

20, January 2021

Malware Analysis Report

SHA256: -

- Sample One: 472b720c0a8b4a0947441f43ce9982fc27f03811d2009f6128b4cd9c90a45286

Presented by: Dakshitha Perera

10519381

pdakshi@our.ecu.edu.au

Table of Contents

.....	1
Malware Sample	2
Executive Summary:.....	2
Identification of malware sample.....	3
Details of architectures targeted by malware.	10
Details of packing or obfuscation	1
Details of malware behaviour	8
Dynamic Analysis:.....	8
Noticeable behaviours:.....	10
Interaction with Files System:.....	11
Created Files:	12
Accessed Files:.....	15
Removed or closed Files:.....	15
Network Behaviours:.....	16
Registry Keys:.....	22
Detection rules	23
YARA Rules	23
Snort Rules	26
Detection and Removal Instructions	27
References:	1

Malware Sample

Executive Summary:

This malware is a Windows operating system targeted browser hijacker. This type of malware enters a PC using patched freeware or shareware and hijacks the settings of the default web browser. The malware in question primarily hijacks the default web

browser's homepage, new tab, and the search engine. Not surprisingly, it gathers the victim's information on to their own remote servers and sells them to third party clients who may use this data for malvertising or identify theft. Essentially, this malware collects the victim's web browser cookies, passwords, usernames, most visited websites, and bookmarked websites.

According to the analysis information, this malware or adware has been created by adexchange guru. It is a professional adware making company based in the USA.

Once the device is infected with this malware,

- The browser redirects sites are different from what is expected.
- The normally visited websites are not shown impeccably by the web browser.
- The browser popups fake updates and software to download. Which can degrade a computer speed and performance completely.

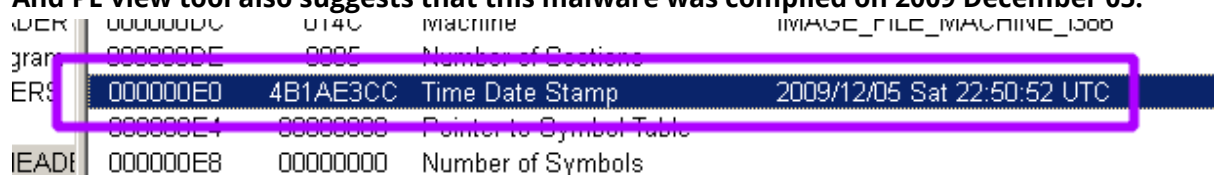
Another doubtful reality of the malware is that the prompted websites by the search engine do not provide privacy policy at all. Keeping the computer update is the best way to avoid these types of malwares.

This malware is a 32-bit executable which was written in 2009. NISI can be identified as a malware packer. According to the Virustotal, 47 anti-malware engines have the capability to detect this malware including Microsoft. Since this is a trojan there is no one name that can be given to identify the malware in question. However, Virustotal uses 976fd3e98a0ce54a10d28a3a87340e56.virus as the malware name. Moreover, this malware checks the victim computer network adapters. And it monitors user input as well.

This report mentions the step to removal and detect this malware using various techniques. However, recommended steps to remove the malware is difficult. To make it easy, this report demonstrates a python script to detect and remove the malware automatically. Moreover, Yara and Snort rules which were developed in this report will also assist to detect the malware in question.

Identification of malware sample

This malware sample is a 32bit PE executable file which only targets Microsoft Windows (GUI) with Intel 368 or later processors. NSIS (Nullsoft Scriptable Install System) installer is used in order to install the malware to the Microsoft platform. According to Virustotal.com it was first created in 2009 and uses English-United states as its Language. And PE view tool also suggests that this malware was compiled on 2009 December 05.



00000000	0140	Machine	IMAGE_FILE_MACHINE_I386
0000000E	0005	Number of Sections	
000000E0	4B1AE3CC	Time Date Stamp	2009/12/05 Sat 22:50:52 UTC
000000E4	00000000	Pointer to Symbol Table	
000000E8	00000000	Number of Symbols	

Figure 1: Pestudio time data stamp

However, this does not mean that the information is true. Some malware authors intentionally change the compiler-stamp in order to mislead malware analysts. Furthermore, 47 malware engines detected this file as malicious in Virustotal. The Nullsoft

PiMP stub SFX can be identified as the packer and compiler.

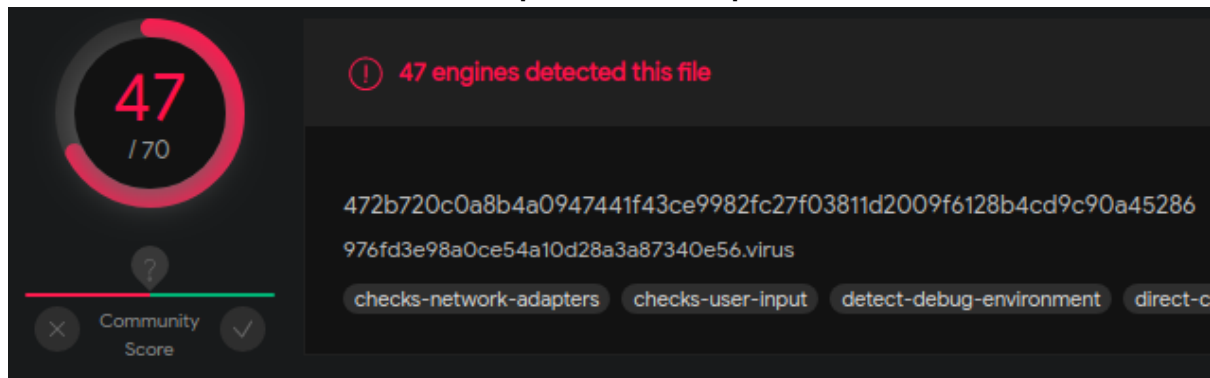


Figure 2: Screenshot of Virustotal

There are quite a few names that can be used to identify this malware. Basically, Virustotal and Hybrid-Analysis have identified this malware under two different names. Nonetheless, virus total named this malware as 976fd3e98a0ce54a10d28a3a87340e56.virus. here are some names that identifies by famous any-malware engines.

BitDefender	Generic.Application.Adload.B8F92FBC
Comodo	ApplicUnwnt@#2l9zt7ignzohz
Alibaba	Trojan:Win32/AdLoad.b2514328
Microsoft	TrojanDownloader:JS/Istbar!atmn
McAfee	Artemis!976FD3E98A0C
Avast	Win32:Downloader-UNP [Drp]

The basic properties of the malware can be categorized as below.

MD5	976fd3e98a0ce54a10d28a3a87340e56
SHA1	f452acf8f2424464b99693e8b140fc9bb88ed8cd
SHA256	472b720c0a8b4a0947441f43ce9982fc27f03811d2009f6128b4cd9c90a45286
VHASH	035056655d5c05509043z8003b7z47z62z3e03dz
PE header hash	81c790eab65ff0be64a9029444dd0f7f

HashMyFile also illustrate the same numbers:

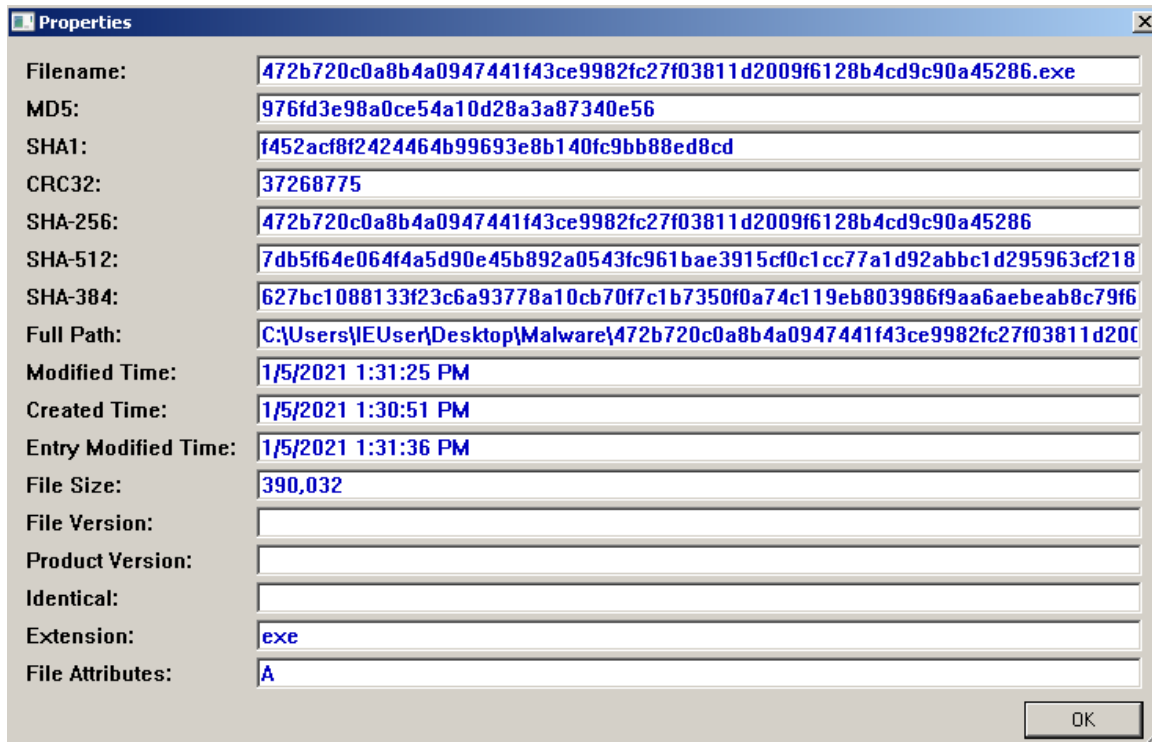
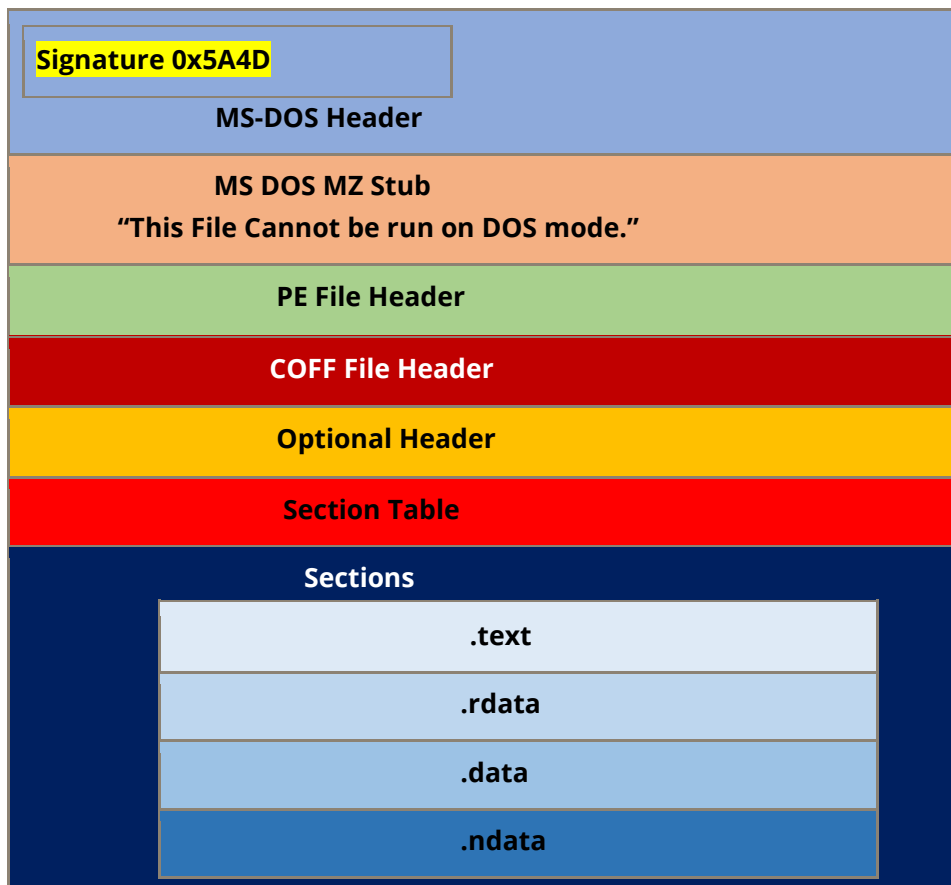
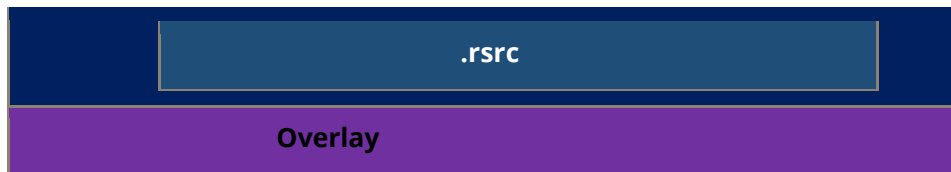


Figure 3: hashmyfile screenshot

This PE file size is around 380.89KB (390032 Bytes). The basic structure of the PE file is very similar to a normal PE file. The file contains:





1. **Dos Header:** This area used to identify MS-DOS compatible executables files. Its e_magic value has a hex value 0x5A4D which is MZ in ASCII (Mark Zbikowski who developed it).

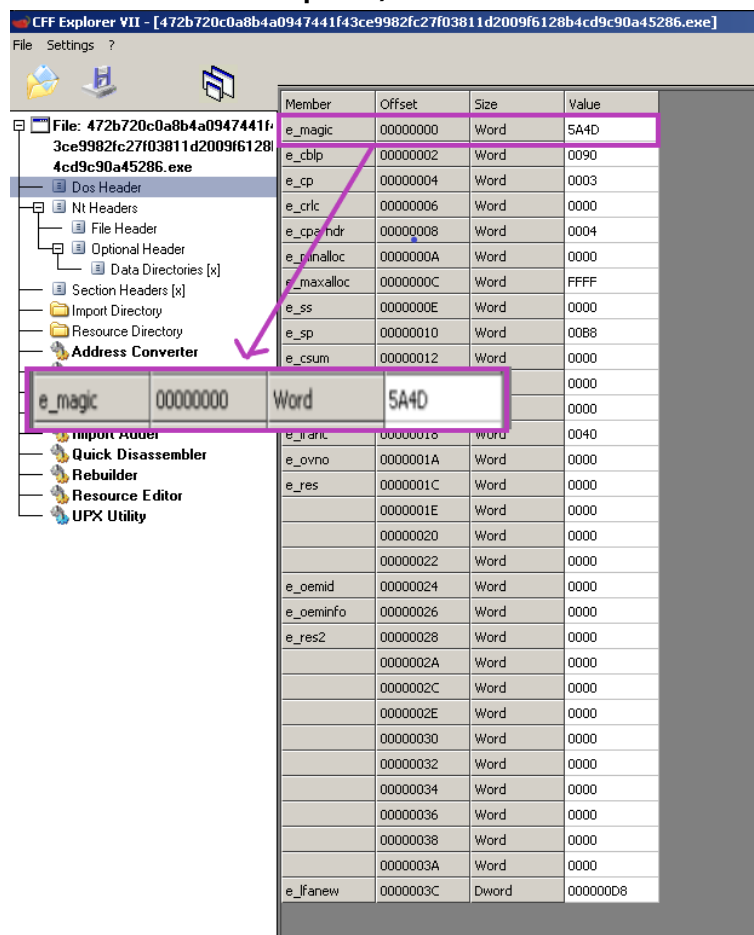


Figure 4: Screenshot of CFF explorer

2. **DOS stub:** This section prints a message of This Program cannot be run in DOS mode.

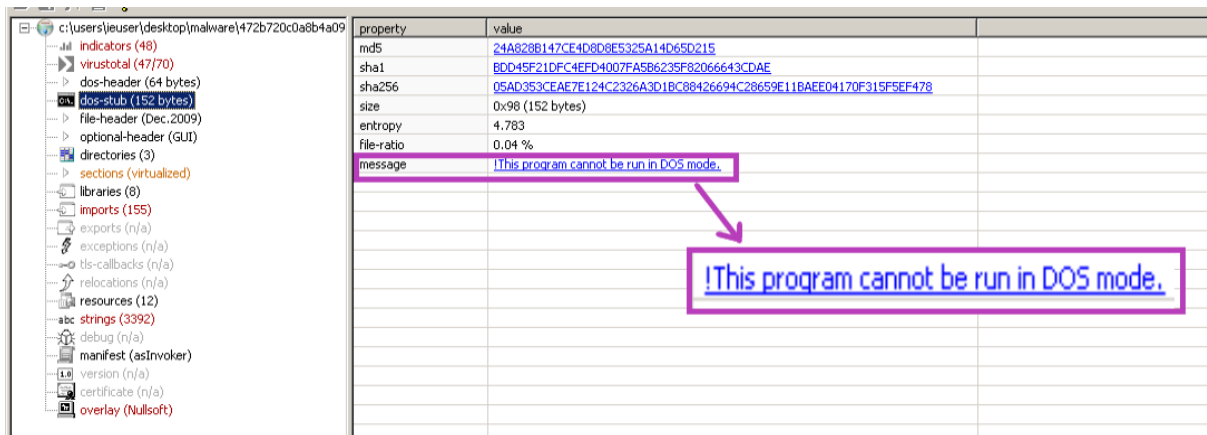


Figure 5: Screenshot of pestudio

3. PE file Header: This section allows linking other executables files.

Furthermore, this contains signatures. Which uniquely identifies the PE file header.

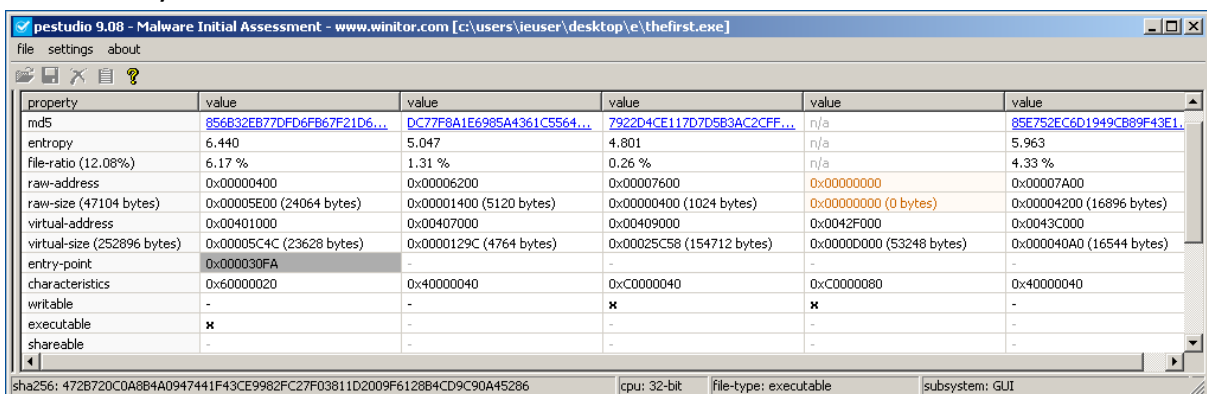
Member	Offset	Size	Value
Signature	000000D8	Dword	00004550

Figure 6: Screenshot of CFF explorer

4. Sections

Moreover, 5 file sections have been used in this malware file. Which are

1. .text
2. .rdata
3. .data
4. .ndata
5. .rsrc,



This malware contains unique threads such as web address, IP address and hard code error messages. So, these unique threads can be used to detect malware.

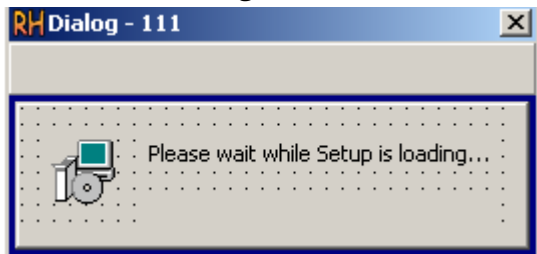
```

VerQueryValueA
GetFileVersionInfoA
GetFileVersionInfoSizeA
VERSION.dll
verifying installer: %d%%
Installer integrity check has failed. Common causes include
incomplete download and damaged media. Contact the
installer's author to obtain a new copy.
More information at:
http://nsis.sf.net/NSIS_Error
Error launching installer
... %d%%
SeShutdownPrivilege
~nsu.tmp
\Temp
NSIS Error
Error writing temporary file. Make sure your temp folder is valid.

```

Moreover, Malware resources can be examine using resource hacker.

- Here is a dialog box in the malware:



- Here is the Icon of the malware:



Nonetheless, these sections do not have a considerable number of entropies.

Section Name	Address	Size	MD5	Entropy
.text	4096	23628	856b32eb77dfd6fb67f21d6543272da5	6.44
.rdata	28672	4764	dc77f8a1e6985a4361c55642680ddb4f	5.05
.data	36964	154712	7922d4ce117d7d5b3ac2cffe4b0b5e4f	4.8
.ndata	19251	53248	d41d8cd98f00b204e9800998ecf8427e	0
.rsrc	245760	16544	85e752ec6d1949cb89f43e1cb6da4b54	5.96

5. Overlay

However, overlay of this file takes most of the storage of the file (normal, Overlay is not a part of a PE file). The Overlay has a high number of entropies as well. Which is around 7.2. Having a considerable size of overlay of the file and higher number of entropies could be

another good evidence to a packed malware.

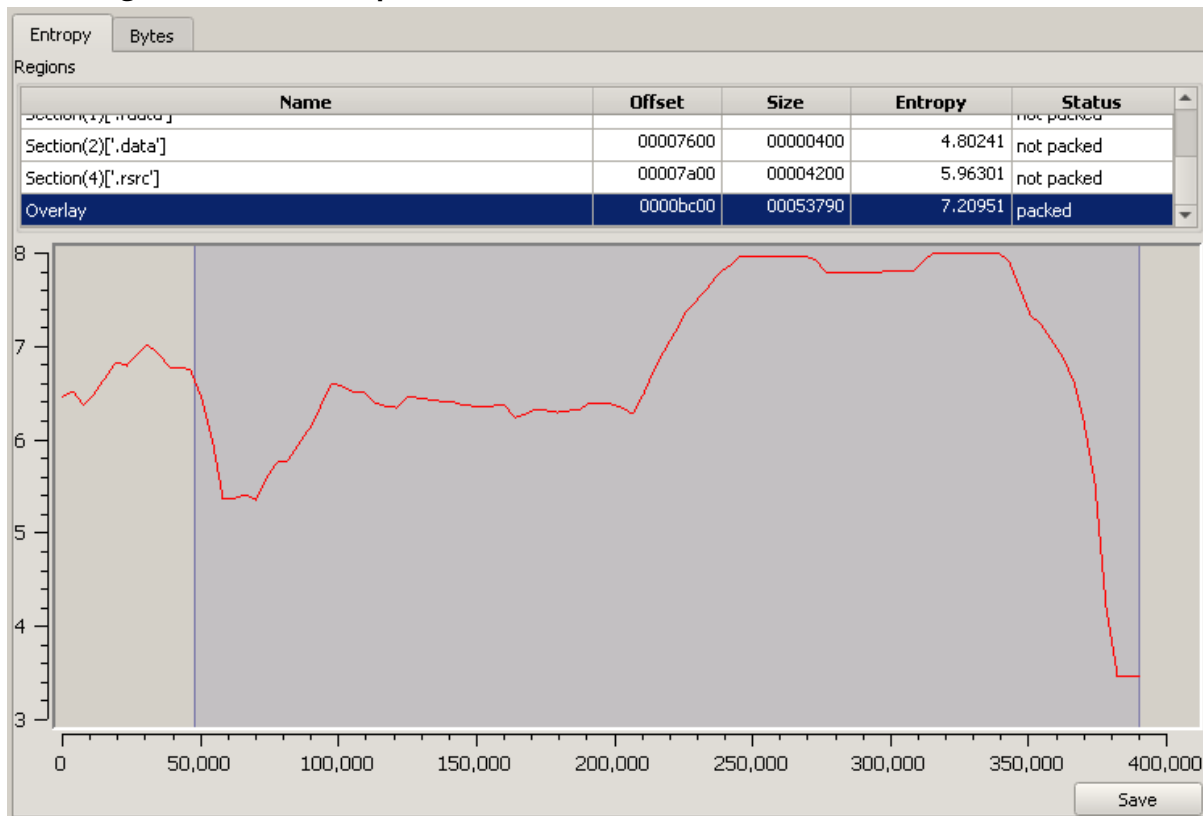


Figure 7: DIE entropy pattern of the malware sample one

And here is the look of the malware with a hex editor.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00@.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	D8	00	00	00@.....
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..°.°.í!..Lí!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$......
00000080	31	B8	84	3A	75	D9	EA	69	75	D9	EA	69	75	D9	EA	69	1,,:uüëiuüëiuüëi
00000090	B6	D6	B5	69	77	D9	EA	69	75	D9	EB	69	EE	D9	EA	69	Qüiüëiuüëiüëi
000000A0	B6	D6	B7	69	64	D9	EA	69	21	FA	DA	69	7F	D9	EA	69	QÖ·idüëi!üüi.üëi
000000B0	B2	DF	EC	69	74	D9	EA	69	52	69	63	68	75	D9	EA	69	*ßitüëiRighuüëi
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00PE..L...
000000E0	CC	E3	1A	4B	00	00	00	00	00	00	00	00	E0	00	0F	01	ïä.K.....ä...
000000F0	0B	01	06	00	00	5E	00	00	00	84	02	00	00	04	00	00^.....
00000100	FA	30	00	00	00	10	00	00	00	70	00	00	00	00	40	00	ü0.....p....@.
00000110	00	10	00	00	00	02	00	00	04	00	00	00	06	00	00	00
00000120	04	00	00	00	00	00	00	00	00	10	04	00	00	04	00	00
00000130	00	00	00	00	02	00	00	80	00	00	10	00	00	10	00	00e.....
00000140	00	00	10	00	00	10	00	00	00	00	00	00	10	00	00	00
00000150	00	00	00	00	00	00	00	00	00	B0	74	00	00	B4	00	00°t..'
00000160	00	C0	03	00	A0	40	00	00	00	00	00	00	00	00	00	00	..ä.. @.....
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001B0	00	70	00	00	8C	02	00	00	00	00	00	00	00	00	00	00	..p..@.....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001D0	2E	74	65	78	74	00	00	00	4C	5C	00	00	00	10	00	00	..text...L\.....
000001E0	00	5E	00	00	00	04	00	00	00	00	00	00	00	00	00	00	..^.....
000001F0	00	00	00	00	20	00	00	60	2E	72	64	61	74	61	00	00'.rdata..
00000200	9C	12	00	00	00	70	00	00	14	00	00	00	62	00	00	00	æ....p.....b..
00000210	00	00	00	00	00	00	00	00	00	00	00	40	00	00	40	00@..@
00000220	2E	64	61	74	61	00	00	58	5C	02	00	00	90	00	00	00	..data...X\.....
00000230	00	04	00	00	00	76	00	00	00	00	00	00	00	00	00	00v.....
00000240	00	00	00	00	40	00	00	C0	2E	6E	64	61	74	61	00	00@..ä.ndata..

Details of architectures targeted by malware.

With the assistance of different tools, a board idea can be gained about the target architecture by the malware. This report mentioned earlier, this malware only targeted architecture was Microsoft Windows with Intel 368 or later processors. And 32-bit can be recognized as the target CPU architecture. Furthermore, according to the pestudio, the aimed subsystem is GUI.

file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	0x4B1AE000 (Sat Dec 05 14:50:52 2009)
debugger-stamp	n/a

The aimed endianness is little endianness. Moreover, malware manifest could be allowed to get detailed information about the aimed architecture. And it allows to identify the malware uniquely.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><assembly
xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"><assemblyIdentity
version="1.0.0.0" processorArchitecture="X86" name="Nullsoft.NSIS.exehead"
type="win32"/><description>Nullsoft Install System v21-Jun-
2014.cvs</description><dependency><dependentAssembly><assemblyIdentity
type="win32" name="Microsoft.Windows.Common-Controls" version="6.0.0.0"
processorArchitecture="X86" publicKeyToken="6595b64144ccf1df" language="*"
/></dependentAssembly></dependency><trustInfo xmlns="urn:schemas-microsoft-
com:asm.v3"><security><requestedPrivileges><requestedExecutionLevel
level="asInvoker"
uiAccess="false"/></requestedPrivileges></security></trustInfo><compatibility
xmlns="urn:schemas-microsoft-com:compatibility.v1"><application><supportedOS
Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/><supportedOS Id="{e2011457-1546-43c5-
a5fe-008deee3d3f0}"/></application></compatibility></assembly>
```

This malware is quite old one. And it mainly targets windows 7 or below versions. One of the evidences for that is some API that malware uses. For example, it uses an API called InitCommonControls. This function is obsolete.

```
param_2 = 0;
pcVar14 = s_Error_writing_temporary_file._Ma_00409160;
uVar13 = 0;
param_1._0_1_ =
InitCommonControls();
SetErrorMode(0x8001);
DAT_0042eb64 = FUN_00405d2e(8);
SHGetFileInfoA((LPCSTR)&lpString_00428f98,0,(SHFILEINF
FUN_00405a0c((LPSTR)&lpchText_0042e360,s_NSIS_Error_00
```

Figure 8: Screenshot of Ghidra

category	Details
Compiler	NSIS
Architecture	I368 [Windows]
Installer	Nullsoft Script Install System
Library	Microsoft Visual C++ 6.0
subsystem	GUI
File-Type	Executable
Endianness	Little
CPU	32-bit
Linker	Microsoft Linker (6.0)
Overlay	NSIS data

Details of packing or obfuscation

As this report mentioned above, according to the Virustotal and Hybrid-Analysis this malware has been packed with Nullsoft PiMP Stub SFX.

As the below screenshot illustrates the DIE (Detect It Easy) also confirms that the malware has been packed. And has 7.13313 entropy. Normally, if the entropy is more than 7 out of 8. The malware is possibly packed.

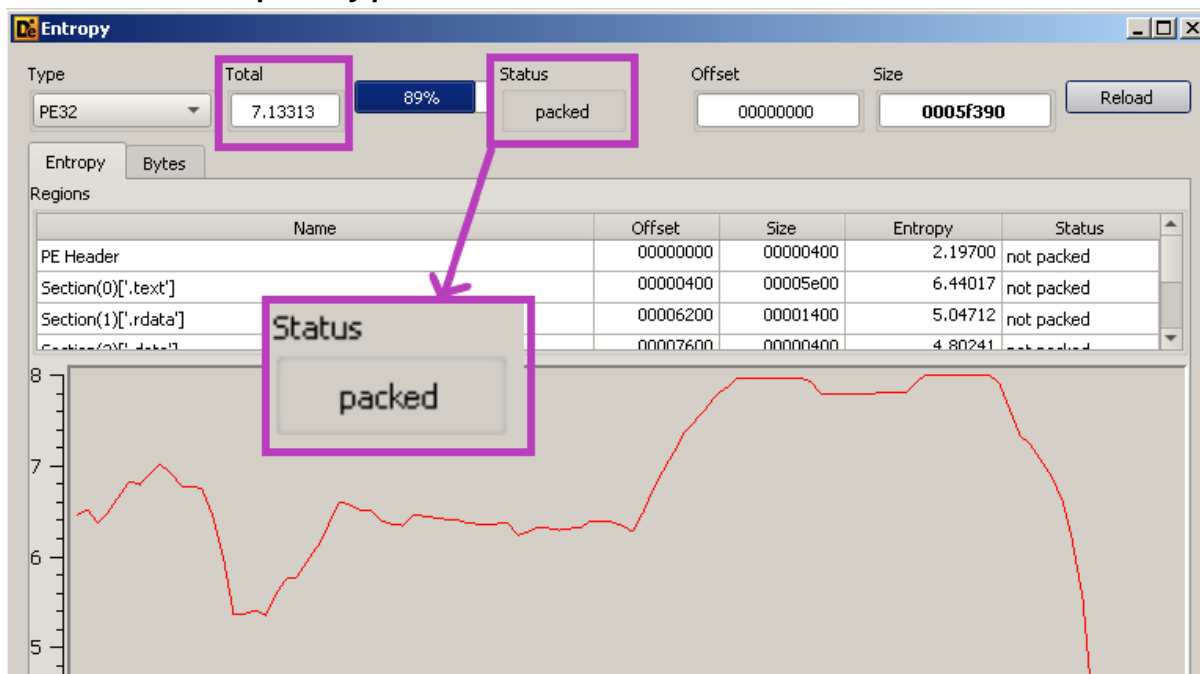


Figure 9: DIE screenshot

Nonetheless, some parts of the file have not been packed. For instance: as DIE presents all the sections have not been packed. But Overlay has almost 7.2 entropy and it is packed.

However, PEiD does not detect any packings.

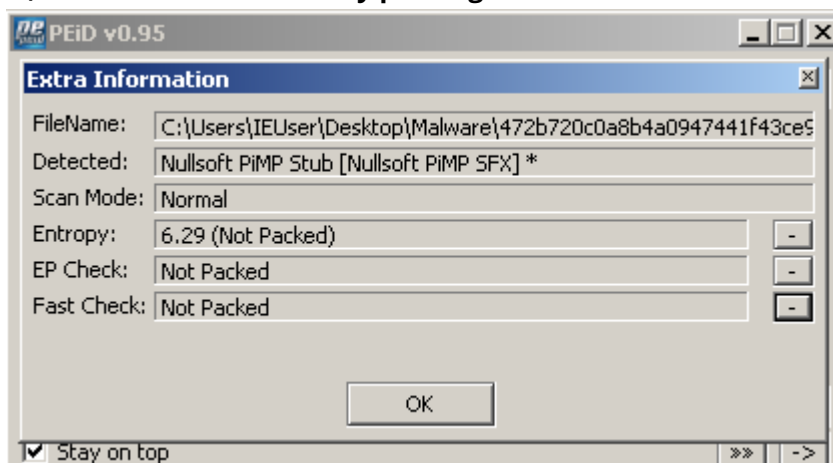


Figure 10: Screenshot of PEiD

RDG Packer Detector does not illustrate anything other than what PEiD shows.



Figure 11: Screenshot of RDG Packer Detector

However, as this report mentioned earlier almost all the imports and strings are visible in this malware sample. Nevertheless, LoadLibrary and GetProcAddress function can be found in the malware sample.

Therefore, it is necessary to prove the malware is actually packed. In order to prove it, PortEx visualization can be used.

```
C:\Users\IEUser\Desktop>java -jar PortexAnalyzer.jar -o FirstMalwareSample.txt -p FirstMalwareSample.png Malware\472b720c0a8b4a094744f43ce9982fc27f03811d2009f6128b4cd9c90a45286.exe
PortEx Analyzer

Creating report file...
Writing header reports...
Writing section reports...
Writing analysis reports...
Report done!
creating visualization...
picture successfully created and saved to C:\Users\IEUser\Desktop\FirstMalwareSample.png

C:\Users\IEUser\Desktop>_
```

Figure 12: Screenshot of the terminal while running PortEx Analyzer

As the PortEx Analyzer illustrates, clearly, there are few parts of the malware that have been packed.

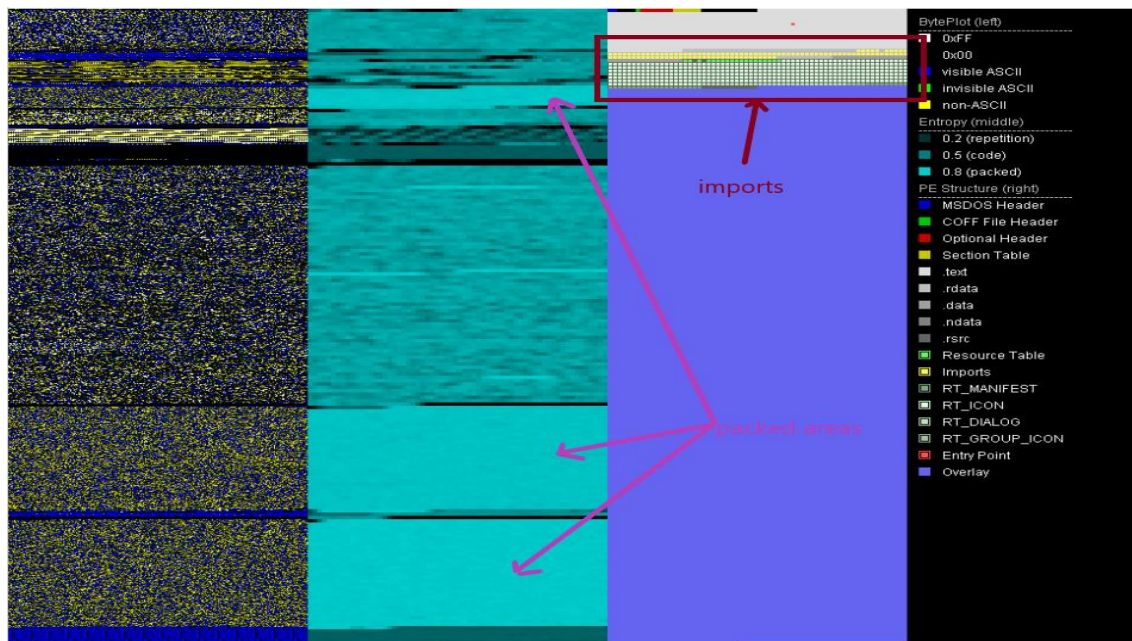


Figure 13: The results of the PortEx

Therefore, it is clear that some part of this malware has been packed. Furthermore, the possible tail jump can also be recognized with a debugger. As the below picture shows, the malware has been opened with x32dbg and x77C05F82 could be the possible OEP (Possible Entry Point).

77C05F71	CC	1nt3
77C05F72	03B6 FDBEC5CA	add esi,dword ptr ds:[esi-353A4103]
77C05F78	0200	add al,byte ptr ds:[eax]
77C05F7A	0000	add byte ptr ds:[eax],al
77C05F7C	77 6E	ja ntdll.77C05FEC
77C05F7E	74 64	je ntdll.77C05FE4
77C05F80	6C	insb
77C05F81	6C	insb
77C05F82	2E:70 64	jo ntdll.77C05FE9
77C05F85	6200	bound eax,qword ptr ds:[eax]
77C05F87	0000	add byte ptr ds:[eax],al
77C05F89	0000	add byte ptr ds:[eax],al
77C05F8B	0000	add byte ptr ds:[eax],al
77C05F8D	0000	add byte ptr ds:[eax],al
77C05F8F	0000	add byte ptr ds:[eax],al
77C05F91	0000	add byte ptr ds:[eax],al
77C05F93	0000	add byte ptr ds:[eax],al
77C05F95	0000	add byte ptr ds:[eax],al
77C05F97	0000	add byte ptr ds:[eax],al
77C05F99	0000	add byte ptr ds:[eax],al

However, the above jumps are also suspicious. However, the final jump leads to nowhere. And Here is the unpacking function in IDA:



Thus, First I put a breaking point there and run the code

until it hits the breaking point. After that, I clicked the step into button, and it jumped to a place where a new function starts.

	004030E2	80 00504300	push 472b720c0a8b4a0947441f43ce9982fc27f03811d2009f6128b4
	004030F3	E8 1A260000	call 472b720c0a8b4a0947441f43ce9982fc27f03811d2009f6128b4
	004030F8	5E	pop esi
	004030F9	C3	ret
	004030FA	81EC 80010000	sub esp,180
EIP →	00403100	53	push ebx
	00403101	55	push ebp
	00403102	56	push esi
	00403103	33DB	xor ebx,ebx
	00403105	57	push edi
	00403106	895C24 18	mov dword ptr ss:[esp+18],ebx
	0040310A	C74424 10 60914000	mov dword ptr ss:[esp+10],472b720c0a8b4a0947441f43ce99821

Figure 14: Obtaining the Original Entry Point

Using Ollydump it can be dumped and using Scylla the PE file can be fixed. The DIE tool detects that the newly fixed executable is now unpacked. However, it now has 6 sections. And the overlay is missing. Furthermore, its total number of entries has been dropped to

5.709.

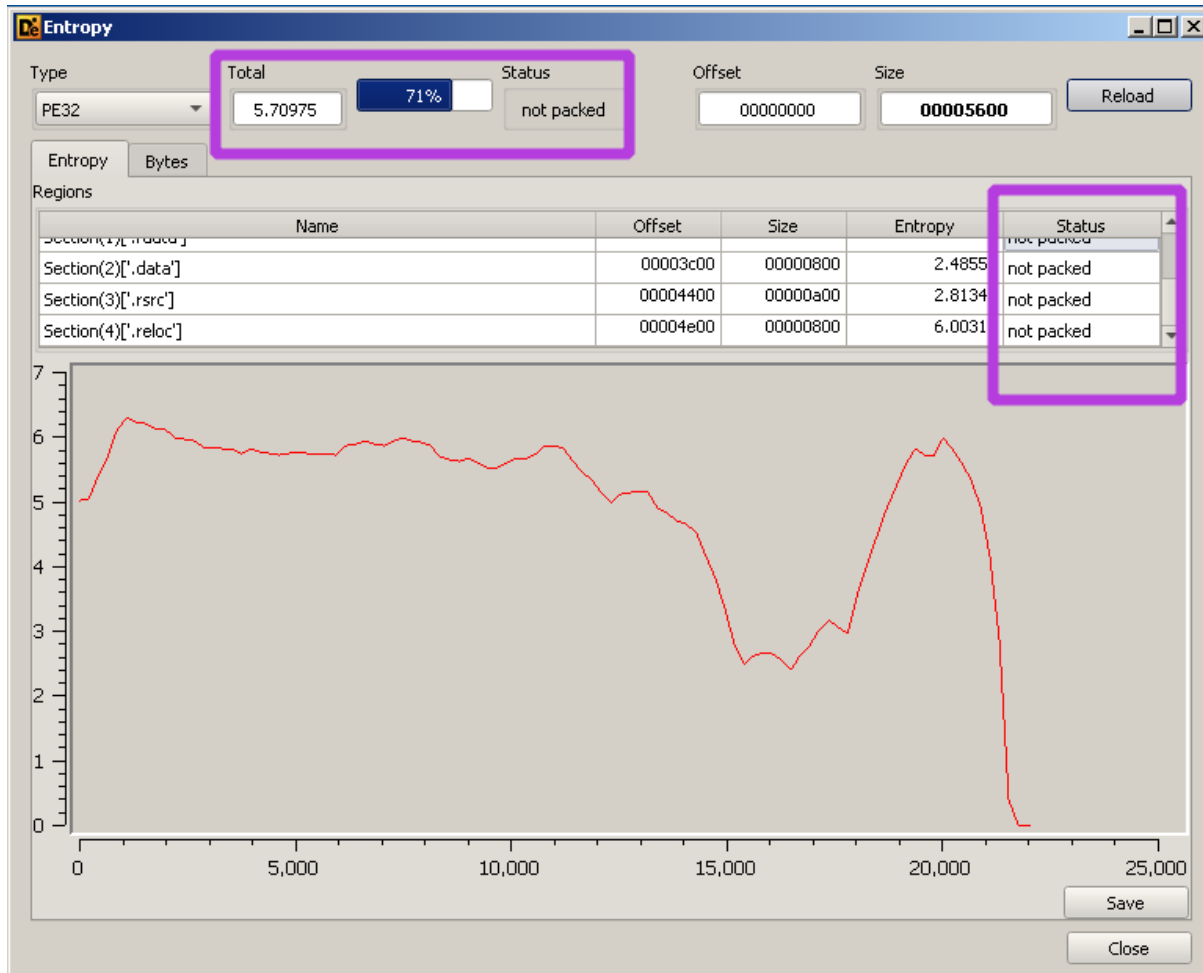


Figure 15: Unpacked malware entropy with DIE

And the new unpacked file sha256 hash is

67eff17c53a78c8ec9a28f392b9bb93df3e74f96f6ecd87a333a482c36546b3e.

Filename	SHA-256
unpackSampleOne.exe	67eff17c53a78c8ec9a28f392b9bb93df3e74f96f6ecd87a333a482c36546b3e
packedSampleOne.exe	472b720c0a8b4a0947441f43ce9982fc27f03811d2009f6128b4cd9c90a45286

Figure 16: Comparing packed and unpacked malware hashes.

Unpacked malware visualization can be viewed like this with PortEx Analyzer.

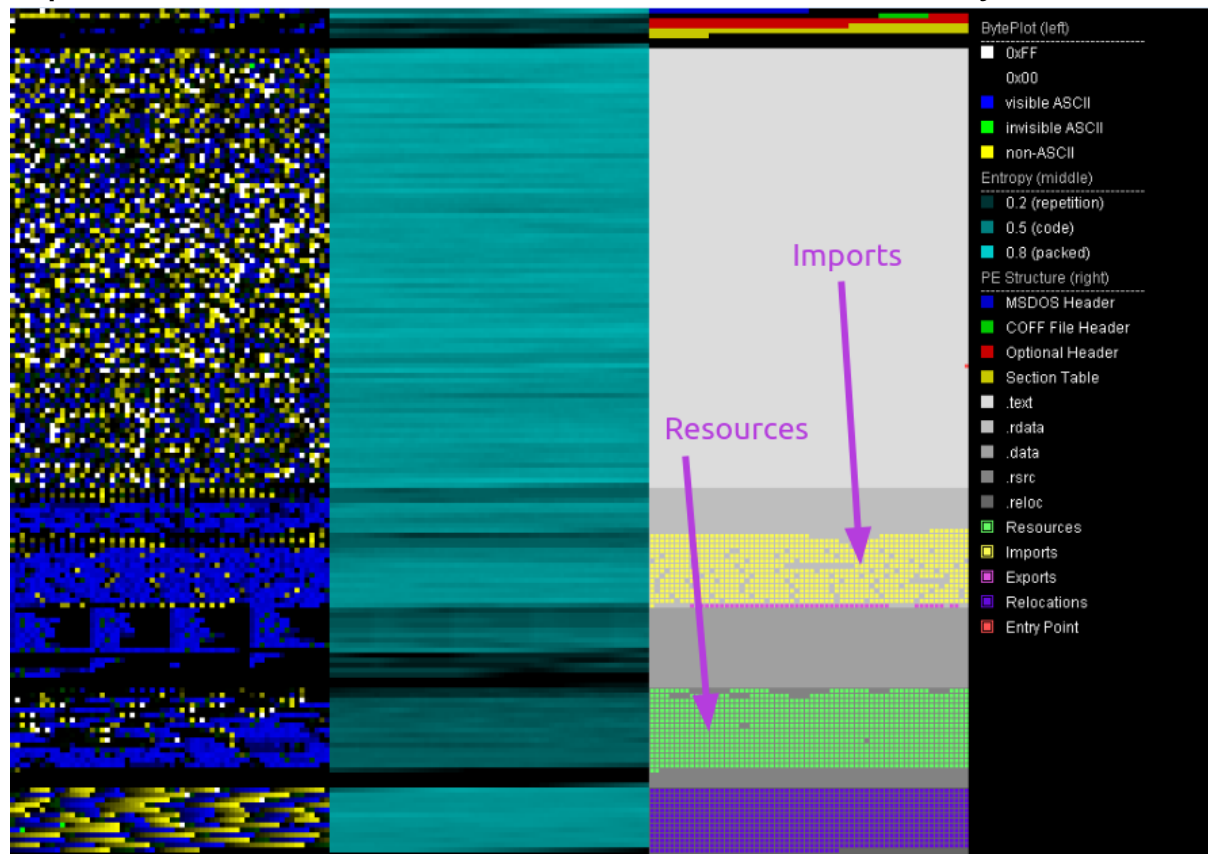


Figure 17: Unpacked malware ProtEx view

A significant difference can be noticed. When unpacked and packed malware visualizations are compared. Above mentioned packed areas are now clearly unpacked. Furthermore, this figure illustrates more imports and resources than the imports and resources were in the packed malware visualization.

Except of some string obfuscation, there does not appear to be any anti-disassembly or anti-debugging techniques. However, this malware was written in way that make difficult to follow the program flow. Thus, understanding of this malware flow by only looking at assembly is nearly impossible.

This malware dynamically builds strings. some stack strings are included in this malware. Stack string is one of the ways to obfuscate strings. This basically pushes one byte at a time to the stack and retrieves all string by referencing to start of the stack.

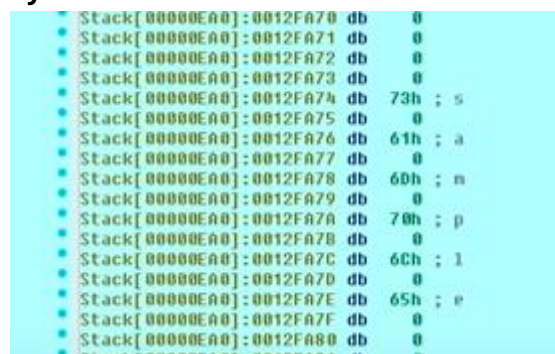


Figure 18: Building stack strings.

FUN_004041fc
MayBeststring0

Figure 19: Checking a special string.

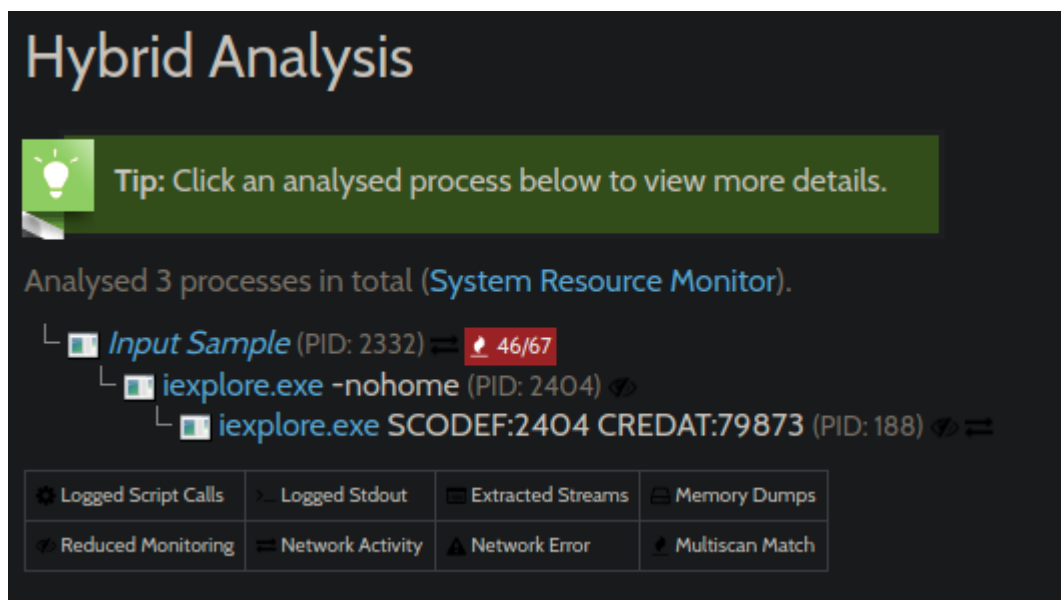
Some 64 base encode strings can be found with malware. For instances

Figure 20: decoding base64 strings with bash language.

Once it is decoded. "GetAdapterAddress" with ascii. This is API which uses by windows to check network adapter on the computer.

Details of malware behaviour

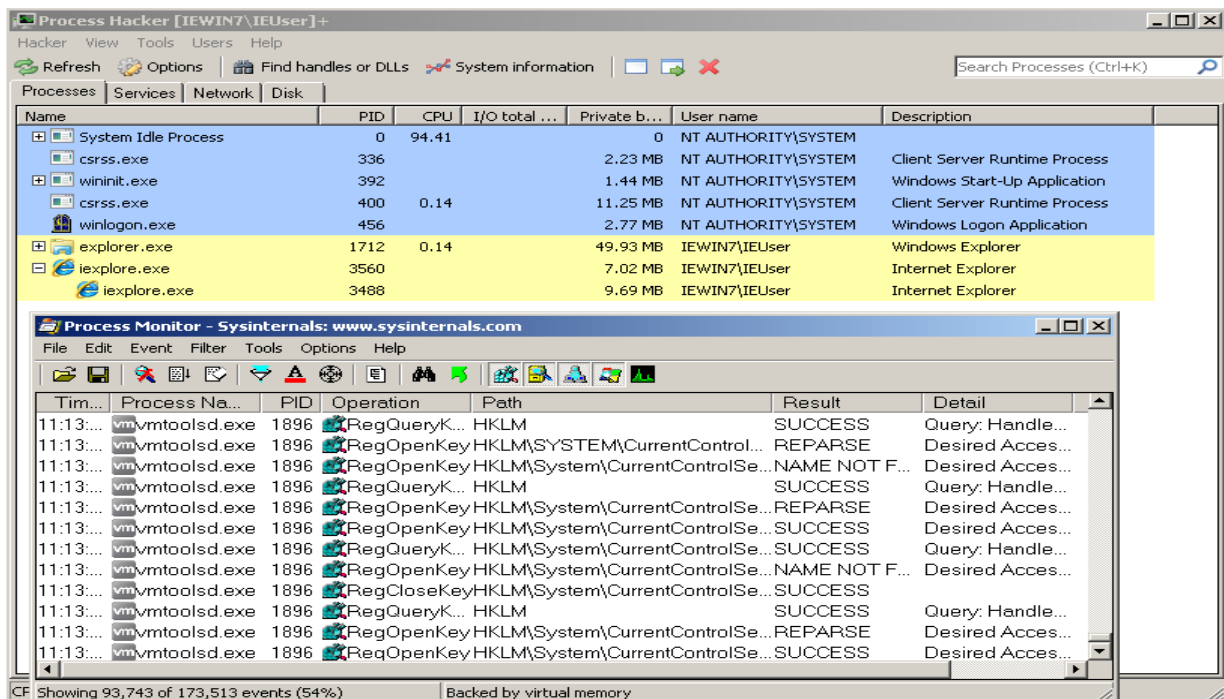
This malware behaviour spreads to a considerable range. For instance, to Creating, deleting, and modifying files and registries to network communication can be experienced with this malware. Therefore, this malware mainly can be identified as an Adware. Moreover, Hybrid-Analysis's system resource monitor shows that malware itself creates the ieexplore.exe.



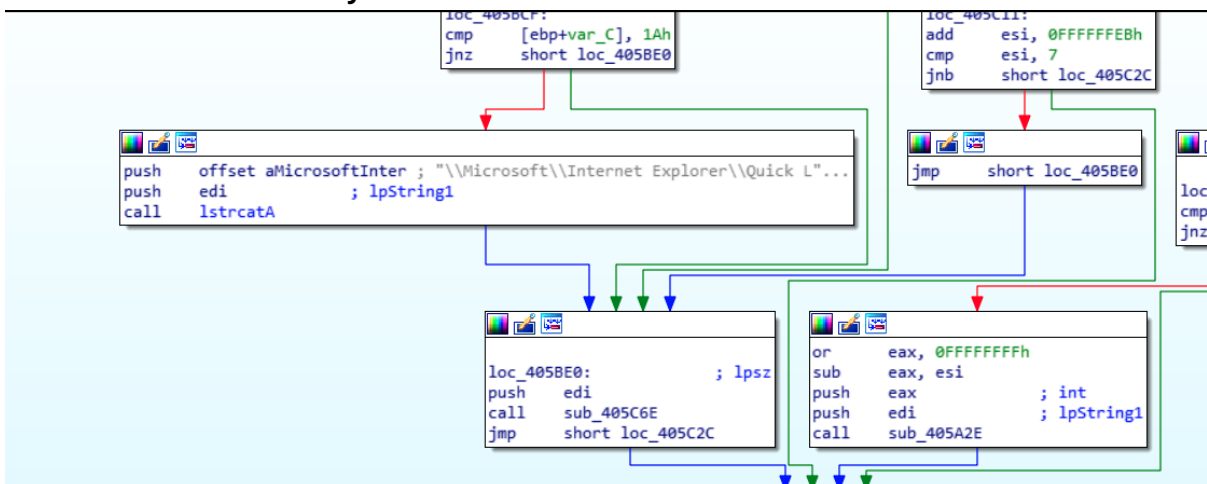
Dynamic Analysis:

the malware can be executed normally. Once it is executed, after roughly 5 to 7 seconds the default browser will automatically open. That is the only action that can be seen.

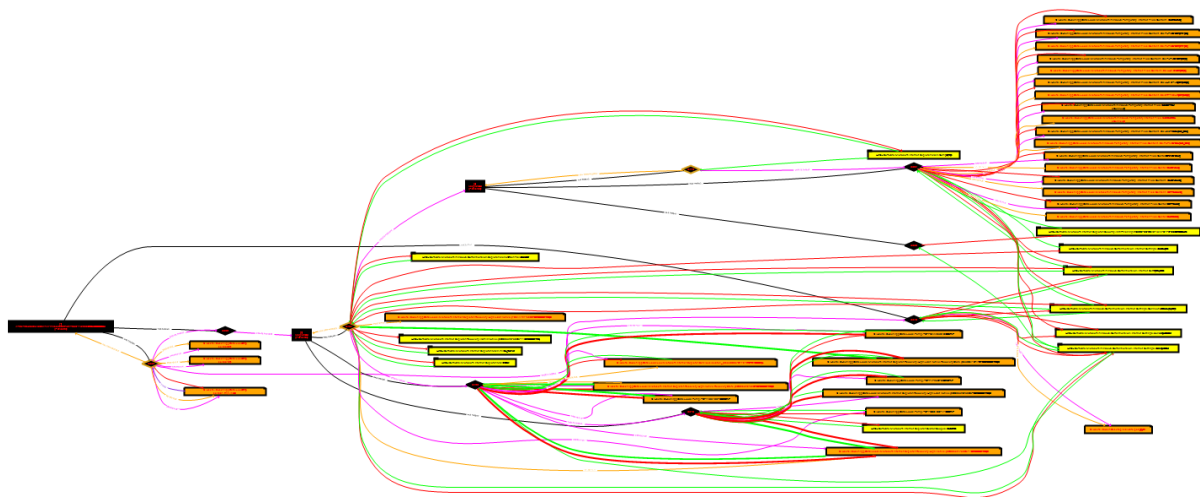
However, if the processes are checked with some tools such as process hackers or procmon some actions can be identified.



As the above screenshot illustrates, the malware launches Internet Explorer by malware itself. Here is the assembly code look from IDA.



When Process monitor(procmon) data is analysed with ProcDot. This chart can find out.



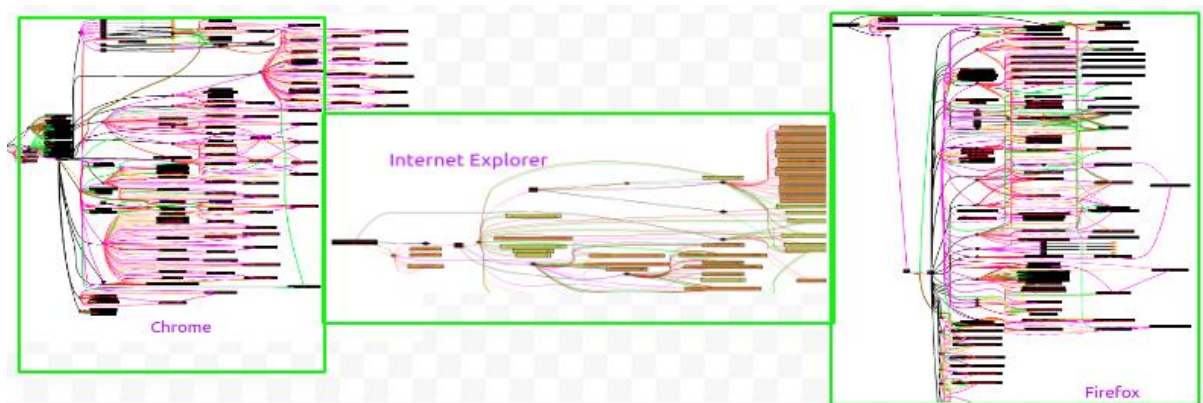
Noticeable behaviours:

Setting some special directory properties can be identified as the distinguished behaviour of this malware. This malware mainly setup below directories on the targeted computer.

- C:\User\<UserName>\AppData\Local\Microsoft\Temporary Internet Files\
- C:\User\<UserName>\AppData\Local\Microsoft\Temporary Internet Files\Content.IE5
- C:\User\<UserName>\AppData\Local\Microsoft\Temporary Internet Files\History
- C:\User\<UserName>\AppData\Local\Microsoft\Windows\Cookies
- C:\User\<UserName>\AppData\Local\Microsoft\Windows\History\History.IE5

5

However, these paths depend on the victim's Windows version and the default web browser.



.3.36.

As this report mentions earlier this malware creates and removes a few files and directories in a temporary directory.

```
1 |
2 | void CreatingFilestoTMP(void)
3 |
4 | {
5 |     int CheckingSpecialCharB;
6 |
7 |     FUN_00405c6e((short *)&DAT_00435400);
8 |     CheckingSpecialCharB = CheckingSpecialChar((short *)&DAT_00435400);
9 |     if (CheckingSpecialCharB == 0) {
10 |         return;
11 |     }
12 |     maybeFindingDeirectory(&DAT_00435400);
13 |     CreateDirectoryA(&DAT_00435400, (LPSECURITY_ATTRIBUTES)0x0);
14 |     maybeCheckingSomethinginTMP((LPSTR)&lpFileName_00435000, &DAT_00435400);
15 |     return;
16 | }
17 |
```

Furthermore, this malware uses a function called AdjustTokenPrivilege. This is a security base api function that enables or disables privileges in an access token. An access token includes security information for the logon session. The window uses these access tokens to control or restrict access to very sensitive objects. Therefore, accessing this function is very risky and noticeable behaviour of this malware.

```
.data:00409224 dd offset aLookupprivileg ; "LookupPrivilegeValueA"
.data:00409228 dd offset aAdvapi32 ; "ADVAPI32"
.data:0040922C dd offset aAdjusttokenpri ; "AdjustTokenPrivileges"
.data:00409230 dd offset aKernel32 ; "KERNEL32"
.data:00409234 dd offset aGetuserdefault ; "GetUserDefaultUILanguage"
.data:00409238 dd offset aShlwapi ; "SHLWAPI"
.data:0040923C dd offset aShautocomplete ; "SHAutoComplete"
.data:00409240 dd offset aShfolder ; "SHFOLDER"
.data:00409244 dd offset aShgetfolderpat ; "SHGetFolderPathA"
.data:00409248 aShgetfolderpat db 'SHGetFolderPathA',0 ; DATA XREF: .data:00409244fo
.data:00409259 align 4
.data:0040925C aShfolder db 'SHFOLDER',0 ; DATA XREF: .data:00409240fo
.data:00409265 align 4
.data:00409268 aShautocomplete db 'SHAutoComplete',0 ; DATA XREF: .data:0040923Cfo
.data:00409277 align 4
.data:00409278 aShlwapi db 'SHLWAPI',0 ; DATA XREF: .data:00409238fo
.data:00409280 aGetuserdefault db 'GetUserDefaultUILanguage',0
.data:00409280 ; DATA XREF: .data:00409234fo
.data:00409299 align 4
.data:0040929C aAdjusttokenpri db 'AdjustTokenPrivileges',
.data:0040929C ; DATA XREF: .data:0040922Cfo
.data:004092B2 align 4
.data:004092B4 aLookupprivileg db 'LookupPrivilegeValueA',0
.data:004092B4 ; DATA XREF: .data:00409224fo
```

Interaction with Files System:

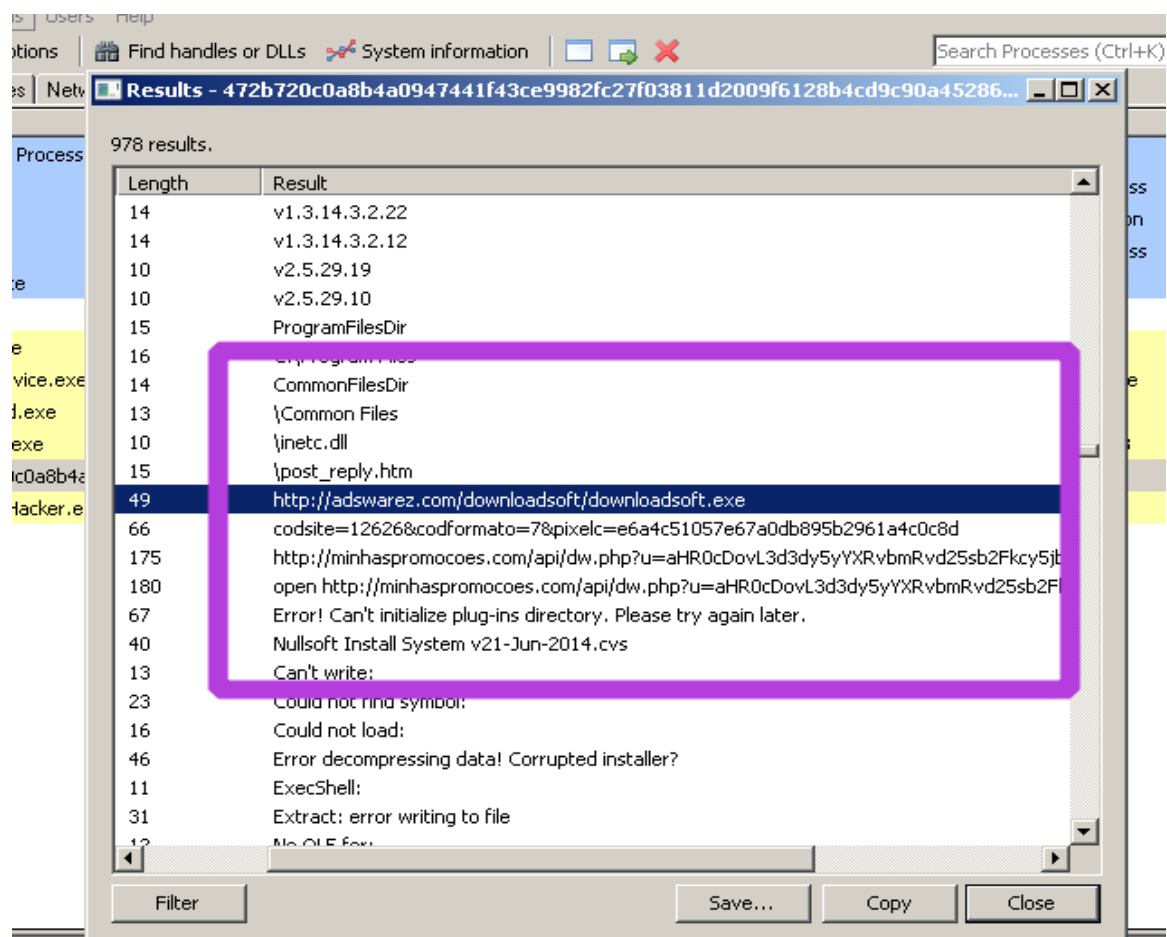
```
InitCommonControls();
SetErrorMode(0x8001);
_DAT_0042ec18 = OleInitialize((LPVOID)0x0);
DAT_0042eb64 = FUN_00405d2e(8);
/* Retrieves information about an object in the file system, such as a file,
   folder, directory, or drive root. */
SHGetFileInfo((LPCSTR)&lpString_00428f98,0,(SHFILEINFO *)&stack0x00000020,0x160,0);
FUN_00405d0e((LPCSTR)&stackText_0042e360,&MSG_00400154);
CmdLineStringPnt = GetCommandLineA();
FUN_00405a0c(&DAT_00434000,CmdLineStringPnt);
hInstance_0042eb60 = (HINSTANCE)GetModuleHandleA((LPCSTR)0x0);
pcVar3 = &DAT_00434000;
```

As I mentioned above, the malware interacts with the victim computer by creating, modifying, and deleting a few files. according to the ProcDot tool, most of the time the malware interacts with the default browser default files.

- Internet Explorer → C:\User\<UserName>\AppData\Local\Microsoft\
- Firefox → C:\User\<UserName>\AppData\Local\Mozilla\Firefox\Profiles
- Chrome → C:\User\<UserName>\AppData\Local\Google\User Data

Furthermore, Process Hacker was used track opening and closing processes. And this malware and spawned processes has interesting strings such as:

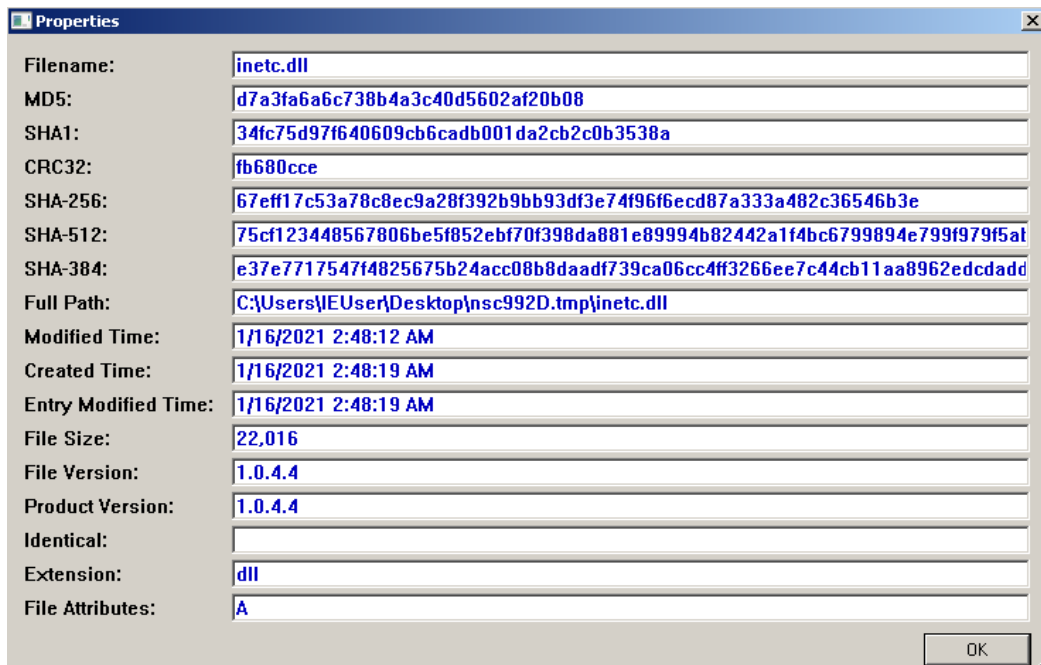
- <http://adswarez.com/downloadsoft/downloadsoft.exe>
- Inetct.dll
- Post_repy.html
- Open <http://minhaspromocoas.com/api/dw.php>



Later, this report will delve deeper into these interesting strings.

Created Files:

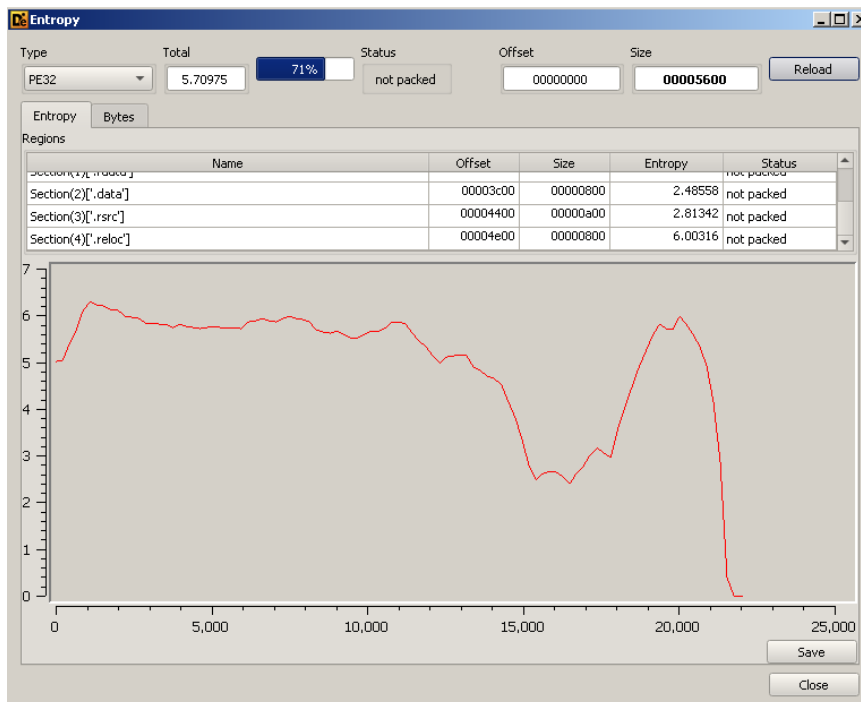
Around 30 files are created by the malware. Most of these files are either deleted or renamed by the malware itself. The below Process Monitor snapshot was taken while the malware Sample one is being run. As this snapshot illustrates, SampleOne.exe (which is the malware sample) created a few files while it was running. The integrity of these created files is high. Therefore, it is clear that the malware interacts with highly confidential data on the victim's computer.



According to pestudio, this is also a 32-bit, GUI subsystem windows dynamic-link-library file. Surprisingly, its compiler stamp is on 20 July 2014. And its size is 22016 bytes.

cpu	32-bit
subsystem	GUI
compiler-stamp	0x53CBDDCB (Sun Jul 20 08:18:35 2014)

And entropy is around 5.7. Which is not packed. Furthermore, this file contains four sections. Which are .text, .rdata, .data, rsrc and .reloc.



Therefore, this is an internet client plug-in. This allows downloading and uploading files into a remote client or server. Furthermore, this internet implementation uses MS windInet API. and it supports a wide range of network protocols such as HTTP/HTTPS and FTP protocols.

The reason why the malware author has used this file because usernames or passwords don't need to be set with this script. Therefore, this script allows to login to a server without revealing any login information to the anyone. (Sikorski, Honig, 2012)

Accessed Files:

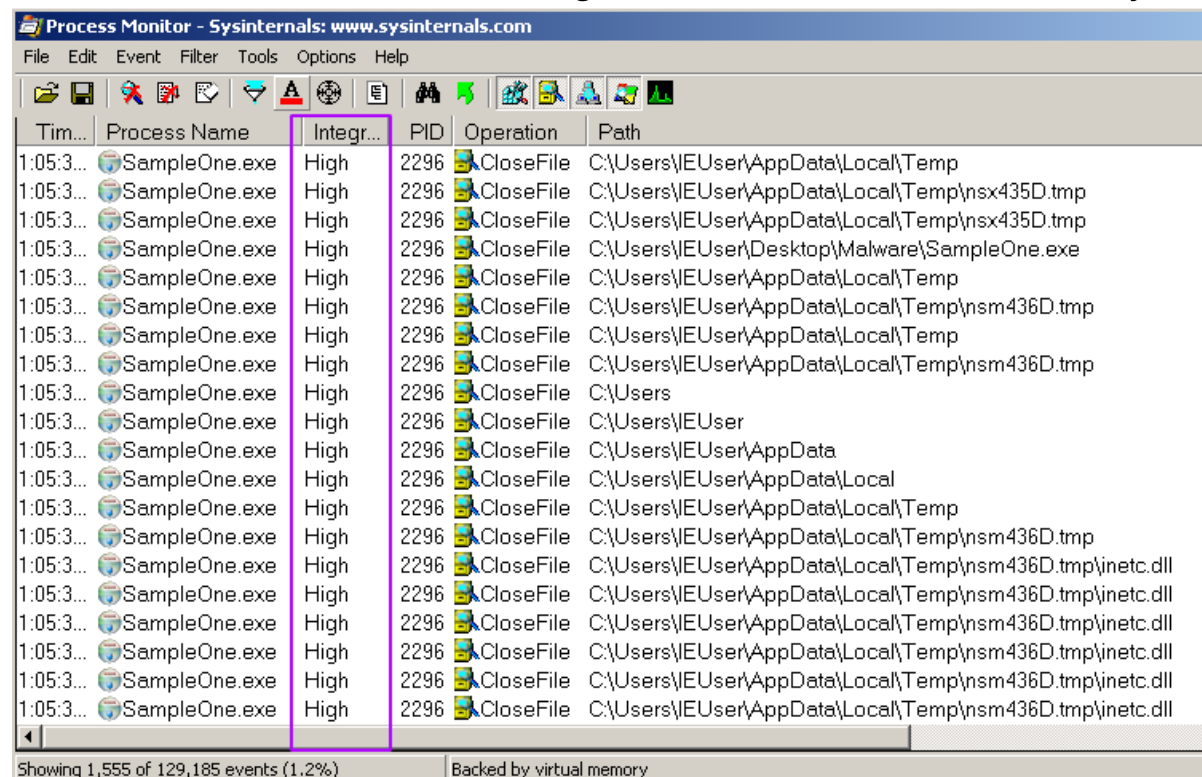
Furthermore, the malware has reached these files in order to obtain more information about the victim and the victim's network communication.

- C:\Users\<Username>\AppData\Local\Administrator\Application Data\Microsoft\Network\Connections\Pbk*.pbk
- C:\All Users\Application Data\Microsoft\Network\Connections\Pbk*.pbk
- C:\WINDOWS\system32\Ras*.pbk

These files allow checking whether the victim is using any VPN or proxy networks.

Removed or closed Files:

As I mentioned above, almost every file that was created by the malware is removed from the file system by malware itself. Below screenshot which was taken with Process Monitor while the malware is being run, illustrates the above statement very well.



Tim...	Process Name	Integr...	PID	Operation	Path
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsx435D.tmp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsx435D.tmp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\Desktop\Malware\SampleOne.exe
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp\inetcd.dll
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp\inetcd.dll
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp\inetcd.dll
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp\inetcd.dll
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp\inetcd.dll
1:05:3...	SampleOne.exe	High	2296	CloseFile	C:\Users\EUser\AppData\Local\Temp\nsm436D.tmp\inetcd.dll

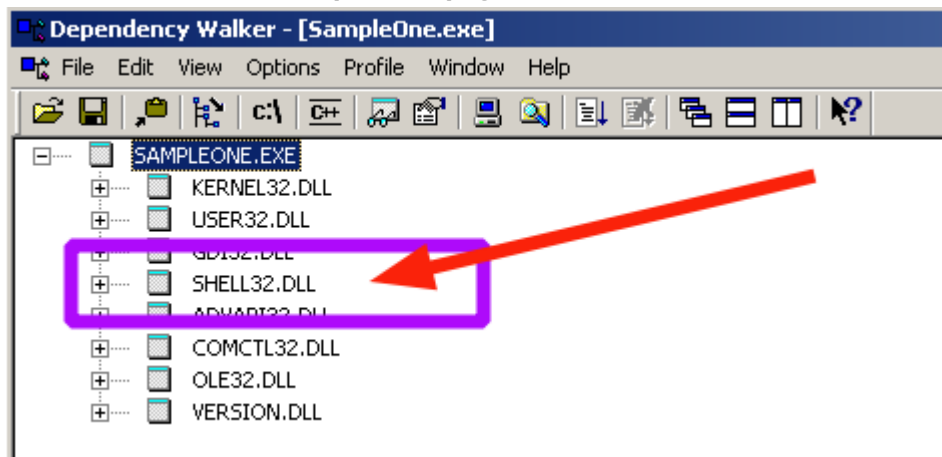
Showing 1,555 of 129,185 events (1.2%) Backed by virtual memory

However, among these deleted files, some files are rather noticeable than others.

- C:\User\<UserName>\ApptData\Local\Temp\nsn2.tmp
- C:\User\<UserName>\ApptData\Local\Temp\nsn3.tmp
- C:\User\<UserName>\ApptData\Local\Microsoft\Temporary Internet Files\Content.IE5\C1OS62RY\downloadsoft [1].exe
- C:\Documents and Settings\Administrator\Local Settings\Temp\nsn3.tmp\inetcd.dll

Network Behaviours:

There are quite a few network activities that can be noticed with malware. According to the Dependency Walker tool. This malware sample has shell32.dll. Which is an API which is used to open webpages.

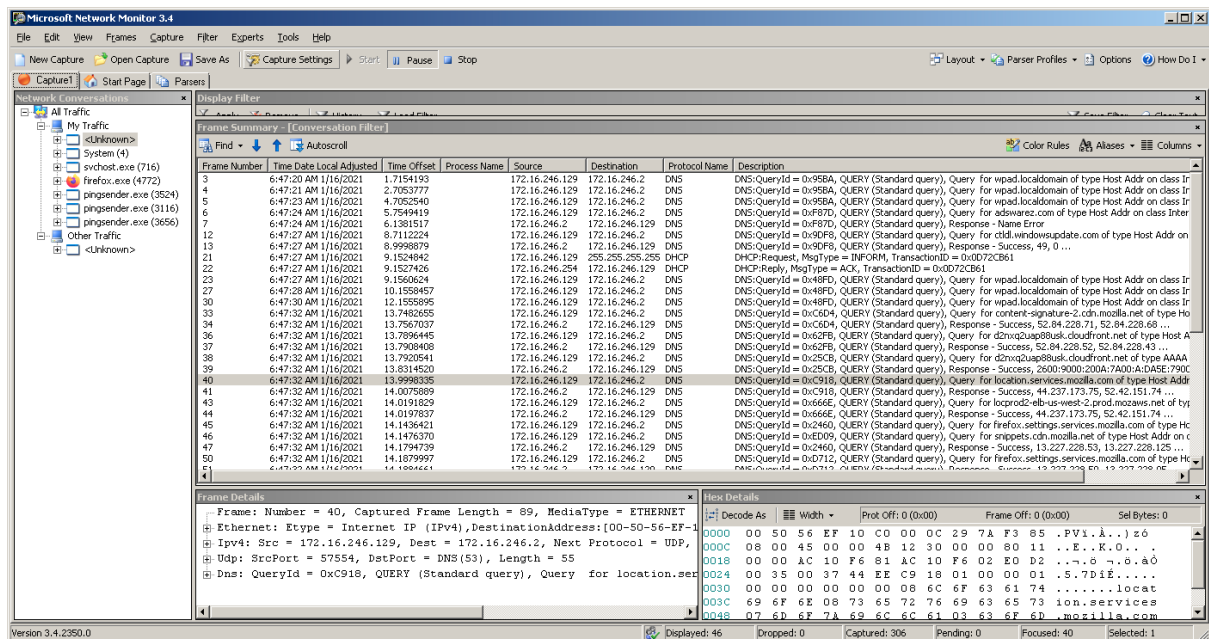


According to the below screenshots with wireshark,

- 99% packets protocol was IPv4. Among that around 88% were TCP and others were UDP. But Interestingly enough, there was no data captured that wireshark cannot understand.

Protocol	Percent Packets	Packets	Percent Bytes
Frame	100.0	101	100.0
Ethernet	100.0	101	3.4
Internet Protocol Version 6	1.0	1	0.1
User Datagram Protocol	1.0	1	0.0
DHCPv6	1.0	1	0.2
Internet Protocol Version 4	99.0	100	4.8
User Datagram Protocol	10.9	11	0.2
Transmission Control Protocol	88.1	89	89.4

- Moreover, Microsoft Network monitor was utilized to check network communications. According to Mc NetMoc an Unknown process makes most of the network communication.



At the first glance, the malware makes three DNS requests. To monitor that I made a fake dns server. Here are the results of it:

1. 35.201.126.110
2. 104.28.0.35
3. 104.28.16.74

Using OSINT, I tried to find who is belonging to these ip addresses. Unfortunately, these ip addresses belong to well-known companies.

- First, I gathered some information about 35.201.126.110 using OSINT. This is IPv4 and public class A ip address. and its host provider is Google. And its mail provider is GoDaddy.com. And geolocation is in the USA. however, these are not real information of the malware authors.

```
NetRange:      35.192.0.0 - 35.207.255.255
CIDR:          35.192.0.0/12
NetName:       GOOGLE-CLOUD
NetHandle:     NET-35-192-0-0-1
Parent:        NET35 (NET-35-0-0-0-0)
NetType:       Direct Allocation
OriginAS:
Organization:  Google LLC (GOOGL-2)
RegDate:       2017-03-21
Updated:       2018-01-24
Comment:       *** The IP addresses under this Org-ID are in use by Google Cloud customers ***
Comment:
```

Screenshot of searching ip address in linux whois

The malware connects to this site several times. Most of the time it uses <http://adexchange guru.com/jump> directory. And it redirects to another website.

SendCancel<>Follow redirection

Request

PrettyRaw\nActions

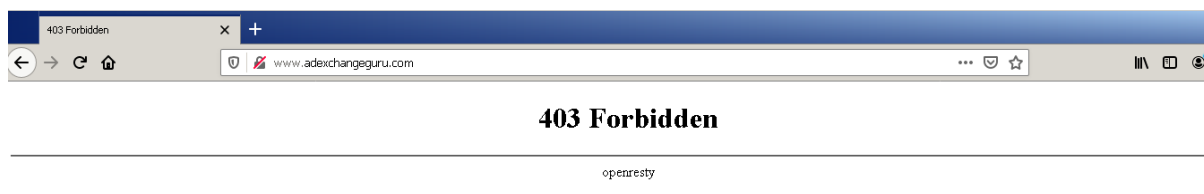
```
1 GET /jump/next.php?r=449015&sub1=itc HTTP/1.1
2 Host: www.adexchangeguru.com
3 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
```

Response

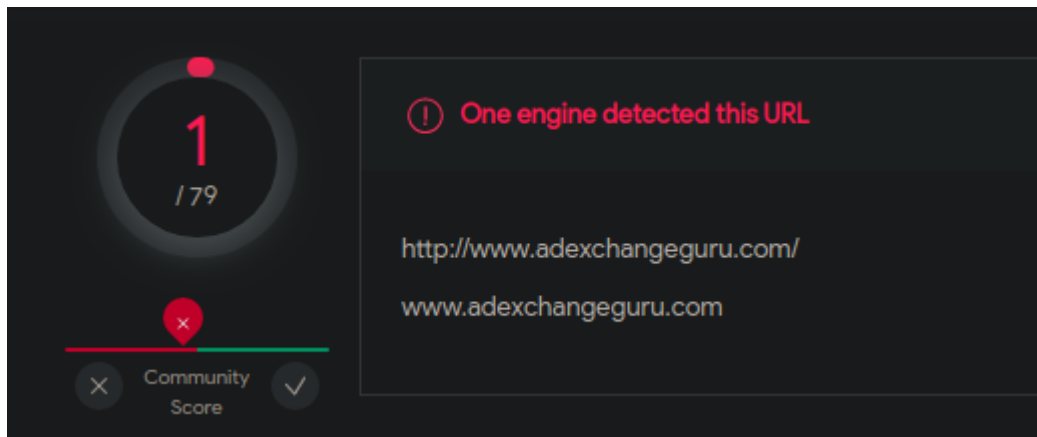
PrettyRawRender\nActions

```
32 unctio checkDocumentBody() {
33     return (typeof document.body != 'undefined' &&
34         ((document.body != null) || (typeof document.getEl
35     ));
36
37
38 / Appends first element in html to body. Works in a
39 unctio documentAsyncWriteElementFromHtml(html)
40
41 if (!checkDocumentBody()) {
42     return setTimeout (documentAsyncWriteElementFromH
43 }
44 else {
45     var tempDiv = document.createElement('div');
46     tempDiv.innerHTML = html;
47     var element = tempDiv.firstChild;
48     document.body.appendChild(element);
49 }
50
51
52 unctio ReopenUrlBuilder(baseUrl) {
53
54     this.baseUrl = baseUrl;
55
56     /**
57      * Get value of content attribute of meta tag
58      * Fallback to top if possible
59      * @return string
60      */
61     this._getMetaContent = function (name) {
62         try {
63             var meta = window.top.document.getElementsByTagName
64             for (var i = 0;
65                 i < meta.length;
66                 i++) {
67                 if (meta[i].hasAttribute('name') && meta[i].
68                     var info = meta[i].getAttribute('content')
69                     return this._getSafeSizeSubString(info);
69             }
69         }
69     }
```

However, this domain registered to a website called, www.adexchangeguru.com. Then I tried access to the website. However, accessing is forbidden.



Then, I used Dirbuster and Gobuster to find any hidden directories on this website. After that, Virustotal was used to search that IP address. Interestingly, only one search engine detects that the IP address is malicious.



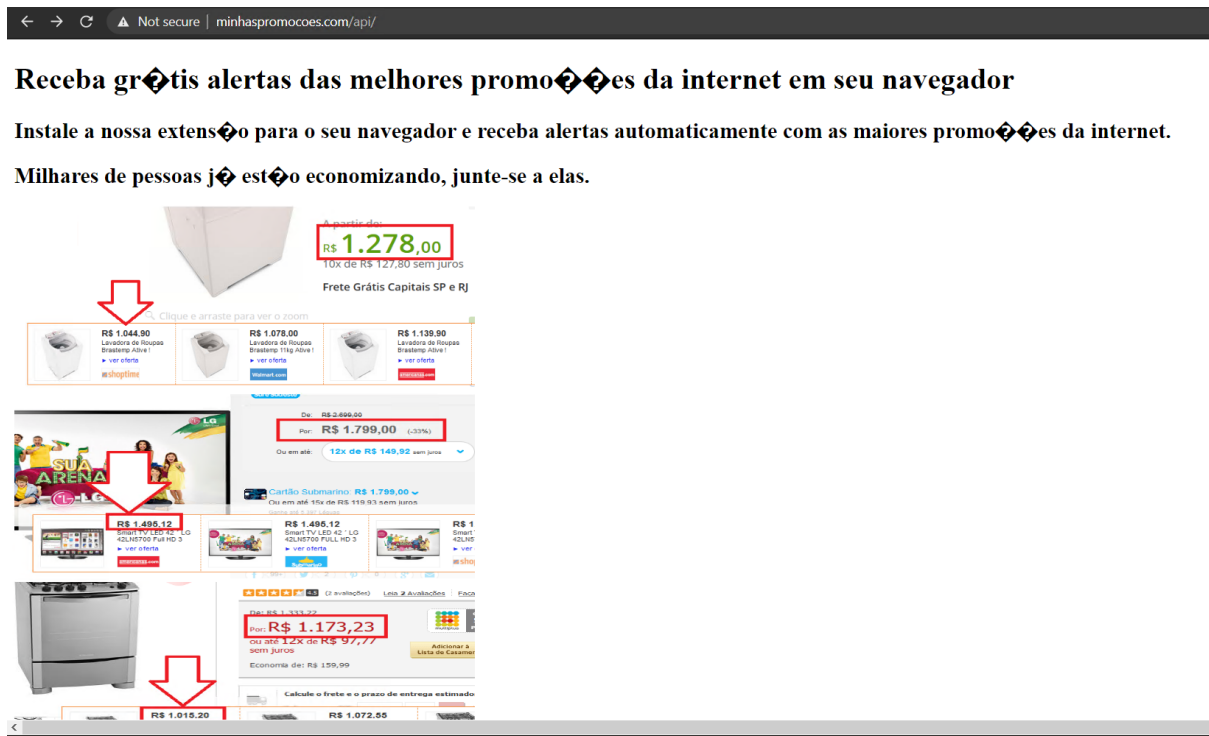
- 104.28.0.35 is also an IPv4 which belongs to Cloudflare. Therefore, the geolocation which is the USA could not be the real location where the network traffics ahead to. Here is the information that you get with whois tool in Linux about the above IP address.

```
OrgName:      Cloudflare, Inc.
OrgId:        CLOUD14
Address:      101 Townsend Street
City:         San Francisco
StateProv:    CA
PostalCode:   94107
Country:      US
RegDate:      2010-07-09
Updated:      2019-09-25
Ref:          https://rdap.arin.net/registry/entity/CLOUD14
```

And this ip address registered as www.minhaspromocoas.com. I navigated to this website. And it was an online shopping website. Accept-Encoding: gzip, deflate.



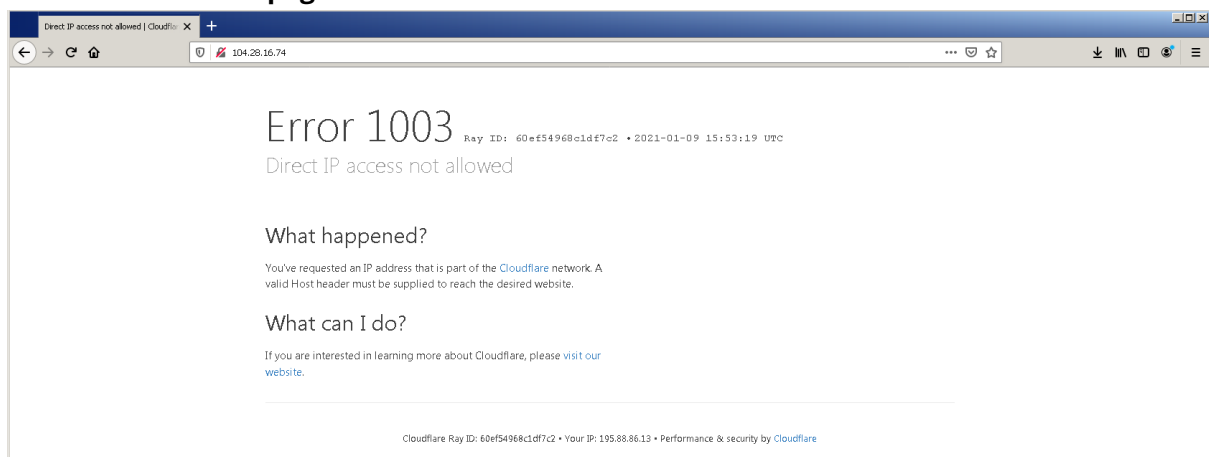
Moreover, with this malware it first accesses this website API.



- The last ip address which is 104.28.16.74 also a Cloudflare IP address. And its domain name is adswarez.com.



This is how the webpage looks like.



Furthermore, Burp Suite was utilized in order to check all the network communications. And some interesting communication was able to find out. Here are some interesting GET requests.

The below Request is my malware to www.adexchange.com.

```
REQUESTS :
GET /jump/next.php?r=449015&sub1=itc HTTP/1.1
Host: www.adexchange.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0)
Gecko/20100101 Firefox/84.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

And This the reply for the above request from the server. This HTML code first checks the browser with the user agent. After that it checks whether the adobe flash is supported with the browser.

```
// Check for the flash support
b.flash_support = false;
try {
    b.flash_support = navigator.mimeTypes['application/x-shockwave-flash'];
}
catch (e) {
}
```

Then it checks the version of the browser.

```
// Get the browser version
b.version = (b.safari) ? (n.match(/.+(?:ri)[\/: ]([\d.]+)/) || [])[1] : (n.match(/.+(?:ox|me|ra|ie)[\/: ]([\d.]+)/) || [])[1];

b.touchable = 'ontouchstart' in document.documentElement;

// Get the major browser version, like Chrome 41 or Firefox 38, from the full version
b.major_version = parseInt(b.version);

/* Detect if the current browser is a mobile browser or not. */
b.is_mobile = b.android || b.ios || b.blackberry || b.ms_mobile || b.opera_mini || b.ucbrowser;
```

Finally, it checks whether the device is a computer or a phone. Nonetheless, this is normal web communication. But this will help the adexchange.com to send more customized advertisements to the victim. For instance, sometimes adexchange.com redirects to the web pages based on the user's location.

```
HTTP/1.1 302 Moved Temporarily
Server: openresty
Date: Sat, 09 Jan 2021 12:35:39 GMT
Content-Type: text/html; charset=utf-8
Access-Control-Allow-Origin: *
Location: https://click.vcommission.com/t/NDkzXzE/?p1=16101957321887891855178840974730184&p2=449015-665822637-0&p3=449015&p4=Adcash&source=SriLanka
Referer-Policy: no-referrer
Via: 1.1 google
Connection: close
Content-Length: 0
```



However, there is no evidence that malware is self-promoting over the Internet.

Furthermore, this malware accesses and either deletes or modifies very important registries of the windows. For example: below Ghidra's screenshot illustrates the function that malware used to retrieve registry values.

Here are the registries that malware used get value from:

-
- The screenshot shows the Windows Registry Editor. The left pane displays the tree structure, with 'Connections' selected under 'Internet Settings'. The right pane shows a list of registry values. The 'Edit Binary Value' dialog box is open, displaying the hex data for 'SavedLegacySettings'.
- | Name | Type | Data |
|---------------------------|------------|--|
| (Default) | REG_SZ | (value not set) |
| DefaultConnectionSettings | REG_BINARY | 46 00 00 00 03 00 00 00 09 00 00 00 00 00 00 00 0... |
| SavedLegacySettings | REG_BINARY | 46 00 00 00 67 00 00 00 09 00 00 00 00 00 00 00 0... |
- Edit Binary Value**
- Value name: SavedLegacySettings
- Value data:
- | | | |
|------|-------------------------|----------|
| 0000 | 46 00 00 00 67 00 00 00 | F...g... |
| 0008 | 09 00 00 00 00 00 00 00 | |
| 0010 | 00 00 00 00 00 00 00 00 | |
| 0018 | 04 00 00 00 00 00 00 00 | |
| 0020 | 20 65 BD 75 9F E3 D6 01 | eku.80. |
| 0028 | 00 00 00 00 00 00 00 00 | |
| 0030 | 00 00 00 00 02 00 00 00 | |
| 0038 | 17 00 00 00 00 00 00 00 | |
| 0040 | FE 80 00 00 00 00 00 00 | b..... |
| 0048 | D5 7D 85 E2 0B 69 BA 33 | 0y.ä.i²3 |
| 0050 | 0E 00 00 00 1C 00 00 00 | |
| 0058 | 00 00 00 00 00 00 00 00 | |
| 0060 | 00 00 00 00 20 00 00 00 | |
- OK Cancel

And it deletes.

- HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer
- HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyOverride

These Registry keys are used to access the binary value of DefaultConnectionSettings which has information about proxy configuration.

Thus, if the user uses proxy or VPN connections. This malware will disable them. This behaviour allows the malware to identify its victims real IP address.

Detection rules

YARA Rules

This malware does not contain many unique strings. Therefore, identifying the malware with these unique strings is quite difficult. However, following Yara rules have the full capability to detect the malware.

```
private global rule IsPE
{
    condition:
        // MZ signature at offset 0 and ...
        uint16(0) == 0x5A4D and
        // ... PE signature at offset stored in MZ header at 0x3C
        uint32(uint32(0x3C)) == 0x00004550
}
```

```

private global rule ExeFiles
{
    strings:
        a = /\.:\ 7ZemK659.exe/ wide ascii
        b = /\.:\ OnUljwXt.exe/ wide ascii
        c = /\.:\ lqb94otL.exe/ wide ascii
        d = /\.:\ AHXjDwFP.exe/ wide ascii
        e = /\.:\ siVFWZ2z.exe/ wide ascii
        f = /\.:\ CZiQpHm_.exe/ wide ascii
        g = /\.:\ 8qZedrTG.exe/ wide ascii

    condition:
        3 of them
}

rule FileSizeExample
{
    condition:
        filesize > 350KB
}

rule SampleOne : Adware
{
    meta:
        author = "Dakshitha Perera"
        description = "Yara rules for malware Sample One"
        data = "02/01/2020"

    strings:
        ErrorNISIPage = "http://nsis.sf.net/NSIS_Error"
        PathDesktop = /\. {2,20} \ ResourceLocale/ nocase
        Debugpath =
        /\. bf0cc9d735b4ffadda494dceb25f93374b9da7fb98d9be188cbdbbbe953619a9/
        wide ascii
        SecurityBaseAPI = "AdjustTokenPrivileges"

    condition:
        all of them and IsPE and ExeFiles
}

```

First rule in this file is IsPE. This rule has the ability to identify all the PE or executable files in the given directory.

```

private global rule IsPE
{
    condition:
        // MZ signature at offset 0 and ...
        uint16(0) == 0x5A4D and
        // ... PE signature at offset stored in MZ header at 0x3C
        uint32(uint32(0x3C)) == 0x00004550
}

```

This rule is looking for the MZ signature which is 0x5A4D. And it searches the first 16 bits of every file in the given directory. After that it checks the PE signature itself. The entrance of the PE signature is always at 0x3C. Therefore, it searches 0x3C and checks the

files, values are equal to 0x00004550. If it does not match. Yara will ignore that file and will not search other rules with the file.

As the following screenshot shows, The Adware includes some of the executable files as its strings.



This is special to this file. Then, the next rule searches whether the above found executable file has any three of these executable files as its strings.

```
private global rule ExeFiles
{
    strings:
        a = /\. : 7ZemK659.exe/ wide ascii
        b = /\. : OnU1jwXt.exe/ wide ascii
        c = /\. : lqb94otL.exe/ wide ascii
        d = /\. : AHXjDwFP.exe/ wide ascii
        e = /\. : siVFWZ2z.exe/ wide ascii
        f = /\. : CZiQpHm_.exe/ wide ascii
        g = /\. : 8qZedrTG.exe/ wide ascii

    condition:
        3 of them
}
```

Furthermore, Regular expressions have been used in order to write this code. Because it is unclear whether these files are stored. It depends on the Windows version and some other facts.

```
rule FileSizeExample
{
    condition:
        filesize > 350KB
}
```

This rule checks whether found executable is more than 350 KB.

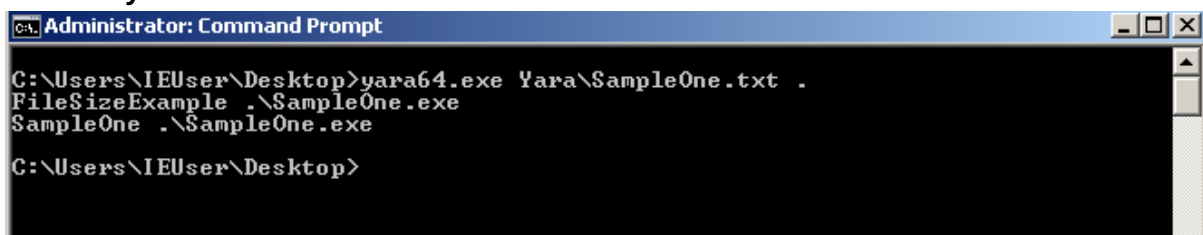
This is the main rule. Mainly this searches the most unique and important features of the malware.

```
rule SampleOne : Adware
{
    meta:
        author = "Dakshitha Perera"
        description = "Yara rules for malware Sample One"
        data = "02/01/2020"

    strings:
        $ErrorNISIPage = "http://nsis.sf.net/NSIS_Error"
        $PathDesktop = /.{2,20} ResourceLocale/ nocase
        $Debugpath =
        /.: bf0cc9d735b4ffadda494dceb25f93374b9da7fb98d9be188cbdbbbe953619a9/
        wide ascii
        $SecurityBaseAPI = "AdjustTokenPrivileges"

    condition:
        all of them and IsPE and ExeFiles
}
```

Generally speaking, the condition is “find an executable file in the given directory and it has to contain at least three of above-mentioned executables as strings and the file should contain AdjustTokenPrivileges API. And finally, the file’s size should be more than 350 KiloBytes.



```
Administrator: Command Prompt
C:\Users\IEUser\Desktop>yara64.exe Yara\SampleOne.txt .
FileSizeExample .\SampleOne.exe
SampleOne .\SampleOne.exe
C:\Users\IEUser\Desktop>
```

Snort Rules

To detect this malware network behaviour, Snort rules could be utilized. There are three IP address are mentioned during the network behaviour part of this report. Thus, detecting these malicious Ips with snort could be assisted to check whether the devices is infected by this malware.

```
alert TCP any 80 -> 35.201.126.110 any (msg: "Code Red: accessing adware website."; \
flags: S; \
reference: "www.adexchange guru.com")
```

```
alert TCP any 80 -> 104.28.0.35 any (msg: "Code Yello: Accessing to a less security website."; \
content: "/api/dw.php"; depth: 2; \
reference: "minhaspromocoes.com")
```

The first rule detects any ongoing traffic to 35.201.126.110. That is the reason why it mentioned any source IP addresses. And it uses the port number 80. Since this connection is a HTTP. It uses port number 80 as the source port. and its destination IP is 35.201.126.110 which is the adexchange guru. And the port could be anything. If these conditions meet, the first rule will present an alert with a warning message. Furthermore, this rule's flags have been set to S, which stands for SYN flag. Hence, this rule has the capability to warn the user in very early stage about malicious web traffics.

Second rule detects any ongoing web traffic to 104.28.0.35 which is the IP of minhaspromocoes.com. This is a website with less secure protocols. Interacting with this website is always a risk. Therefore, it will warn the user once these conditions meet. In order to warn user, web traffic content must include /api/dw.php. This is a web file it uses to redirect traffic to adexchange.exe.

Detection and Removal Instructions

According to the analysed information, this program can be rather identified as an adware. Adware stands for Advertisement-support software. Every adware is not harmful or malicious. However, this clearly steals user's information without user's knowledge. Therefore, this act of malvertising can be harmful for users. Thus, beyond the adware, the name of the spyware or a web browser hijacker can also be used for this.

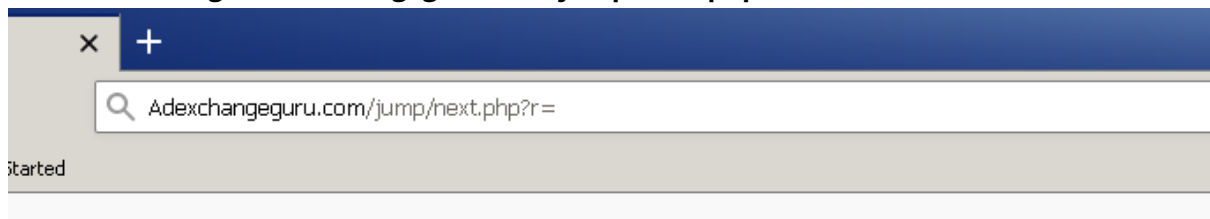
Adware and Spywares presence is a hard question to answer. However, there are two main ways in which spyware could reach into the computer.

1. Through free installation and share installation. These adware or spywares come as a trojan from freeware or shareware. Thus, the best way to avoid trojans is by downloading and installing software from trusted and well-known sources.
2. Infected websites are another way of reaching adware to a device. If the user accessed an infected website. It can cause a malicious installation on the device.

Instead of these ways, social engineering is another way of accessing these types of adware to the computer.

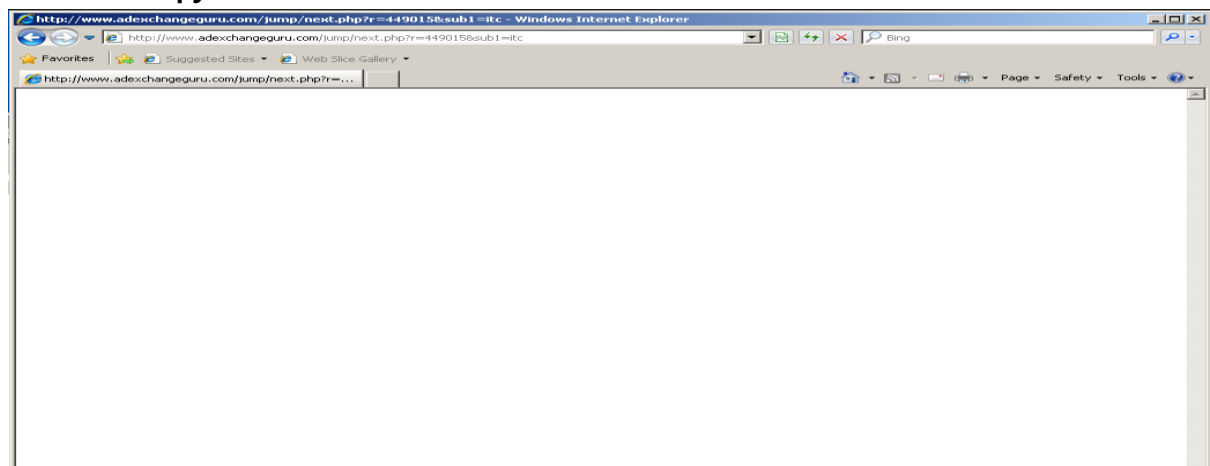
However, if the computer gets infected with this malware. There are few easy ways of detecting that the computer is infected.

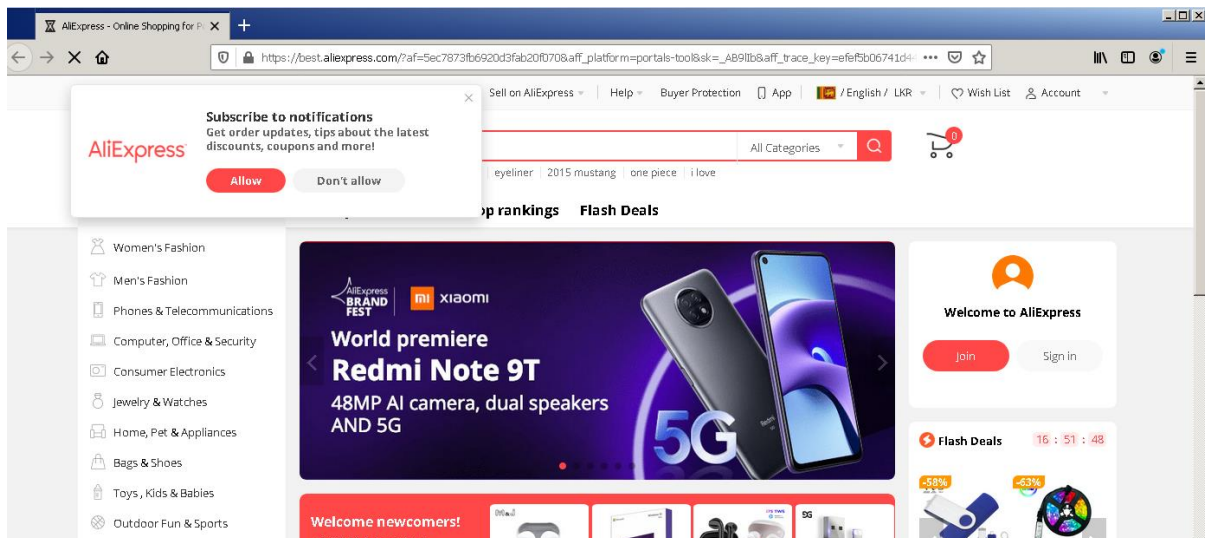
1. Most noticeable difference of the web browser is that its homepage and default search engine could be changed or modified. Normally, adexchange guru changes them by adding a value identified onto them. For instance: the search engine could be changed adexchange guru.com/jump/next.php?r=.



This link allows adexchange guru to redirect all your search queries to their remote servers. This helps them show advertisements according to your preferences.

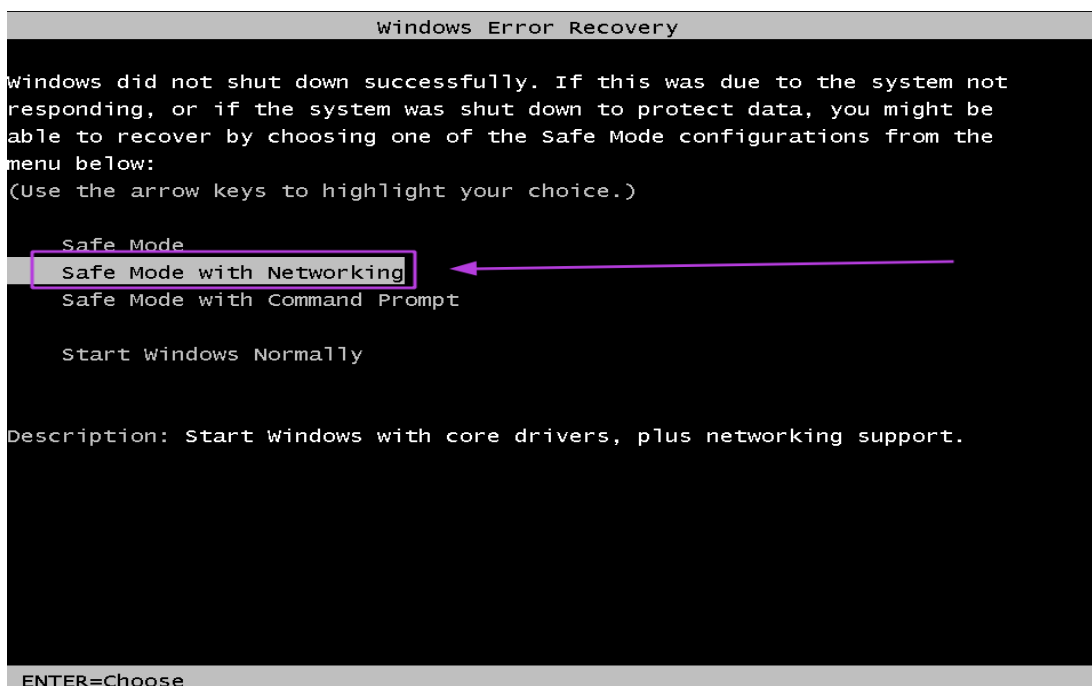
2. If the browser (Chrome, Internet Explorer or Firefox) pops up advertisements from Adexchange guru.com time to time. Or redirecting to some advertisements or even blank webpages from the website. So, this could be a sign of an infected victim of this Spyware.



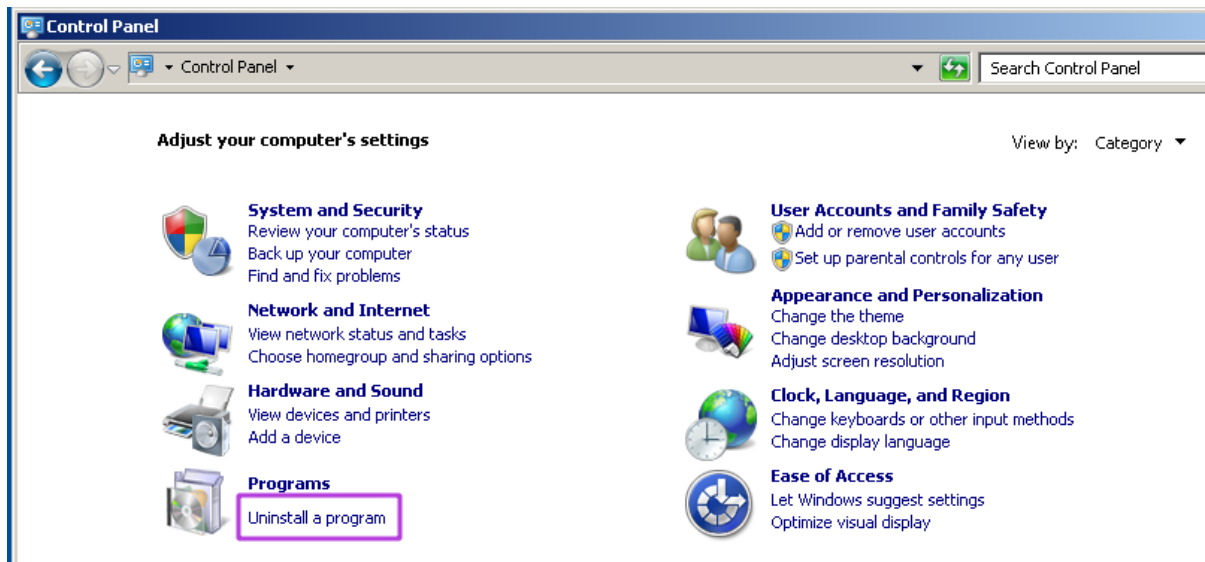


This malware can be removed in a few ways.

1. Using the Windows Defender - If the windows defender was turned off, first activate it. Because Windows defender has the full capability to identify this malware and remove it from the system. And do a virus scan. The windows defender is totally capable of detecting and removing this malware. However, this step does not remove unwanted configuration that occurred to the web browser.
2. Manually removing the malware is the other way.
 - a. First reboot the machine into "Safe mode with Networking".




- b. Uninstall unwanted Programs from the computer. - Navigate to the control panel on the windows machine. And then go to the Uninstall program.

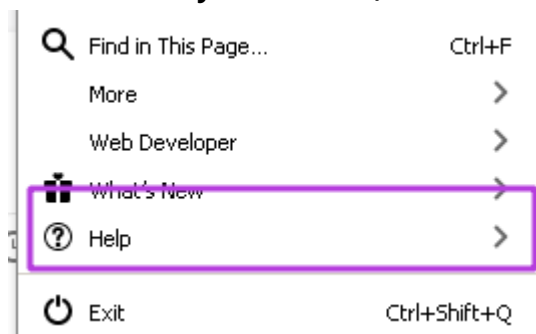


After that you can delete unwanted programs from your computer.

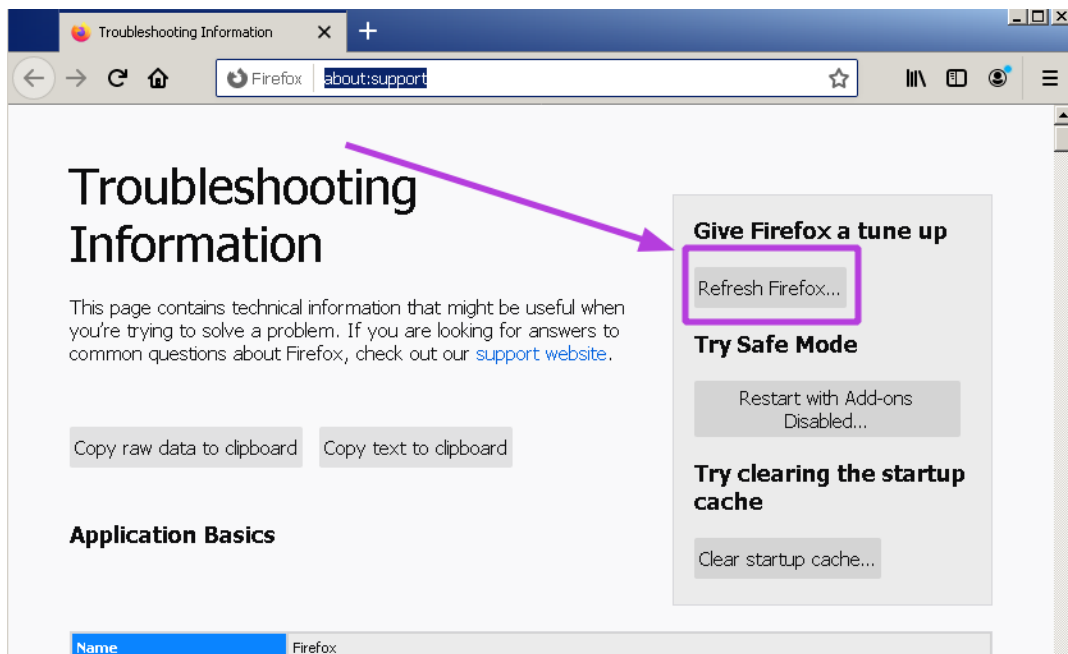
c. Remove ads from the default browser -

Firefox: -

If your default browser is firefox, click on this hand burger menu  in the right-side corner of your browser, and then click the help button right before the Exit.



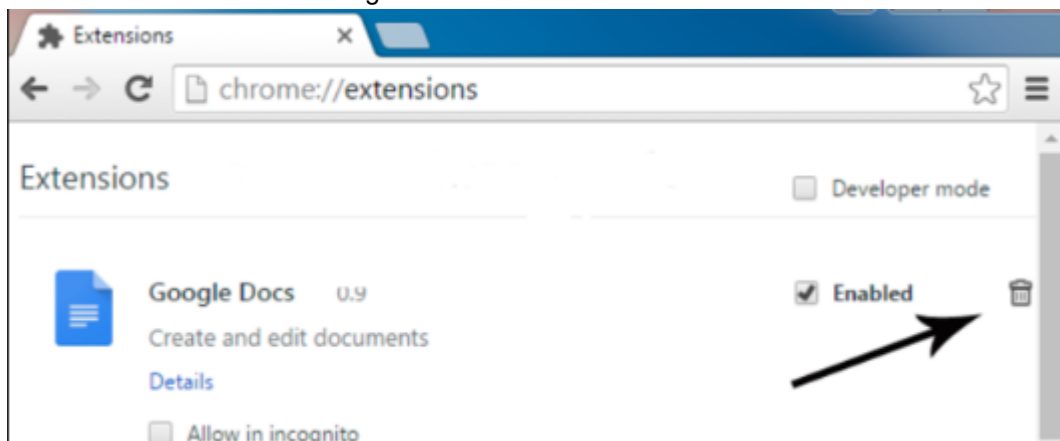
After that go to the troubleshooting information. And it will direct you to a webpage like this.



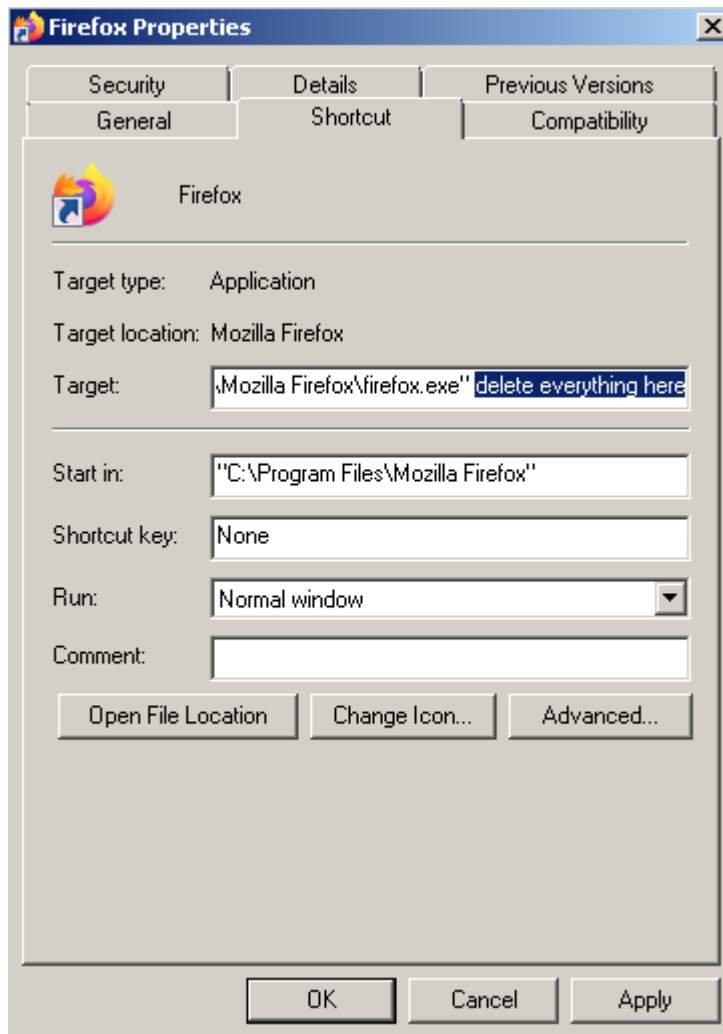
Finally click on Refresh Firefox. This will create a new and refreshed Firefox. And user ad blocking extension. And make sure that unwanted extensions are removed. Or remove them manually by going to



→ Settings → Extensions

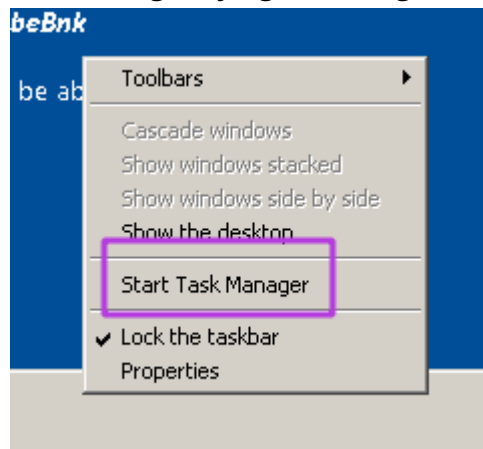


- d. Remove shortcuts for Adexchangeguru.com. First go to the default browser properties by right clicking on its icon on Windows.



After that delete everything after "<webbrowser>.exe".

- e. Kill suspicious processes from the task manager. To do that first go to the task manager by right clicking on the Windows taskbar.



And terminate unwanted programs from the task manager.

3. Or this python program can be used to remove this malicious program from the computer.

```
4. # Autor: Dakshitha Perera
5. # Data : 19/01/2021
6. # Title : program to remove Malware Sample One.
```

```

7.
8. from winreg import *
9. import sqlite3
   import winreg
   import os
   import re
10.
11. def ChromeHistoryDelete():
12.     username = os.getlogin()
13.
14.     db =
sqlite3.connect('c:\\Users\\{ }\\AppData\\Local\\Google\\Chrome\\User
Data\\Default\\History'.format(username))
15.     c = db.cursor()
16.
17.     URLPattern = re.compile(r"https?:\/\/([^\/]+)\/")
18.     domains = {}
19.
20.     result = True
21.     id = 0
22.     while result:
23.         result = False
24.         ids = []
25.
26.         for row in c.execute('SELECT id, url, title FROM urls WHERE id > ?
LIMIT 1000', (id,)):
27.             result = True
28.             match = URLPattern.search(row[1])
29.             id = row[0]
30.
31.             if match:
32.                 domain = match.group(1)
33.                 domains[domain] = domains.get(domain, 0) + 1
34.                 # clean if this is true
35.                 if "adexchange" in domain:
36.                     ids.append((id,))
37.
38.         c.executemany('DELETE FROM urls WHERE id=?', ids)
39.         db.commit()
40.
41.     db.close()
42.
43. def ClearChromeCookies():
44.     username = os.getlogin()
45.
46.     db =
sqlite3.connect('c:\\Users\\{ }\\AppData\\Local\\Google\\Chrome\\User
Data\\Default\\Cookies'.format(username))
47.     c = db.cursor()
48.
49.     result = True
50.     while result:
51.         result = False
52.         ids = []
53.
54.         for row in c.execute('SELECT creation_utc, host_key, name FROM
Cookies'):

```

```

55.         id = row[0]
56.         if row[1] == '.adexchange.com':
57.             ids.append((id, ))
58.
59.         c.executemany('DELETE FROM Cookies WHERE creation_utc=?', ids)
60.         db.commit()
61.
62.     db.close()
63.
64. def ClearChromeCache():
65.     username = os.getlogin()
66.
67.     db = ('C:\\Users\\{}\\AppData\\Local\\Google\\Chrome\\User
Data\\Default\\Cache'.format(username))
68.
69.     for root, dirs, files in os.walk(db):
70.         for filename in files:
71.             pattern = re.compile(r'^f.')
72.             matches = pattern.finditer(filename)
73.             for match in matches:
74.
75.                 os.remove('C:\\Users\\{}\\AppData\\Local\\Google\\Chrome\\User
Data\\Default\\Cache\\{}'.format(username, filename))
76.                 print("Deleting {}".format(filename))
77.
78. ChromeHistoryDelete()
79. ClearChromeCache()
80. ClearChromeCache()

```

References:

1. Yason, M. V. (2007). The art of unpacking. Retrieved Feb, 12, 2008.
2. Peter, F. (2011). The Ultimate Anti-Debugging Reference
3. Kendall, K., & McMillan, C. (2007, August). Practical malware analysis. In Black Hat Conference, USA (p. 10).
4. Sikorski, M., & Honig, A. (2012). Practical malware analysis: the hands-on guide to dissecting malicious software. no starch presses.
5. Sharif, M., Lanzi, A., Giffin, J., & Lee, W. (2009, May). Automatic reverse engineering of malware emulators. In 2009 30th IEEE Symposium on Security and Privacy (pp. 94-109). IEEE.
6. Debray, S., & Patel, J. (2010, October). Reverse engineering self-modifying code: Unpacker extraction. In 2010 17th Working Conference on Reverse Engineering (pp. 131-140). IEEE.