MALWARE ANALYSIS
AND
REVERSE ENGINEERING

Project Report

Prepared By: Deepak Kushwaha

Institution - Lloyd Institute of Engineering and Technology

Course - IBM Cybersecurity PBEL training

Date – 16 July 2025

Supervisor: Hrushikesh Dinkar

Abstract

In the modern digital era, the frequency and sophistication of malware attacks have increased significantly, posing serious threats to individuals, enterprises, and government systems. Traditional signature-based antivirus solutions are no longer sufficient to detect and prevent advanced and evolving malware strains. This project aims to bridge that gap by performing in-depth malware analysis and reverse engineering to understand malicious behavior, uncover attack patterns, and contribute to more intelligent cybersecurity defense mechanisms.

Our approach involved both static and dynamic analysis techniques conducted within a safe and isolated virtual environment. Tools such as Triage Sandbox powered by CrowdStrike Falcon, PEStudio, Ghidra, and Wireshark were used for extracting technical information from malware samples. Static analysis focused on disassembling binaries to reveal internal structures, strings, and code logic. In parallel, dynamic analysis allowed us to observe real-time system behavior, such as process creation, registry modifications, file system activity, and external network communications. Indicators of Compromise (IOCs) were collected to track infection methods and persistence strategies.

The analysis revealed several tactics employed by malware, including code obfuscation, anti-virtualization, and attempts to connect with command-and-control servers. Through reverse engineering, we were able to partially reconstruct malware logic and detect its payload delivery methods. These insights assist in developing more robust detection rules and understanding modern threat actors' techniques.

This project demonstrates the critical role of malware analysis and reverse engineering in modern cybersecurity. By decoding how malware operates, organizations can enhance their threat detection, response capabilities, and overall security posture.

Table of Contents

1. Introduction

1.1. What is the Project About?

This project, titled **"Malware Analysis and Reverse Engineering,"** is focused on understanding how malware operates at both the system and code level. Malware refers to malicious software designed to disrupt, damage, or gain unauthorized access to computer systems. The goal of this project is to perform in-depth analysis of malware samples using both static and dynamic techniques in a secure, virtualized environment. Through reverse engineering, we aim to uncover the internal logic, behavior, and indicators of compromise associated with different types of malwares. The project emphasizes both practical analysis and ethical handling of malware in a controlled setup.

1.2. Why Did I Choose This Project?

I chose this project because malware continues to be a significant and evolving threat in the field of cybersecurity. With the increasing sophistication of cyberattacks, traditional detection methods are often insufficient. Understanding how malware works from the inside is essential for developing better defense mechanisms and detection systems. Personally, I wanted to build strong foundational skills in binary analysis, behavioral tracking, and threat investigation. This project offered an opportunity to work hands-on with real-world malware samples in a safe and educational manner, helping bridge the gap between theoretical knowledge and practical cybersecurity applications.

1.3. What Problem Am I Solving?

The key problem being addressed is the lack of understanding and visibility into how modern malware behaves once it infects a system. Many malware strains use obfuscation, anti-analysis techniques, or operate in stealth to avoid detection. By analyzing malware through reverse engineering, this project attempts to demystify the internal logic, payloads, and behavioral patterns of malicious software. The end goal is to generate insights that can help improve detection mechanisms, develop behavioral signatures, and educate cybersecurity learners on threat response and analysis techniques.

1.4. How Will I Solve It?

To address the problem, I followed a step-by-step methodology:

- **Malware samples** were collected and prepared for analysis in an isolated virtual environment.
- **Static analysis** was conducted using tools to extract metadata, embedded strings, and examine file structures without executing the malware.
- **Dynamic analysis** was performed by executing the malware in a sandbox environment to observe real-time behavior, such as process creation, file modifications, network activity, and registry changes.
- **Reverse engineering tools** were used to decompile or disassemble the binary code to understand deeper functionality and payload delivery mechanisms.
- Findings were documented in structured reports, focusing on behavioral indicators and possible detection methods.

1.5. Tools and Methods used

A variety of industry-standard tools and platforms were used during this project:

- **REMnux:** A Linux-based reverse engineering and malware analysis distribution used for setting up the secure analysis environment.
- **Triage Sandbox:** An online sandboxing tool powered by CrowdStrike Falcon used for executing and monitoring malware behavior in real-time.
- **PEStudio:** Used for static analysis of Windows executable files to extract metadata, API imports, and indicators.
- **Ghidra**: A powerful reverse engineering tool used to disassemble and analyze binary code at the instruction level.
- These tools enabled a comprehensive view of how malware behaves and how it can be analyzed effectively for educational and defensive purposes.

2. Literature Review

2.1. Malware Analysis

Malware analysis is a structured approach used to examine malicious software to understand how it operates, spreads, and impacts computer systems. It is one of the most critical components of modern cybersecurity, as it allows analysts to uncover the internal logic of the malware, detect infection vectors, and identify the types of systems it targets. Through this process, analysts can generate detailed reports, build detection signatures, and create response strategies. During my research, I came across several real-world cases where malware analysis played a major role in uncovering ransomware campaigns, banking trojans, and remote access tools (RATs).

2.2. Need for Malware Analysis

The need for malware analysis has grown rapidly due to the increasing frequency and complexity of cyber threats. Cybercriminals now use techniques like code obfuscation, polymorphism, and encryption to avoid detection by traditional antivirus solutions. Therefore, analyzing malware at a deeper level helps organizations and researchers identify these hidden behaviors and adapt their defense mechanisms accordingly. My research showed that without proper analysis, most organizations are only treating the symptoms of attacks, not understanding their root causes. Malware analysis also contributes to threat intelligence, which is shared among cybersecurity communities to improve global defense systems.

2.3. Reverse Engineering

Reverse engineering is the process of taking a compiled executable or binary file and analyzing it to retrieve high-level insights about its logic, structure, and behavior. In malware analysis, reverse engineering is used when static or dynamic analysis alone does not reveal enough information. It involves disassembling or decompiling the code using tools like Ghidra or IDA Pro. While researching this area, I found that reverse engineering is especially useful in identifying embedded payloads, encrypted communication routines, and anti-analysis techniques. It also helps in reconstructing the behavior of malware that may disable itself in virtual environments or sandboxes.

2.4. Static Analysis

Static analysis refers to the examination of a malware sample without executing it. This involves looking into the file's metadata, embedded strings, file structure, imports/exports, and opcode sequences. The key advantage of static analysis is that it can be performed safely without any risk of infecting the host system. During my exploration, I used tools like PEStudio and Ghidra to analyze .exe files, revealing useful information such as suspicious API calls, encrypted strings, and unusual section headers. Static analysis is particularly valuable in identifying indicators of compromise (IOCs) like domain names, IP addresses, and registry paths used by malware.

2.5. Dynamic Analysis

Dynamic analysis involves running the malware in a controlled and isolated environment to observe its behavior in real-time. This method reveals how the malware interacts with the operating system, what changes it makes to files and the registry, and whether it communicates over the network. Through dynamic analysis, one can detect payloads that are only activated under certain conditions or monitor how malware escalates privileges and persists after a reboot. In my research, I used tools such as REMnux and Wireshark within virtual machines to capture these behaviors. This practical approach gave me insights into how malware can evade detection by only activating under certain system conditions or time delays.

3. Methodology

This section outlines the step-by-step approach that I followed to conduct malware analysis and reverse engineering. The methodology includes the setup of a safe analysis environment, selection of tools, and execution of both static and dynamic analysis techniques. As the project focuses on learning purposes, the process will be carried out in a controlled virtual environment to prevent any unintentional harm to the host system or network.

3.1. Environment Setup

To ensure the safety and isolation of the analysis process, all malware samples will be examined inside a virtual machine (VM). The following setup will be used:

- **Operating System:** REMnux Linux (for analysis) and Windows 10 VM (for malware execution), Android VM
- **Virtualization Platform:** VirtualBox or VMware
- **Network Mode:** Host-only / Bridged with internet disabled (to prevent malware from reaching the internet)
- **Snapshot Enabled:** VM snapshots will be taken before each test to allow easy rollback

This sandbox setup ensures malware does not affect the host system or spread to external networks.

3.2. Static Analysis Methodology

Static analysis involves inspecting the malware without executing it. This will be the first phase of analysis to extract preliminary information from the sample.

Steps:

1. **File Inspection:**
    - Examine file type, size, and metadata using tools like file, strings, and exiftool.
2. **Header and Structure Analysis:**

- Use PEStudio to examine the Portable Executable (PE) file format and detect anomalies.
- Identify suspicious sections, imports, and embedded resources.

3. **String Extraction:**
    - Extract readable ASCII and Unicode strings to find URLs, IPs, commands, etc.
4. **Disassembly & Decompilation:**
    - Use Ghidra to disassemble the binary and analyze code flow, functions, and API calls.
5. **Detection of Obfuscation or Packing:**
    - Look for signs of packed or encrypted content using entropy checks.

This phase will provide indicators of compromise (IOCs) and help form hypotheses about malware's behavior.

3.3. Dynamic Analysis Methodology

Dynamic analysis is the second phase, where the malware will be executed in a controlled environment to observe its real-time behavior.

Planned Steps:

1. **Sandbox Execution:**
    - The malware will be executed inside an offline Windows VM with monitoring tools active.
2. **Process and File Monitoring:**
    - Tools like Procmon, Process Explorer, and RegShot will be used to track changes to files, processes, and the registry.
3. **Network Monitoring:**
    - Use Wireshark to capture any network traffic generated by the malware, even in offline scenarios.
    - Analyze DNS requests, attempted connections, or C2 communication patterns.
4. **System Changes Observation:**
    - Monitor for creation of scheduled tasks, persistence mechanisms, or service changes.

4. Results and Discussion

Result

The malware analysis process reveals several critical behaviors and characteristics of the sample. During static analysis, the structure of the malware file is examined, and important details such as embedded strings, suspicious functions, and file metadata are identified. This helps in detecting potentially harmful operations like file modification, code injection, or system manipulation even before execution.

In dynamic analysis, the execution of the malware in a controlled environment allows observation of its real-time behavior. The malware often performs actions such as creating or modifying files, adding registry entries for persistence, spawning new processes, and attempting to connect to external addresses or domains. The changes made to the system during execution provide clear evidence of the malware's purpose and the impact it could have on an unprotected system.

Discussion

The results show that malware commonly uses a combination of stealth and persistence techniques to avoid detection and maintain long-term control over a system. Static analysis is useful for identifying hidden components and understanding how the malware is built, especially when the malware uses obfuscation or packing. However, dynamic analysis gives a clearer picture of how the malware behaves in real conditions.

By combining both types of analysis, a more complete understanding is achieved. This approach helps in identifying not only the technical structure of the malware but also its objectives, such as data theft, surveillance, or system disruption. Overall, the analysis demonstrates the importance of a layered approach in cybersecurity to detect, understand, and defend against modern malware threats.

5. Conclusion

This project provides a comprehensive understanding of malware analysis and reverse engineering by exploring both static and dynamic analysis techniques. Through static analysis, the internal structure of the malware is examined to uncover hidden strings, suspicious code patterns, and potential indicators of compromise. Dynamic analysis allows the observation of real-time behavior, including system modifications, process activity, and network communications.

The combination of both methods ensures a deeper and more accurate insight into the functioning and intentions of the malware. This dual approach not only highlights the technical aspects of malware but also emphasizes the importance of using secure environments and systematic methodologies in cybersecurity research. Overall, the project reinforces the critical role of malware analysis in detecting, understanding, and defending against modern cyber threats.

6. Recommendations

Based on the analysis and understanding gained during this project, the following recommendations are proposed:

- **Implement Isolated Analysis Environments:** Always conduct malware analysis in secure, sandboxed virtual environments to prevent accidental infection or spread.
- **Combine Static and Dynamic Techniques:** Relying on only one analysis method may provide incomplete results. A combination of static and dynamic approaches offers a more accurate assessment of malware behavior.
- **Stay Updated with Threat Intelligence:** Cyber threats evolve rapidly. Regularly referring to security advisories, threat intelligence feeds, and malware databases can help analysts stay informed about the latest attack patterns and techniques.
- **Use Well-Documented and Legal Samples:** Ensure that all malware samples used for learning or research are sourced from trusted repositories and used ethically for educational purposes only.
- **Enhance Reporting and Documentation:** Proper documentation of every analysis step helps in future reference, peer collaboration, and continuous improvement in detection techniques.

References

- Remnux Documentation
- Ghidra Documentation
- PEStudio Website
- Youtube – Malware Analysis

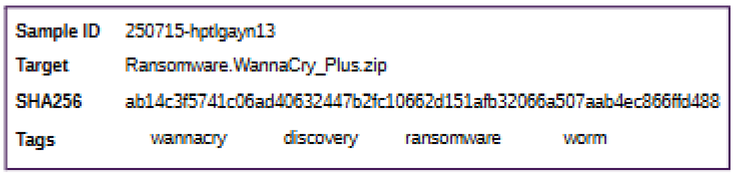
- Cybersecurity & Infrastructure security agency guidance
- Thezoo – Malware Repository

Appendix

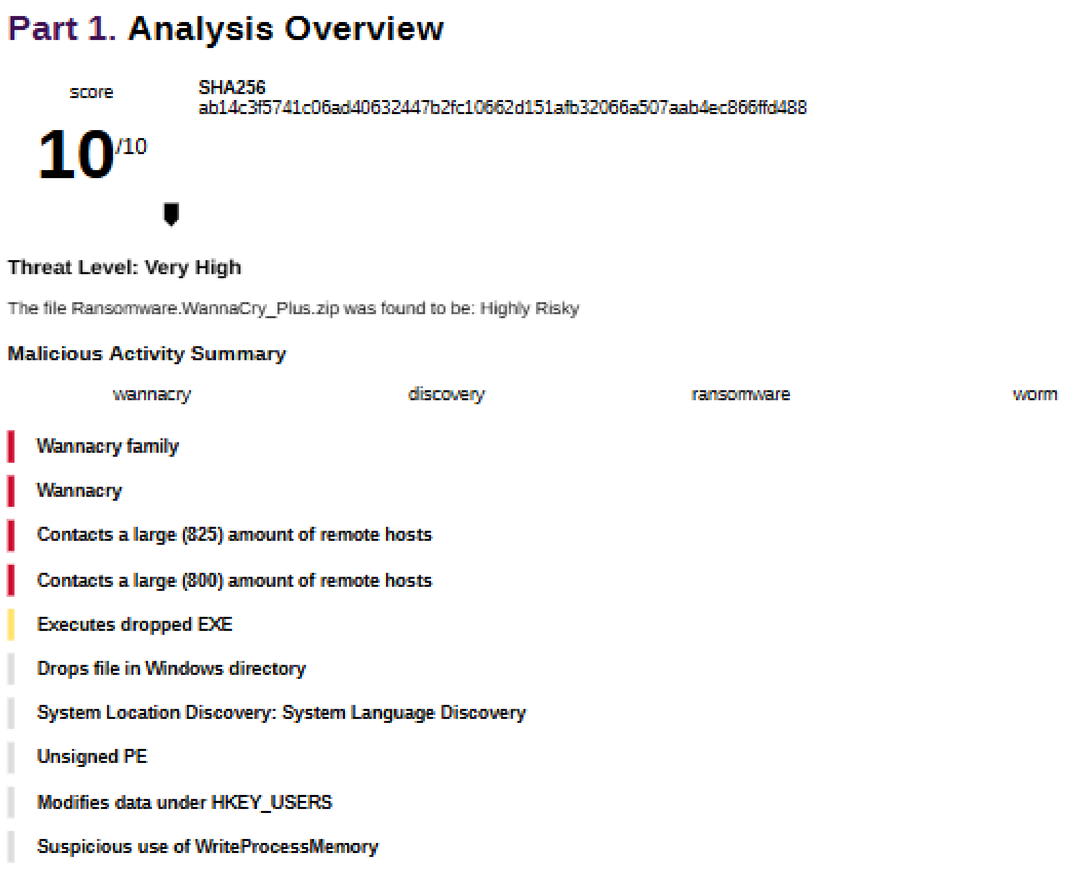
Malware Report :

Malware name : WannaCry ransomware cyrptoworm

Target OS : Windows



Sample ID	250715-hptlgayn13
Target	Ransomware.WannaCry_Plus.zip
SHA256	ab14c3f5741c06ad40632447b2fc10662d151afb32066a507aab4ec866ffd488
Tags	wannacry discovery ransomware worm



Part 1. Analysis Overview

score

10/10

SHA256
ab14c3f5741c06ad40632447b2fc10662d151afb32066a507aab4ec866ffd488

Threat Level: Very High

The file Ransomware.WannaCry_Plus.zip was found to be: Highly Risky

Malicious Activity Summary

wannacry discovery ransomware worm

- Wannacry family
- Wannacry
- Contacts a large (825) amount of remote hosts
- Contacts a large (800) amount of remote hosts
- Executes dropped EXE
- Drops file in Windows directory
- System Location Discovery: System Language Discovery
- Unsigned PE
- Modifies data under HKEY_USERS
- Suspicious use of WriteProcessMemory

Part 2. MITRE ATT&CK

2. 1. Enterprise Matrix V16

Reconnaissance TA0043		
Resource Development TA0042		
Initial Access TA0001		
Execution TA0002		
Persistence TA0003		
Privilege Escalation TA0004		
Defense Evasion TA0005		
Credential Access TA0006		
Discovery TA0007	Network Service Discovery T1046	System Location Discovery T1614
		System Language … T1614.001
Lateral Movement TA0008		
Collection TA0009		
Command and Control TA0011		
Exfiltration TA0010		
Impact TA0040		

Part 4. Analysis: behavioral1

4. 1. Detonation Overview

Submitted	Reported	Platform	Max time kernel	Max time network
2025-07-15 06:55	2025-07-15 06:56	win10v2004-20250610-en	60s	61s

4. 2. Command Line

```
rundll32.exe C:\Users\Admin\AppData\Local\Temp\Win32.Wannacry.dll,#1
```

4. 3. Signatures

Wannacry

wannacry ransomware worm

Wannacry family

wannacry

Contacts a large (825) amount of remote hosts

discovery

Executes dropped EXE

Description	Indicator	Process	Target
N/A	N/A	C:\WINDOWS\mssecsvc.exe	N/A
N/A	N/A	C:\WINDOWS\mssecsvc.exe	N/A
N/A	N/A	C:\WINDOWS\tasksche.exe	N/A

Drops file in Windows directory

Description	Indicator	Process	Target
File created	C:\WINDOWS\mssecsvc.exe	C:\Windows\SysWOW64\rundll32.exe	N/A
File created	C:\WINDOWS\tasksche.exe	C:\WINDOWS\mssecsvc.exe	N/A

System Location Discovery: System Language Discovery

discovery

Description	Indicator	Process	Target
Key opened	\REGISTRY\MACHINE\SYSTEM\ControlSet001\Control\NLS\Language	C:\WINDOWS\mssecsvc.exe	N/A
Key opened	\REGISTRY\MACHINE\SYSTEM\ControlSet001\Control\NLS\Language	C:\WINDOWS\mssecsvc.exe	N/A
Key opened	\REGISTRY\MACHINE\SYSTEM\ControlSet001\Control\NLS\Language	C:\Windows\SysWOW64\rundll32.exe	N/A

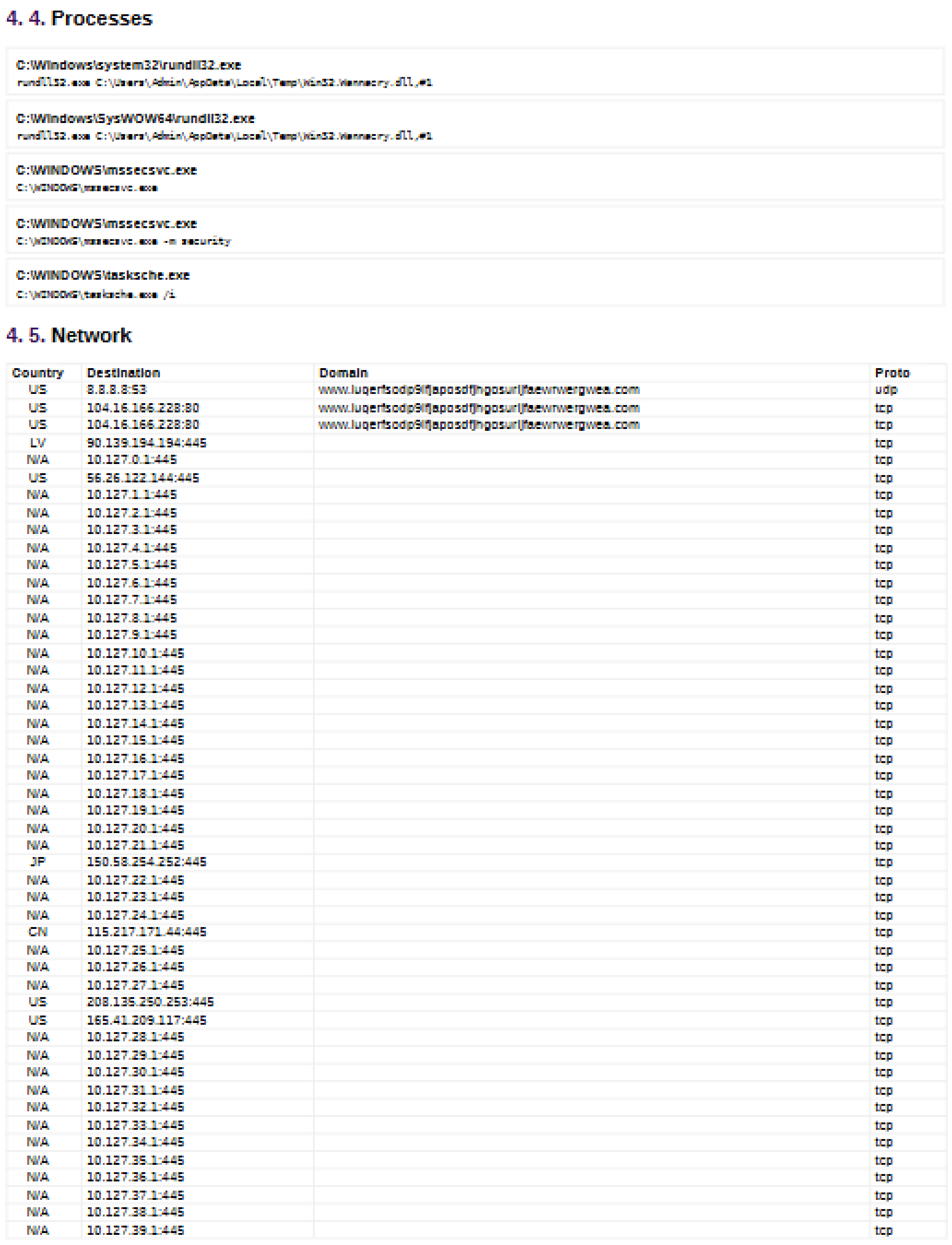
Modifies data under HKEY_USERS

Description	Indicator	Process	Target
Set value (int)	\REGISTRY\USER\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect = "0"	C:\WINDOWS\mssecsvc.exe	N/A
Key created	\REGISTRY\USER\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\	C:\WINDOWS\mssecsvc.exe	N/A
Set value (int)	\REGISTRY\USER\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass = "1"	C:\WINDOWS\mssecsvc.exe	N/A
Set value (int)	\REGISTRY\USER\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName = "1"	C:\WINDOWS\mssecsvc.exe	N/A
Set value (int)	\REGISTRY\USER\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet = "1"	C:\WINDOWS\mssecsvc.exe	N/A

Suspicious use of WriteProcessMemory

Description	Indicator	Process	Target
PID 4168 wrote to memory of 5180	N/A	C:\Windows\system32\rundll32.exe	C:\Windows\SysWOW64\rundll32.exe
PID 4168 wrote to memory of 5180	N/A	C:\Windows\system32\rundll32.exe	C:\Windows\SysWOW64\rundll32.exe
PID 4168 wrote to memory of 5180	N/A	C:\Windows\system32\rundll32.exe	C:\Windows\SysWOW64\rundll32.exe
PID 5180 wrote to memory of 5268	N/A	C:\Windows\SysWOW64\rundll32.exe	C:\WINDOWS\mssecsvc.exe
PID 5180 wrote to memory of 5268	N/A	C:\Windows\SysWOW64\rundll32.exe	C:\WINDOWS\mssecsvc.exe
PID 5180 wrote to memory of 5268	N/A	C:\Windows\SysWOW64\rundll32.exe	C:\WINDOWS\mssecsvc.exe

Total of 502 Network (UDP+TCP) connections were made to different locations



4. 4. Processes

C:\Windows\system32\rundll32.exe
```
rundll32.exe C:\Users\Admin\AppData\Local\Temp\Win32.Wannacry.dll,#1
```

C:\Windows\SysWOW64\rundll32.exe
```
rundll32.exe C:\Users\Admin\AppData\Local\Temp\Win32.Wannacry.dll,#1
```

C:\WINDOWS\mssecsvc.exe
```
C:\WINDOWS\mssecsvc.exe
```

C:\WINDOWS\mssecsvc.exe
```
C:\WINDOWS\mssecsvc.exe -m security
```

C:\WINDOWS\tasksche.exe
```
C:\WINDOWS\tasksche.exe /i
```

4. 5. Network

Country	Destination	Domain	Proto
US	8.8.8.8:53	www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	udp
US	104.16.166.228:80	www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	tcp
US	104.16.166.228:80	www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	tcp
LV	90.139.194.194:445		tcp
N/A	10.127.0.1:445		tcp
US	56.26.122.144:445		tcp
N/A	10.127.1.1:445		tcp
N/A	10.127.2.1:445		tcp
N/A	10.127.3.1:445		tcp
N/A	10.127.4.1:445		tcp
N/A	10.127.5.1:445		tcp
N/A	10.127.6.1:445		tcp
N/A	10.127.7.1:445		tcp
N/A	10.127.8.1:445		tcp
N/A	10.127.9.1:445		tcp
N/A	10.127.10.1:445		tcp
N/A	10.127.11.1:445		tcp
N/A	10.127.12.1:445		tcp
N/A	10.127.13.1:445		tcp
N/A	10.127.14.1:445		tcp
N/A	10.127.15.1:445		tcp
N/A	10.127.16.1:445		tcp
N/A	10.127.17.1:445		tcp
N/A	10.127.18.1:445		tcp
N/A	10.127.19.1:445		tcp
N/A	10.127.20.1:445		tcp
N/A	10.127.21.1:445		tcp
JP	150.58.254.252:445		tcp
N/A	10.127.22.1:445		tcp
N/A	10.127.23.1:445		tcp
N/A	10.127.24.1:445		tcp
CN	115.217.171.44:445		tcp
N/A	10.127.25.1:445		tcp
N/A	10.127.26.1:445		tcp
N/A	10.127.27.1:445		tcp
US	208.135.250.253:445		tcp
US	165.41.209.117:445		tcp
N/A	10.127.28.1:445		tcp
N/A	10.127.29.1:445		tcp
N/A	10.127.30.1:445		tcp
N/A	10.127.31.1:445		tcp
N/A	10.127.32.1:445		tcp
N/A	10.127.33.1:445		tcp
N/A	10.127.34.1:445		tcp
N/A	10.127.35.1:445		tcp
N/A	10.127.36.1:445		tcp
N/A	10.127.37.1:445		tcp
N/A	10.127.38.1:445		tcp
N/A	10.127.39.1:445		tcp