2_32.dll	Windows Socket 2.0 32-Bit DLL	Microsoft Corporation	C:\Windows\SysWOW64\ws2_32.dll
wsock32.dll	Windows Socket 32-Bit DLL	Microsoft Corporation	C:\Windows\SysWOW64\wsock32.dll

CPU Usage: 0.00% | Commit Charge: 33.10% | Processes: 117 | Physical Usage: 41.50%

Figure 36 DLLs loaded by the sample

Regshot

- Regshot analysis shows that the malware adds the registry key **HKU\<SID>\Software\WinRAR\HWID**

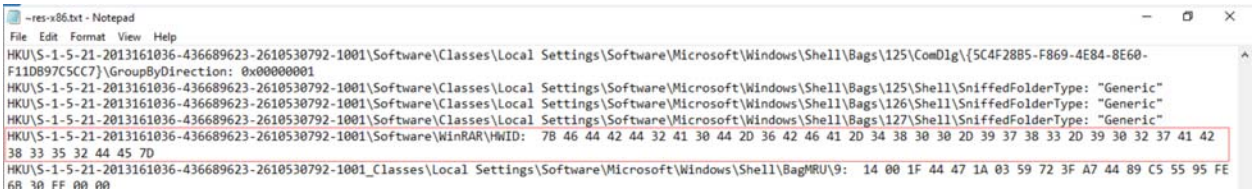


Figure 37 WinRAR registry key added by malware

- The malware adds the path to sample3.exe as a value for the key **HKU\<SID>\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store**

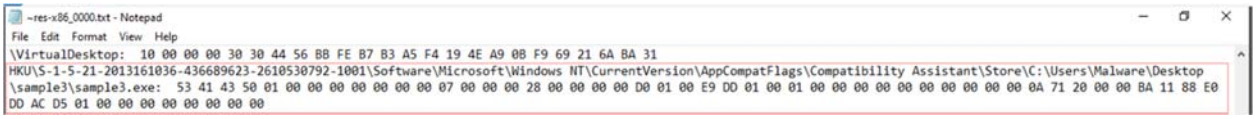
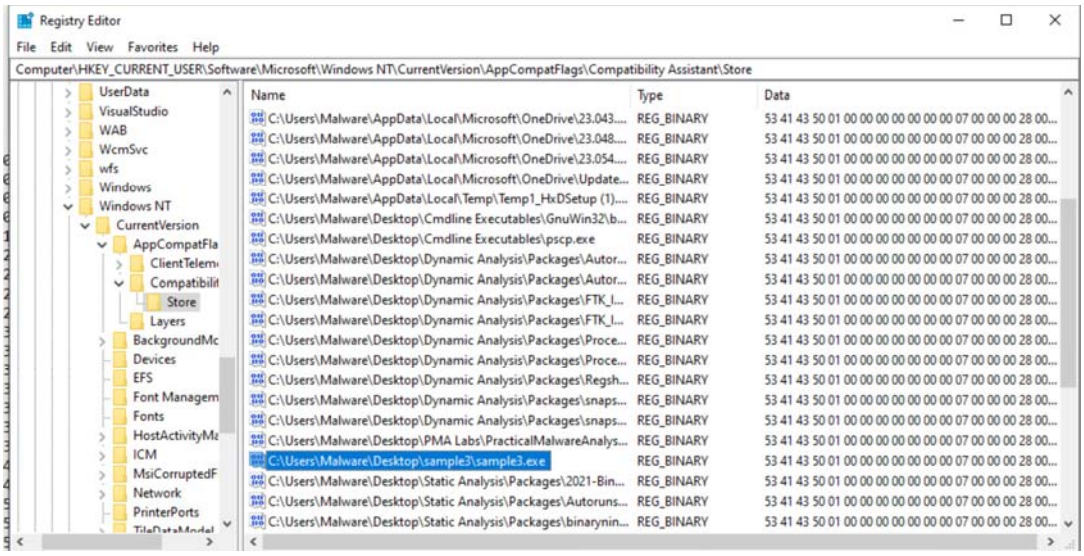


Figure 38 Value added for a registry key by the malware



- A lot of **shellbag** related registry keys are modified by the malware which indicates that it might be accessing multiple folders.

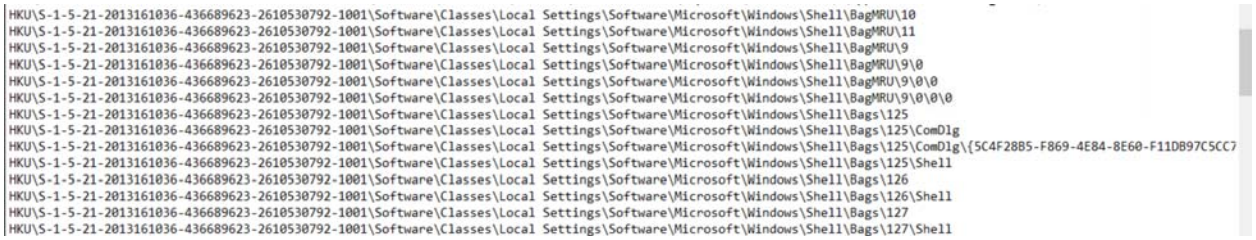


Figure 39 Shellbag related registry keys modified by the malware

Wireshark

- Wireshark is a tool used to capture and analyze packets.
- Other than some standard DNS queries, Wireshark shows TCP communication on port 10015.

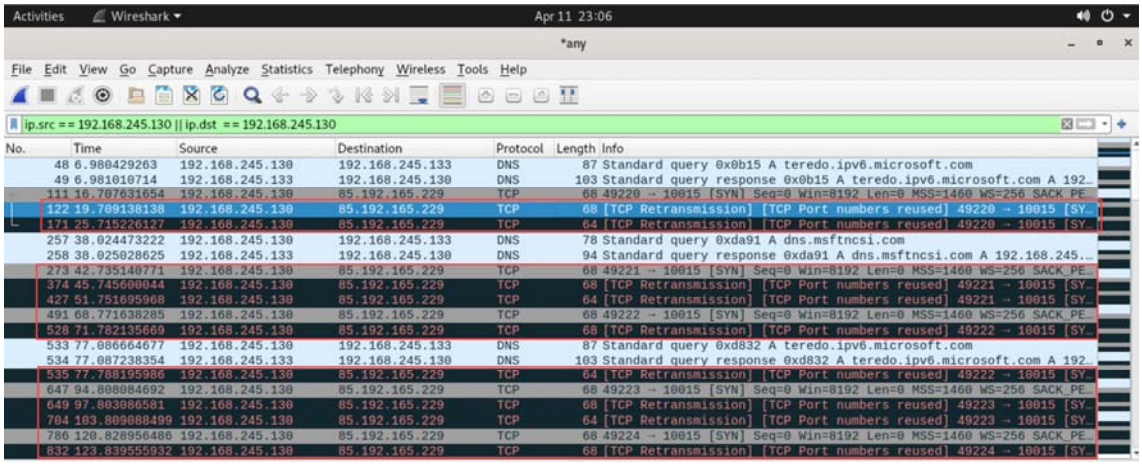


Figure 40 Wireshark shows suspicious TCP traffic on port 10015

- Analyzing the TCP frame further shows that the packet was destined for IP address 85.192.16.229 and destination port 10015. It originated from the Windows machine we were analyzing from port 49220. These are not standard ports.
- The data being sent seems to be encrypted.
- The malware's CnC server must be at 85.192.16.229 and it must be exfiltrating data to it. The trojan is trying to hide its communication by communicating on a non-standard port.

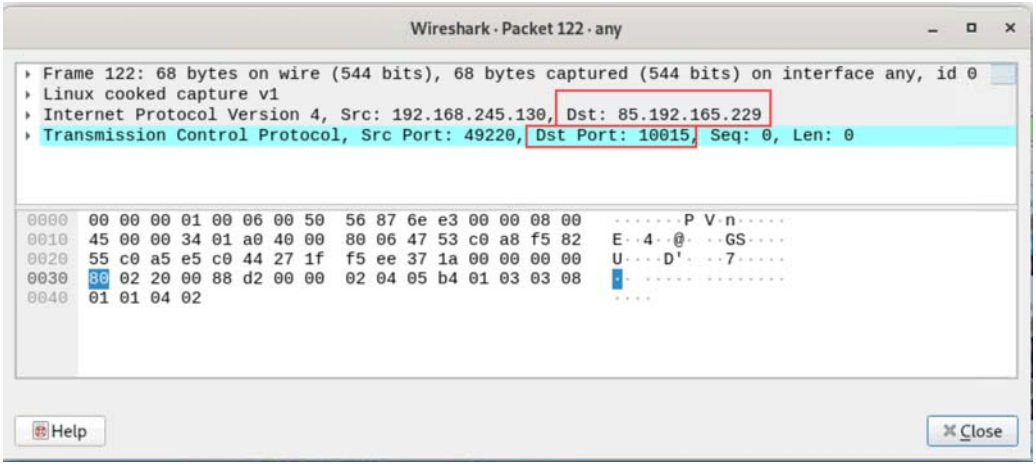


Figure 41 The malware exfiltrates encrypted data

FakeDNS

FakeDNS is a command line utility on Remnux which listens to DNS resolution requests and resolves them by sending the requester its own IP address. As can be seen from the following screenshot, the Windows10 machine on which the malware sample was run makes quite a few DNS requests. Requests to ctldl.windowsupdate.com, slscr.update.microsoft.com, config.teams.microsoft.com, oneclient.sfx.ms, etc. are all requests to legitimate domains for services like Windows Update, Email and Microsoft Teams which are used by most Windows machines. Therefore, we can conclude that the malware makes no DNS requests to resolve any suspicious domains.

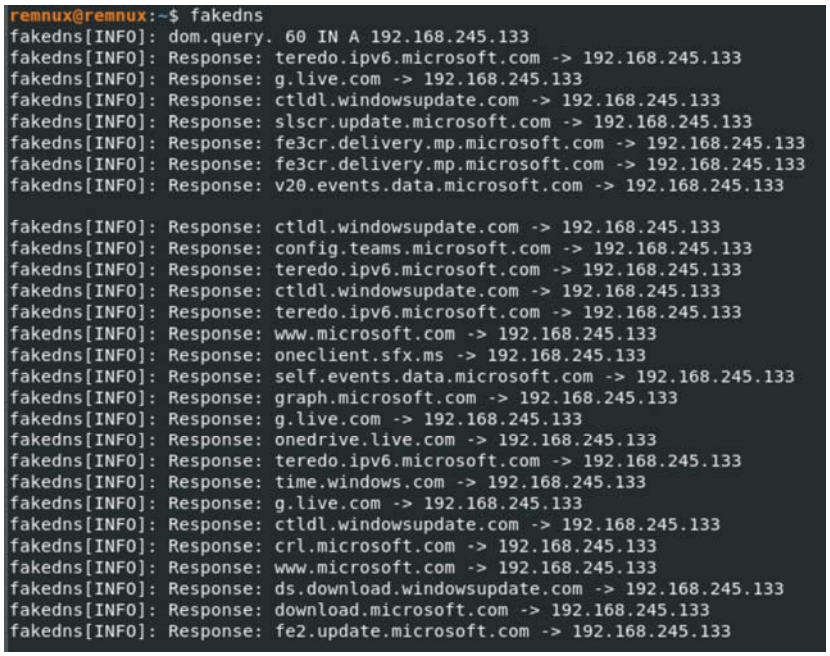


Figure 42 FakeDNS results don't show anything suspicious

Inetsim

InetSim is a command line utility on Remnux that can simulate a range of Internet services, including HTTP, FTP and SMTP. It is used to create a safe environment for malware analysis, honeypot deployment, and general network testing. When a client tries to access any of these services, InetSim replies with a dummy response in the format the client expects. For example, for HTTP GET requests, InetSim responds with a dummy HTML webpage. As can be seen from the screenshot below, the Windows10 machine on which the malware was run makes only two types of requests: an HTTP request to authrootstl.cab service and another HTTP request to disallowedcertstl.cab service. Both these requests are standard and do not indicate any malicious activity.

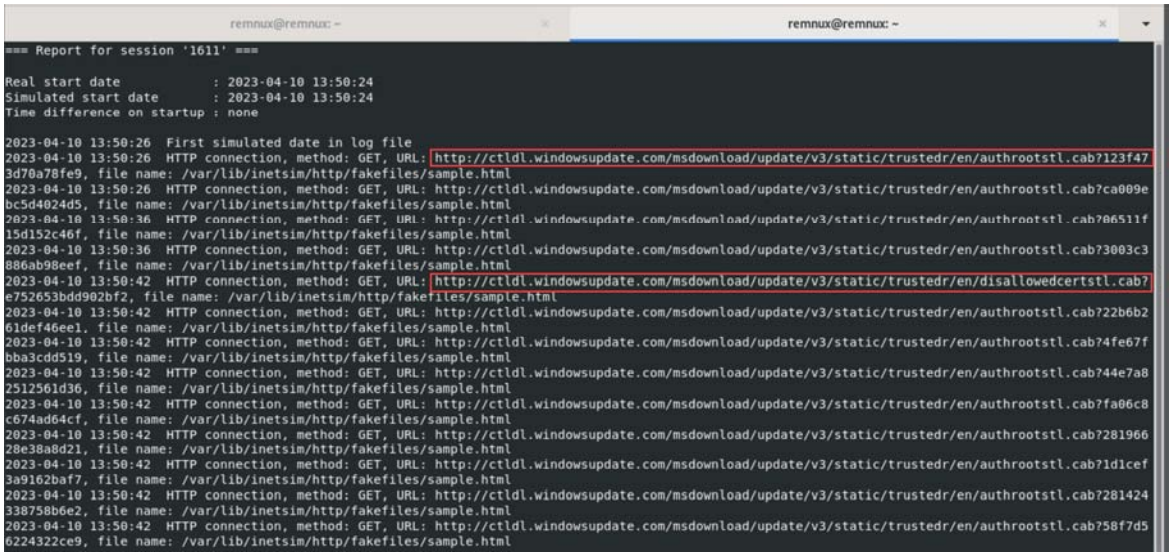


Figure 43 InetSim logs don't show anything suspicious

Autoruns

Autoruns is a Microsoft tool which allows us to view and manage all the programs which run on startup. After running sample3.exe, Autoruns doesn't show anything suspicious. Malware usually sets up persistence by adding a value to the HKCU or HKLM Run registry key but this sample doesn't do that. Hence, we can conclude that that **malware is not persistent**.

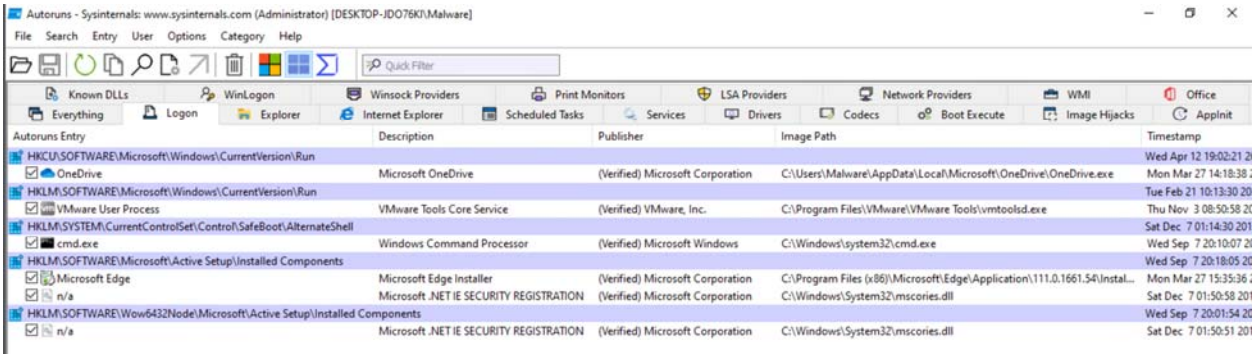


Figure 44 No signs of persistence shown by Autorun

Volatility

- I ran the sample, suspended its process and took a memory dump using Volatility. Then I extracted the process from this memory dump using Volatility's **pslist** command.

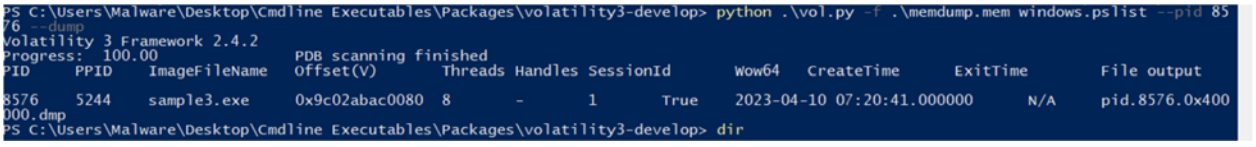


Figure 45 Extracting the process from memory dump using Volatility

- Analyzing the extracted process in PEStudio revealed a lot of information. The signature now shows UPX which is a packer. The compilation timestamp stays the same. The entropy is now reduced to 5.738.

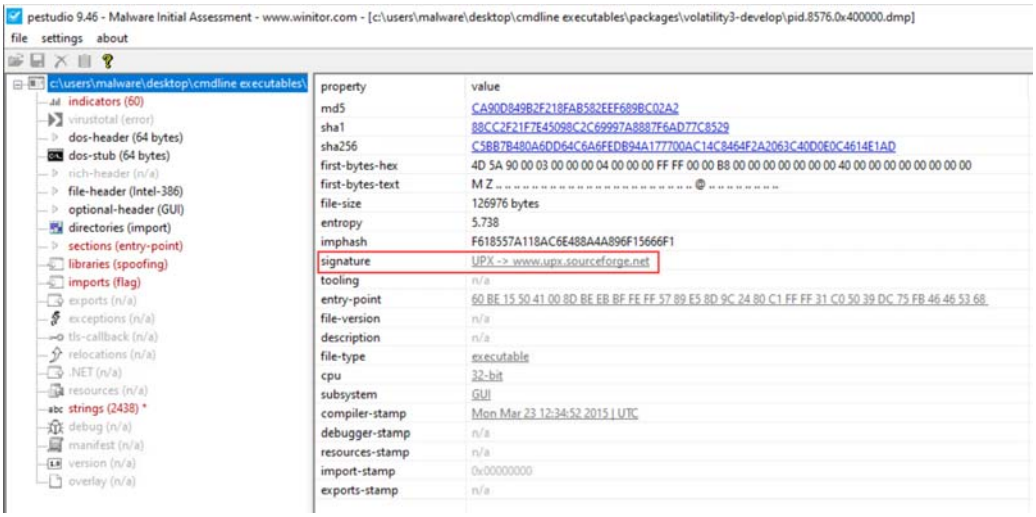


Figure 46 Program signature changed to UPX

- The names of the PE sections have also changed in the dump file.

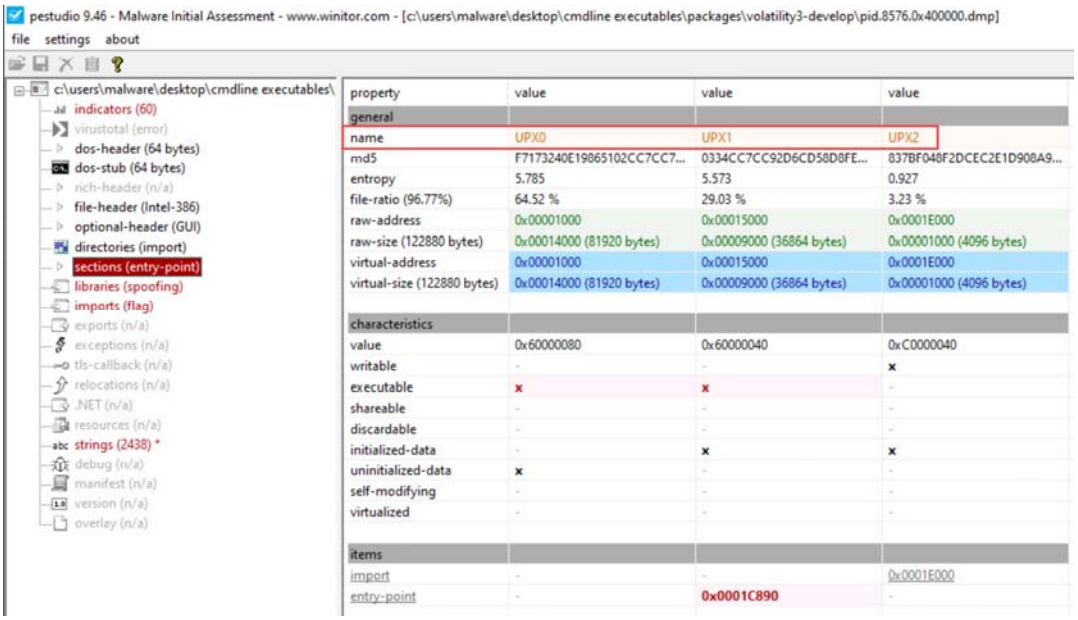


Figure 47 New section names

- **Libraries list:** The dumped file imports libraries like urlmon.dll, wininet.dll and wsock32.dll statically. These are network libraries and indicate that the malware is communicating with some other machine to exfiltrate data, infect other machines or receive commands from a C&C server.

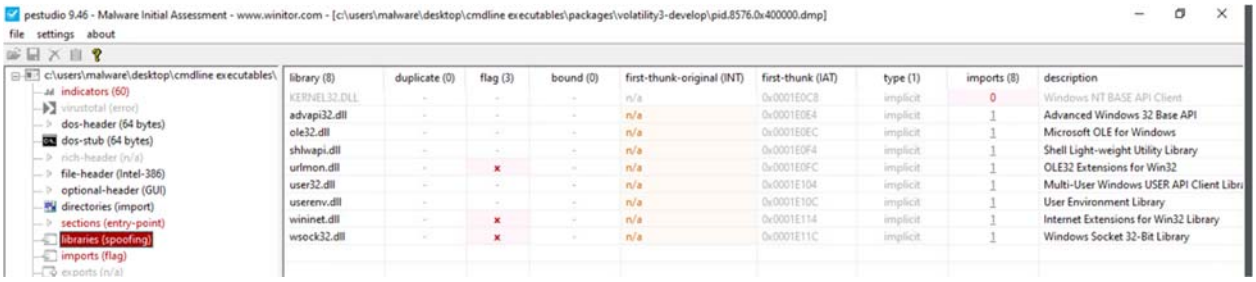


Figure 48 Imported libraries

- **Imported functions list:** The imports list is quite small and all these imports were not present in the original .exe. Following functions are suspicious:
    - *LoadUserProfile*: Loads a local or roaming user's profile in the current process. The malware might be using this for privilege escalation.
    - *ObtainUserAgentString*: This function fetches the UserAgent from the HTTP request. The malware might be using this to detect which application the request is coming from. It might be using this in some function which gets commands from the C&C server.
    - *InternetCrackUrl*: This function breaks a URL into its components. This indicates that the malware does some HTTP networking operations.
    - *Send*: This function is used to send data to another machine. The malware probably uses this to exfiltrate data to its command server.

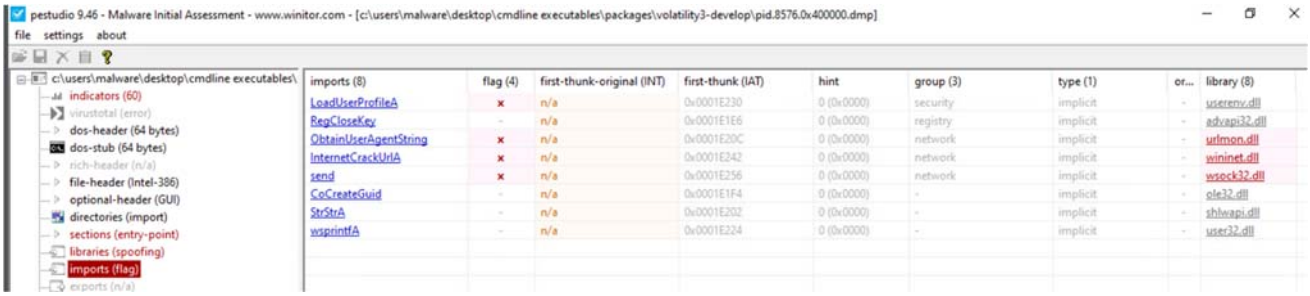


Figure 49 Imported functions

Interesting strings in the dump file

The dump file has a ton of interesting strings.

1. **File operations:** These strings indicate that the malware iterates over the file system and reads and writes files to steal data. CreaeFileMapping and MapViewOfFile are suspicious as malware authors often use them to load a file into memory and perform operations at arbitrary locations.

ascii	15	0x000154F6	-	-	file	SHGetFolderPath
ascii	10	0x0001B009	-	-	file	CreateFile
ascii	8	0x0001B016	-	-	file	ReadFile
ascii	9	0x0001B02D	x	-	file	WriteFile
ascii	17	0x0001B095	-	-	file	GetFileAttributes
ascii	11	0x0001B0C4	-	-	file	GetFileSize
ascii	17	0x0001B0D1	-	-	file	CreateFileMapping
ascii	13	0x0001B0E5	x	-	file	MapViewOfFile
ascii	15	0x0001B0F4	x	-	file	UnmapViewOfFile
ascii	11	0x0001B123	-	-	file	GetTempPath
ascii	15	0x0001B131	-	-	file	CreateDirectory
ascii	10	0x0001B143	x	-	file	DeleteFile
ascii	13	0x0001B1D6	x	-	file	FindFirstFile
ascii	12	0x0001B1F1	x	-	file	FindNextFile
ascii	9	0x0001B200	-	-	file	FindClose

2. **Process related strings:** These strings indicate that the malware iterates through all the active processes. The malware might also be creating another process through no such process was seen during dynamic analysis.

ascii	19	0x00015356	x	-	execution	CreateProcessAsUser
ascii	22	0x0001543A	x	-	execution	CreateEnvironmentBlock
ascii	23	0x00015451	x	-	execution	DestroyEnvironmentBlock
ascii	12	0x00018768	x	-	execution	ShellExecute
ascii	24	0x0001B190	x	-	execution	CreateToolhelp32Snapshot
ascii	14	0x0001B1AA	x	-	execution	Process32First
ascii	11	0x0001B1BA	x	-	execution	OpenProcess
ascii	13	0x0001B1C7	x	-	execution	Process32Next

3. **Network related strings:** These provide more evidence that the malware does some network activity and probably exfiltrates data.

ascii	16	0x000151E8	x	-	network	NetApiBufferFree
ascii	11	0x000151F9	x	-	network	NetUserEnum
ascii	17	0x0001B53B	x	-	network	InternetCreateUrl
ascii	9	0x0001B558	x	-	network	inet_addr
ascii	13	0x0001B563	x	-	network	gethostbyname
ascii	6	0x0001B572	x	-	network	socket
ascii	11	0x0001B583	x	-	network	closesocket
ascii	4	0x0001B59E	x	-	network	recv
ascii	10	0x0001B5A4	x	-	network	setsockopt
ascii	10	0x0001B5B0	x	-	network	WSAStartup
ascii	12	0x000151DB	-	file	network	netapi32.dll
ascii	21	0x0001B4C6	x	import	network	ObtainUserAgentString
ascii	16	0x0001B528	x	import	network	InternetCrackUrl
ascii	21	0x0001E20E	x	import	network	ObtainUserAgentString
ascii	16	0x0001E244	x	import	network	InternetCrackUrl
ascii	10	0x0001E154	-	library	network	urlmon.dll
ascii	11	0x0001E176	-	library	network	wininet.dll
ascii	11	0x0001E182	-	library	network	wsock32.dll
ascii	7	0x0001B57A	x	utility	network	connect
ascii	4	0x0001B590	x	utility	network	send
ascii	6	0x0001B596	x	utility	network	select
ascii	4	0x0001E258	x	utility	network	send

4. **Memory related strings:** Strings like VirtualProtect and VirtualAlloc are interesting as malware often uses these functions to change the privileges of a memory section to make it executable or allocate memory in a process for process injection.

ascii	10	0x0001B042	-	-	memory	GlobalLock
ascii	12	0x0001B04E	-	-	memory	GlobalUnlock
ascii	9	0x0001B05C	-	-	memory	LocalFree
ascii	10	0x0001B067	-	-	memory	LocalAlloc
ascii	13	0x0001B446	-	-	memory	CoTaskMemFree
ascii	14	0x0001E1AE	x	-	memory	VirtualProtect
ascii	12	0x0001E1BE	-	-	memory	VirtualAlloc
ascii	11	0x0001E1CC	-	-	memory	VirtualFree

5. **Registry related strings:** Show that the malware modifies the registry.

ascii	23	0x0001B261	-	-	registry	GetPrivateProfileString
ascii	29	0x0001B291	x	-	registry	GetPrivateProfileSectionNames
ascii	12	0x0001B359	-	-	registry	RegOpenKeyEx
ascii	15	0x0001B368	-	-	registry	RegQueryValueEx
ascii	10	0x0001B387	-	-	registry	RegOpenKey
ascii	12	0x0001B394	-	-	registry	RegEnumKeyEx
ascii	12	0x0001B3A3	x	-	registry	RegCreateKey
ascii	13	0x0001B3B2	x	-	registry	RegSetValueEx
ascii	18	0x0001B3D1	x	-	registry	RegOpenCurrentUser
ascii	12	0x0001B3E5	-	-	registry	RegEnumValue
ascii	11	0x0001B37A	-	import	registry	RegCloseKey
ascii	11	0x0001E1E8	-	import	registry	RegCloseKey

6. **Possible anti-analysis related strings:** The malware might be getting the information about the system using GetNativeSystemInfo as an anti-VM mechanism. GetTickCount is suspicious as it is often used for anti-debugging.

ascii	19	0x000158E5	x	-	reconnaissance	GetNativeSystemInfo
ascii	12	0x0001B073	-	-	reconnaissance	GetTickCount
ascii	12	0x0001B21D	-	-	reconnaissance	GetVersionEx
ascii	13	0x0001B23C	-	-	reconnaissance	GetSystemInfo
ascii	19	0x0001B24B	-	-	reconnaissance	GetWindowsDirectory
ascii	11	0x0001B3F4	-	-	reconnaissance	GetUserName

7. **Cryptography related strings:** All these cryptography related functions indicate that the sample probably encrypts the data it sends out on the network.

ascii	13	0x00015263	x	-	cryptography	CredEnumerate
ascii	8	0x00015272	x	-	cryptography	CredFree
ascii	15	0x0001527B	x	-	cryptography	CryptGetUserKey
ascii	14	0x0001528B	x	-	cryptography	CryptExportKey
ascii	15	0x0001529A	x	-	cryptography	CryptDestroyKey
ascii	19	0x000152AA	x	-	cryptography	CryptReleaseContext
ascii	18	0x00015378	x	-	cryptography	CryptUnprotectData
ascii	19	0x0001538B	x	-	cryptography	CertOpenSystemStore
ascii	27	0x000153A0	x	-	cryptography	CertEnumCertificatesInStore
ascii	14	0x000153BC	x	-	cryptography	CertCloseStore
ascii	33	0x000153CB	x	-	cryptography	CryptAcquireCertificatePrivateKey
ascii	20	0x00015418	x	-	cryptography	PStoreCreateInstance
ascii	11	0x0001536C	-	file	cryptography	crypt32.dll
ascii	11	0x0001540C	-	file	cryptography	pstorec.dll

8. **Password strings:** The malware has a lot of readable strings which look like common passwords and names of people and common things. These probably are values the malware tries for cracking passwords.

encoding (2)	size (bytes)	location	flag (...	label (297)	g	value (2438)
ascii	6	0x00018135	-	-	-	cassie
ascii	7	0x0001813C	-	-	-	victory
ascii	8	0x00018144	-	-	-	passw0rd
ascii	6	0x0001814D	-	-	-	foobar
ascii	8	0x00018154	-	-	-	ilovegod
ascii	6	0x0001815D	-	-	-	nathan
ascii	6	0x00018164	-	-	-	blabla
ascii	7	0x0001816B	-	-	-	digital
ascii	7	0x00018173	-	-	-	peaches
ascii	9	0x0001817B	-	-	-	football1
ascii	8	0x00018185	-	-	-	11111111
ascii	5	0x0001818E	-	-	-	power
ascii	7	0x00018194	-	-	-	thunder
ascii	7	0x0001819C	-	-	-	gateway
ascii	9	0x000181A4	-	-	-	iloveyou!
ascii	8	0x000181AE	-	-	-	football
ascii	6	0x000181B7	-	-	-	tigger
ascii	8	0x000181BE	-	-	-	corvette
ascii	5	0x000181C7	-	-	-	angel
ascii	6	0x000181CD	-	-	-	killer
ascii	8	0x000181D4	-	-	-	creative
ascii	9	0x000181DD	-	-	-	123456789
ascii	6	0x000181E7	-	-	-	google
ascii	7	0x000181EE	-	-	-	zxcvbnm
ascii	8	0x000181F6	-	-	-	startrek
ascii	6	0x000181FF	-	-	-	ashley
ascii	6	0x00018206	-	-	-	cheese
ascii	8	0x0001820F	-	-	-	sunshine
ascii	6	0x00018218	-	-	-	christ
ascii	6	0x0001821F	-	-	-	000000
ascii	6	0x00018226	-	-	-	soccer
ascii	7	0x0001822D	-	-	-	qwerty1

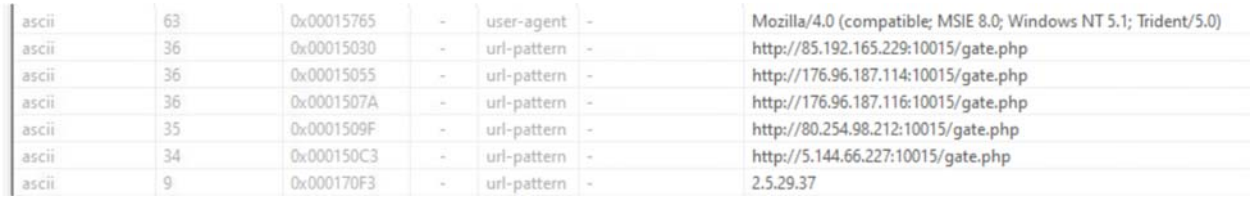
9. **FTP, SMTP, POP3 and browser related strings:** These hint at the malware stealing credentials and data from FTP, email client and browser applications.

value (2438)
HostName
FTP CONTROL
WinFTP
hostname
FTP Now
FTP Count
FTP File%d
HostName
FTP destination server
FTP destination user
FTP destination password
FTP destination port
FTP destination catalog
FTP profiles
Program
Dir #%d
SMTP Email Address
SMTP Server
POP3 Server
POP3 User Name
SMTP User Name
POP3 User
SMTP User
POP3 Port
SMTP Port
POP3 Password2
SMTP Password2
POP3 Password
SMTP Password

value (2438)
Favorites.dat
History.dat
addrbk.dat
quick.dat
profiles.xml
FtpSite.xml
.ini
\sites.xml
RushSite.xml
\drives.js
ftplast.osd
\SharedSettings.sqlite
\SharedSettings_1_0_5.sqlite
unleap.exe
sites.dat
sites.ini
\32BitFtp.ini
NDSites.ini
.prf
wand.dat
wiseftpsrvs.bin
wiseftpsrvs.ini
wiseftp.ini
FTPVoyager.ftp
FTPVoyager.ftp.backup
FTPVoyager.ftp.old.backup
FTPVoyager.qc
\RhinoSoft.com
nss3.dll
profiles.ini
prefs.js
signons.sqlite

Software\BulletProof Software\BulletProof FTP Client\Main
Software\BPFTP\Bullet Proof FTP\Options
Software\BulletProof Software\BulletProof FTP Client\Options
Software\BPFTP
Software\TurboFTP
Software\Sota\FFFTP
Software\Sota\FFFTP\Options
Software\CoffeeCup Software\Internet\Profiles
Software\FTPWare\COREFTP\Sites
Software\FTP Explorer\FTP Explorer\Workspace\MFCToolBar-224
Software\FTP Explorer\Profiles
Software\VanDyke\SecureFX
Software\Cryer\WebSitePublisher
Software\ExpanDrive\Sessions
Software\ExpanDrive
Software\NCH Software\ClassicFTP\FTPAccounts
SOFTWARE\NCH Software\Fling\Accounts
Software\FTPClient\Sites
Software\SoftX.org\FTPClient\Sites
SOFTWARE\LeapWare
Software\Martin Prikryl
Software\South River Technologies\WebDrive\Connections
Software\AceBIT
SOFTWARE\Classes\TypeLib\{CB1F2C0F-8094-4AAC-BCF5-41A64E27F777}
SOFTWARE\Classes\TypeLib\{9EA55529-E122-4757-BC79-E4825F80732C}
Software\Mozilla
Software\LeechFTP
Software\Microsoft\Internet Explorer\IntelliForms\Storage2
Software\Adobe\Common
Software\ChromePlus

10. **URL strings:** The malware might be exfiltrating data to these URLs and using Firefox as the UserAgent.



ascii	63	0x00015765	-	user-agent	-	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/5.0)
ascii	36	0x00015030	-	url-pattern	-	http://85.192.165.229:10015/gate.php
ascii	36	0x00015055	-	url-pattern	-	http://176.96.187.114:10015/gate.php
ascii	36	0x0001507A	-	url-pattern	-	http://176.96.187.116:10015/gate.php
ascii	35	0x0001509F	-	url-pattern	-	http://80.254.98.212:10015/gate.php
ascii	34	0x000150C3	-	url-pattern	-	http://5.144.66.227:10015/gate.php
ascii	9	0x000170F3	-	url-pattern	-	2.5.29.37

There's also a POST request string. The sample must be using this to send data to its C&C server.

ascii	245	0x0001578F	-	format-st...	-	POST %s HTTP/1.0\r\nHost: %s\r\nAccept: */*\r\nAccept-Encoding: identity, *;q=0\r\nAccept-Language: en-US...

There is a string pointing to Facebook's URL.

unicode	25	0x030B6B78	-	-	-	2http://www.facebook.com/

Ghidra and x32dbg analysis

- **Populates the memory with ASCII characters:** In the function at memory location 415af8, the malware has a loop that populates the memory with a series of ASCII characters representing digits, upper and lowercase letters and special characters. The malware might be using these in combination with the GetKeyState function to do keylogging.

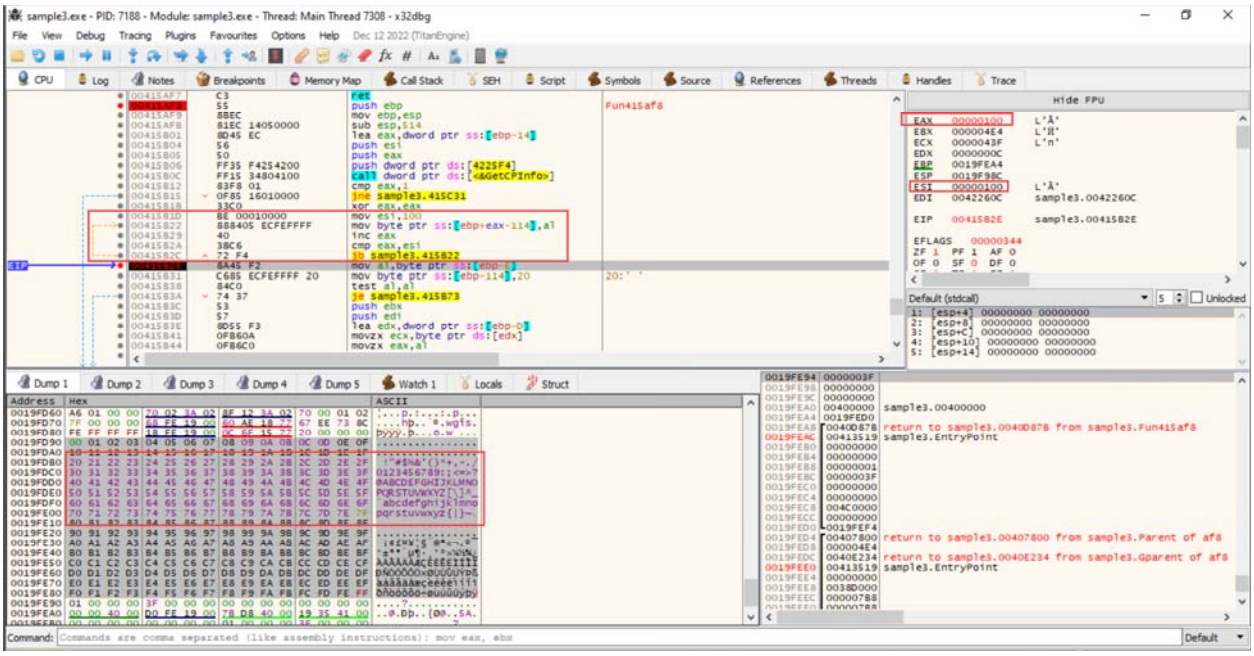


Figure 50 Populates ASCII characters in memory

Call to MultiByteToWideChar in Fun416d24 (which is a child function of Fun415af8) converts the ASCII string to Unicode.

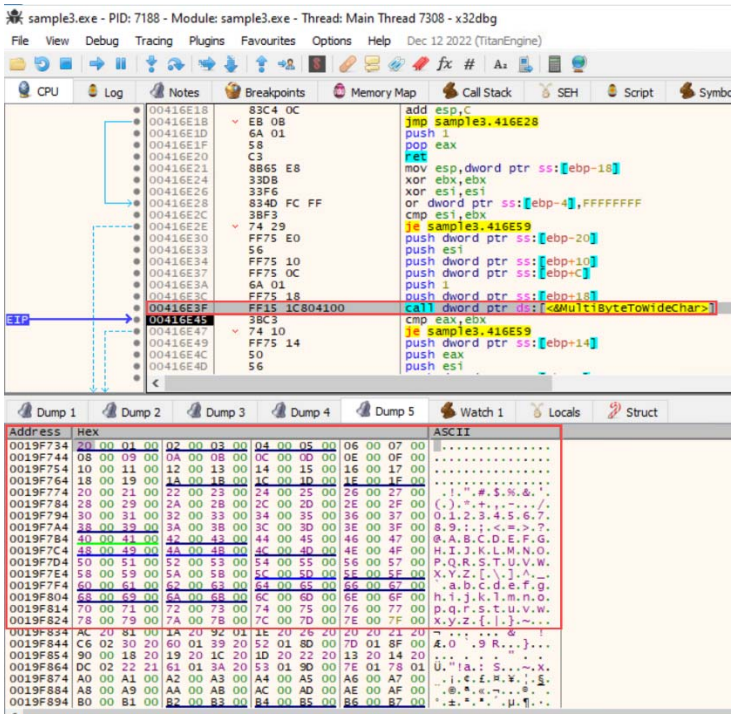


Figure 51 ASCII string converted to Unicode

- **Anti-debugging:** The malware goes into an infinite loop to implement an anti-debugging strategy using **GetTickCount**.

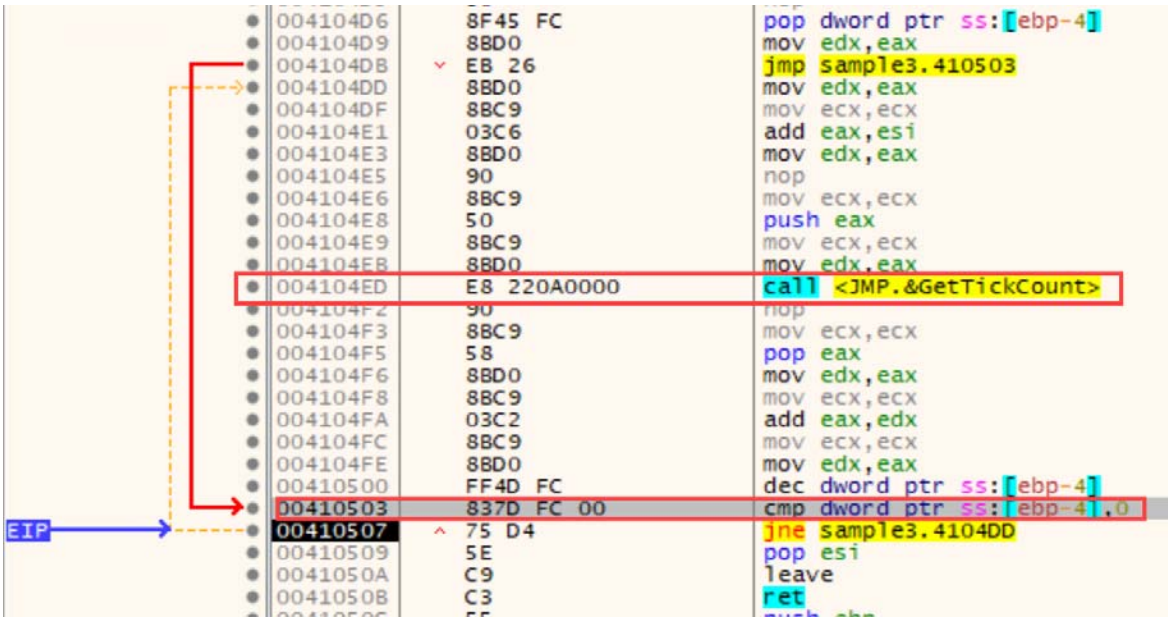


Figure 52 Anti-debugging using GetTickCount

- **Network communication:** The malware sends POST requests to multiple URLs on port 10015. I was able to capture traffic to various IPs by putting breakpoints in the code and rerunning netcat after the socket to each IP was closed. Without the breakpoints and rerunning netcat, I could only capture traffic on the first port as netcat kept closing the socket was closed. The contents of the post request are encrypted. It sends requests to the following URLs:
    - http://85.192.165.229:10015/gate.php
    - http://176.96.187.114:10015/gate.php
    - http://176.96.187.116:10015/gate.php
    - http://80.254.98.212:10015/gate.php
    - http://5.144.66.227:10015/gate.php

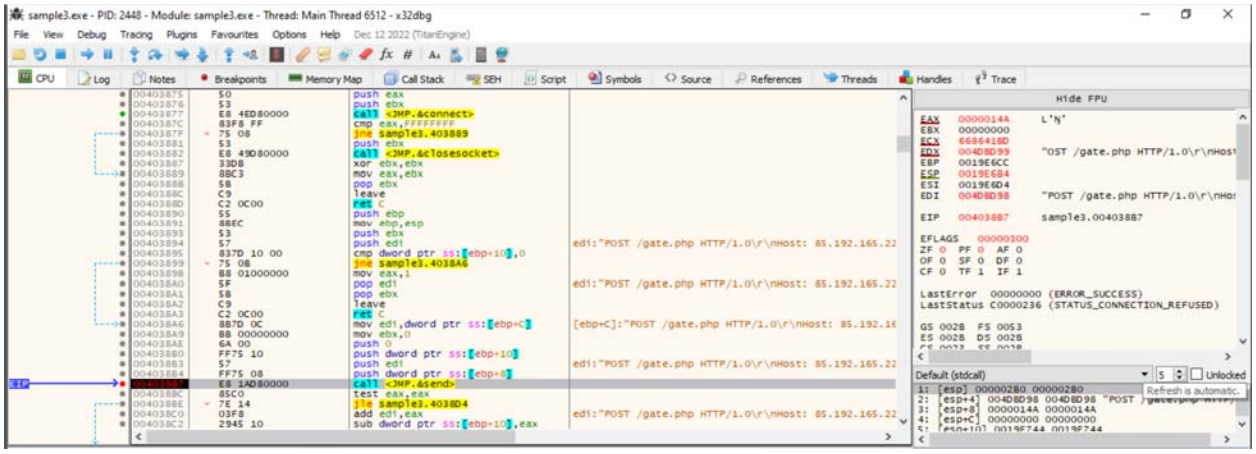


Figure 53 Network communication

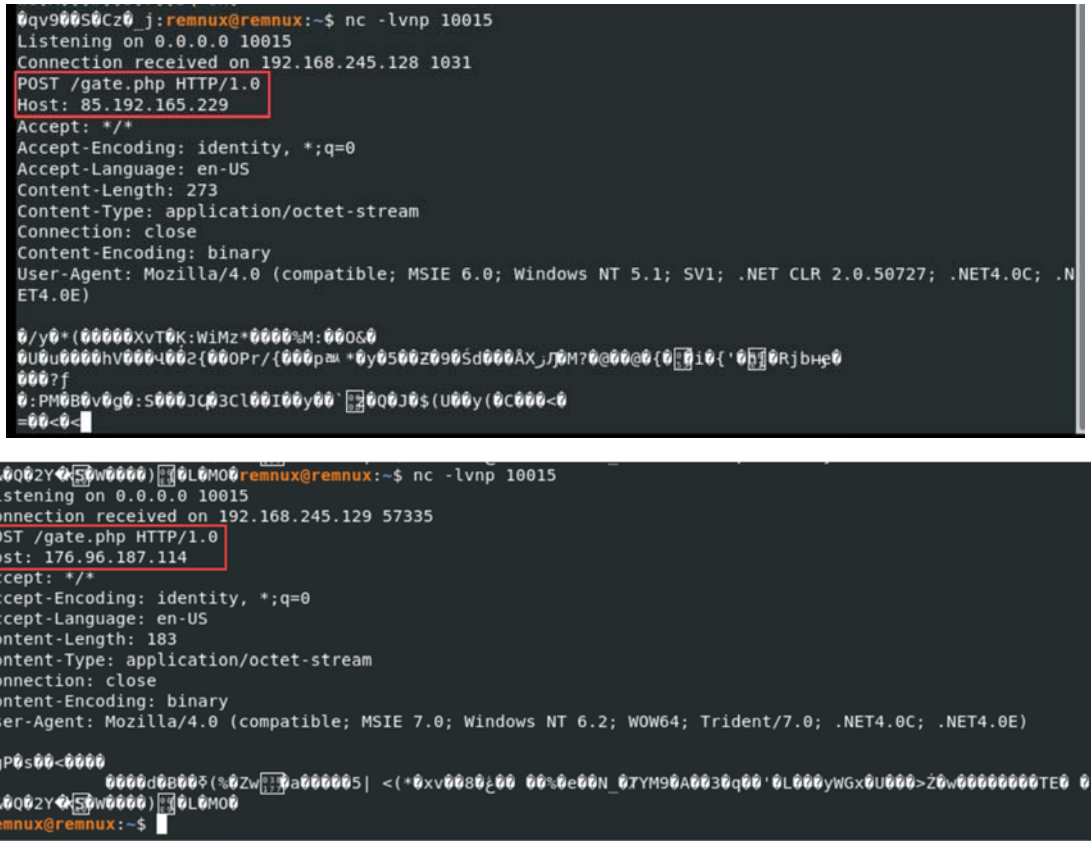


Figure 54 Encrypted data sent to multiple IPs

Indicators of Compromise

Host Based Indicators

The sample is very good at hiding itself. It does not create any files on the infected system so it is really difficult to detect.

The sample adds the key registry *HKU\<SID>\Software\WinRAR\HWID* which can be used as a host-based IoC. Only using this key as an IoC will give false positives as it will fire in case of legitimate machines which have WinRAR installed. Therefore, this registry key is not a sufficient IoC to determine the presence of the malware by itself and needs to be combined with other indicators to avoid false positives.

The malware has the following 2 mutants but they cannot be used by themselves as an IOC. They need to be combined with other indicators to avoid false positives.

\Sessions\1\BaseNamedObjects\SM0:<4-digits>:<3-digits>:WilStaging_02
\Sessions\1\BaseNamedObjects\SM0:<4-digits>:<2-digits>:WilError_03

Network Based Indicators

Since the malware attempts to connect to the following URLs, traffic to all these IPs can act as network-based indicator of compromise.

- 85.192.165.229
- 176.96.187.114
- 176.96.187.116
- 80.254.98.212
- 5.144.66.227

These might be the IPs of the malware's command and control server.

YARA Rule

```
rule ManyBytesMalware {
   meta:
      description = "Rule to detect practical3 malware"
      author = "Rachana"
      date = "4/7/2023"
   strings:
      $s1 = "Msisoft" ascii wide nocase
      $s2 = "ManyBytes program" ascii wide nocase
      $s3 = "ManyBytes.exe" ascii wide nocase
      $s4 = "OriginalFilename" ascii wide nocase
      $s5 = "screenssanges" ascii wide nocase
      $s6 = "guikas.txt" ascii wide nocase
      $s7 = "@Od@ddx0x0" ascii wide nocase
      $s8 = "zolupalim" ascii wide nocase
      $s9 = "Add page %s to favorites with (optional) name" ascii wide nocase

      $x1 = "GPLv3" ascii wide nocase
      $x2 = "<program name unknown>" ascii wide nocase
      $x3 = "SING error" ascii wide nocase
      $x4 = "VarFileInfo" ascii wide nocase
      $x5 = "1.1.21.11" ascii wide nocase

   condition:
      uint16(0) == 0x5a4d and filesize < 120KB and (6 of ($s*) and 3 of ($x*))
}
```

The YARA rule only fires on sample3.exe and does not fire on any other PMA Lab samples.

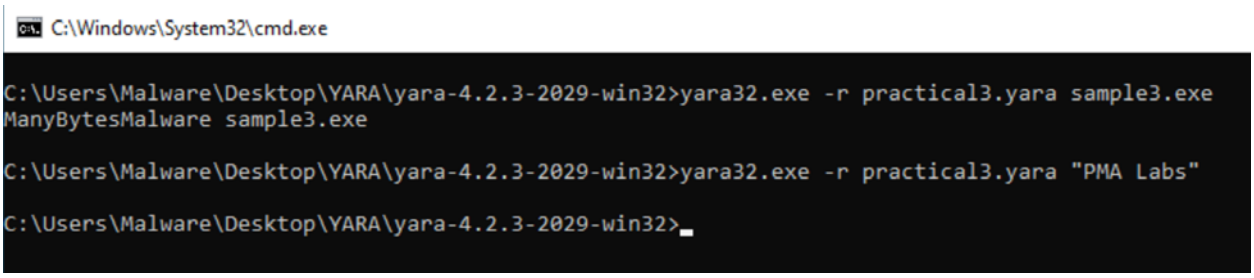


Figure 55 Executing the YARA rule on sample3.exe and all PMA Labs