MALWARE ANALYSIS OF ZEUS BANKING TROJAN

“Static and Dynamic Analysis of Zeus Banking Trojan.”

A project report

submitted in partial fulfillment of the requirements

for the award of the degree of

POST GRADUATE DIPLOMA IN CYBERSECURITY AND FORENSICS (PG-DCSF).

Submitted by

Dinde Jayraj Vijay (230960940017)

Ritwik Mungale (230960940029)

Nikita Pakhale (230960940031)

Omkar Ingawale (230960940032)

CENTRE OF DEVELOPMENT FOR ADVANCE COMPUTING,

THIRUVANANTHAPURAM

Under the supervision of

<u>Hiron Bose</u>

(Section Head-STDC, CDAC Thiruvananthapuram)

Index

Introduction.

What is Malware? - Malware is a malicious program or software that inserts into a system, with the intention of compromising CIA (Confidentiality, Integrity, Availability).

Types of Malwares: -

- Virus: Malicious software inserted into a program or data file.
- Trojan: disguised as legitimate files but something else, attacker embedded malicious software into legitimate file.
- Worm: self-replicating program, spear over network.
- Backdoor: open backdoor to C&C instruction like RAT(remote access tool)
- Ransomware: malicious program that encrypts victim files and asks for money to decrypt. -APT (Advanced persistent threat): state-sponsored group-create malware to remain undetected for an extended period.

What is Malware Analysis?

Malware Analysis is the study or process of determining the functionality, origin and potential impact of a given malware sample and extracting as much information from it. The information that is extracted helps to understand the functionality and scope of malware, how the system was infected and how to defend against similar attacks in future.

Objectives:

- To understand the type of malware and its functionality.
- Determine how the system was infected by malware and define if it was a targeted attack or a phishing attack.
- How malware communicates with attacker.
- Future detection of malware and generating signatures.

Types of Malware Analysis:

- Static analysis – It is a process of analysing the malware without executing or running it. This analysis is used to extract as much metadata from malware as possible like P.E headers strings etc.
- Dynamic analysis – It is process of executing malware and analysing its functionality and behaviour. This analysis helps to know what malware does during its execution using debugger.

What is the Zeus Trojan?

The Zeus Trojan, Zbot, or ZeuS: all these names refer to a devious collection of malware that can infect your computer, spy on you, and collect sensitive personal details. Zeus also conscripts your computer into a botnet, which is a massive network of enslaved computers that can be controlled remotely.

Though Zeus peaked in the early 2010s, its source code leaked in 2011, making Zeus available for anyone to use as a template for their own malware. Many Zeus-based malware strains have gone on to cause widespread damage and become notorious examples of malware in their own right.

How does the Zeus Trojan work?

The Zeus Trojan is a package that contains multiple elements of malicious code that work together to infect your computer. Like all Trojan malware, Zeus must trick you into installing it — mistakenly thinking the malware is helpful, you welcome it onto your device. Once it gets inside, it unleashes its malicious payload — just like the soldier-filled wooden horse of Greek legend.

How does Zeus get on my computer?

Zeus infects its victims through two primary vectors: phishing emails and malicious downloads. The phishing attacks fool people into downloading and opening malicious attachments. Once opened, the attachments install the Zeus malware package. Other phishing emails may contain links to infected websites. Zeus can also hide in malicious online ads, which when clicked download malware onto a victim's computer. Infected websites can automatically download Zeus to your computer when you visit, and Zeus can also hide in otherwise legitimate product downloads.

What does Zeus do?

Since Zeus is available as open-source malware, its effects can vary widely. Historically, it's had two consistent roles:

Steal sensitive information. Zeus is known as a banking Trojan, but it can steal anything its operator wants it to steal: system information, stored passwords, online account credentials, and more.

Build a botnet. Zeus maintains contact with its operator through a command-and-control (C&C) server so that it can remotely receive additional instructions. The operator can hijack the victim's computer and install more malware.

Zeus originally stole passwords via Internet Explorer's Password Store feature: Zeus simply helped itself to any passwords stored in the browser. If Zeus detected that the victim was visiting a banking site, it would use keylogging or form-grabbing methods from within the browser to capture usernames and passwords.

What is Zeus used for?

Zeus was originally designed to **steal sensitive banking information**. As early as 2009, Zeus had hit computers at Bank of America, NASA, Amazon, and many other organizations, infecting an estimated 3.6 million computers that year.

The cybercriminals behind Zeus would transfer funds out of their victims' accounts and funnel the money back to themselves via intermediaries known as *money mules*. These mules would receive the stolen funds and redirect them onward, obscuring the final destination of the money.

Zeus would also give remote access to the machines it infected. This led to the creation of the **Gameover ZeuS botnet**, Zeus's most infamous successor. Botnets are often used to send spam or phishing emails, or to conduct DDoS attacks.

In 2010, the FBI successfully penetrated the Zeus cybercrime ring, arresting over 100 people in the US, the UK, and Ukraine. By that time, the group had managed to pilfer **over $70 million** from victims of Zeus attacks.

Zeus's legacy.

Gameover ZeuS might be the most famous malware to use Zeus code, but it's far from the only one. Here's a quick look at other Zeus-inspired malware:

Cthonic can access a victim's webcam and microphone in addition to their personal information.

Citadel targets a victim's password manager by attempting to access its master password, and it can also block the websites of various antivirus providers.

Atmos emerged in 2015 and targeted banks directly, harvesting financial data and leaving ransomware in its wake.

Terdot hunts for social media and email credentials in addition to a victim's banking information.

Methodology.

Common Steps in Malware Analysis:

1. Identification: Determining the presence of malware and understanding its characteristics.
2. Acquisition: Obtaining a copy of the malware for analysis, ensuring proper handling and containment.
3. Preliminary Analysis: Conducting initial assessments to gather basic information about the malware.
4. Static Analysis: Examining the malware without executing it to extract metadata and understand its structure.
5. Dynamic Analysis: Executing the malware in a controlled environment to observe its behaviour and effects.
6. Code Analysis: Analysing the malware’s code to understand its functionality, logic, and potential vulnerabilities.
7. Behavioural Analysis: Monitoring the malware’s actions during execution to identify its interactions with the system and network.
8. Reverse Engineering: Unpacking and decompiling the malware to understand its inner workings and algorithms.
9. Post-Analysis: Documenting findings, generating reports, and deriving insights for future prevention and detection.

Sandbox Environment:

A sandbox is an isolated environment where users can safely test suspicious code without risk to the device or network. Another term used to describe a sandbox is an automated malware analysis solution and it is a widely employed method of threat and breach detection.

Sandboxes most often come in the form of a software application, though, hardware alternatives do exist. Methods for implementation include third-party software, virtual machines, embedded software, or browser plug-ins. Sandboxing can detect the newest and most critical threats, foster collaboration, minimize risks, and facilitate IT governance. Antivirus software is notable for its ability to scan programs being transferred, downloaded, and stored. However, a general scan of a program’s binary only tells so much. By processing programs in a sandbox environment, we fill the security gap that existing solutions miss.

Lab Setup

Tools used for creation of sandbox:

1) Virtual Box- Downloads – Oracle VM VirtualBox
2) REMnux- Download remnux-v7-focal-virtualbox.ova (REMnux) (sourceforge.net)
3) FlareVM enabled WINDOWS 10- GitHub - mandiant/flare-vm: A collection of software installations scripts for Windows systems that allows you to easily setup and maintain a reverse engineering environment on a VM.

Steps to setup lab:

1) Install Virtual box in your host machine.
2) Install fresh Win10 VM :-

- Use Microsoft windows Media Creation Tool to download latest Windows 10 ISO file.
- Create a new Windows 10 VM using this ISO file.
- Completely disable Windows firewall and Windows defender by using How to Disable Defender Antivirus & Firewall in Windows 10 - WinTips.org
- Take a VM snapshot so you can always revert to a state before the FLARE-VM installation

3) Installing Flare-Vm: –

- Open Windows PowerShell and carefully follow the procedure mentioned under this repository GitHub - mandiant/flare-vm: A collection of software installations scripts for Windows systems that allows you to easily setup and maintain a reverse engineering environment on a VM.

4) Installing Remnux -vm: -

- Download the .OVA file associated with virtual box from Downloads – Oracle VM VirtualBox.
- Create a VM using this .OVA file and run the REMnux

5) Configuring Inetsim on Remnux: -

- Type in console **Cd /etc/inetsim** to go to inetsim directory.
- Type **sudo nano inetsim.conf** this will open GNU editor.
- Scroll down and uncomment **START_SERVICE DNS**.
- Change **service_bind_address** to 0.0.0.0
- Change **dns_default_ip** to REMnux Ip.
- Save the changes.

6) Creating Private network and connecting both the VM:-

- Go to Machine >settings>Set network adapter to ‘Host only adapter #3’ in both the VMs.
- In FlareVM Go to ethernet settings>properties>Ipv4 properties and change the Ip address of dns server to REMnux Ip address.
- Ping the respective machines to check for successful creation of private network.

- Take snapshot of both the machines. These will be treated as base machines for malware download.

7) Download ZEUS from theZoo/malware/Binaries/ZeusBankingVersion_26Nov2013 at master · ytisf/theZoo · GitHub on the DESKTOP of FlareVM. Password for extraction is 'infected'

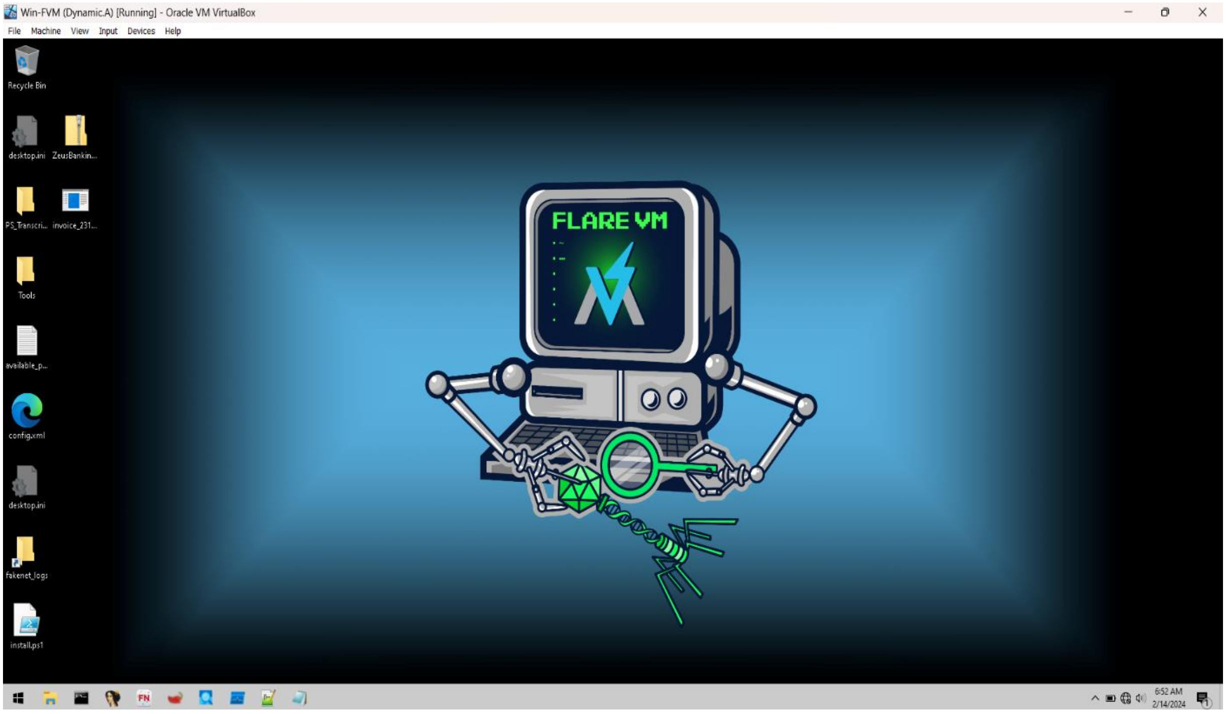


Fig.1

Finger Printing: -

Cybersecurity fingerprinting refers to a set of information that can be used to identify network protocols, operating systems, hardware devices, software among other things.

1]Hashes:-

MD5 - ea039a854d20d7734c5add48f1a51c34

SHA-1 - 9615dca4c0e46b8a39de5428af7db060399230b2

SHA-256 - 69e966e730557fde8fd84317cdef1ece00a8bb3470c0b58f3231e170168af169
2]Name:- invoice_2318362983713_823931342io.pdf.exe
The sample is trying to impersonate invoice of a company. The name extension is deliberately named .pdf prior to .exe to confuse the user.

3]Virus Total Findings-

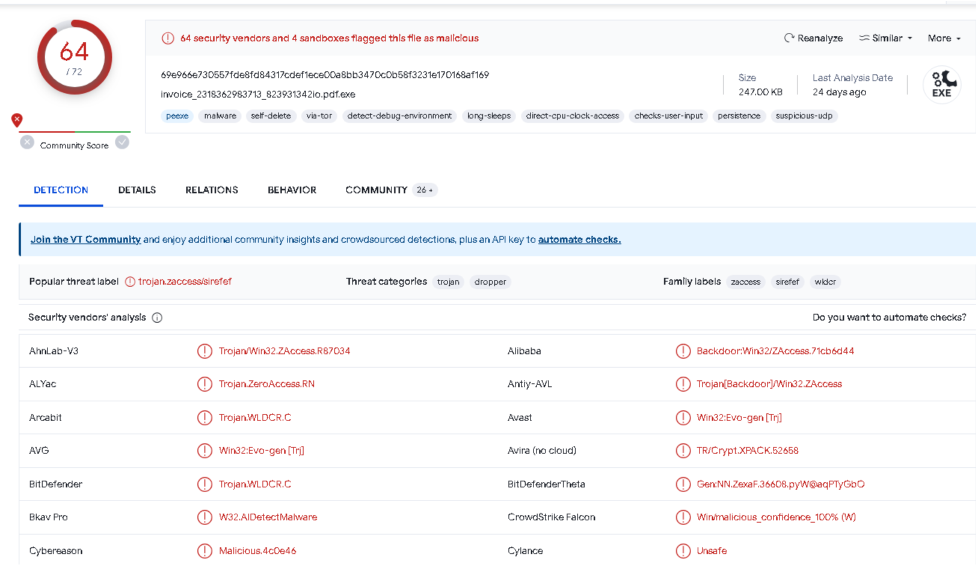


Fig. 2

The file invoice_2318362983713_823931342io.pdf.exe is flagged as malicious by 64 vendors.

4]Initials:- Below findings indicate that the initials are "MZ". The MZ signature, found in the first byte of certain files, holds historical significance. It serves as a magic number or file signature. Let's delve into its origins:

1. MZ Signature: The ASCII string "MZ" (hexadecimal: 4D 5A) appears at the beginning of these files. The initials "MZ" stand for Mark Zbikowski, one of the leading developers of MS-DOS.
2. File Format: This signature is associated with the DOS MZ executable format, used for .EXE files in DOS. When viewed as text, the contents of such files are unintelligible, as they are not intended to be read in that manner.

property	value
footprint > sha256	69E966E730557FDE8FD84317CDEF1ECE00A8BB3470C0B58F3231E170168AF169
first-bytes > hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes > text	M Z .. .. @ .. ..

Fig.3

This gives us another reason to examine the file under Portable Executable checking tools like Pestudio. Pestudio is already installed with FlareVM. We shall see the same further in Basic dynamic analysis.

Basic Static Analysis

1]Virus Total

Contacted URLs (2)

Scanned	Detections	Status	URL
2023-12-18	1 / 90	404	http://fpdownload.macromedia.com/get/flashplayer/update/current/install/install_all_win_cab_64_ax_sgn.z
2023-09-29	0 / 90	-	http://j.maxmind.com/app/geoip.js

Contacted Domains (185)

Domain	Detections	Created	Registrar
088156176110.warszawa.vectranet.pl	0 / 89	-	-
095160103115.warszawa.vectranet.pl	0 / 89	-	-
1-174-58-84.dynamic-ip.hinet.net	0 / 89	1994-03-19	Network Solutions, LLC
10-171-136-186.fibertel.com.ar	0 / 89	-	nicar
103.31.177.202.deltainfocom.com	0 / 89	2012-08-08	CloudFlare, Inc.
111-246-83-123.dynamic-ip.hinet.net	0 / 89	1994-03-19	Network Solutions, LLC
114-36-177-168.dynamic-ip.hinet.net	0 / 89	1994-03-19	Network Solutions, LLC
115-167-67-101.wi-tribe.net.pk	0 / 89	-	-
116-64-206-157.rev.home.ne.jp	0 / 89	-	-
118-169-229-231.dynamic-ip.hinet.net	0 / 89	1994-03-19	Network Solutions, LLC

• • •

Fig.4

- The file appears to contact a suspicious domain fpvdownload.com to install flashplayer.

2]PEstudio findings:-

1)Names and Domains.

- The findings hint at a domain corect.com.
- However, upon inspection the domain corect.com yielded no interesting results

names	
file	c:\users\alpha\desktop\invoice_2318362983713_823931342io.pdf.exe
debug	n/a
export	corect.com
version	n/a
manifest	n/a
.NET > module	n/a

Fig.5

2) Virtual size and raw size of the malware are same so the possibility of obfuscation is ruled out.

property	value
section	section[0]
name	.text
footprint > sha256	8309B5D320B3D392E25AFD5...
entropy	6.707
file-ratio (99.60%)	18.42 %
raw-address (begin)	0x00000400
raw-address (end)	0x0000BA00
raw-size (251904 bytes)	0x0000B600 (46592 bytes)
virtual-address	0x00001000
virtual-size (250379 bytes)	0x0000B571 (46449 bytes)

Fig.6

3)Strings Findings in PE studio:-

- The below findings indicate the malicious programs that have been flagged by Pestudio. These are basically windows APIs.

encoding (2)	size (bytes)	location	flag (17)	label (110)	group (11)	technique (7)	value
ascii	24	.itext	x	import	windowing	-	AllowSetForegroundWindow
ascii	22	.itext	x	import	reconnaissance	-	GetEnvironmentVariable
ascii	22	.itext	x	import	reconnaissance	-	GetEnvironmentVariable
ascii	9	.itext	x	import	input-output	-	VkKeyScan
ascii	16	.itext	x	import	input-output	T1056 | Input Capture	GetAsyncKeyState
ascii	19	.itext	x	import	file	-	PathRenameExtension
ascii	9	.itext	x	import	file	-	WriteFile
ascii	12	.itext	x	import	file	T1083 | File and Directory Discovery	FindNextFile
ascii	16	.itext	x	import	execution	-	GetCurrentThread
ascii	7	.itext	x	-	execution	T1106 | Execution through API	WinExec
ascii	13	.itext	x	import	data-exchange	-	GlobalAddAtom
ascii	17	.itext	x	import	data-exchange	T1115 | Clipboard Data	GetClipboardOwner
ascii	16	.itext	x	import	data-exchange	T1115 | Clipboard Data	GetClipboardData
ascii	20	.itext	x	import	data-exchange	T1115 | Clipboard Data	EnumClipboardFormats
ascii	18	.itext	x	import	data-exchange	-	DdeQueryNextServer
ascii	25	.itext	x	import	console	-	GetConsoleAliasExesLength
ascii	19	.itext	x	import	-	-	SetCurrentDirectory

Fig.7

The API calls: - They are a set of functions and data structures that a Windows program can use to ask Windows to do something, like opening a file, displaying a message, etc.

Pretty much everything that a Windows program does involves calling various API functions.

Collectively, all the API functions that Windows makes available are called "The Windows API".

Below are the API calls the file is executing in windows 10: -

- AllowSetForegroundWindow
- GetEnvironmentVariable
- GetEnvironmentVariable
- VkKeyScan
- Input Capture, GetAsyncKeyState
- PathRenameExtension
- WriteFile
- File and Directory Discovery, FindNextFile
- GetCurrentThread

- ➢ Execution through API, WinExec
- ➢ GlobalAddAtom

- ➢ Clipboard Data, GetClipboardOwner
- ➢ Clipboard Data, GetClipboardData
- ➢ Clipboard Data, EnumClipboardFormats
- ➢ DdeQueryNextServer
- ➢ GetConsoleAliasExesLength
- ➢ SetCurrentDirectory
- ➢ CallWindowProc
- ➢ UpdateWindow
- ➢ GetCapture
- ➢ IsWindowEnabled
- ➢ Window Discovery, GetWindowTextLength
- ➢ DeleteCriticalSection
- ➢ SizeofResource
- ➢ GetLogicalDrives
- ➢ System Time Discovery, GetTickCount
- ➢ GetDriveType
- ➢ LocalUnlock
- ➢ HeapFree
- ➢ Process Injection, VirtualQueryEx
- ➢ LocalAlloc
- ➢ LocalFree
- ➢ CopyAcceleratorTable
- ➢ SwapMouseButton
- ➢ PathQuoteSpaces
- ➢ PathCombine
- ➢ GetCompressedFileSize
- ➢ CreateFileMapping
- ➢ GetPrivateProfileInt
- ➢ FreeLibrary
- ➢ GetModuleHandle

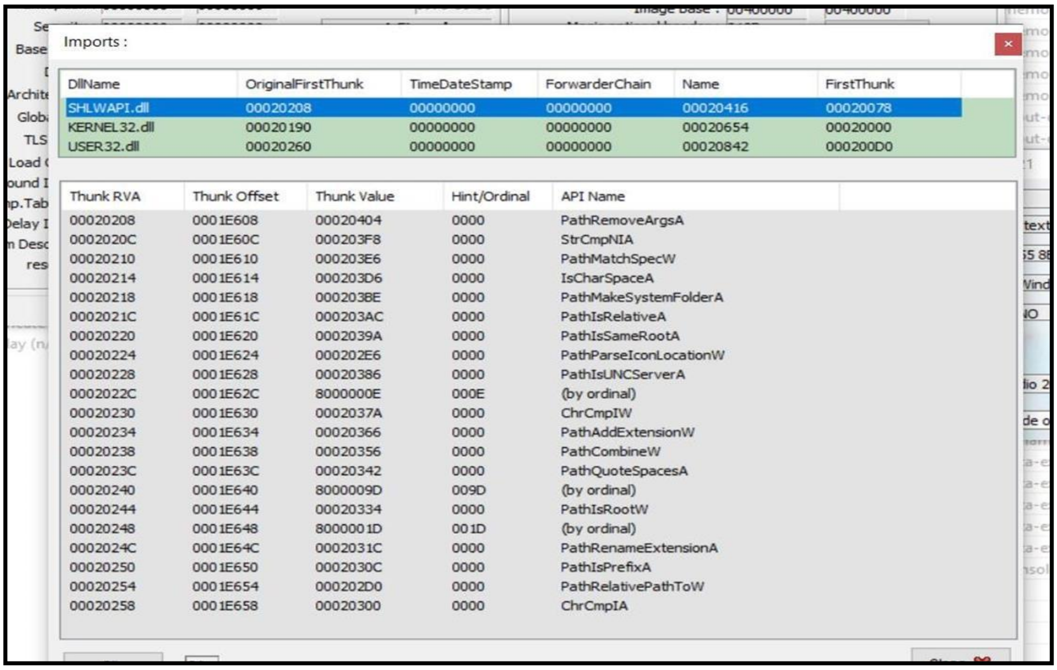
- Based on examination of libraries section three different .dll files are used by the malware, namely **SHLWAPL.dll**, **KERNEL32.dll** and **USER32.dll.**

library (3)	duplicate (0)	flag (0)	first-thunk-original (INT)	first-thunk (IAT)	type (1)	imports (77)	group	description
SHLWAPI.dll	-	-	0x00020208	0x00020078	implicit	21	-	Shell Light-weight Utility Library
KERNEL32.dll	-	-	0x00020190	0x00020000	implicit	29	-	Windows NT BASE API Client
USER32.dll	-	-	0x00020260	0x000200D0	implicit	27	-	Multi-User Windows USER API Client Library

Fig.8

API calls from each individual libraries –

- SHLWAPI.dll



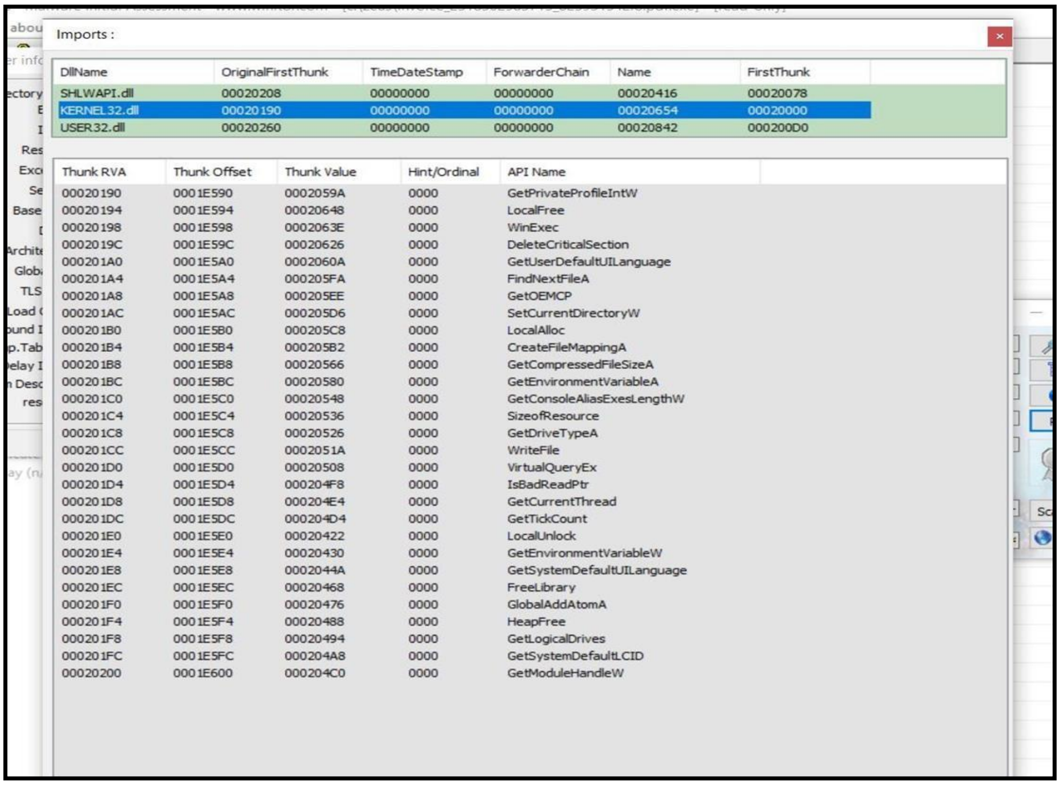
Imports :

DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
SHLWAPI.dll	00020208	00000000	00000000	00020416	00020078
KERNEL32.dll	00020190	00000000	00000000	00020654	00020000
USER32.dll	00020260	00000000	00000000	00020842	000200D0

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
00020208	0001E608	00020404	0000	PathRemoveArgsA
0002020C	0001E60C	000203F8	0000	StrCmpNIA
00020210	0001E610	000203E6	0000	PathMatchSpecW
00020214	0001E614	000203D6	0000	IsCharSpaceA
00020218	0001E618	000203BE	0000	PathMakeSystemFolderA
0002021C	0001E61C	000203AC	0000	PathIsRelativeA
00020220	0001E620	0002039A	0000	PathIsSameRootA
00020224	0001E624	000202E6	0000	PathParseIconLocationW
00020228	0001E628	00020386	0000	PathIsUNCServerA
0002022C	0001E62C	8000000E	000E	(by ordinal)
00020230	0001E630	0002037A	0000	ChrCmpIW
00020234	0001E634	00020366	0000	PathAddExtensionW
00020238	0001E638	00020356	0000	PathCombineW
0002023C	0001E63C	00020342	0000	PathQuoteSpacesA
00020240	0001E640	8000009D	009D	(by ordinal)
00020244	0001E644	00020334	0000	PathIsRootW
00020248	0001E648	8000001D	001D	(by ordinal)
0002024C	0001E64C	0002031C	0000	PathRenameExtensionA
00020250	0001E650	0002030C	0000	PathIsPrefixA
00020254	0001E654	000202D0	0000	PathRelativePathToW
00020258	0001E658	00020300	0000	ChrCmpIA

Fig.9

- KERNEL32.dll –



Imports :

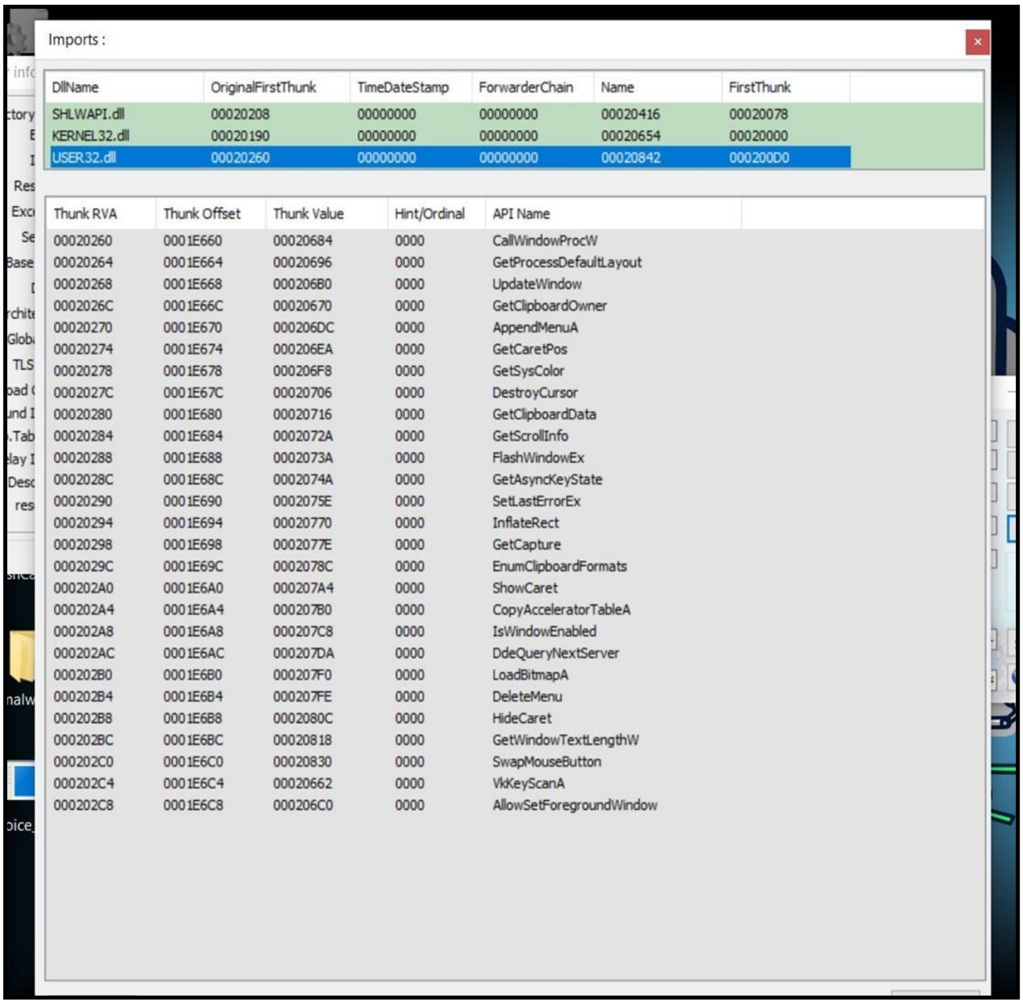
DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
SHLWAPI.dll	00020208	00000000	00000000	00020416	00020078
KERNEL32.dll	00020190	00000000	00000000	00020654	00020000
USER32.dll	00020260	00000000	00000000	00020842	000200D0

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
00020190	0001E590	0002059A	0000	GetPrivateProfileIntW
00020194	0001E594	00020648	0000	LocalFree
00020198	0001E598	0002063E	0000	WinExec
0002019C	0001E59C	00020626	0000	DeleteCriticalSection
000201A0	0001E5A0	0002060A	0000	GetUserDefaultUILanguage
000201A4	0001E5A4	000205FA	0000	FindNextFileA
000201A8	0001E5A8	000205EE	0000	GetOEMCP
000201AC	0001E5AC	000205D6	0000	SetCurrentDirectoryW
000201B0	0001E5B0	000205C8	0000	LocalAlloc
000201B4	0001E5B4	000205B2	0000	CreateFileMappingA
000201B8	0001E5B8	00020566	0000	GetCompressedFileSizeA
000201BC	0001E5BC	00020580	0000	GetEnvironmentVariableA
000201C0	0001E5C0	00020548	0000	GetConsoleAliasExesLengthW
000201C4	0001E5C4	00020536	0000	SizeofResource
000201C8	0001E5C8	00020526	0000	GetDriveTypeA
000201CC	0001E5CC	0002051A	0000	WriteFile
000201D0	0001E5D0	00020508	0000	VirtualQueryEx
000201D4	0001E5D4	000204F8	0000	IsBadReadPtr
000201D8	0001E5D8	000204E4	0000	GetCurrentThread
000201DC	0001E5DC	000204D4	0000	GetTickCount
000201E0	0001E5E0	00020422	0000	LocalUnlock
000201E4	0001E5E4	00020430	0000	GetEnvironmentVariableW
000201E8	0001E5E8	0002044A	0000	GetSystemDefaultUILanguage
000201EC	0001E5EC	00020468	0000	FreeLibrary
000201F0	0001E5F0	00020476	0000	GlobalAddAtomA
000201F4	0001E5F4	00020488	0000	HeapFree
000201F8	0001E5F8	00020494	0000	GetLogicalDrives
000201FC	0001E5FC	000204A8	0000	GetSystemDefaultLCID
00020200	0001E600	000204C0	0000	GetModuleHandleW

Fig.10

Here some important api calls are analysed such as GETickCount which will related to the on time of the host machine for VM detection.

- USER32.dll



Imports :

DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
SHLWAPI.dll	00020208	00000000	00000000	00020416	00020078
KERNEL32.dll	00020190	00000000	00000000	00020654	00020000
USER32.dll	00020260	00000000	00000000	00020842	000200D0

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
00020260	0001E660	00020684	0000	CallWindowProcW
00020264	0001E664	00020696	0000	GetProcessDefaultLayout
00020268	0001E668	000206B0	0000	UpdateWindow
0002026C	0001E66C	00020670	0000	GetClipboardOwner
00020270	0001E670	000206DC	0000	AppendMenuA
00020274	0001E674	000206EA	0000	GetCaretPos
00020278	0001E678	000206F8	0000	GetSysColor
0002027C	0001E67C	00020706	0000	DestroyCursor
00020280	0001E680	00020716	0000	GetClipboardData
00020284	0001E684	0002072A	0000	GetScrollInfo
00020288	0001E688	0002073A	0000	FlashWindowEx
0002028C	0001E68C	0002074A	0000	GetAsyncKeyState
00020290	0001E690	0002075E	0000	SetLastErrorEx
00020294	0001E694	00020770	0000	InflateRect
00020298	0001E698	0002077E	0000	GetCapture
0002029C	0001E69C	0002078C	0000	EnumClipboardFormats
000202A0	0001E6A0	000207A4	0000	ShowCaret
000202A4	0001E6A4	000207B0	0000	CopyAcceleratorTableA
000202A8	0001E6A8	000207C8	0000	IsWindowEnabled
000202AC	0001E6AC	000207DA	0000	DdeQueryNextServer
000202B0	0001E6B0	000207F0	0000	LoadBitmapA
000202B4	0001E6B4	000207FE	0000	DeleteMenu
000202B8	0001E6B8	0002080C	0000	HideCaret
000202BC	0001E6BC	00020818	0000	GetWindowTextLengthW
000202C0	0001E6C0	00020830	0000	SwapMouseButton
000202C4	0001E6C4	00020662	0000	VkKeyScanA
000202C8	0001E6C8	000206C0	0000	AllowSetForegroundWindow

Fig.11

This user32.dll calling some suspicious API such as GetCapture, GetClipboardData which give clipboard data and screen capture ability.

- Interesting pattern in strings and suspected functions associated with the Libraries.

AsksmaceaglyBubuPulsKaifTeasMistPeelGhisPrimChaoLyreroeno
KERNEL32.MulDiv
BagsSpicDollBikeAzonPoopHamsPyasmap
KERNEL32.SetCurrentDirectory
BardHolyawe
SHLWAPI.SHFreeShared
BathEftsDawnvilepughThroCymakohloverMitefuzerat
SHLWAPI.PathMakeSystemFolder
BemaCadsPodsWavyCedeRadsbrioOustPerefenom
USER32.SetDlgItemText
BullbonyaweeWaitsnugTierDriblibye
KERNEL32.VirtualQuery
CameValeWauler
USER32.IsIconic
CedeSalsshullLimyThroliraValeDonabox
USER32.CreateCaret
CellrotoCrudUntohighCols
KERNEL32.CreateFile
DenyLubeDunssawsOresvarut
SHLWAPI.PathRemoveFileSpec
DragRoutflusCrowPeatmownNewsyaksSerfmare
USER32.DestroyIcon
Dumpcotsavo
USER32.SetDlgItemInt
DungBadebankBangGelthoboCocaBozotsksWheyVaryShoghoseNipsCadisi
USER32.EndPaint
ExitRollWoodGumsgamaSloerevsWussletssinkYearZitiryesHypout
USER32.GetClassInfo
FociTalcileador
KERNEL32.ConvertDefaultLocale
GeneAilshe
KERNEL32.FindFirstFile
GhisGoodHowlCoonCigscateged
KERNEL32.GetWindowsDirectory
GimpWadsdashHoraYardSeatDeanScanscowRantKeasfib
KERNEL32.LCMapString
Haesourfe
USER32.GetKeyNameText

Fig.12

USER32.GetClassInfo
MarkMokeOsesShwaSkegpornlimemim
KERNEL32.GetStartupInfo
MeanOrrabirogirtWorkGawpSassPirnVinoLotaPledEidefe
SHLWAPI.SHLockShared
NextLoveOralwanySurfhm
KERNEL32.VerSetConditionMask
NisiBoyolineJiaoveryObiaowedblamHaetMaulweensky
SHLWAPI.PathCanonicalize
OastcabskamiKartDumbInksSomsMass
KERNEL32.SetCurrentDirectory
PeckQuinFillrillsaw
KERNEL32.GetThreadPriority
RamilimaputtHastJobs
KERNEL32.FindNextFile
RemsSlaySoreAnoaaxalbuffusesemeuMapsyogaHangLoud
SHLWAPI.PathMakePretty
RidsFineZingMickMomsdue
USER32.GetMonitorInfo
SeminerdsoloseenYaginobox
SHLWAPI.PathIsLFNFileSpec
SiretomsbritGrewIckyNapaLumsBoaren
KERNEL32.OpenFileMapping
SlabKitsSlayseptPfftjiffSabsdeskOafsNowtMemsKirnKepiMiffDunt
KERNEL32.OpenSemaphore
SoldKartAgueiliaRushWauldhal
SHLWAPI.PathIsUNC
SuitplieGunsMaidBaitFeusJiaotodycolyAlbsLuneToyspe
USER32.GetProp
SungActaKopsMaarposyparefuzedeck
SHLWAPI.PathIsDirectory
ToeaTailecusGeesSoliCadeSpueEndsPlaykaphall
SHLWAPI.PathRemoveArgs
Vavsrubepodsjadebrooli
USER32.GetUpdateRgn
VeerCrawFlateel
SHLWAPI.PathParseIconLocation
WainMeekPinyWonkpooflaudsir
KERNEL32.GetWindowsDirectory
WhopTestrangrapsdebsTzarNipaYins

Fig.13

The strings contain some suspicious program and known function calls are deliberately inserted between them. These function calls cumulatively can be harmful and further examination is needed as the extent of damage is unknown.

- By using pestudio tool we can extract sections and information about sections that which sections contains readable and executable. We can also analyze virtual size and raw size of each section which help to understand is the sample is packed or not.

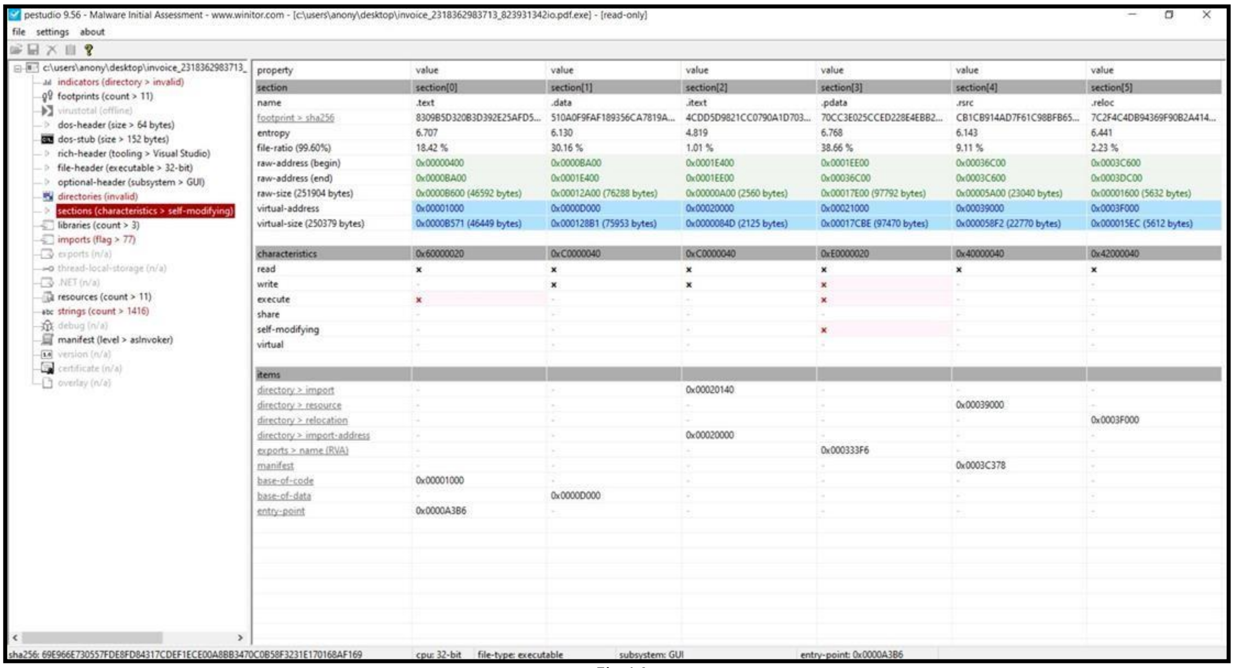


Fig.14

Here we can see first section .text having raw size and virtual size is comparativelysame that indicate sample is not packed with any packer. Sections actual contains data and PE headers having headers information.

PE (Portable Executable) file structure -

1. Dos header – Defines file as an executable binary also contains magic numbers.

2. Dos stub – Exist for backward compatibility. Its function is to print message.

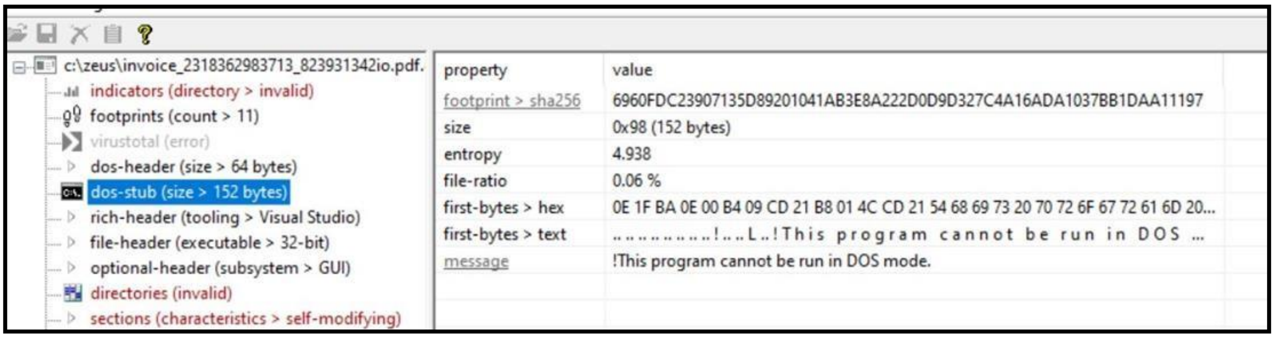


Fig.15

Here we can see the message that “This program cannot be run in DOS mode”

3.PE header – Defines the executable as PE, Holds signature to represent as a PE file, Contain machine type.

4.Optional Header – Size of the code, Address of entry point, Preferred base address.

5.Section table – Virtual size, Size of raw data

6.Sections – section contains .text, .bss, .rdata, .data, .rsrc, .edata, .idata, .debug sections.

In pestudio there is section called indicator gives suspicious indicators based onindicator level.

Fig.16

Here we can see it is giving all possible suspicious information about the sample with indicator level. It is showing possible MITRE ATT&CK framework techniques alsowhich is more useful for analysis.

3]Capa tool findings.

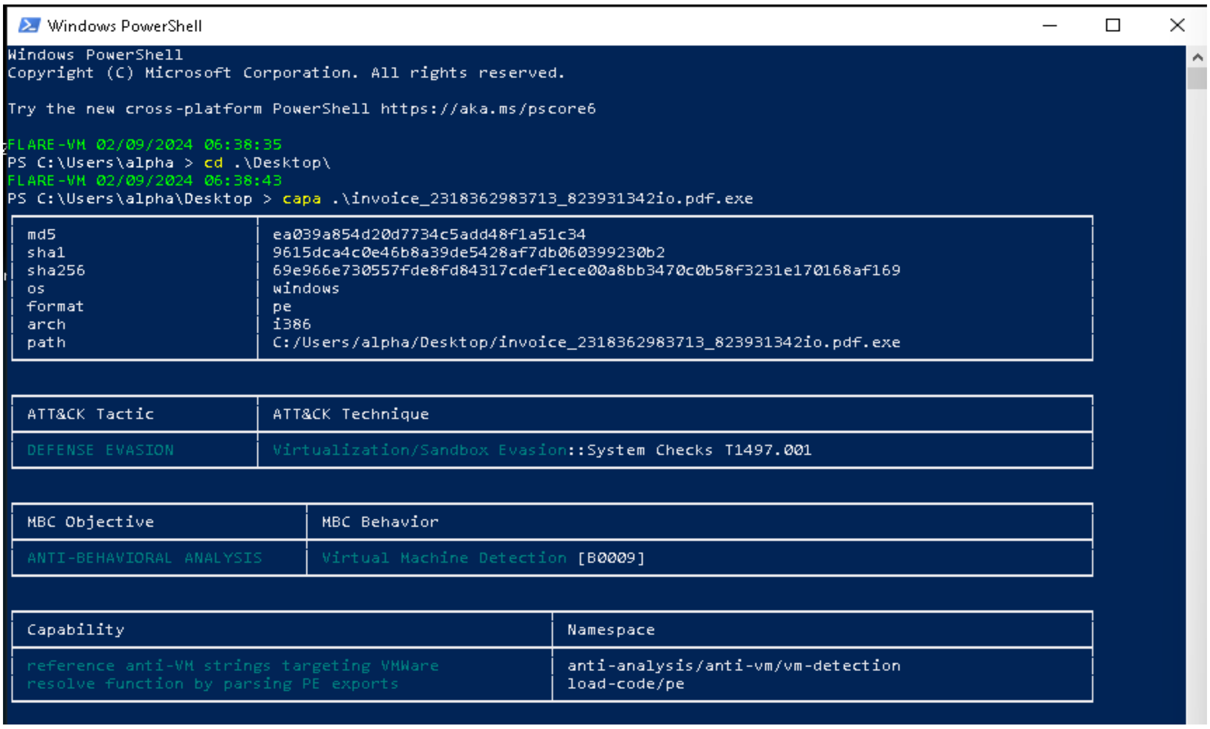


Fig.17

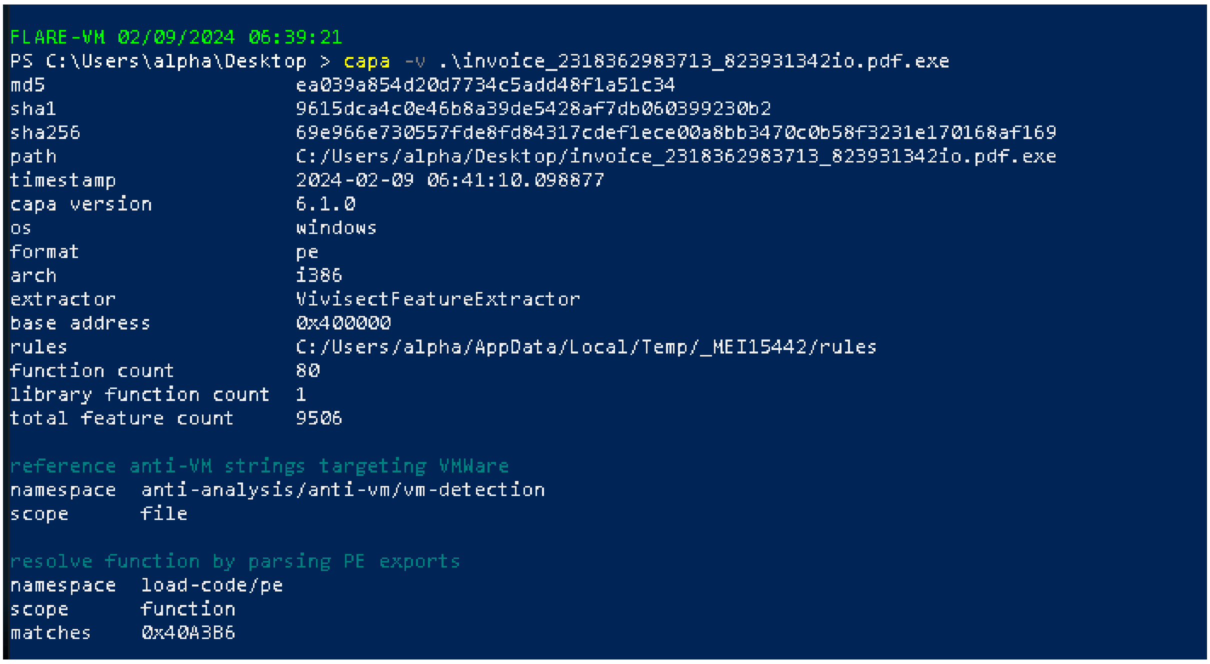


Fig.18

- Findings reveal that the file has **Virtual Machine Evasion Techniques** programmed into it. T.1497.001 is know to MITRE ATT&CK framework.
- Strings suggesting **anti-Vm, anti-VM-detection** and **anti-analysis** were found.

- **T.1497.001** is a sub-technique categorized under **Virtualization/Sandbox Evasion** in the **MITRE ATT&CK framework**.

T.1497.001 :

1) **Purpose**:

   - Adversaries employ various system checks to detect and avoid virtualization and analysis environments.
   - They change their behaviour based on the results of checks for artifacts indicative of a virtual machine environment (VME) or sandbox.
   - If a VME is detected, adversaries may alter their malware to disengage from the victim or conceal the core functions of the implant.
   - They may also search for VME artifacts before dropping secondary or additional payloads.

2) **Specific Checks**:

   - These checks vary based on the target and adversary but may involve behaviours such as:
     - **Windows Management Instrumentation (WMI)** queries.
     - **PowerShell** commands.
     - **System Information Discovery**.
     - **Registry queries** to obtain system information and search for VME artifacts.
   - Adversaries may search for VME artifacts in memory, processes, file systems, hardware, and/or the Registry.
   - Scripting is often used to automate these checks into one script, which exits if it determines the system to be a virtual environment.

Advance Static Analysis

1. Cutter Tool Analysis: -

As we go for advance static analysis, we need to go deep into assembly level from where we can analyze how functions are getting called and executable code as well as address of every function this can be analyze through the tool called cutter. As we went into deep by using this tool we can get overall basic information about sample like format, size, programming language used while constructing malware, hashes, OS etc.

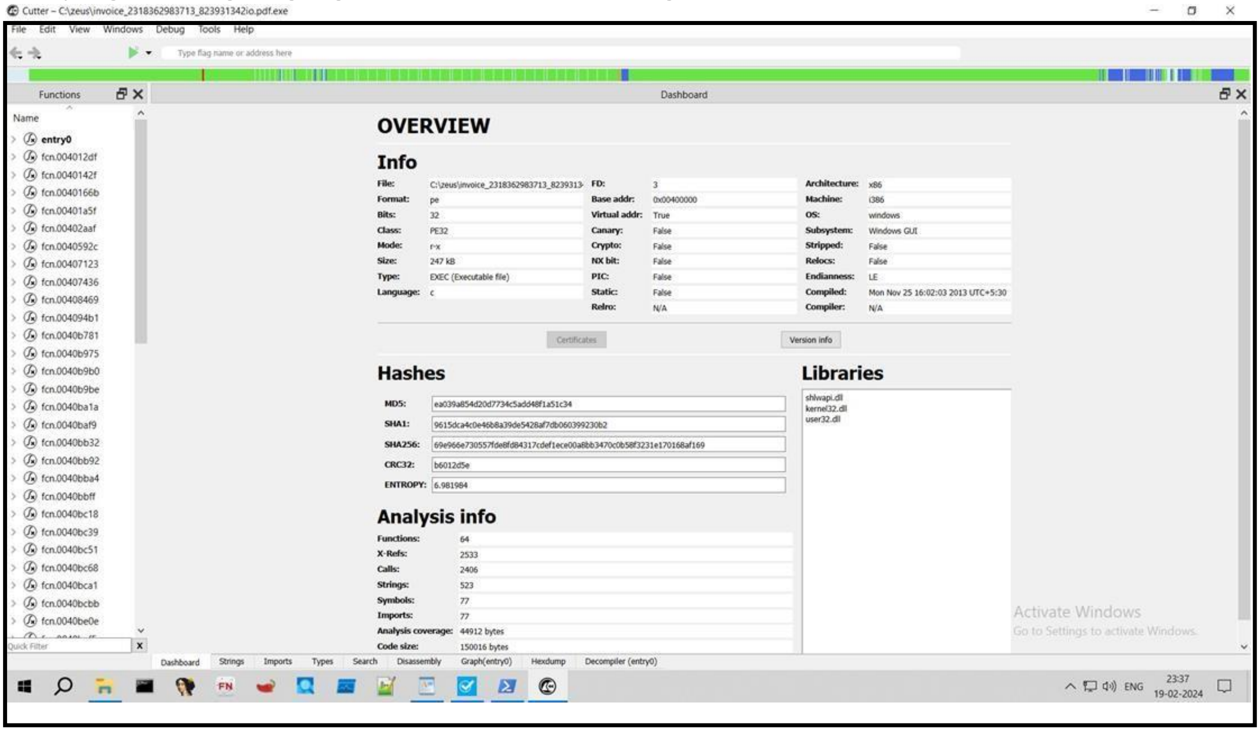


Fig.19

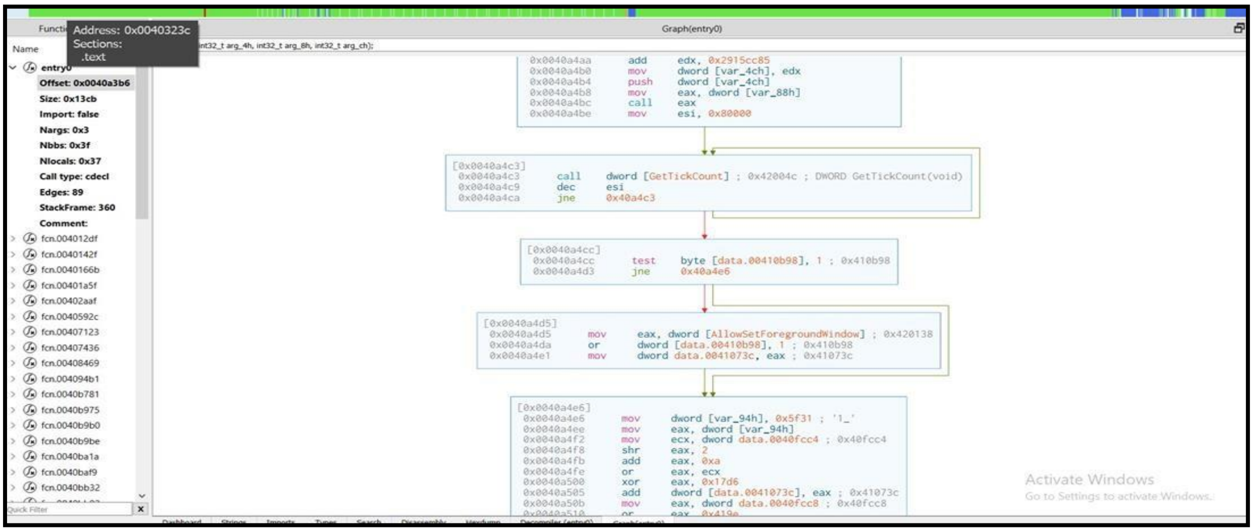


Fig.20

From this tool we can analyze starting point of the code execution process and firstfunction that is called by this executable

Here we can analyze that first api called from user32.dll library that is **GetTickCount** and calling function **AllowSetForegroundWindows**

```
eax = var_88h;
void (*0x6b29)(uint32_t, uint32_t, uint32_t, ui
esi = 0x80000;
do {
    GetTickCount ();
    esi--;
} while (esi != 0);
if ((*(data.00410b98) & 1) == 0) {
    eax = imp.AllowSetForegroundWindow;
    *(data.00410b98) |= 1;
    *(data.0041073c) = eax;
}
eax = 0x5f31;
ecx = *(data.0040fcc4);
eax >>= 2;
eax += 0xa;
eax |= ecx;
```

fig.21

at decompiler section we can see code and how it is calling different functions and analyze assembly level instructions.

As we analyzed functions called by this malware earlier we analyze that some of this functions are obfuscated with random strings and this functions are getting called in assembly code with some of the characters from strings. Here we found **CellrotoCrudUntohighCols** this random string is function while disassembling the sample.

```
0x00433972    je       0x4339e3
0x00433974    inc      ebx
0x00433975    jb       0x4339ec
0x00433977    push     ebp
0x00433979    outsb    dx, byte [esi]
0x0043397a    je       0x4339eb
0x0043397c    push     0x43686769 ; 'ighC'
0x00433981    outsd    dx, dword [esi]
0x00433982    insb     byte es:[edi], dx
0x00433983    jae      0x433985
0x00433985    dec      ebx
0x00433986    inc      ebp
0x00433987    push     edx
0x00433988    dec      esi
```

Fig.22

as we can see it is calling this function called as ‘ighC’ which are the characters from **CellrotoCrudUntohighCols.**

Basic Dynamic Analysis

In dynamic analysis we actually execute the sample and analyze for processes that malware sample will create. First we configured our **REMnux** as our c2 server and configured network adaptors to protect our host machine. We used **procmon** tool for dynamic analysis

1] process monitoring: -

Steps: -

1) Start INETSIM service in REMnux VM. (fig.13)
2) Make sure the both the VMs are connected privately as mentioned earlier.
3) Open **procmon** utility from **sysinternals** in Flare-vm.
4) Run the binary and record observations.

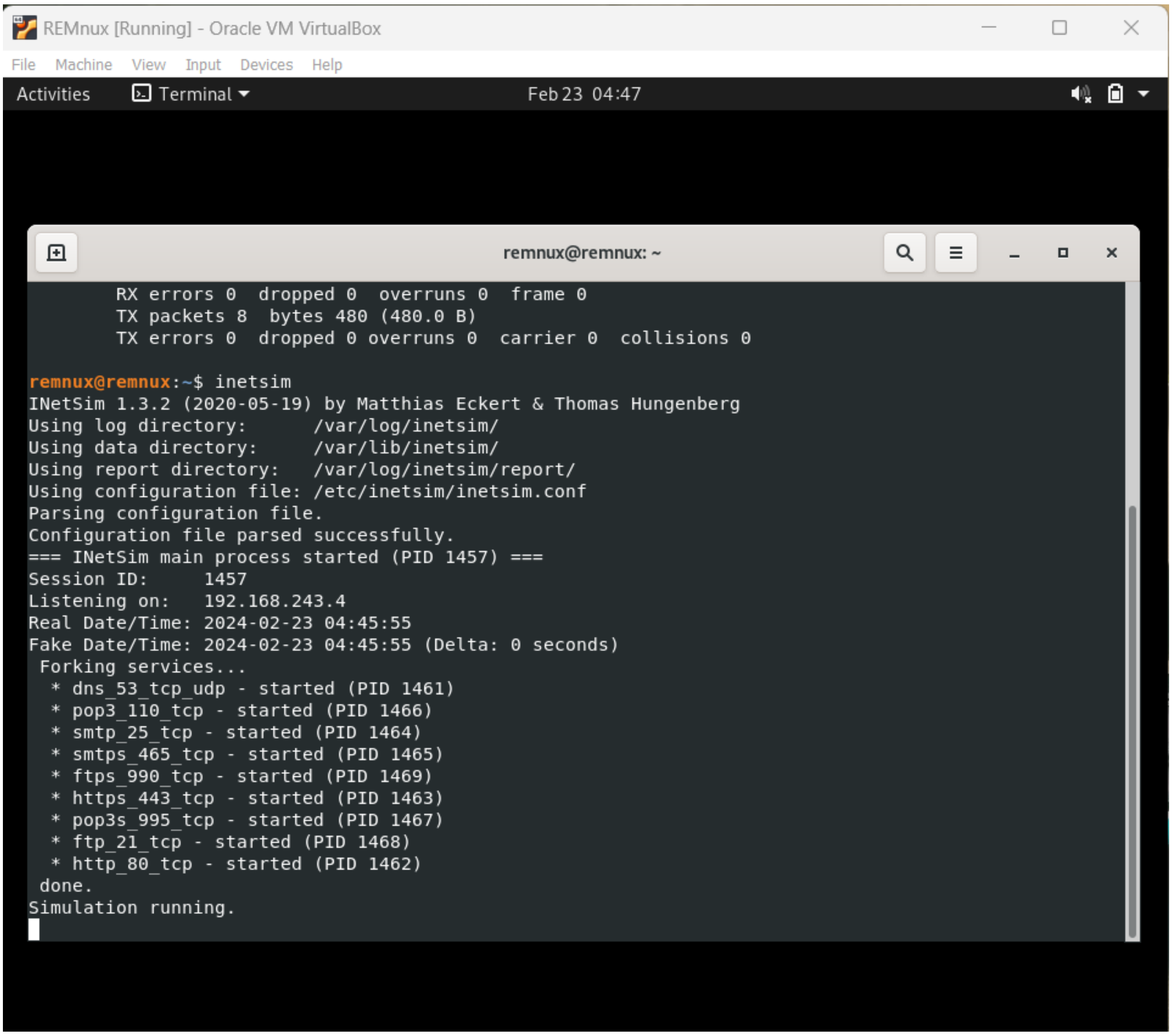


Fig.23

Findings:-

1) Initial Observation is that the binary deleted itself. This response suggests that the binary is most probably trying to establish persistence.

2) The binary attempts to install illegitimate copy of flashplayer.exe and once it is installed the binary deletes itself.

3) The Binary spawns a console host and starts an illegitimate session. The command that was run is mentioned in Fig.15. This is creating a suspended terminal hidden session and executing some kind of command.

4) The binary installs the installflashplayer.exe into the temp folder.

5) The file also installs msmg32.dll i.e a suspicious file.

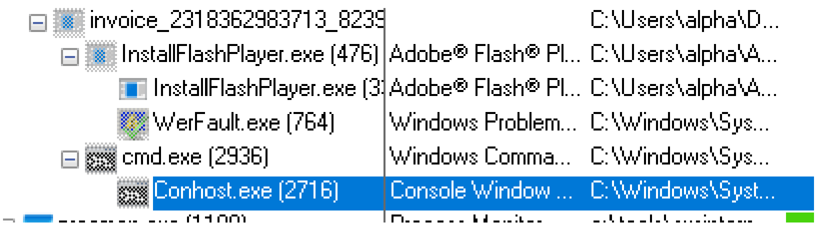


Fig.24

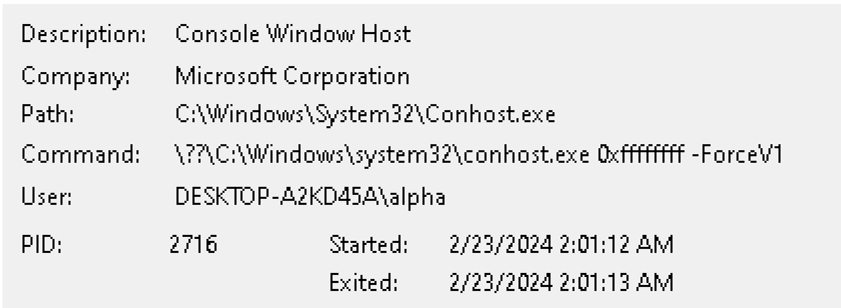


Fig.25

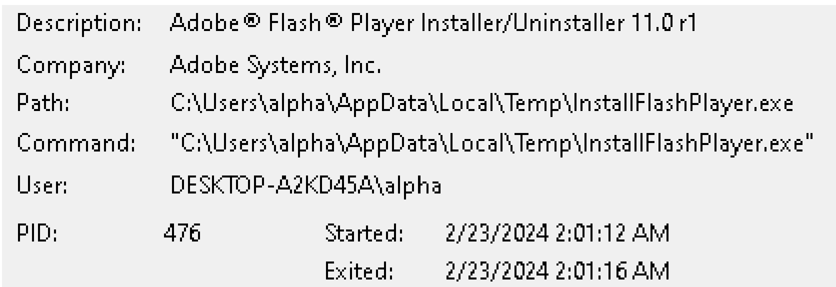


Fig.26

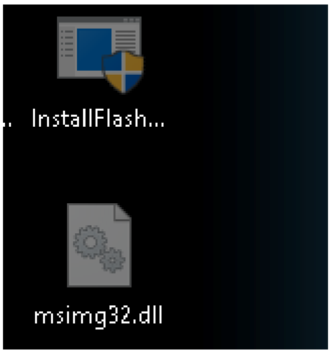


Fig.27

6)We ran a search for any registry changes and found out suspicious changes to browser update(fig.18). The process microsoftedgeupdate.exe is running under parent process called winint.exe(fig20). This is responsible for startup of the windows. It is safe to conclude that every time a msedgeupdate.exe runs the binary is installed again. This is how it is able to establish **Persistence.** (fig.18,19,20)

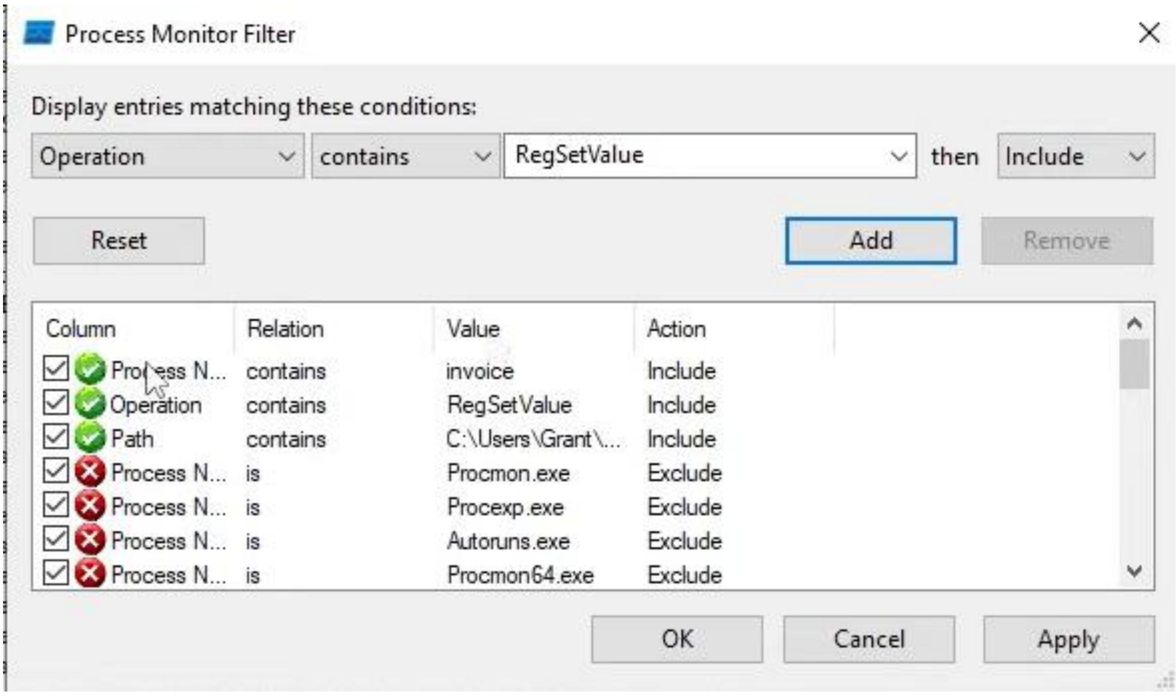


Fig.28

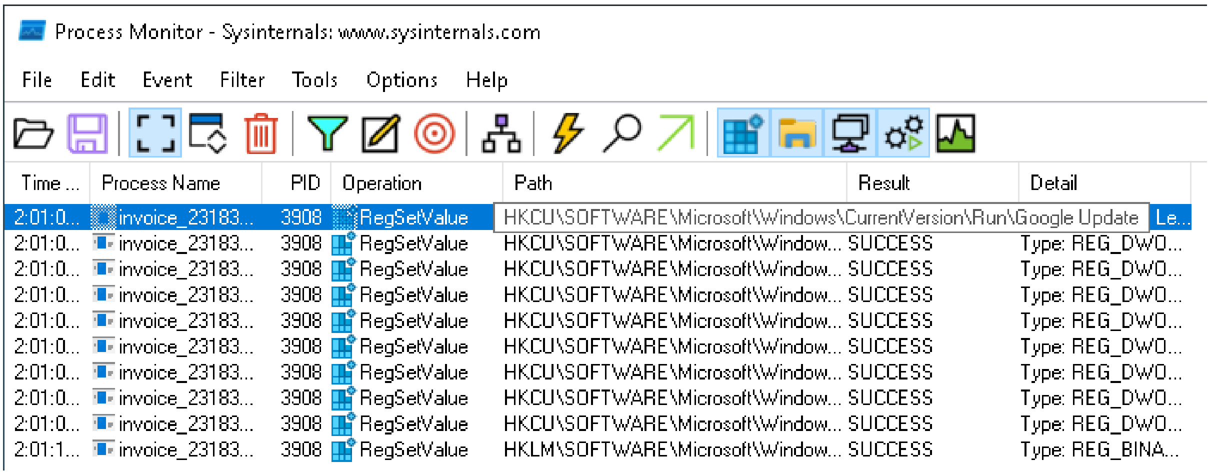


Fig.29

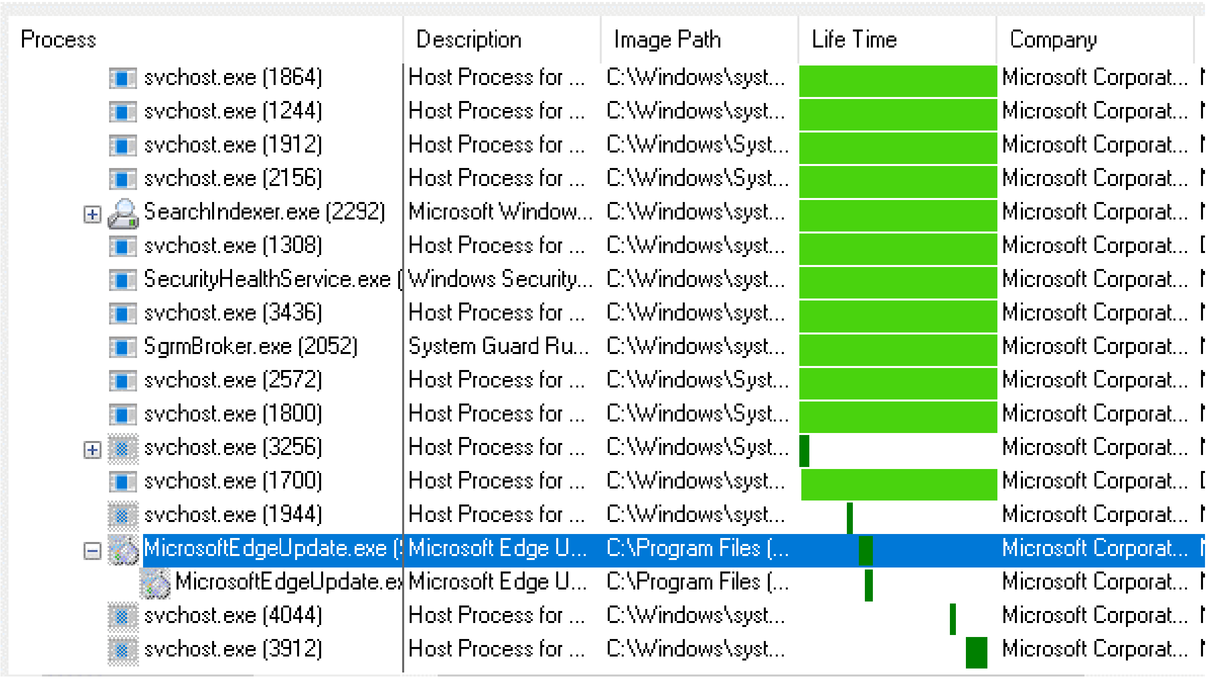


Fig.30

2]Monitoring Netwok Traffic: -

Capturing the traffic in Wireshark we found out the following findings.

1). A suspicious GET request was observed to download a binary from a suspicious domain.

2). Upon following the packet in TCP stream a domain called fpdownload.macromedia.com was revealed. (fig21,22,23)

3) Inspecting the domain in virus total only one security vendor flagged it as malicious. (fig.24)

```
0000  08 00 27 a8 81 d0 08 00  27 45 89 72 08 00 45 00   ··'·····  'E·r··E·
0010  00 dd ec e6 40 00 80 06  00 00 c0 a8 f3 05 c0 a8   ····@···  ········
0020  f3 04 c2 af 00 50 6c 9c  06 47 67 da 01 7b 50 18   ·····Pl·  ·Gg··{P·
0030  04 00 68 2b 00 00 47 45  54 20 2f 67 65 74 2f 66   ··h+··GE  T /get/f
0040  6c 61 73 68 70 6c 61 79  65 72 2f 75 70 64 61 74   lashplay  er/updat
0050  65 2f 63 75 72 72 65 6e  74 2f 69 6e 73 74 61 6c   e/curren  t/instal
0060  6c 2f 69 6e 73 74 61 6c  6c 5f 61 6c 6c 5f 77 69   l/instal  l_all_wi
0070  6e 5f 63 61 62 5f 36 34  5f 61 78 5f 73 67 6e 2e   n_cab_64  _ax_sgn.
0080  7a 20 48 54 54 50 2f 31  2e 31 0d 0a 55 73 65 72   z HTTP/1  .1··User
0090  2d 41 67 65 6e 74 3a 20  46 6c 61 73 68 20 50 6c   -Agent:   Flash Pl
00a0  61 79 65 72 20 53 65 65  64 2f 33 2e 30 0d 0a 48   ayer See  d/3.0··H
00b0  6f 73 74 3a 20 66 70 64  6f 77 6e 6c 6f 61 64 2e   ost: fpd  ownload.
00c0  6d 61 63 72 6f 6d 65 64  69 61 2e 63 6f 6d 0d 0a   macromed  ia.com··
00d0  43 61 63 68 65 2d 43 6f  6e 74 72 6f 6c 3a 20 6e   Cache-Co  ntrol: n
00e0  6f 2d 63 61 63 68 65 0d  0a 0d 0a                  o-cache·  ···
```

Fig.31

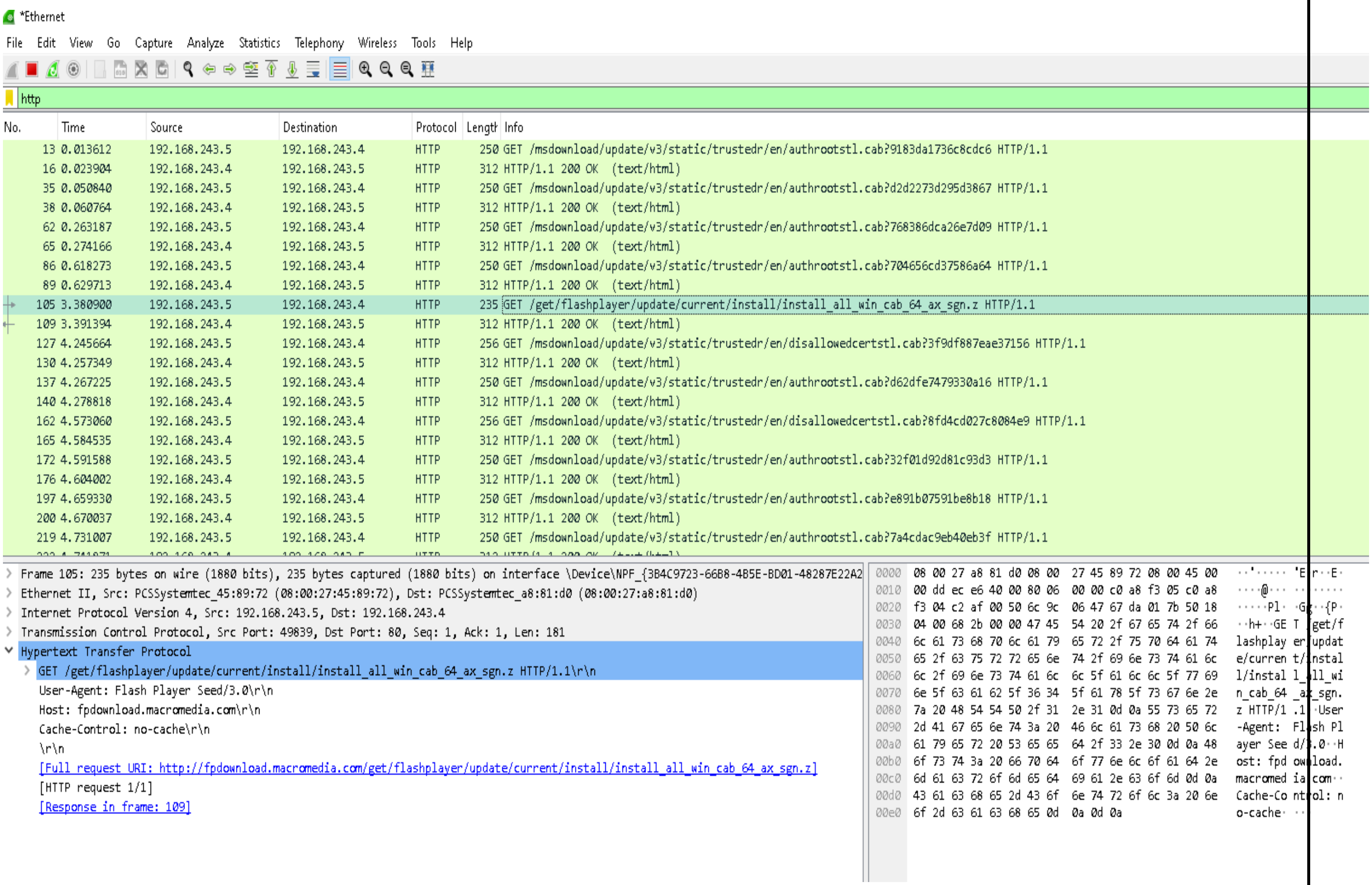


Fig.32

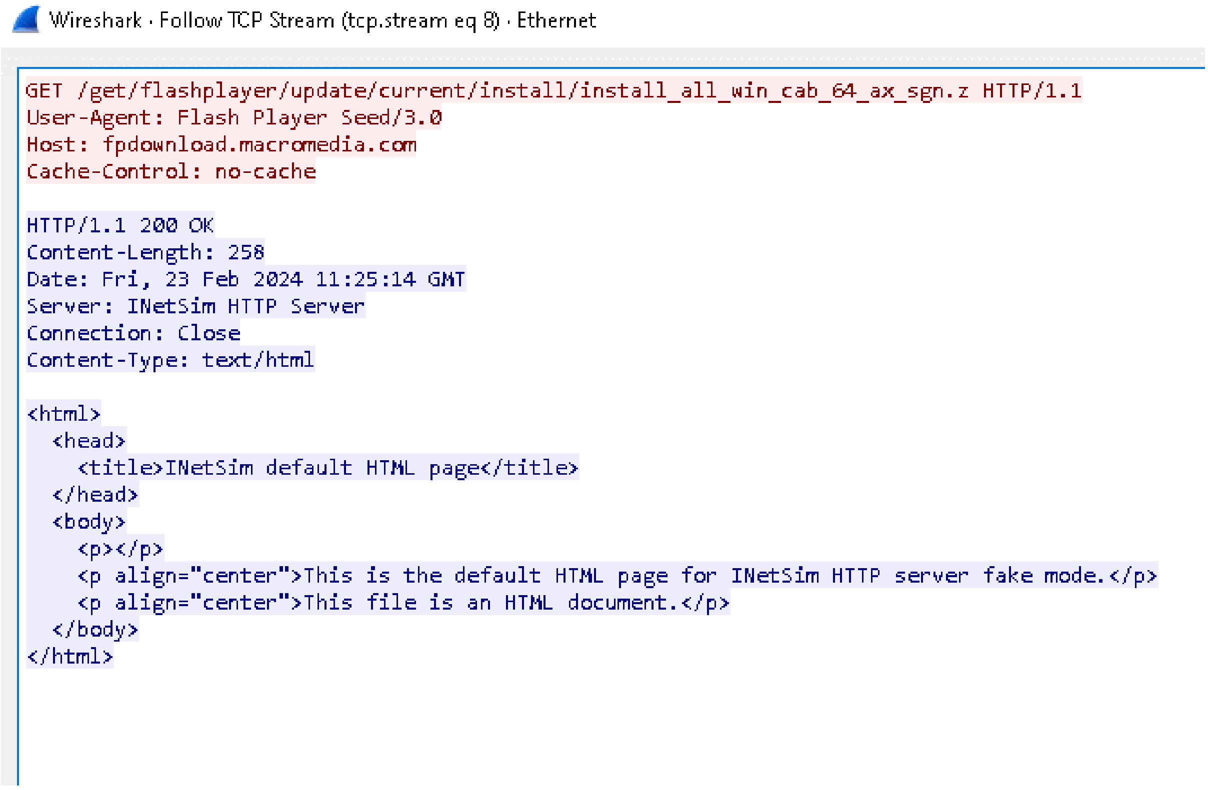

Wireshark · Follow TCP Stream (tcp.stream eq 8) · Ethernet
GET /get/flashplayer/update/current/install/install_all_win_cab_64_ax_sgn.z HTTP/1.1
User-Agent: Flash Player Seed/3.0
Host: fpdownload.macromedia.com
Cache-Control: no-cache
HTTP/1.1 200 OK
Content-Length: 258
Date: Fri, 23 Feb 2024 11:25:14 GMT
Server: INetSim HTTP Server
Connection: Close
Content-Type: text/html
<html>
<head>
<title>INetSim default HTML page</title>
</head>
<body>
<p></p>
<p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>
<p align="center">This file is an HTML document.</p>
</body>
</html>

Fig.33

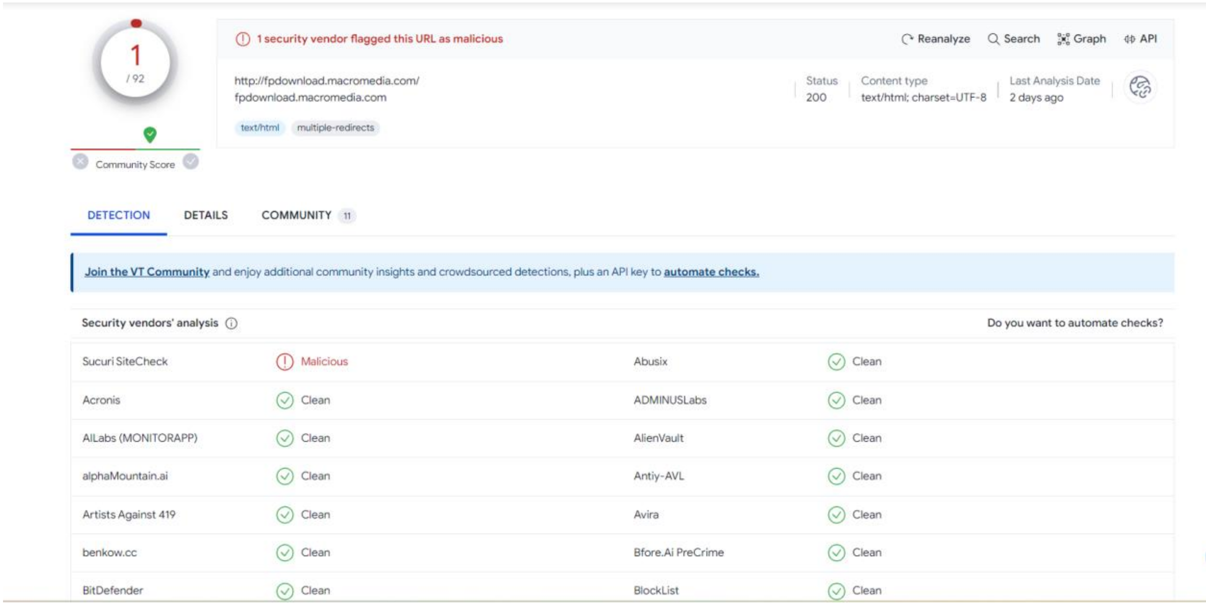

1
/ 92
1 security vendor flagged this URL as malicious
Reanalyze
Search
Graph
API
http://fpdownload.macromedia.com/
fpdownload.macromedia.com
Status
200
Content type
text/html; charset=UTF-8
Last Analysis Date
2 days ago
text/html
multiple-redirects
Community Score
DETECTION
DETAILS
COMMUNITY 11
Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Security vendors' analysis
Do you want to automate checks?
Sucuri SiteCheck Malicious
Abusix Clean
Acronis Clean
ADMINUSLabs Clean
AILabs (MONITORAPP) Clean
AlienVault Clean
alphaMountain.ai Clean
Antiy-AVL Clean
Artists Against 419 Clean
Avira Clean
benkow.cc Clean
Bfore.Ai PreCrime Clean
BitDefender Clean
BlockList Clean

Fig.34

YARA rules

YARA rules are used to identify sample based on specific strings or binary data. We created simple yara rule for this sample also.

rule zeus { meta:

description="Malware analysis of zeus banking trojan"

strings:

$File_name = "invoice_2318362983713_823931342io.pdf.exe" ascii

$function_name_KERNEL32_CreateFileA = "CellrotoCrudUntohighCols" ascii

$PE_magic_byte = "MZ"

$exports ="corect.com" ascii

condition:

$PE_magic_byte at 0 or $File_name or $function_name_KERNEL32_CreateFileA or

$exports

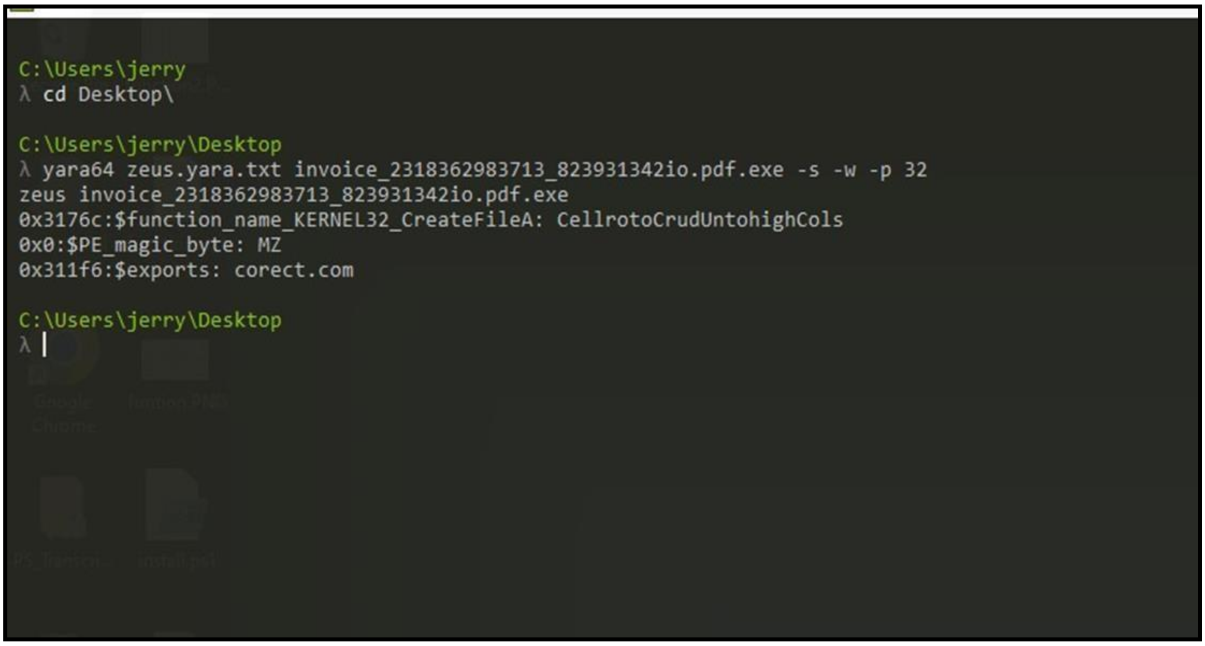


Fig.35

The Conditions were Established as per the malware and a Yara rule was deployed

Conclusion.

The ZEUS banking trojan is indeed a well-crafted malware and serves up to its evil functionalities. After a thorough analysis it is safe to conclude that the malware tricks users into believing that the binary is used to install flashplayer and the through established persistence carries out the malicious activities it was programmed for. The evasion techniques include deleting itself right after the binary is executed and install a malicious file in the temp folder to establish persistence.

Basic Static analysis revealed a lot of things. The binary also tries to contact corect.com but the analysis met a dead end there. The file appears to contact a suspicious domain fpvdownload.com to install flashplayer. The initials reveal a MZ signature that stands for Mark Zbikowski a developer who possibly created the malware. The binary is executing a set windows APIs to fulfill it's malicious intent. The strings contain some suspicious program and known function calls are deliberately inserted between them. The strings possess suspected function calls. These function calls cumulatively can be harmful and further examination is needed as the extent of damage is unknown. Findings reveal that the file has Virtual Machine Evasion Techniques programmed into it. T.1497.001 is known to MITRE ATT&CK framework. Strings suggesting anti-Vm, anti-VM-detection and anti-analysis were found. T.1497.001 is a sub-technique categorized under Virtualization/Sandbox Evasion in the MITRE ATT&CK framework

Basic Dynamic analysis suggests that the binary deleted itself. This response suggests that the binary is most probably trying to establish persistence. The binary attempts to install illegitimate copy of flashplayer.exe and once it is installed the binary deletes itself. The Binary spawns a console host and starts an illegitimate session. Another interesting behavior of the malware is that the execution installs some malicious files into the temp folder. The binary installs the installflashplayer.exe into the temp folder. The file also installs msmg32.dll i.e a suspicious file. Upon network traffic analysis an uncommon domain was found that was flagged malicious by only one security vendor. Further examinations are needed to appropriately conclude whether the domain is really malicious or not.

References.

1. Flare-VM Sandbox Guide: Creating an Isolated Lab Environment for Malware Analysis & Reverse Engineering | by Muhammad Haroon | Medium
2. Installing and Configuring InetSim (techanarchy.net)
3. What is the Zeus Trojan? How to Prevent and Remove it | Avast
4. Sysinternals Suite - Sysinternals | Microsoft Learn
5. Memory Analysis of Zeus with Volatility | by Neeraj | Medium
6. Zeus_Malware_Analysis_Case_Study/Zeus_Malware_Analysis_Case_Study.pdf at main · Dulanaka/Zeus_Malware_Analysis_Case_Study · GitHub