Deepak --project report-final.pdf

by deepak kumar

Submission date: 21-Dec-2023 12:59PM (UTC+0530)
Submission ID: 2263550959
File name: Deepak --project report-final.pdf (1.68M)
Word count: 4379
Character count: 26866

Hybrid Malware Analysis

A Project Report or Thesis submitted in partial fulfillment of the requirements for the award of degree of

Bachelor's degree in computer science with Specialization in Cyber Security

December – January (2023-24)

Supervised by: -
Dr. Vivek Dabra
Assistant Professor

Submitted by: -
Mr. Deepak (2820431)
7th Sem

PANIPAT INSTITUTE OF ENGINEERING AND TECHNOLOGY
SAMALKHA, PANIPAT-132103
(Approved by AICTE and Affiliated to the Kurukshetra University, Kurukshetra)

DECLARATION

I certify that

i The work presented in this project report is an authentic record of my own work under the guidance of my supervisor. It has not been submitted to any other Institute for the award of any other degree or diploma.

ii Whenever I have used information (text, data, figure, photograph, chart, analysis, inference, etc.) from other sources, I have given due credit by citing it in the text of the report and providing its details in the references.

iii I have followed the guidelines provided by the department for preparing thereport.

_____

Name: Mr. Deepak

Roll no: 2820431

Project title: Hybrid Malware Analysis
Semester: 7th

Date:

APPROVAL FROM SUPERVISOR

This is to certify that the project report entitled "*Hybrid Malware Analysis*" presented by "Mr. *Deepak 2820431"* under my supervision is an authentic work. To the best of my knowledge, the content of this report has not been submittedfor the award of any previous degree to anyone else.

It is recommended that the report be accepted as fulfilling this part of the requirements for the award of the degree.

_____

Name: Dr. Vivek Dabra
Designation: Assistant Professor
Department: Cyber Security
Date:

_____

(Counter Signed by)
Head Department of ….

CERTIFICATE

This is to certify that the work embodied in this report, entitled "*Hybrid Malware Analysis*" carried out by "Mr. *Deepak 2820431*" is approved for the degree of "*bachelor's degree*" at the department of "*Computer Science and Engineering*", Panipat Institute of Engineering and Technology.

_____

Internal Examiner

_____

External Examiner

Date:

Place: Panipat

ACKNOWLEDGEMENTS

Primarily, we would like to thank our Mentor, Prof. Pooja Dahiya, who was a constant source of inspiration for us. He encouraged us to think creatively and to complete our homework without hesitation. His vast knowledge, extensive experience, and professional competence in Cyber Security enabled us to complete this project successfully. This project would not have been possible without his assistance and supervision. We could not have had a better mentor in our studies. Without the contributions of each individual, this initiative would not have been a success. We were always there to encourage one another, and it was this that kept us together until the end.

I'd like to express my gratitude to Panipat Institute of Engineering & Technology for giving me the opportunity to work on the project (Packet Sniffing). Last but not least, I would like to thank my family, siblings, and friends for their invaluable assistance, and I would like to thank everyone who contributed to the successful completion of this project.

Supervisor: I am deeply thankful to our supervisor, Dr. Vivek Dabra , for their guidance, support, and invaluable insights throughout the duration of this project. Their expertise and encouragement have been crucial in shaping the trajectory of our research.

Name :Mr. Deepak
Roll No: 2820431
Date:

ABSTRACT

Malware analysis is an important field in cybersecurity because it assists security professionals in identifying and understanding the dangers they face, as well as developing tactics for mitigating those threats. Python's adaptability, simplicity of usage, and abundance of libraries and tools make it a popular programming language for malware investigation.

This work investigates the use of feature engineering and sandboxing approaches in Python-based dynamic malware analysis.

The suggested method entails doing dynamic analysis on malware samples using Python scripts, which enables a deeper comprehension of the behavior and intents of the infection. The paper covers several Python libraries and tools that can be used for malware research, including the dynamic binary instrumentation framework Frida-Python and Pyew, a Python-based command-line tool for malware analysis.

The research also emphasizes how crucial feature engineering and feature learning are to the analysis of malware. Through the extraction and analysis of malware samples' core code, analysts may obtain valuable insights into the functionality of the malware and identify new possibilities for attack.

Having this information, more reliable methods for detecting and decreasing malware risks can be created. In the final analysis, the study discusses some of the challenges and obstacles associated with utilizing Python for malware analysis and makes recommendations for future research directions. With its suggested methodology, Python is shown to be an effective instrument for dynamic malware analysis, and the cybersecurity community gains important knowledge.

TABLE OF CONTENTS

S.No	Title		Page No
1.	**Declaration**	----------	**ii**
2.	**Approval From Supervisor**	----------	**iii**
3.	**Certificate**	----------	**iv**
4.	**Acknowledgements**	----------	**v**
5.	**Abstract**	----------	**vi**
6.	**Introduction**	----------	**1**
i.	What is Malware Analysis	----------	**1**
ii.	Project Goals	----------	**1-2**
iii.	About Project	----------	**2**
iv.	Analysis Technique	----------	**3**
v.	Topology	----------	**4**
7.	**Methodology**	----------	**5**
8.	**Zeus (ZBOT) Banking Trojan**	----------	**6**
i.	Background Information	----------	**7**
9.	**Lab (Sandboxing)**	----------	**8**
i.	Basic Static Analysis	----------	**9**
ii.	Strings	----------	**10 - 13**
iii.	Virtual Size Vs Raw Data	----------	**14**
iv.	Defense Evasion	----------	**15 - 16**
v.	Basic Dynamic Analysis	----------	**17 - 18**
vi.	Network Based Indicators	----------	**19 - 21**
vii.	Detection Rules (YARA)	----------	**22 – 23**

10. Lab (With Scripting) ---------- **24**

i. Architecture Design ---------- **24**

ii. User Interface ---------- **25**

iii. Technical Coding and code Solution ---------- **26**

i. PE Header Based Malware Detection ---------- **26**

ii. Malicious URL Detection ---------- **26 - 27**

iv. Test And Validation ---------- **27 - 29**

v. Performance Analysis ---------- **29**

11. Conclusion ---------- **30**

12. Future Scope ---------- **31**

13. Resources ---------- **32**

LIST OF FIGURES

Fig No	Fig Title		Page No
1.	Flow Chart	----------	5
2.	Virus Total	----------	9
3.	Strings	----------	10 - 12
4.	XML Code	----------	13
5.	Virtual Size Vs Raw Data	----------	14
6.	File Header	----------	14
7.	Basic Capa Output	----------	14
8.	Defense Evasion	----------	15
9.	String Address Location	----------	15 - 16
10.	Host Based Indicator	----------	17 -18
11.	Network Based Indicator	----------	19
12.	Wireshark	----------	19
13.	Nslookup	----------	21
14.	Behavior Activities	----------	21
15.	Detection Using YARA	----------	22 - 23
16.	Architecture Design	----------	24
17.	User Interface	----------	25
18.	ROC curve for URL	----------	27
19.	Confusion Matrix For URL	----------	28
20.	Correlation Matrix for PE Header	----------	28
21.	Predicated Label for PE Header	----------	29

1. Introduction

Analyzing malware dynamically with a Python script and sandbox is a crucial step in the process. Using dynamic analysis, researchers may watch malicious code operate and behave in a controlled setting.

However, manual dynamic analysis can be laborious, error-prone, and time-consuming. Researchers can automate routine processes and workflows using scripting tools and APIs to streamline the process.

This study aims to investigate the use of popular tools and frameworks for automating dynamic analysis tasks, including network traffic analysis, virtual machine and sandbox setup and management, malware sample execution and monitoring, extraction and processing of indicators of compromise, and result generation and reporting.

We will concentrate on writing scripts that can run malware samples, communicate with the system, and carry out a variety of tasks using Python, a strong and flexible scripting language. We will also look at how to use other modules and libraries, such pefile, pydbg, and volatility, that might be useful for malware research.

Lastly, we'll talk about how crucial it is to record and examine the network traffic that the malware sample generates in order to determine the virus's communication pathways and the architecture that powers its command and control.

1.1 What is Malware Analysis?

- Malware analysis is the process of understanding the behavior and purpose of malicious files, applications, or suspicious executables.
- Effective analysis allows for uncovering hidden indicators of compromise (IOCs), triage of incidents, improving threat alerts and detection, and provide additional context into the latest exploits and defense evasion techniques.

1.2 Project Goals

Establishes a completely novice perspective when it comes to the field of malware analysis and investigation.

- **Overarching Objective**: To improve general understanding and expose users to malware infection methods andwell-liked technologies that security professionals utilize to facilitate malware research.

1. **Sub-Goals**: Discover the fundamentals of common indications of compromise (also known as IOCs) and malware infection methodologies.

2. Acquire knowledge of the fundamentals of static and dynamic malware analysis methods.

3. Analyse malware samples from the past, using instructions in writing if needed.

1.3 About Project

Step 1: - Practical Malware Analysis

Utilized as the primary training course for learning the fundamentals of malware, analysis methods, setting up a private malware lab, and instruments for malware research.

Step 2: -Creating Isolated Malware Lab Environment

Using self-hosted virtual machines from Vm-Ware and cloud-provisioned instances of EC2 from AWS, create isolated malware lab settings. It will be possible to securely investigate harmful executables and programs in an isolated lab setting.

Step 3: -Malware Analysis CTF

Play CTFs to gain experience with malware analysis. Play the PE Injection Study and MacOS DyLib Injection CTFs with Malware Unicorn. Plus extra!

Step 4: -Investigate Live Malware Sample

Examine real-world malware samples and produce reports detailing your findings. Install malicious software using vx-underground.org, then gather artifacts using malware analysis tools. Report your findings in writing. Use additional articles as needed.

1.4 Analysis Technique

In malware analysis combines the static analysis and the dynamic analysis methods to detect and analyze unknow threats and malware is called **Hybrid Malware Analysis.**
This method of analysis provides greater advantages over both static analysis and the dynamic analysis techniques, such as lightweight static malware analysis that can be performed on a user's device & dynamic analysis that can detect evasive malware.

Some Key Features of Hybrid Malware Analysis include:

File Analysis: Even for the most complex malware, it extracts every potential path of execution by combining runtime data with memory dump analysis.

Integration of Threat Intelligence: Hybrid Analysis provides actionable indicators of compromise (IOCs) and enriches results with threat intelligence, helping security teams better comprehend advanced malware attacks.

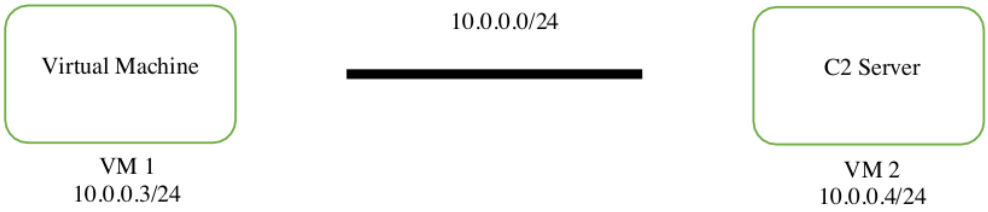
Community Support: The Hybrid Analysis engine instantly processes all data and integrates it into the analysis, and users can download samples and IOCs or browse through thousands of previously published malware reports.

Types of Malware Analysis: -

Static Malware Analysis: - In static malware analysis it does not require the code, program, or executable to run. Examines signs of malicious contents, including strings, file names, hashes, domains, IP addresses and file header data.

Dynamic Malware Analysis: - In Dynamic malware analysis it requires the code, program, or executable in question is run in real-time.
It also examines what happens when malware is executed and provides details in more effective ways and shows how the malware operates and work.

1.5 Topology



Note: - Host only Adapter (No Internet Access)

Will require to pause while installation and setup is taking place for:

- Download VirtualBox
- Download Windows 10 Enterprise ISO
- Setup FlareVM
- Download Remnux
- Setup EC2 Instance
- Setup AMI
- Setup Terraform

2. Methodology

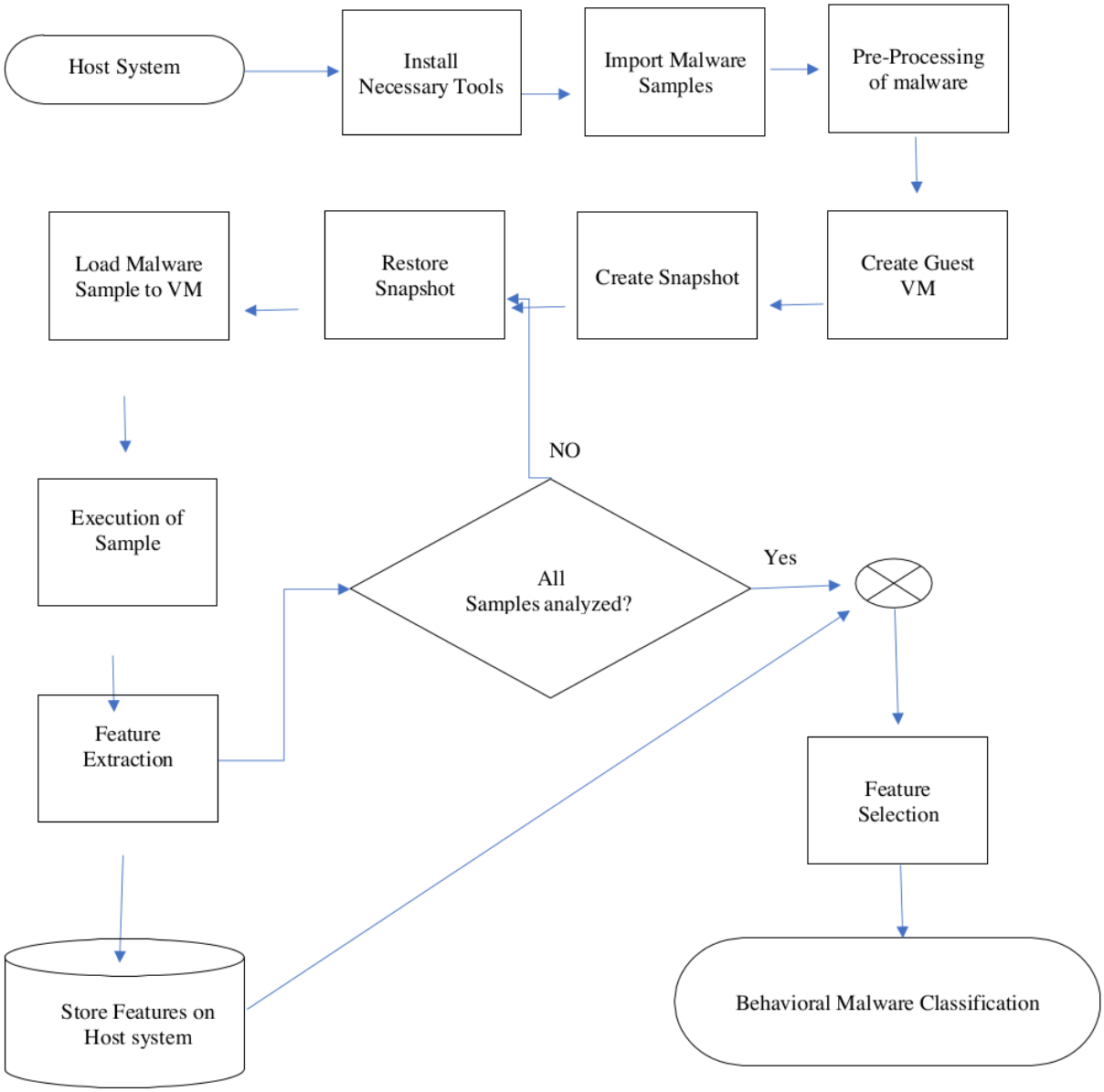


Fig – 1 Flow Chart

3. Zeus (Zbot) Banking Trojan - Malware Analysis

Report Overview

- Resources
- Background Information
- Malware Composition
- Basic Static Analysis
- Basic Dynamic Analysis
- Advanced Static Analysis
- Indicators of Compromise (IOC)
- Callback URLs
- Network Indicators
- Host-based Indicators
- Detection Rules (Yara)

3.1 Background Information

- Primarily created to be a financial banking trojan.
- Zeus Trojan was first discovered in the wild in 2007 after it was discovered stealing private data from US Department of Transportation computers..
  - According to the website https://zeusmuseum.com/, there have been 573+ known variations of the Zeus Trojan since then, divided into 36 distinct familie.
- 2011 saw the release of malicious code due to a leak. Evgeniy Mikhailovich Bogachev is the suspected author of the malware (source: https://krebsonsecurity.com/2015/02/fbi-3m-bounty-for-zeus-trojan-author/).
- Delivery Methods:
  - Drive-by downloads: Need the user to go to a website that contains the trojan backdoor.

    By default, contemporary web browsers prevent these downloads. Most attack vectors are out of date..
  - Spam and Phishing Campaigns: Primary Infection Method.
- Primary Goals:
  - To obtain financial information, steal people's financial data.
  - Add machines to a globally distributed P2P botnet (depends on the family).
- Crackdown History:
  - FBI cracked down on Gameover Zeus (which was a prolific variant of the Zeus Trojan) in 2014. ∘ An estimated 25% of computers were infected in the United States.
  - $100+ million in financial damages due to Gameover Zeus.
  - Evgeniy Mikhailovich Bogachev had a $3 million dollar bounty from the FBI. Continues to be one of the most wanted hackers.
- Impact:
  - Inspired hundreds of additional variants which use parts of source code.
  - Millions of infected machines with associated costs in damages in the millions.

4. Lab (Sandboxing)

Malware Composition

Filename: `invoice_2318362983713_823931342io.pdf.exe`

Hashes:

- md5: `EA039A854D20D7734C5ADD48F1A51C34`
- sha1: `9615DCA4C0E46B8A39DE5428AF7DB060399230B2`
- sha256: `69E966E730557FDE8FD84317CDEF1ECE00A8BB3470C0B58F3231E170168AF169`

4.1 Basic Static Analysis

65/71 security vendors flagged as malicious.

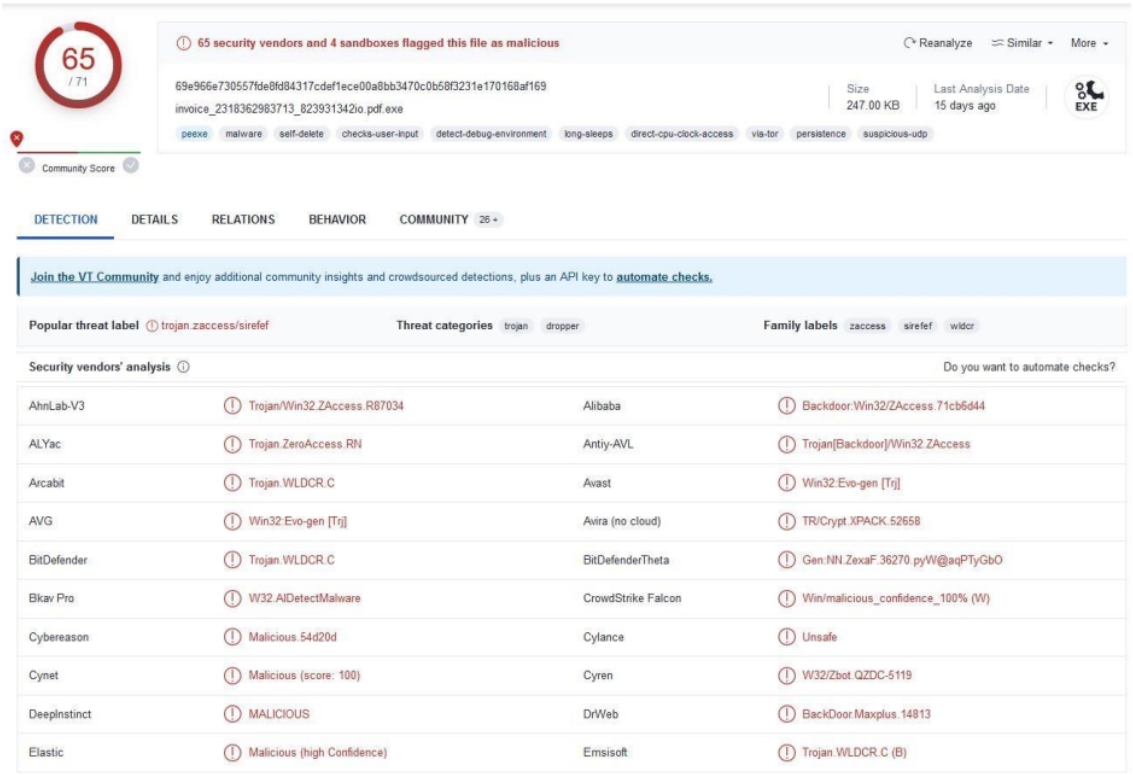


Fig – 2 ***Virus Total***

4.2 Strings

Tools Used: PeStudio, Floss.exe, & Capa

corect.com

```
AsksmaceaglyBubuPulsKaifTeasMistPeelGhisPrimChaoLyreroeno
KERNEL32.MulDiv
BagsSpicDollBikeAzonPoopHamsPyasmap
KERNEL32.SetCurrentDirectoryA
BardHolyawe
SHLWAPI.SHFreeShared
BathEftsDawnvilepughThroCymakohloverMitefuzerat
SHLWAPI.PathMakeSystemFolderW
BemaCadsPodsWavyCedeRadsbrioOustPerefenom
USER32.SetDlgItemTextW
BullbonyaweeWaitsnugTierDriblibye
KERNEL32.VirtualQuery
CameValeWauler
USER32.IsIconic
CedeSalsshulLimyThroliraValeDonabox
USER32.CreateCaret
```

```
CellrotoCrudUntohighCols
KERNEL32.CreateFileA
DenyLubeDunssawsOresvarut
SHLWAPI.PathRemoveFileSpecA
DragRoutflusCrowPeatmownNewsyaksSerfmare
USER32.DestroyIcon
Dumpcotsavo
USER32.SetDlgItemInt
DungBadebankBangGelthoboCocaBozotsksWheyVarysShoghoseNipsCadisi
USER32.EndPaint
ExitRollWoodGumsgamaSloerevsWussletssinkYearZitiryesHypout
USER32.GetClassInfoW
FociTalcileador
KERNEL32.ConvertDefaultLocale
GeneAilshe
KERNEL32.FindFirstFileA
GhisGoodHowlCoonCigscateged
KERNEL32.GetWindowsDirectoryA
GimpWadsdashHoraYardSeatDeanScanscowRantKeasfib
KERNEL32.LCMapStringW
Haesourfe
USER32.GetKeyNameTextA
HoggSoonLasstwaeNapeCeilBawlscopdub
KERNEL32.SystemTimeToFileTime
Icontellnoway
SHLWAPI.PathRemoveBlanksW
ImidslatJokyCombdrubChefBilkSale
USER32.GetShellWindow
IzararfsFlamWostAirsconsMouefemelallPoretweeSacsOxidMinx
SHLWAPI.PathAddExtensionA
JabsNaveFateLariManyLeeksecshiesBawlwoo
KERNEL32.CreateIoCompletionPort
KatsDoreOmerBetsKoraKeef
KERNEL32.GetShortPathNameA
KineChamLows
KERNEL32.SetCurrentDirectoryW
LeerMiff
KERNEL32.LeaveCriticalSection
MaarSectFiscNextMattbamsErasnimstoeaBadshon
USER32.GetClassInfoW
MarkMokeOsesShwaSkegpornlimemim
```

```
SHLWAPI.SHLockShared
NextLoveOralwanySurfhm
KERNEL32.VerSetConditionMask
NisiBoyolineJiaoveryObiaowedblamHaetMaulweensky
SHLWAPI.PathCanonicalizeW
OastcabskamiKartDumbInksSomsMass
KERNEL32.SetCurrentDirectoryA
PeckQuinFillrillsaw
KERNEL32.GetThreadPriority
RamilimaputtHastJobs
KERNEL32.FindNextFileW
RemsSlaySoreAnoaaxalbuffusesemeuMapsyogaHangLoud
SHLWAPI.PathMakePrettyA
RidsFineZingMickMomsdue
USER32.GetMonitorInfoW
SeminerdsoloseenYaginobox
SHLWAPI.PathIsLFNFileSpecA
SiretomsbritGrewIckyNapaLumsBoaren
KERNEL32.OpenFileMappingA
SlabKitsSlayseptPfftjiffSabsdeskOafsNowtMemsKirnKepiMiffDunt
KERNEL32.OpenSemaphoreW
SoldKartAgueiliaRushWauldhal
SHLWAPI.PathIsUNCW
SuitplieGunsMaidBaitFeusJiaotodycolyAlbsLuneToyspe
USER32.GetPropW
SungActaKopsMaarposyparefuzedeck
SHLWAPI.PathIsDirectoryA
ToeaTailecusGeesSoliCadeSpueEndsPlaykaphall
SHLWAPI.PathRemoveArgsW
Vavsrubepodsjadebrooli
USER32.GetUpdateRgn
VeerCrawFlateel
SHLWAPI.PathParseIconLocationA
WainMeekPinyWonkpooflaudsir
KERNEL32.GetWindowsDirectoryW
WhopTestrangrapsdebsTzarNipaYins
KERNEL32.DeleteFileA
YeukMags
KERNEL32.GlobalHandle
ZetaBeduPirnhipsjailTingSrisTeleAposhuskNameHoerflagemuwo
USER32.LoadIconA
```

Fig - 3

```xml
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level='asInvoker' uiAccess="false"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Fig - 4

Interesting API Calls

- Allow Set Foreground Window: Allow a specified process to set the foreground window even if the process does not currently own the foreground window focus.
- Get Capture
- Get Window Text Length
- Get Environment Variable
- Vk Key Scan: Translate a character to the corresponding virtual-key code.
- Get Async Key state: Allows you to determine whether a particular key is currently pressed or released.
- Path rename extension
- Write File
- Find Next File
- Get Current Thread
- Win Exec: Legacy Function used to launch an application or execute a command line. Available in earlier versions of Windows.
- Global Add atom: Adds a string to the atom table. Atom table are used for storing small pieces of string-based data.
- Get Clipboard Data
- Enum Clipboard Formats
- Dde Query Next Server
- Get console Alias Exes Length**:** Retrieves executables files.
- Set Current Directory

4.3 Virtual Size Vs Raw Data

raw-address	0x00000400	0x0000BA00	0x0001E400	0x0001EE00	0x00036C00	0x0003C600
raw-size (251904 bytes)	0x0000B600 (46592 bytes)	0x00012A00 (76288 bytes)	0x00000A00 (2560 bytes)	0x00017E00 (97792 bytes)	0x00005A00 (23040 bytes)	0x00001600 (5632 bytes)
virtual-address	0x00001000	0x0000D000	0x00020000	0x00021000	0x00039000	0x0003F000
virtual-size (250379 bytes)	0x0000B571 (46449 bytes)	0x000128B1 (75953 bytes)	0x0000084D (2125 bytes)	0x00017CBE (97470 bytes)	0x000058F2 (22770 bytes)	0x000015EC (5612 bytes)

Fig - 5

Likely not packed (compressed).

Libraries

- KERNEL32.dll
- SHLWAPI.dll
- USER32.dll

File Header

```
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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 D8 00 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01
4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 65
20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24
```

Fig - 6

Basic Capa Output

```
+-----------------------+-----------------------------------------------------------------------------+
| ATT&CK Tactic         | ATT&CK Technique                                                            |
|-----------------------+-----------------------------------------------------------------------------|
| DEFENSE EVASION       | Virtualization/Sandbox Evasion::System Checks T1497.001                     |
+-----------------------+-----------------------------------------------------------------------------+

+---------------------------+-----------------------------------------------------------------------+
| MBC Objective             | MBC Behavior                                                          |
|---------------------------+-----------------------------------------------------------------------|
| ANTI-BEHAVIORAL ANALYSIS  | Virtual Machine Detection [B0009]                                     |
+---------------------------+-----------------------------------------------------------------------+

+-----------------------------------------------------+-----------------------------------------------+
| CAPABILITY                                          | NAMESPACE                                     |
|-----------------------------------------------------+-----------------------------------------------|
| reference anti-VM strings targeting VMWare          | anti-analysis/anti-vm/vm-detection            |
| contain a resource (.rsrc) section                  | executable/pe/section/rsrc                    |
| resolve function by parsing PE exports              | load-code/pe                                  |
+-----------------------------------------------------+-----------------------------------------------+
```

Fig - 7

Appears this Zeus variant uses VM / Sandbox Evasion techniques to avoid inspection.

4.4 Defense Evasion

```
[0x0040a4c3]
 0x0040a4c3      call    dword [GetTickCount] ; 0x42004c ; DWORD GetTickCount(void)
 0x0040a4c9      dec     esi
 0x0040a4ca      jne     0x40a4c3

[0x0040a4cc]
 0x0040a4cc      test    byte [data.00410b98], 1 ; 0x410b98
 0x0040a4d3      jne     0x40a4e6

[0x0040a4d5]
 0x0040a4d5      mov     eax, dword [AllowSetForegroundWindow] ; 0x420138
 0x0040a4da      or      dword [data.00410b98], 1 ; 0x410b98
 0x0040a4e1      mov     dword data.0041073c, eax ; 0x41073c
```

Fig - 8

Tests to see how long the Windows machine has been running with the GetTickCount() function. (MITRE ATT&CK sub-technique 3 T1497.003).

String Address Location

Looking into a set of extracted strings (random gibberish followed by a DLL function):

```
CellrotoCrudUntohighCols
KERNEL32.CreateFileA
```

I suspect each of these random blobs of strings is the programs function name for the specific function called within the DLLs function.

```
Address     String
0x0043396a    je      0x43396c
0x0043396c    inc     ebx
0x0043396d    insb    byte es:[edi], dx
0x0043396f    insb    byte es:[edi], dx
0x00433970    jb      0x4339e1
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
```

The string `icgH` has an offset of 19 address space away from the `KERNEL32.MulDiv` function, meaning they are relatively close.

Address	String
0x00433985	KERNEL32.CreateFileA

```
0x0043397c    push    0x43686769 ; 'ighC'
0x00433981    outsd   dx, dword [esi]
0x00433982    insb    byte es:[edi], dx
0x00433983    jae     0x433985
0x00433985    dec     ebx
```

Fig - 9

4.5 Basic Dynamic Analysis

Host-based Indicators

Tools Used: Procmon, INetSim, & Wireshark

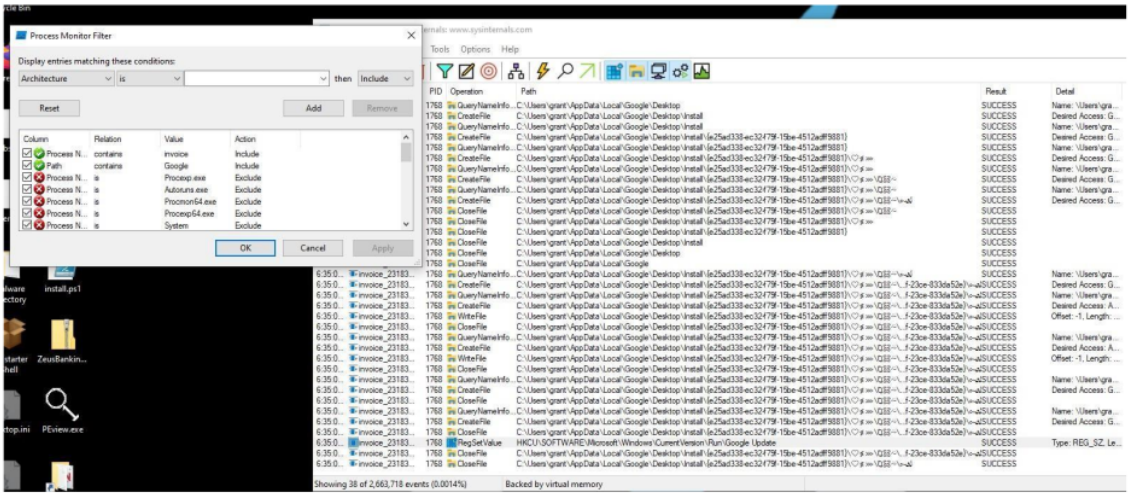
Explorer.EXE (4068)	Windows Explorer	C:\Windows\Expl...	Microsoft Corporat...	DESKTOP-6RRK...	C:\Windows\Expl...	6/22/2023 11:08:...	n/a
SecurityHealthSystray.exe (520	Windows Security...	C:\Windows\Syst...	Microsoft Corporat...	DESKTOP-6RRK...	"C:\Windows\Sys...	6/22/2023 11:10:...	n/a
VBoxTray.exe (5236)	VirtualBox Guest ...	C:\Windows\Syst...	Oracle and/or its ...	DESKTOP-6RRK...	"C:\Windows\Sys...	6/22/2023 11:10:...	n/a
7zFM.exe (5500)	7-Zip File Manager	C:\Program Files\...	Igor Pavlov	DESKTOP-6RRK...	"C:\Program Files...	7/18/2023 1:47:1...	n/a
pestudio.exe (7232)	Malware Initial Ass...	C:\Tools\pestudio...	www.winitor.com	DESKTOP-6RRK...	"C:\Tools\pestudi...	7/18/2023 4:18:3...	n/a
notepad.exe (6640)	Notepad	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	"C:\Windows\syst...	7/18/2023 4:48:0...	n/a
HxD.exe (3468)	HxD Hex Editor	C:\Program Files\...	Maël Hörz	DESKTOP-6RRK...	"C:\Program Files...	7/18/2023 4:59:1...	n/a
WindowsTerminal.exe (6004)		C:\Program Files\...		DESKTOP-6RRK...	"C:\Program Files...	7/18/2023 5:04:1...	n/a
OpenConsole.exe (5920)		C:\Program Files\...		DESKTOP-6RRK...	"C:\Program Files...	7/18/2023 5:04:2...	n/a
powershell.exe (6376)	Windows PowerS...	C:\Windows\Syst...	Microsoft Corporat...	DESKTOP-6RRK...	C:\Windows\Syst...	7/18/2023 5:04:2...	n/a
Procmon.exe (2232)	Process Monitor	C:\Tools\sysintern...	Sysinternals - ww...	DESKTOP-6RRK...	"C:\Tools\sysinter...	7/18/2023 6:33:5...	n/a
Procmon64.exe (7852)	Process Monitor	C:\Users\grant\A...	Sysinternals - ww...	DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:34:0...	n/a
invoice_2318362983713_8239		C:\Users\grant\D...		DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:35:0...	7/18
InstallFlashPlayer.exe (4440	Adobe® Flash® Pl...	C:\Users\grant\A...	Adobe Systems, I...	DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:35:3...	7/18
InstallFlashPlayer.exe (54	Adobe® Flash® Pl...	C:\Users\grant\A...	Adobe Systems, I...	DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:35:5...	7/18
WerFault.exe (8056)	Windows Problem...	C:\Windows\Sys...	Microsoft Corporat...	DESKTOP-6RRK...	C:\Windows\Sys...	7/18/2023 6:35:5...	7/18
cmd.exe (6860)	Windows Comma...	C:\Windows\Sys...	Microsoft Corporat...	DESKTOP-6RRK...	"C:\Windows\syst...	7/18/2023 6:35:3...	7/18
conhost.exe (4228)	Console Window ...	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	\??\C:\Windows\...	7/18/2023 6:35:3...	7/18
invoice_2318362983713_8239		C:\Users\grant\D...		DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:54:0...	7/18
InstallFlashPlayer.exe (8188	Adobe® Flash® Pl...	C:\Users\grant\A...	Adobe Systems, I...	DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:54:4...	7/18
InstallFlashPlayer.exe (6	Adobe® Flash® Pl...	C:\Users\grant\A...	Adobe Systems, I...	DESKTOP-6RRK...	"C:\Users\grant\...	7/18/2023 6:54:4...	n/a
WerFault.exe (7300)	Windows Problem...	C:\Windows\Sys...	Microsoft Corporat...	DESKTOP-6RRK...	C:\Windows\Sys...	7/18/2023 6:54:4...	7/18

Under invoice parent process has two child processes, including a suspended with a child

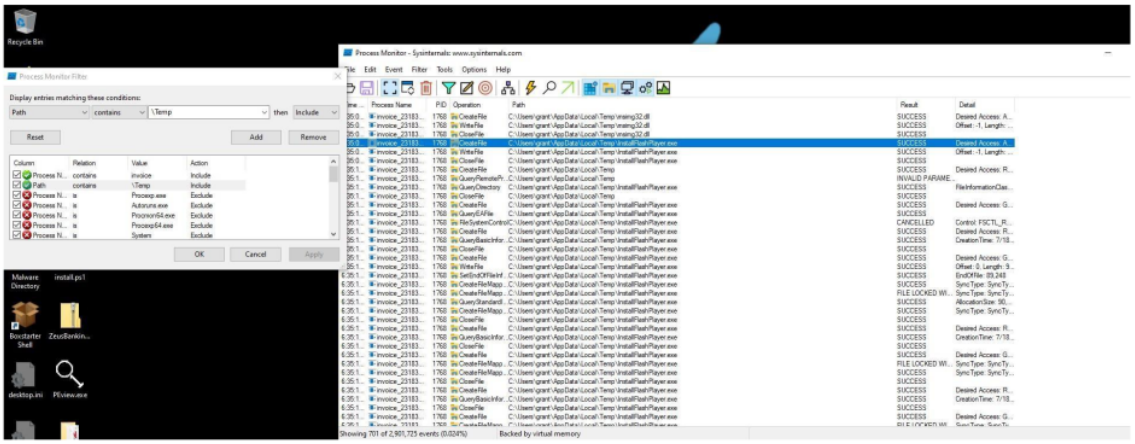
`conhost.exe` .

wininit.exe (504)	Windows Start-Up...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	wininit.exe	6/22/2023 12:15:...	n/a
services.exe (560)	Services and Cont...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\syst...	6/22/2023 12:15:...	n/a
svchost.exe (872)	Host Process for ...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\syst...	6/22/2023 10:15:...	n/a
svchost.exe (404)	Host Process for ...	C:\Windows\Syst...	Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\Syst...	6/22/2023 10:15:...	n/a
svchost.exe (1048)	Host Process for ...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\syst...	6/22/2023 10:15:...	n/a
sihost.exe (1812)	Shell Infrastructur...	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	sihost.exe	6/22/2023 11:08:...	n/a
taskhostw.exe (3884)	Host Process for ...	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	taskhostw.exe {22...	6/22/2023 11:08:...	n/a
GoogleUpdate.exe (516	Google Installer	C:\Program Files (...	Google LLC	NT AUTHORITY\...	"C:\Program Files ...	7/18/2023 1:27:2...	n/a
consent.exe (7848)	Consent UI for ad...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	consent.exe 1048...	7/18/2023 6:35:2...	7/18
taskhostw.exe (6008)	Host Process for ...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	taskhostw.exe	7/18/2023 6:35:5...	7/18
wermgr.exe (3076)	Windows Problem...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\syst...	7/18/2023 6:36:1...	7/18
GoogleUpdate.exe (808	Google Installer	C:\Program Files (...	Google LLC	NT AUTHORITY\...	"C:\Program Files ...	7/18/2023 6:47:1...	7/18
MicrosoftEdgeUpdate.ex	Microsoft Edge U...	C:\Program Files (...	Microsoft Corporat...	NT AUTHORITY\...	"C:\Program Files ...	7/18/2023 6:50:0...	7/18
consent.exe (3936)	Consent UI for ad...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	consent.exe 1048...	7/18/2023 6:54:2...	7/18
wermgr.exe (6124)	Windows Problem...	C:\Windows\syst...	Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\syst...	7/18/2023 6:54:5...	7/18
taskhostw.exe (7600)	Host Process for ...	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	taskhostw.exe Ev...	7/18/2023 7:04:1...	7/18
taskhostw.exe (5508)	Host Process for ...	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	taskhostw.exe Ins...	7/18/2023 7:04:1...	7/18
cleanmgr.exe (1000)	Disk Space Clean...	C:\Windows\syst...	Microsoft Corporat...	DESKTOP-6RRK...	C:\Windows\syst...	7/18/2023 7:04:1...	7/18

Under `wininit.exe` → `svchost.exe` → `GoogleUpdate.exe` is dropped.



Creates a new registry value in the Google Update location. It is likely this means each time Chrome updates, the malware will be executed.



Drops `msimg32.ddl` and `InstallFlashPlayer.exe` into

Fig - 10

`C:\Users\grant\AppData\Local\Temp\InstallFlashPlayer.exe`.

4.6 Network-based Indicators

```
> remnux@remnux:~$ INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 6241) ===
Session ID:     6241
Listening on:   10.0.0.4
Real Date/Time: 2023-07-20 14:23:12
Fake Date/Time: 2023-07-20 14:23:12 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 6245)
  * smtp_25_tcp - started (PID 6248)
  * http_80_tcp - started (PID 6246)
  * https_443_tcp - started (PID 6247)
  * pop3_110_tcp - started (PID 6250)
  * smtps_465_tcp - started (PID 6249)
  * ftp_21_tcp - started (PID 6252)
  * ftps_990_tcp - started (PID 6253)
  * pop3s_995_tcp - started (PID 6251)
 done.
Simulation running.
```

Fig - 11

Started `inetsim` to simulate DNS server and Wireshark for packet capture.

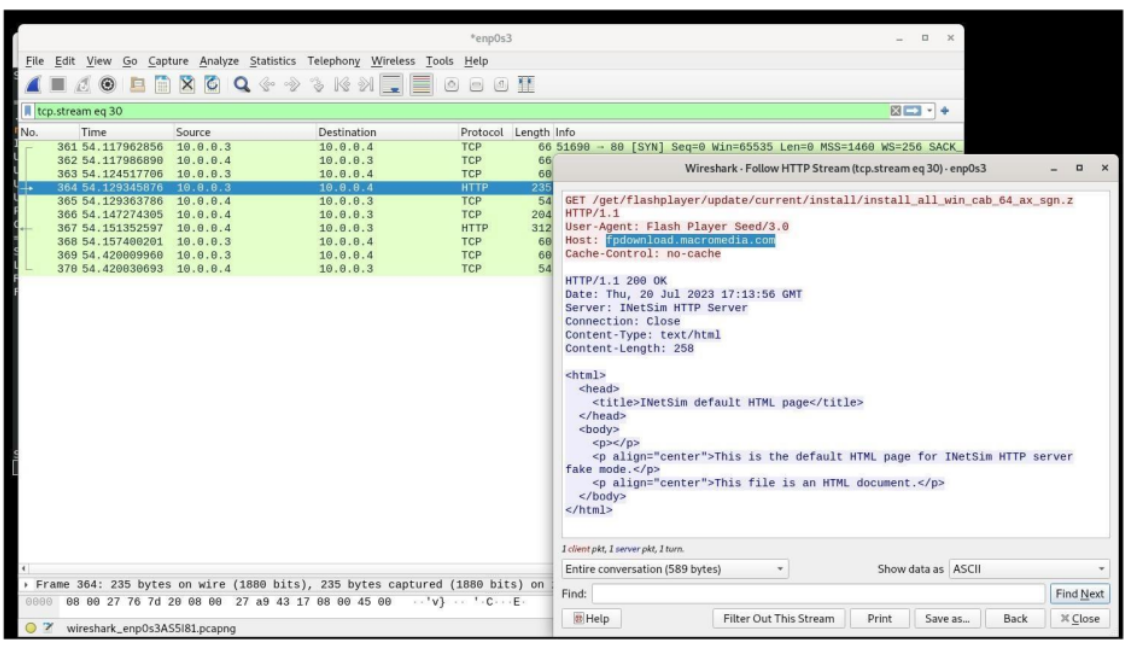


Fig - 12

Captured a suspicious HTTP request out to `fpdownload.macromedia.com` .

```
remnux@remnux:~/dnschef$ sudo ./dnschef.py -i 10.0.0.4
```

Downloaded [DNSChef]() to serve as a DNS proxy to identify if any other domains were being reached out to. Listened on the `ensp0` interface.

```
[*] 10.0.0.3: proxying the response of type 'A' for fpdownload.macromedia.com
fpdownload.macromedia.com request: [Errno 101] Network is unreachable
```

```
C:\Users\colli>nslookup fpdownload.macromedia.com
Server:  RAC2V1S
Address:  192.168.1.1

Non-authoritative answer:
Name:    e13914.dscd.akamaiedge.net
Addresses:  2600:1407:e800:586::365a
          2600:1407:e800:58b::365a
          23.192.235.212
Aliases:  fpdownload.macromedia.com
          fpdownload.macromedia.com.edgekey.net
```

Fig - 13

Issuing `nslookup` on `fpdownload.macromedia.com` shows an IP address of `23.192.235.212`,

which is not flagged as malicious on VirusTotal.

Fig - 14

Source: Any.Run

Searching for the domain `fpdownload.macromedia.com` displays an Any.Run analysis web page,

which displays a similar behavior as seen when executing the binary.

4.7 Detection Rules (YARA)

YARA rule used to detect the Zeus Banking Trojan Version 26-Nov-2013.

```
// import pe

rule Zeus {

    meta:
        author="Grant C."
        description="A detection rule against ZuesBankingVersion_26Nov2013"

    strings:
```

```
        $file_name="invoice_2318362983713_823931342io.pdf.exe"  ascii
        // Suspected name of functions and DLL functionalities.

$function_name_KERNEL32="AsksmaceaglyBubuPulsKaifTeasMistPeelGhisPrimChaoLyr
eroeno" ascii
        $function_name_KERNERL32_CreateFileA="CellrotoCrudUntohighCols"
ascii
        $function_name_KERNEL32_FINDFIRSTFILEA="GeneAilshe"  ascii

        // PE Magic Byte.

        $PE_magic_byte="MZ"// Hex String Function Name + DLL.
        $hex_string_SHLWAPI_PATHREMOVEFILESPECA= {44 65 6E 79 4C 75 62 65 44
     75 6E 73 73 61 77 73 4F 72 65 73 76 61 72 75 74 00 53 48 4C 57 41 50 49}

     condition:
     // Use the pe library to create fine-grained rules for PE files.
     // pe.ispie
     $PE_magic_byte at 0 and $filenameand
     $function_name_KERNEL32
     or $function_name_KERNERL32_CreateFileA or
     $function_name_KERNEL32_FINDFIRSTFILEA
     and $hex_string_SHLWAPI_PATHREMOVEFILESPECA}
```

```
C:\Users\grant\Desktop
λ yara64 zeus_rule.yara invoice_2318362983713_823931342io.pdf.exe -s -w -p 32
Zeus invoice_2318362983713_823931342io.pdf.exe
0x315a2:$function_name_KERNEL32: AsksmaceaglyBubuPulsKaifTeasMistPeelGhisPrimC
haoLyreroeno
0x3176c:$function_name_KERNERL32_CreateFileA: CellrotoCrudUntohighCols
0x318fa:$function_name_KERNEL32_FINDFIRSTFILEA: GeneAilshe
0x0:$PE_magic_byte: MZ
0x3179a:$hex_string_SHLWAPI_PATHREMOVEFILESPECA: 44 65 6E 79 4C 75 62 65 44 75
 6E 73 73 61 77 73 4F 72 65 73 76 61 72 75 74 00 53 48 4C 57 41 50 49
```

Fig - 15

Run the `yara64 zeus_rule.yara invoice_2318362983713_823931342io.pdf.exe -s -w -p 32` to detect this malware variant based on unique strings.

- `-s`: Print matched strings to stdout.
- `-w`: Ignore warnings.
- `p 32`: Allocate 32 threads.

5. Lab (With Scripting)

5.1 Architecture Design

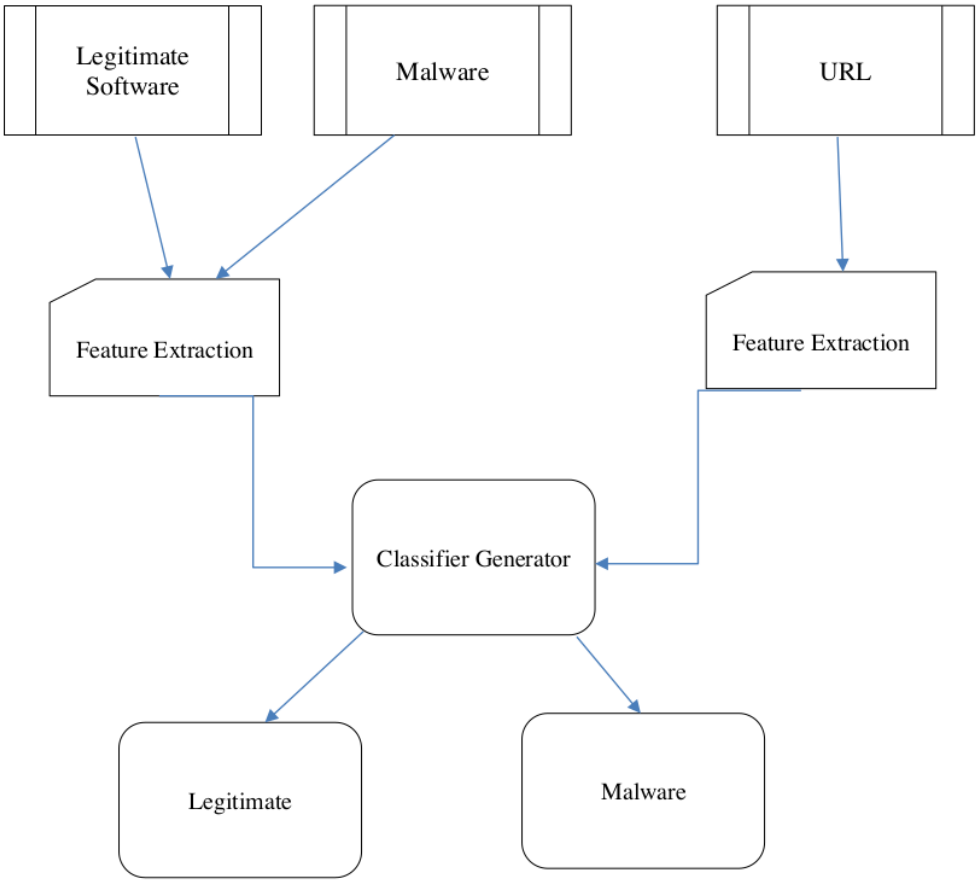


Fig - 16

5.2 User Interface

```
(base) kabir@kabir:~/Desktop/zzzzzzz/Malware-Detection-using-Machine-Learning$ python3 main.py

 Welcome to antimalware detector

 1. PE scanner
 2. URL scanner
 3. Exit

Enter your choice :
```

```
(base) kabir@kabir:~/Desktop/zzzzzzz/Malware-Detection-using-Machine-Learning$ python3 main.py

 Welcome to antimalware detector

 1. PE scanner
 2. URL scanner
 3. Exit

Enter your choice : 1
Enter the path and name of the file : LoJaxSmallAgent.exe
Features used for classification:  [332, 224, 270, 7, 10, 13824, 2560, 0, 13453, 4096, 20480, 4194304, 409
6]
The file LoJaxSmallAgent.exe is malicious
Do you want to search again? (y/n)
```

```
 Welcome to antimalware detector

 1. PE scanner
 2. URL scanner
 3. Exit

Enter your choice : 2
Input the URL that you want to check (eg. google.com) : google.com

The entered domain is:  good


If you feel that this prediction is wrong, or if you are not so sure about this output
you can contact us at kabirdhruw@protonmail.com we'll check the URL and update the machine
accodingly. Thank you.
Do you want to search again? (y/n)
```

Fig - 17

5.3 Technical Coding and Code Solution

1. PE-Header Based Malware Detection

This project's primary component is a machine learning model that classifies malware and benign files using a Random Forest classifier tree. We are utilizing a dataset that has 29.9% benign files and 70.1% malware.

In terms of the splitting process, we split the data into 70% training and 30% testing. Next, we used the extratrees.feature_importances_ function to identify the critical features needed for the classification. Finally, we compared the scores of the Random Forest and Decision Tree classifiers and determined that the Random Forest classifier outperformed the Decision Tree classifier (99.04%), so we chose it for training. After training, we saved the data as Classifier.pkl and additionally saved the features that we deemed crucial as features.pkl to maintain track of the necessary function when extracting it from any actual file.

After the Machine Learning part comes the file extraction part in which we extract the features that are required for the classification. The major challenge that we faced while coding this project was to extract the features of the PE Header file and then storing it for the saved machine learning model. For extracting the content of the PE Header we used pefile library (https://pypi.org/project/pefile/). The selection of those features is done using the feature.pkl model that stores all the important features for the Random Forest Classifier. The extraction of the PE Header files is done using pefile library in python and then the selected features are then given to the classifier.pkl machine and it predicts the output.

2. Malicious URL Detection

This portion of the program is divided into two stages: first, cleaning the data in preparation for the logistic regression, and second, teaching the computer to determine whether or not the data is malicious. The model's architecture is designed to enable it to interpret the training data sets in a meaningful way and attempt to derive a stated behavior from them. Since data sets are the foundation of any model, they must contain accurate and sufficient information on both good and bad URLs in order for the model to be trained on them. The initial step in using machine learning to detect dangerous websites will be data cleansing inside the data sets. We created our own vectorizer using pandas.

To extract the pertinent information from raw URLs, a sanitization technique is applied because the URLs differ from our typical text documents.

To filter the URLs, we will create our sanitization function in Python. As a result, we will get the appropriate URL data-set values for testing and training the model. Two column structures will be present in the data-set: one for URLs and another for labels (malicious or not).

Next, we employed the Tf-idf machine learning text feature extraction method from the Python module of Sklearn. Prior to using the term-frequency and inverse document frequency text extraction approaches, we must first read the data sets into data frames and matrices that the vectorizer we created can understand. For that assignment, we used the Python pandas module. As previously said, we trained our model using the logistic regression strategy. However, prior to training and testing the machine learning model, we used the Tf-idf methodology to send the data to our custom vectorizer function.

We employed the traditional approach of URL filtering in conjunction with the machine learning model, namely the Whitelist Filter, because the model was still unable to anticipate all the excellent URLs and thus combined good URLs with bad URLs. It is a list of websites that we are confident are trustworthy, safe, or won't hurt our users. Therefore, we permit these certain websites to be accessed through our internet traffic; the contrary is known as blacklisting. We also used this extremely basic yet effective method of segmenting our network traffic in our machine learning model.

5.4 Test And Validation

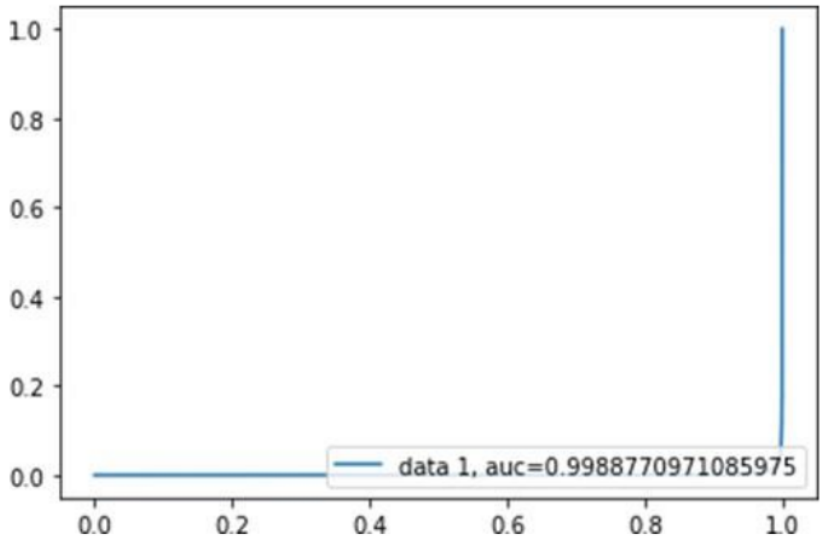


Fig – 18 ROC Curve For URL

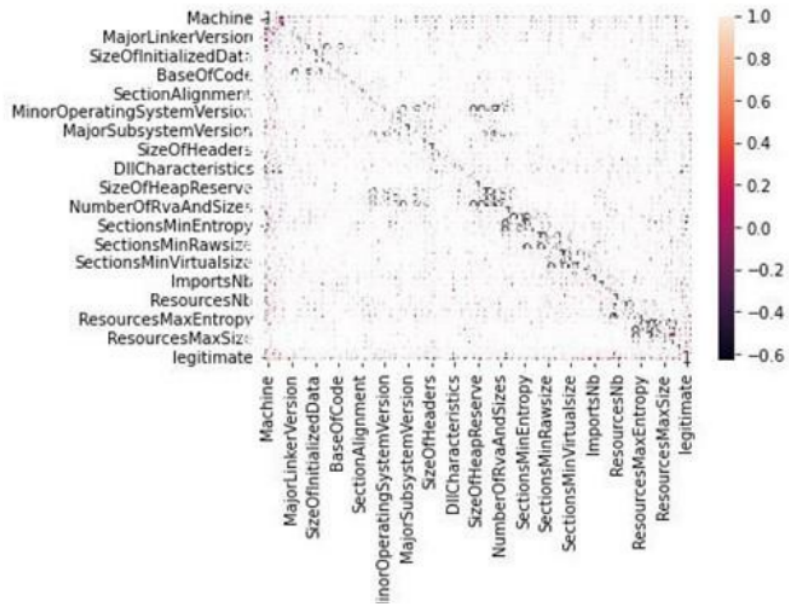


Fig – 19 Confusion Matrix for URL

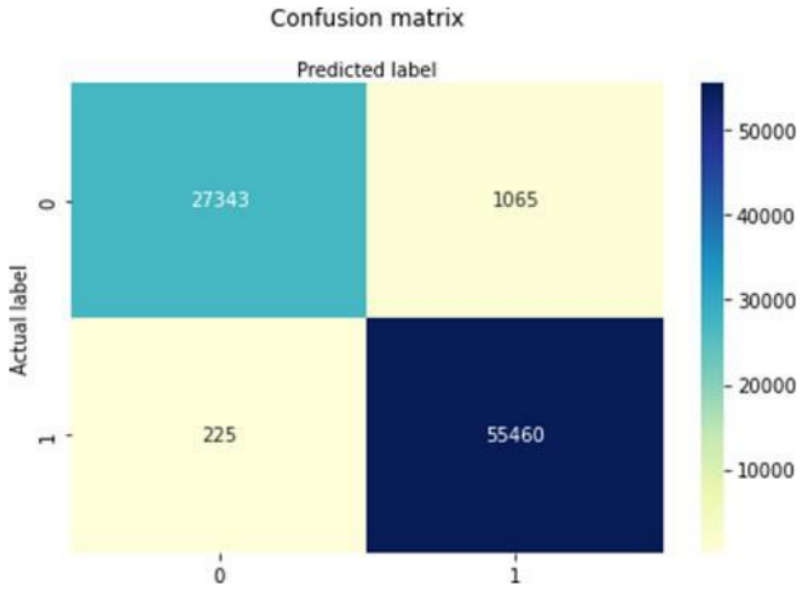


Fig – 20 Correlation Matrix For PE Header

Fig – 21 Confusion Matrix for PE Header

5.4 Performance Analysis

1. The Accuracy percentage of Random Forest Classifier: 99.37%
2. The Accuracy of Logistic Regression: 98.46%
3. The Precision of Logistic Regression: 99.18%
4. The Recall of Logistic Regression: 96.25%

5. Conclusion

To sum up, hybrid malware analysis is a thorough method that combines static and dynamic analysis methodologies for more accurate malware detection and classification. This approach has a number of benefits, including:

Expanding on the subject: By merging the advantages and disadvantages of static and dynamic analysis techniques, hybrid analysis overcomes the shortcomings of both and offers a more thorough comprehension of malware activity.

Detecting Unknown Threats: Hybrid analysis analyzes both the static and dynamic components of the virus in order to discover unknown risks, including those originating from complex malware.

Enhancing context: Contextualization is improved when using hybrid analysis to find malicious infrastructure, libraries, and technical indications when conducting threat hunting.

Adaptability: Hybrid analysis may be customized to the unique dangers that a company faces and is able to adapt to various malware variants.

It is imperative to take into account the efficacy and precision of hybrid analysis vis-à-vis the requirements and assets of the firm. Studies indicate that distinct approaches and combinations of approaches are more effective in identifying particular malware kinds, therefore it's critical to keep up with the latest analysis techniques. All things considered, hybrid malware analysis provides a strong and effective method for identifying and classifying malware, making it an invaluable tool for businesses to safeguard their infrastructure and digital assets.

6. Future Scope

Future developments in increasingly sophisticated and effective methods to counteract emerging threats will determine the extent of malware analysis. This comprises the subsequent:

Improved Hybrid Analysis: Static, dynamic, and hybrid analysis methods are further refined and integrated to increase detection capabilities, particularly against complex and evasive malware variants.

Adaptation to New Malware Types: Constant research and development is needed to handle newly developing malware classifications, like ransomware and cryptominers, while maintaining the efficacy of analytic methods against these dynamic threats.

Memory Forensics and Side-Channel Analysis: To improve the breadth and precision of malware analysis, methods such volatile memory forensics and side-channel analysis should be further explored and used.

Adapted Analysis Practices: Adapting analysis techniques to the unique risks that businesses confront, taking into consideration the gravity and characteristics of those threats.

Continuous Improvement and Optimization: The continuous search for ways to make malware analysis methods better and more efficient so they can continue to be useful in the face of a dynamic threat environment.

In result, new methods for addressing growing threats and maintaining the efficacy of malware detection and classification will need to be developed in addition to the continual evolution and improvement of current techniques.

7. Resources

- https://samples.vx-underground.org/root/Papers/Malware Defense/Malware Analysis/2023-03-14 -/
- The Zeus Trojan Malware - Definition and Prevention.pdf/
- https://www.malwarebytes.com/blog/news/2021/07/the-life-and-death-of-the-zeus-trojan/
- https://krebsonsecurity.com/2015/02/fbi-3m-bounty-for-zeus-trojan-author/
- https://zeusmuseum.com/actors/
- https://github.com/ytisf/theZoo/tree/master/malware/Binaries/ZeusBankingVersion_26Nov2013
- https://web.archive.org/web/20210416013527/https://pfarrside.com/zeus-malware-analysis-remnux/

Deepak --project report-final.pdf

ORIGINALITY REPORT

15%	15%	6%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	github.com Internet Source	8%
2	www.joesecurity.org Internet Source	3%
3	filter.micron.ri.cmu.edu Internet Source	1%
4	www.researchgate.net Internet Source	1%
5	moonlightofnepal.blogspot.com Internet Source	1%
6	quieora.ink Internet Source	<1%
7	www.coursehero.com Internet Source	<1%
8	Alankrita Aggarwal, Deepak Chatha. "To Design a Mammogram Edge Detection Algorithm Using an Artificial Neural Network (ANN)", International Journal of Distributed Artificial Intelligence, 2019 Publication	<1%

9	scholar.mzumbe.ac.tz Internet Source	<1 %
10	Luca Caviglione, Michal Choras, Igino Corona, Artur Janicki, Wojciech Mazurczyk, Marek Pawlicki, Katarzyna Wasielewska. "Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection", IEEE Access, 2021 Publication	<1 %
11	psasir.upm.edu.my Internet Source	<1 %
12	www.igi-global.com Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		