An Empirical Study of Data Disruption by Ransomware Attacks

Yiwei Hou
BNRist, Tsinghua University
Beijing, China
houyw22@mails.tsinghua.edu.cn

Lihua Guo
BNRist, Tsinghua University
Beijing, China
glh22@mails.tsinghua.edu.cn

Chijin Zhou
BNRist, Tsinghua University
Beijing, China
tlock.chijin@gmail.com

Yiwen Xu
BNRist, Tsinghua University
Beijing, China
xuyiwen14@gmail.com

Zijing Yin
BNRist, Tsinghua University
Beijing, China
yzjaurora@gmail.com

Shanshan Li
National University of Defense Technology
Changsha, China
shanshanli@nudt.edu.cn

Chengnian Sun
Waterloo University
Waterloo, Canada
cnsun@uwaterloo.ca

Yu Jiang*
BNRist, Tsinghua University
Beijing, China
jiangyu198964@126.com

ABSTRACT

The threat of ransomware to the software ecosystem has become increasingly alarming in recent years, raising a demand for large-scale and comprehensive ransomware analysis to help develop more effective countermeasures against unknown attacks. In this paper, we first collect a real-world dataset MarauderMap, consisting of 7,796 active ransomware samples, and analyze their behaviors of disrupting data in victim systems. All samples are executed in isolated testbeds to collect all perspectives of six categories of runtime behaviors, such as API calls, I/O accesses, and network traffic. The total logs volume is up to 1.98 TiB. By assessing collected behaviors, we present six critical findings throughout ransomware attacks' data reconnaissance, data tampering, and data exfiltration phases. Based on our findings, we propose three corresponding mitigation strategies to detect ransomware during each phase. Experimental results show that they can enhance the capability of state-of-the-art anti-ransomware tools. We report a preliminary result of a 41%-69% increase in detection rate with no additional false positives, showing that our insights are helpful.

CCS CONCEPTS

• **Security and privacy** → **Software security engineering**; • **General and reference** → **Empirical studies**.

KEYWORDS

Ransomware, Data Disruption, Runtime Behaviors, Mitigation Strategies, Empirical Study

*Yu Jiang is the corresponding author.

ICSE '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0217-4/24/04…$15.00
https://doi.org/10.1145/3597503.3639090

ACM Reference Format:
Yiwei Hou, Lihua Guo, Chijin Zhou, Yiwen Xu, Zijing Yin, Shanshan Li, Chengnian Sun, and Yu Jiang. 2024. An Empirical Study of Data Disruption by Ransomware Attacks. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3597503.3639090

1 INTRODUCTION

In recent years, the prevalence of ransomware has seen a marked increase, with a growing number of sample variants, heightened attack frequency, and an expanding range of victims. The substantial economic losses and negative consequences of ransomware attacks have been documented, with the average payout in the first quarter of 2023 reaching $408,644k, a 58% increase from the previous quarter [1]. The consequences are even more dire for medical institutions that have experienced loss of life due to their inability to regain control of computer equipment [2–4].

To defend against ransomware, researchers focus on ransomware analysis to comprehend its characteristics, which is crucial in designing effective detection strategies. While many existing research studies on ransomware rely on static analysis [5–7], this approach cannot fully unveil the behaviors of ransomware. Firstly, comprehending ransomware behavior is challenging due to the impracticality of acquiring the source code and the obstacles encountered in reverse-engineering binary files, such as code obfuscation and packing. As a result, a significant amount of manual effort is required to analyze ransomware samples. In addition, static analysis is inadequate in identifying dynamically generated and utilized code and data, and it cannot thoroughly assess malicious behavior that depends on environmental factors, leading to incomplete results.

Dynamic analysis is applied to enhance the ransomware study. However, some studies lack generality and pervasiveness, as they solely concentrate on a single sample's entire lifecycle or kill chain. For example, Almashhadani et al. [8] only analyze the Locky family, Umar et al. [9] only examine Conti, and Caroscio et al. [10] focus on Babuk. Other studies lack comprehensiveness, analyzing multiple samples from limited perspectives. For example, UShallNotPass [11]

only considers pseudo-random number generator functions as critical resources, RansomSpector [12] identifies ransomware through monitoring file system and network activities, and RTrap [13] focuses on file I/O to detect ransomware. Therefore, there is a growing need for a large-scale and comprehensive ransomware analysis.

Undertaking such an analysis is a challenging task. Firstly, the absence of qualified datasets presents a challenge. Available ransomware datasets only contain a limited number of samples ranging between 7 to 582 [14–17]. This limitation hinders researchers from gaining a broader comprehension of ransomware. Secondly, gathering diverse perspectives on ransomware runtime behaviors remains an open problem. To obtain accurate ransomware behaviors, a completely isolated environment should be provided for the samples, and runtime information must be collected without detection. The more comprehensive the behaviors, the more likely it is to identify intriguing ransomware characteristics for detection purposes.

To fill this gap, we construct a ransomware dataset in this paper and conduct experiments in a controllable environment. Our primary objective is to shed light on the disruptive techniques utilized by ransomware to compromise the accessibility of host data and local networks. Notably, unlike previous analysis studies that solely focus on the user file space, our study examines both the user file space and system file space, providing a comprehensive analysis of the data at risk.

Our study workflow consists of four phases: dataset construction, testbed establishment, log collection, and threat assessment. Firstly, we collect ransomware samples from the last three years through multiple sources, select active ones through test running, and build the structured dataset MarauderMap, which contains 7,796 active samples from 95 families targeting the Microsoft Windows system. To the best of our knowledge, this sample analysis size is an order of magnitude more than previous studies. Then, we build a testbed containing components such as a workload tracker and workload monitor to run and analyze ransomware samples while ensuring an efficient and controllable experimental environment. Moreover, we gather dynamic runtime information through continuous log collection and divide them into six categories and three phases to facilitate later assessment. Finally, we elaborate on procedures that ransomware employs to disrupt data in the system and user file space from perspectives of the data reconnaissance, data tampering, and data exfiltration phases. We further draw the following insights.

In the data reconnaissance phase, ransomware tends to disrupt data in the system file space to create a conducive environment for operating, and profile file data in the user file space to prepare for later manipulation. The first data disruption procedures happen in the system file space without end-users' perceptions. These actions include injecting process, keeping persistence, identifying network environment, and locating C&C servers. Specifically, 39.14% of ransomware samples carry out process injections, and 42.88% modify certain system registries to keep persistence. Besides, in the user file space, data is not disrupted in this phase but has already been inventoried and profiled. The user's home folder is the most targeted file path for 90.48% of all samples, and the availability of PowerShell, CNG (i.e., Cryptography API: Next Generation) service, and recovery tools are also very likely to be examined. These observations alert researchers to track certain out-of-scope file-scanning traces to detect ransomware attacks.

In the data tampering phase, ransomware tends to carry out a series of preparatory disruption actions in the system file space to facilitate subsequent data encryption and exhibit preferences in its encryption algorithm implementation and file encryption pattern when manipulating data in the user file space. Ransomware tends to modify firewalls, download payloads, prevent rollback, and finally encrypt victims' private data in this phase, after which users are eventually aware of the attacks. Data in both the system file space and user file space are disrupted in this phase. To hinder user data restoration, 89.97% of ransomware samples delete system backups, and 24.29% go further to turn off system recovery functionalities. Then, when carrying out their encryption operations, 93.38% of ransomware implement algorithms from scratch, and 82.40% choose to replace the original file content by directly overwriting onto the target. This alerts us about high-risk behaviors when detecting potential ransomware attacks.

In the data exfiltration phase, ransomware leaks data for double extortion without users' notice in the system file space and disrupts data of other devices' user file space within the same network after expanding impact. We notice that double extortion has become increasingly popular in recent ransomware attacks. In addition to the data on the local host being disrupted, it will also be leaked out, whereby the attackers make "the recovery of data on the local host" and "the non-disclosure of leaked data" two conditions for demanding ransom. After leaking data to attackers' servers by cloud file sharing tools (18.07%) or through system APIs (21.73%) with mostly (90.02%) HTTP, UDP, and TCP protocols, ransomware asks for ransom to recover files in compromised hosts and keep exfiltrated files not public. In addition, ransomware also seeks to impact other devices within the same network to expand its attack scope. 3.36% of ransomware samples exploit vulnerabilities in Microsoft's SMB service (port 445) for shared data disruption, and 12.66% attempt to discover other targets through WSDAPI (port 5357). This serves as a reminder that the potential impacts of ransomware attacks extend beyond making data inaccessible on compromised hosts. They can also significantly compromise the confidentiality of user's data and the security of other devices within the same network.

Our paper makes the following contributions:

- **Real-world Dataset.** We construct the open-source dataset MarauderMap[1], comprising 7,796 active ransomware samples and analysis metadata from the past three years.
- **Thorough Analysis.** We systematically examine how ransomware disrupts data accessibility through three phases, supported by experimental runtime logs.
- **Practical Insights and Mitigation.** Based on our findings, we propose three mitigation strategies. Preliminary results show that they can help state-of-the-art anti-ransomware tools achieve a 41%-69% increase in detection rate with no additional false positives.

The rest of the paper is organized as follows. We give an outlook of ransomware attacks and analysis in Section 2. We describe our workflow in Section 3 and explain results and insights in Section 4. We discuss corresponding mitigation strategies in Section 5. After discussing ethics, threats to validity, and limitations in Section 6, we finally conclude our study in Section 7.

[1]The ransomware samples: https://github.com/THU-WingTecher/MarauderMap.

2 BACKGROUND AND RELATED WORK

Ransomware Attacks. Ransomware primarily extorts ransom by disrupting the accessibility of the victim's data. Upon infiltrating the victim's system, ransomware attempts to install itself and keep persistence [18, 19]. Then, it communicates with its C&C server to retrieve the payload necessary for an attack [20]. During the data disruption phase, ransomware scans data resources on the host, encrypts valuable files, and deletes data backups [15, 21–23]. Victims can only recover their valuable data after paying the ransom. Additionally, more and more ransomware engages in data leakage for dual extortion and, as with traditional malware, moves laterally through an organization's network to expand its impact [24–26].

Ransomware Analysis. Many researchers have analyzed ransomware through static and dynamic methods. Static analysis primarily involves employing reverse engineering to examine ransomware samples [27] and extracting signatures from captured portable executables [6, 7]. The dynamic analysis mainly focuses on the runtime behaviors of ransomware. For example, Unveil [23] and ShieldFS [15] analyze ransomware's I/O pattern when accessing files. Beaman et al. [28] and Morato et al. [29] analyze ransomware's network traffic when communicating with its C&C server. Kok et al. [30] and Bae et al. [31] analyze API sequence ransomware used to achieve attack. FlashGuard [32], SSD-Insider++ [33], RansomBlocker [34], RSSD [35], DeepWare[36] and RansomTag [37] detect and defend ransomware attacks through hardware characteristics related to the SSD, hypervisor, or hardware performance counters. Huang et al. [38] and Connolly et al. [39] measure ransom payments and involved victims. There are also surveys [20, 28, 40, 41] that systematically summarize the evolution, taxonomy, and countermeasures of ransomware. They mainly summarize existing publications with no newly conducted experiments.

Main Difference. Firstly, previous research on ransomware analysis either focuses on a limited number of samples or studies them from a narrow perspective, while this paper presents a large-scale measurement with comprehensive perspectives. Secondly, unlike ransomware survey articles that often lack a unified dataset and are constrained by limited sources of first-hand analytical results, this paper conducts experiments on a real-world dataset and provides dynamic runtime logs with a unified data structure. In this work, we provide complete analytical perspectives of ransomware data disruption procedures based on these runtime logs. With the insights we gain, we also offer corresponding strategies to detect and defend ransomware effectively.

3 STUDY WORKFLOW AND METHODOLOGY

This study aims to understand how ransomware disrupts the accessibility of data in system file space and user file space and to what extent it causes Denial-of-Resources attacks. In this section, we first describe details of the self-built dataset MarauderMap. Then, we present how to build a controllable testbed with a runtime logging mechanism and how to parallelize the deployment. Moreover, we introduce what kind of logs are collected and how to process them for later assessment. Finally, we describe how to divide two kinds of file space and three phases of data disruption, from what perspectives we categorize and analyze runtime logs. The general workflow is shown in Figure 1.

3.1 Dataset Construction

Recent years have seen a marked acceleration in the speed of ransomware evolution. This hinders the analysis, detection, and defense efforts against the rising ransomware threat. Ransomware attacks target various operating systems, but the most prevalent and notorious ones are those against the Microsoft Windows platform. Thus, we construct a ransomware dataset with sufficient samples targeting the Windows system. Firstly, we collect samples as extensively as possible from various sources to cover the most widespread ransomware variants in the real world. Our focus includes malware analysis platforms [42], cybersecurity forums [43–47], underground hacker forums [48], the dark web, and black market communication groups, which provide a diverse set of ransomware samples.

Next, we conduct tests for each sample to determine whether they are still active. This step involves running each sample in the testbed introduced in Section 3.2, observing its behaviors, confirming observations of file encryption, system lock, ransom note, or desktop changes, and verifying whether at least two security vendors listed by VirusTotal [42] flag it. Then, we use AVClass [49] to categorize samples into different families and perform deduplication by calculating their SHA-256 hash values.

After filtering, MarauderMap contains 7,796 active and unique ransomware samples from 95 families. They are all Win32 EXE file types. They are collected from the following sources: VirusTotal [42] (3,611), VX Vault [43] (3,273), InTheWild [45] (350), Bazaar [46] (323), tutorialjinni [44] (170), vx-underground [48] (46), and theZoo [47] (23). In terms of the distribution of ransomware families, the top ten are as follows: LockBit (29.32%), Conti (12.87%), REvil (7.93%), Cerber (3.83%), WannaCry (1.64%), BlackCat (1.33%), GandCrab (1.22%), Hive (0.89%), Maze (0.73%), and Jigsaw (0.64%). These samples were collected over a period ranging from November 2022 to March 2023. Regarding the first-seen timestamp distribution of the samples, 66 samples (0.85%) were detected before 2021, while the remaining 7,730 samples (99.15%) were first identified between 2021 and 2023. This distribution can reflect the recent evolution trends in ransomware.

3.2 Testbed Establishment

Setting up a controllable and efficient testbed to execute ransomware samples poses several challenges. For example, specific ransomware comes equipped with self-protection mechanisms, such as determining whether it is initiated through a command line interface or mouse clicks, and the former potentially leads to an incomplete capture of malicious behaviors, indicating that the testbed must reflect closely real-world machines with end-users operations; the large-scale study is resource-intensive, requiring secure management and efficient scheduling for a vast number of ransomware while also expediting the time needed to run all samples and collect runtime logs; while investigating ransomware behaviors accurately, the experiments should not cause any harm in the real world.

Therefore, we design a testbed which consists of the following two parts after careful deliberation: (1) The *running guests* are targets of ransomware intrusion. They include the virtual machine and the workload tracker. The virtual machine (Windows 10) provides an isolated environment for ransomware execution, ensuring the consequences of running malicious code are manageable. It is also

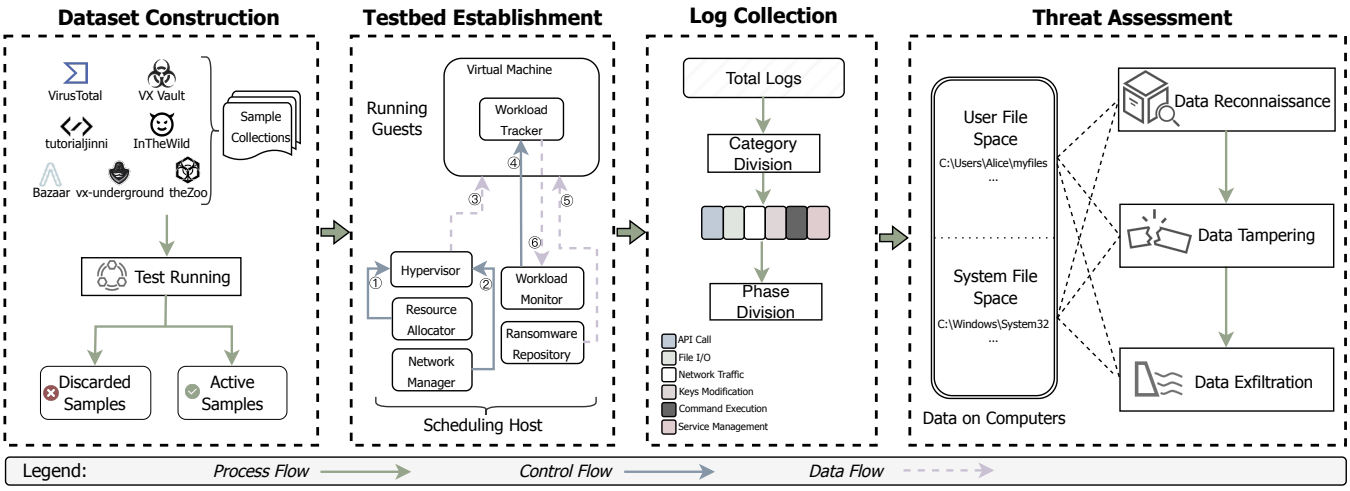


Figure 1: Overall workflow of our study. Firstly, we collect ransomware samples and select active ones through test running. Then, we build up the testbed, run ransomware samples, and gather their runtime logs. After that, we divide the logs into six categories within three phases and elaborate on ransomware's procedures to disrupt data.

equipped with automatic mouse movement to mirror real-world machines. Inside the virtual machine, a workload tracker captures and records ransomware behaviors in real-time. (2) The *scheduling host* contains five components to manage all running guests and aggregate runtime logs: the hypervisor (we use KVM in Ubuntu 22.04) is responsible for creating, managing, and monitoring virtual machine instances; the resource allocator allocates computing resources such as CPU and memory to paralleled virtual machines; the network manager manages network connections between the host and guests; the workload monitor uses CAPE [50] to receive and aggregate all runtime logs; and the ransomware repository schedules all samples.

These components interact through control flows and data flows, and the "Testbed Establishment" diagram in Figure 1 illustrates the sequence of interactions. The flow begins with the resource allocator sending resource allocation instructions to the hypervisor. Next, the network manager sends network connection instructions to the hypervisor. Then, the hypervisor monitors and manages the virtual machines. Once instances are set up, the workload monitor sends monitoring and control instructions to the workload tracker within each virtual machine, and the ransomware repository deploys samples to the virtual machines. As samples execute, the workload tracker captures and records their behaviors and runtime logs, sending information back to the workload monitor.

Experiments were undertaken on a server with a 12th Gen Intel(R) Core(TM) i7-12700KF CPU with 20 cores and 64 GiB of memory. During sample execution, we launched eight running guests in parallel, each allocated 2 cores and 4 GiB of memory. Each running guest operates within an isolated environment that does not affect or interfere with the others.

3.3 Log collection

Within the aforementioned testbed, we comprehensively gather a substantial amount of ransomware runtime logs. The volume of total logs is up to 1.98 TiB. We perform cleaning and standardization processes to enhance the readability further and ease the analysis of runtime logs. We remove redundant, irrelevant, or erroneous information and convert the log content into a unified format. Then, we divide the logs into six categories: API call, file I/O, network traffic, keys modification, command execution, and service management. They will help us delve into the data disruption procedures of ransomware attacks.

Specifically, by analyzing *API call* logs, we can understand the system functionalities exploited by the ransomware and its interactions with the operating system and other programs. By analyzing *file I/O* logs, we can track the file operations involved in the infection process, such as file encryption, deletion, and renaming. By analyzing *network traffic* logs, we can reveal the communication patterns between the ransomware and external servers, such as communication with C&C servers, data leakage, and ransom note transmission. By analyzing *registry modification* logs, we can understand how ransomware utilizes the registry to achieve persistence, hiding, and launching purposes. By analyzing *command execution* logs, we can understand how ransomware exploits system commands for its attack activities, such as process creation, file operations, and privilege escalation. By analyzing *service modification* logs, we can understand how ransomware achieves its malicious objectives by manipulating system services, such as disabling security software and creating backdoor services. Finally, each category of logs is further divided into three phases of data disruption, and details are introduced in Section 3.4.

3.4 Threat Assessment

We provide an exhaustive analysis of data disruption procedures using the empirical data obtained from six log categories. We examine its effects on data accessibility within the system file space and user file space across three phases and evaluate the severity of the Denial-of-Resources attacks.

We propose that the process of ransomware disrupting data accessibility can be divided into three phases: (1) *Data Reconnaissance*: before initiating the disruption process, ransomware consolidates its process and identifies sensitive data within the victim's computer system for subsequent extortion or exfiltration purposes. (2) *Data Tampering*: the data on the local machine are disrupted by harmful operations, such as encrypting, compressing, partitioning, or directly locking the system; thereby, the data accessibility of the compromised device is threatened. (3) *Data Exfiltration*: encrypted, locked, or deleted data is illicitly transmitted to the attacker's server or other devices to execute double extortion. In addition to the local machine, data in mapped network drives may also be affected.

Moreover, we divide data space on computers into two parts: *system file space* and *user file space*. Their fundamental distinction lies in the nature of the files contained within each domain. The system file space encompasses built-in files essential for the system's regular operation, with the primary concern of developers and administrators, and generally remains transparent to the end user. In contrast, the user file space comprises files introduced by the user upon utilizing the system, representing valuable private data requiring protection and thus garnering the intense attention of the user. Ransomware can disrupt data to different degrees in both file spaces and cause different severity of consequences, so we believe a separate investigation is necessary. Still, previous research has generally focused only on user file spaces, while in this paper, we think both of the file spaces are critical and make investigations.

Specifically, the *system file space* includes OS files (kernel objects, device drivers, and dynamic link libraries), configuration files (registries, INI files, system policy files, host files, and application configuration files), backup files (copies of user data, system configurations, and virtual machine files), supporting files (log files, cache files, icons, images, and fonts), localization files (multilingual user interface files and language packs), programs and scripts (executables, batch files, and PowerShell scripts), and security-related files (certificates, encryption keys, and access control lists).

The *user file space* includes document files (word documents, spreadsheets, presentations, PDFs, and text files), media files (images, videos, and audio files), database files (files from database management systems, e.g., SQL, Access, or Oracle databases), email files (archives or individual files from email clients, e.g., Outlook PST files), and project files (files created by development or design software, e.g., Visual Studio, PyCharm, or Photoshop projects). In our testbed, each running guest contains 294,166 files in the system file space and 10,419 files in the user file space. To conclude, these are exactly the "data" our study focuses on.

4 MEASUREMENT RESULT

In this section, we unveil how ransomware causes Denial-of-Resources attacks by systematically analyzing the six categories of runtime logs on aspects of three phases of data disruption.

4.1 Data Reconnaissance

Inject Process. Upon deployment on the victim's host, ransomware initiates a technique known as process injection. By injecting malicious code into another legitimate process, ransomware can bypass security defenses, obfuscate its malicious activities, elevate its privileges, or achieve persistence. This critical step lays the groundwork for the execution of subsequent destructive actions.

After investigating API call chains of process injection techniques, we have the following observations: (1) 951 samples perform APC injection: inject malicious code into the asynchronous procedure call queue of one or more threads in the target process, and the code will automatically execute when the thread enters a wait state. (2) 864 samples perform thread execution hijacking: modify the context of existing threads to hijack the target process's thread execution flow and ultimately execute malicious code within the target process. (3) 860 samples perform classic DLL injection: load the malicious DLL into the target process's address space and execute its exported functions by creating a remote thread or other methods. (4) 185 samples perform system-wide hooks: load malicious code into all existing and newly-created processes by setting system-wide global hooks and facilitating malicious activities across multiple processes. (5) 174 samples perform process hollowing: create a new process (typically a legitimate one) in a suspended state, replace its in-memory code with malicious code, and then resume execution, masquerading as a legitimate program. (6) 17 samples perform process doppelganging: leverage Windows file transaction services to create a file transaction, load malicious code into a transactional view of a legitimate file, create a new process, and then circumvent security checks by utilizing the malicious code concealed within the transaction.

Table 1: Windows registry keys that ransomware modifies. Keys 1-5 are related to persistence keeping, and Keys 6-10 are related to firewall setting.

ID	Registry Key
1	HKEY_LOCAL_MACHINE\...\CurrentVersion\Run
2	HKEY_CURRENT_USER\...\CurrentVersion\Run
3	HKEY_CURRENT_USER\...\ CurrentVersion\RunOnce
4	Policies\System\DisableRegistryTools
5	Policies\System\DisableTaskMgr
6	FirewallPolicy\StandardProfile\AuthorizedApplications\List
7	FirewallPolicy\RestrictedServices\Static\System
8	FirewallPolicy\StandardProfile\EnableFirewall
9	FirewallPolicy\StandardProfile\DisableNotifications
10	FirewallPolicy\StandardProfile\DoNotAllowExceptions

Keep Persistence. After ensuring its ability to run, ransomware proceeds with a series of actions to maintain its vitality and persistence, striving to remain within the victim's host without being killed. As shown in Table 1, 1,620 samples tamper with Key 1, 1,410 samples tamper with Key 2, and 45 samples tamper with Key 3. By altering these registry entries, ransomware automatically launches upon every system restart, achieving user-level persistence. Furthermore, 30 samples modify Key 4, and one changes Key 5. By disabling the registry editor or task manager, ransomware prevents users from checking system risks and terminating malicious processes, thus bypassing security checks.

Finding 1: Data disruption in the system file space precedes data disruption in the user file space but is not yet noticeable to the user. 39.14% of ransomware samples require process injection to initiate and operate, while 42.88% ensure their own process integrity and achieve persistence by modifying Windows registries.

Identify Network Environment. Four samples utilize commands such as `ipconfig`, `netstat` and `systeminfo` to obtain the host's IP, and 41 samples query IP address from external websites like *whatismyipaddress.com* and *whatismyip.everdot.org*. IP information helps attackers locate the victims' host, convey ransom notes, or launch further attacks. Moreover, 320 samples utilize APIs like `GetComputerNameA` or `GetComputerNameW` to get victims' NETBIOS names, helping ransomware navigate the internal network.

Locate C&C Server. Locating the address of the C&C server is an essential step for ransomware in the reconnaissance phase since it needs to communicate with the C&C server to acquire the resources necessary for an attack or to report the victims' information to generate ransom notification pages. This process has two main methods: finding an active server through a hard-coded IP address list or DNS lookups.

Regarding connecting through hard-coded IP addresses, 762 samples attempt to connect to a server directly through *IP:port*. Among them, 506 samples attempt to connect to multiple dead unique *IP:port*, resulting from the expiration of their C&C servers.

The DNS-based approach can be classified into three types. (1) Normal DNS query. 556 samples make DNS queries to less than ten domain names, which indicates that they hard-code several C&C server domain names in their binaries. However, a blocklist easily notices and blocks such a normal DNS query. (2) Domain generation algorithm (i.e., DGA). 70 samples employ DGA to generate domain names and try to look up the active ones. These samples produce many nonsensical domain names with a shallow DNS successful rate (0.73%), many of which returned a "no such name" error. This method can help ransomware evade static detection since it does not store any information about its C&C server, and ransomware attackers can frequently change its C&C server's real IP to circumvent network traffic inspection. However, communication directly by IP is also suspicious. (3) Reverse DNS. 84 samples utilize reverse DNS lookups to resolve an IP address to its corresponding domain name. This means ransomware can communicate with its C&C server through a domain name, which appears more legitimate. However, ransomware servers frequently change their IP addresses to evade blocklist detection. As a result, many queries in the reverse DNS return unsuccessful results, with a success rate of only 5.27%.

Due to the low success rate of the second and third types, ransomware needs to conduct lots of queries to get the desired results. As shown in Figure 2, the queries are voluminous and uniformly distributed over a specific time duration. Furthermore, when DNS lookup returns a successful response, 34 samples immediately send an ICMP or TCP packet to the corresponding address, likely to test their connectivity, and 14 samples try to use `WNetUseConnectionW` to connect to it, as shown in Figure 3. Within a few seconds, they make a multitude of invocations and endeavor to establish connections using different passwords.

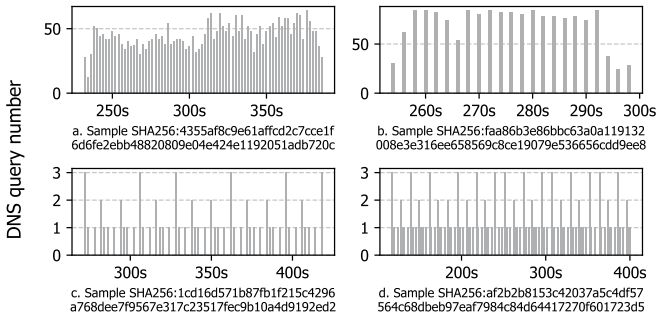


Figure 2: DNS query frequency of four samples. Axis X refers to the samples' execution time; Y refers to queries count every two seconds. Sample a and b use DGA, sample c and d use reverse DNS.

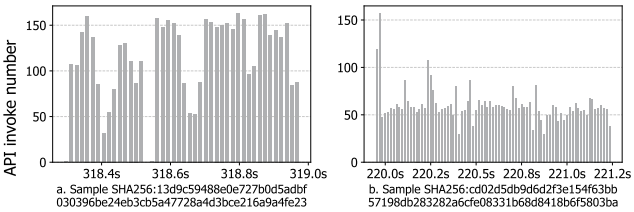


Figure 3: Invoke frequency of `WNetUseConnectionW` after getting a successful DNS response. Axis X refers to samples' execution time; Y refers to API requests per microsecond.

Profile Crucial Data. Finding crucial data before disrupting data accessibility is a significant step for ransomware. It usually scans the disks, pinpointing essential files within the compromised machine. These files include personal documents and more, the value of which is immeasurable to the user. All these activities are conducted beyond the user's awareness, leaving them vulnerable. The presence and accessibility of specific system files are also ascertained, impacting system protection mechanisms.

Table 2 provides a detailed overview of file scanning, illustrating the sensitive file paths most susceptible to ransomware attention. A total of 6,624 samples access Path 1, with an overall access count of 970,333. This path is the primary user folder in Windows systems, containing a vast array of user files. Once identified, these files risk being encrypted, deleted, or exfiltrated. *C:\Windows* is another high-risk path; among its numerous sub-directories, Path 2 is accessed 2,649,172 times by 6,492 samples. This folder contains numerous system files and programs crucial for the regular operation of the operating system, with ransomware targeting this directory to ascertain system versions and running services. Path 4 is accessed 4,307 times by 3,494 samples, enabling ransomware to determine whether the victim's host is located in a specific country or region. Path 5 is accessed 3,915 times by 3,211 samples, allowing ransomware to inspect which applications are running on the target system and pinpoint valuable files for the victim. Path 6 is accessed 3,039 times by 3,039 samples, providing ransomware insight into the state of the Windows shell within the system. Of particular note,

2,858 samples access Path 7, drawing the attention of ransomware to this kernel object file for subsequent utilization of CNG-provided encryption functions. Moreover, 2,693 samples access Path 9, and 450 samples access Path 12, enabling ransomware to discern the recovery features and options of the target system.

Finally, besides data on the compromised host, mapped network drives in the local network are also targeted. A total of 1,780 samples modify the *Windows\CurrentVersion\Explorer\MountPoints2* registry to obtain information on shared folders.

Table 2: The list of most targeted file system paths and kernel object paths. Sample count refers to the number of samples that access a particular path, while access count refers to the total count of all samples access.

ID	Path	Sample Count	Access Count
1	*C:\Users\Alice\myfiles*	6,624	970,333
2	*C:\Windows\System32*	6,492	2,649,172
3	*C:\Program Files*	3,689	3,449,456
4	*C:\Windows\Globalization*	3,494	4,307
5	*C:\Windows\apppatch*	3,211	3,915
6	*C:\Windows\WindowsShell.Manifest*	3,039	3,039
7	*\Device\CNG*	2,858	2,858
8	*C:\PerfLogs*	2,761	3,630
9	*C:\$WinREAgent*	2,693	7,060
10	*C:\$Recycle.Bin*	2,558	16,779
11	*C:\Windows\SystemResources*	1,957	2,274
12	*C:\Recovery*	450	998

Finding 2: The data has already been inventoried before any observable signs of data disruption in the user file space. 90.48% of ransomware samples target personal files within the user's home folder, making it the most critical and vulnerable private data. 41.51% of ransomware samples examine the availability of PowerShell, CNG service, and recovery tools, as these are tools that ransomware is highly likely to manipulate during later phases.

4.2 Data Tampering

Modify Firewall. Some ransomware modify the system firewall settings to allow their malicious network traffic to bypass security checks. This is achieved by changing the corresponding registry keys in Table 1 and can be categorized into three primary types. Firstly, 13 samples circumvent inspection by adding themselves to the trusted application list in the registry Key 6. Secondly, 11 samples work by altering the firewall rules to permit their network traffic, which are set in Key 7. Last, three samples turn off the firewall by setting the Key 8 value to 0 and the Key 9 and Key 10 values to 1. By these means, the firewall will not work, and users cannot see any notifications or messages of exceptions.

Download Payload. Some ransomware fetches additional payloads to complete their attacks. Our observations reveal that 35 samples use PowerShell scripts to acquire the payload. In addition, 269 samples utilize Windows APIs for payload retrieval. These statistics are illustrated in Table 3.

The payload usually contains malicious binary files essential for the attack. Upon completing file downloads, we notice that

Table 3: Statistics of receive data API.

API	Total Count	Sample Count	Avg. per Sample
`recv`	6,435	50	128.70
`WSARecv`	6,377	131	48.68
`recvfrom`	4,951	1	4,951
`HttpOpenRequestA` (Get)	313	124	2.52
`InternetOpenUrlA`	270	20	13.50
`WinHttpOpenRequest` (Get)	146	6	24.33
`HttpOpenRequestW` (Get)	144	113	1.27
`InternetOpenUrlW`	55	10	5.50

15 samples invoke the `deleteFileW` API to eliminate the file's *Zone.Identifier*. The *Zone.Identifier* is a security feature in the Windows platform that helps to track the security zone from which a file is downloaded. When a file is downloaded from the Internet or another untrusted source, the system adds a *Zone.Identifier* alternate data stream (i.e., ADS) to the file, marking it as potentially unsafe. This triggers a security warning when the user tries to execute the file, allowing them to decide whether to trust the file or not. Some ransomware deletes the *Zone.Identifier* of their malicious files to bypass these security warnings. Through this method, they deceive users into presuming the file originates from a trusted source, thus heightening the probability of a successful ransomware infection.

In other cases, the payload contains encryption keys utilized by ransomware to execute encryption functionalities. These keys are used to either directly encrypt data or encrypt encryption keys generated offline by ransomware on the victim's machine. Consequently, victims can not obtain decryption keys through reverse engineering, and they are compelled to pay ransoms to attackers to get their data back.

Table 4: Aspects that ransomware concerns to cut off the opportunity of system rollback and data recovery.

Action	Sample Count
Delete System Backups	6,587
Clean Event Logs	1,886
Disable Recovery	1,778
Kill Processes in the Blacklist	364
Kill Services in the Blacklist	15

Prevent Rollback. As shown in Table 4, by deleting system backups, cleaning event logs, disabling recovery, and killing specific processes and services, ransomware tends to cut off the opportunity for rollback, which makes subsequent data encryption operations irreparable. (1) 6,587 samples delete system backup files, whose procedures include using *vssadmin delete* commands, *wmic shadowcopy delete* commands, and so on. Volume shadow copies are typically used for backing up and restoring files; thus, deleting them can be considered a means of eliminating backups. (2) 1,886 samples clear event logs of four categories. By clearing application, security, system, and PowerShell logs, ransomware covers its attack traces, conceals attack activities, evades detection of data disruption behavior, and decreases the risk of being discovered. (3) 1,778 samples disable system restore or backup recovery capabilities by modifying Master Boot Record (i.e., MBR), modifying

\Policies\Microsoft\Windows NT\SystemRestore registry, or using *bcdedit /set* command. As a result, users have one less avenue for data recovery after the ransomware has damaged valuable data in the user file space. (4) 364 samples kill specific processes in a hard-coded blocklist, while 15 samples kill certain services such as *vss*, *mepocs*, *svc$*, and *sql*. There are several advantages to these two actions: maximizing the system resource usage of ransomware to increase the subsequent encryption speed, freeing up file usage to maximize the encryption scope, further preventing anti-virus and firewall intervention, and so on.

Finding 3: Ransomware carries out a series of preparatory disruption actions in the system file space to facilitate subsequent encryption of user files. 89.97% of samples delete system backups, and 24.29% go further to disable system recovery functions, hindering user data restoration.

Encrypt Data. When encrypting files, ransomware focuses on two crucial aspects: the encryption activity should be challenging to detect, and the encryption speed should be fast. Ensuring the stealthiness of encryption activities allows ransomware to evade detection, providing more time to encrypt additional data. A fast encryption speed enables ransomware to encrypt more data within a given time frame.

Table 5: Three patterns of ransomware's encryption tasks. There are 6,392 cryptographic ransomware in our dataset.

Encryption Pattern	Operation Sequence	Sample Portion
Overwrite	Open, Read, Encrypt, Write, Close.	82.40%
Smash and Rewrite	Open, Read, Encrypt, Close, Open, Write, Close, Create, Write, Close.	14.88%
Delete and Rewrite	Open, Read, Close, Delete, Encrypt, Create, Write, Close.	2.72%

The statistics reveal that: (1) Contrary to relying on system-provided libraries, implementing encryption algorithms from scratch is a favored method for ransomware to circumvent detection. A mere 485 samples utilize encryption APIs from the Crypto++ library, CryptoAPI (i.e., Microsoft Cryptographic API) or BCrypt (from CNG), with the rest resorting to self-implemented algorithms such as Curve25519, Elliptic-curve Diffie-Hellman for keys generation, AES-128-CBC, XSalsa20, ChaCha8, ChaCha20 for encryption, Poly1305 for signing, and SHA256, Blake2b for hashing. (2) Ransomware strategically selects patterns for its encryption task to speed up encryption. We classify ransomware encryption tasks into three patterns based on previous studies [12, 23]: Overwrite, Delete and Rewrite, and Smash and Rewrite. We monitor the encryption patterns through their corresponding operation sequence as shown in Table 5. Results reveal that in 5,267 samples (82.40%), the encrypted data directly supplants the original file content. Conversely, 951 samples (14.88%) initially read the original file content, smash the original file using random data, and finally write the encrypted data into a new file, replacing the original. Lastly, 174 samples (2.72%) first read the original file content, then delete the original file, and finally write the encrypted data into a new file to replace the original one. Generally, Overwrite pattern facilitates a comparatively faster encryption speed, while Smash and Rewrite offers a balanced trade-off between speed and stealthiness.

Finding 4: Ransomware exhibits preferences in its encryption algorithm implementation and file encryption patterns to disrupt data in the user file space safely and swiftly. 93.38% of ransomware samples implement their encryption algorithms rather than directly utilizing existing libraries provided by the system. 82.40% of cryptographic ransomware samples employ the Overwrite encryption pattern, directly overwriting the original file to increase encryption speed.

4.3 Data Exfiltration

Leak Data. Ransomware transmits data to its C&C server for double extortion (attackers threaten to disclose the victims' data if the demanded ransom is not paid). We observe 38 samples that use PowerShell scripts and 258 that utilize Windows APIs to transmit data, as indicated in Table 6. In addition, we notice 32 samples attempting to set up cloud file-sharing tools such as FileZilla [51] and WinSCP [52] for file transmission purposes.

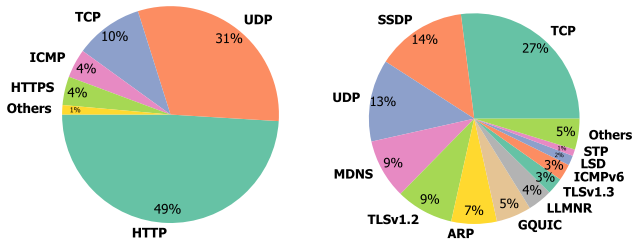
Table 6: Statistics of send data API.

API	Total Count	Sample Count	Avg. per Sample
`send`	3,371	38	88.71
`WSASend`	3,061	129	23.73
`sendto`	1,491	3	497.00
`HttpOpenRequestW` (POST)	108	7	15.43
`HttpOpenRequestA` (POST)	56	8	7.00
`WinHttpOpenRequest` (POST)	35	7	8.00

During the communication process, 90.02% of the interactions are facilitated by elementary protocols, including HTTP, UDP, and TCP; HTTPS enables 4.26% to prevent its payload from being checked by security checkers, and 4.31% are ICMP packets, possibly to test the connectivity. In contrast, benign executables (resources are mentioned in Footnote 3) use more diversified network protocols, as shown in Figure 4. Moreover, we observe that 20.89% of the TCP packets are retransmitted for ransomware. This ratio is only 7.00% for benign executables. This is attributed to the instability of the connection between ransomware and its C&C server.

Finding 5: Ransomware transmits the victim's data to its C&C server for double extortion. Among the samples exhibiting this, 18.07% samples utilize cloud file sharing tools, and 21.73% samples invoke send data APIs to steal victims' data. These communications mainly rely on fundamental network protocols, with 90.02% employing HTTP, UDP, and TCP and a mere 4.26% incorporating the more secure HTTPS protocol.

Ask for Ransom. Upon successful encryption, ransomware typically demands a ransom from its victims, seeking financial profit. Given the anonymity of cryptocurrencies, attackers usually utilize them for ransom requests. Our analysis uncovers Bitcoin-related keywords in the runtime logs of 37 samples and Monero-related keywords in 11 samples. Attackers leave a wallet link of



(a) Ransomware Samples (b) Benign Executables

Figure 4: Runtime network traffic protocol usage for ransomware samples and benign executables.

cryptocurrency for ransom collection, thereby circumventing the disclosure of their identities. Moreover, to provide the victims with more specific instructions, the attackers generate a dedicated page for each victim on the Tor [53] anonymous network, allowing them to understand the subsequent steps for ransom payment and acquisition of decryption tools. We detect Tor-related URLs in the runtime logs of 68 samples, with 28 samples even attempting to install the Tor browser for victims directly.

Expand Impact. Ransomware disrupts data on individual hosts and impacts additional hosts within the same network, intending to broaden its attack scope and demand a higher ransom. We observe two primary strategies for such impact expansion: the disruption of shared network resources and the lateral movement to other hosts. Regarding the first method, 246 samples exploit vulnerabilities in Microsoft's SMB2 service (port 445) to compromise shared resources. As shown in Figure 5, these samples generate a vast number of SMB2 requests within a short period. In the first few seconds, they announce its presence and attempt to discover other hosts within the same network. Upon locating another host with an open SMB server, they attempt to establish a connection and perform massive I/O operations such as querying information, traversing, reading, and writing to encrypt shared files, and even duplicating themselves to other hosts to facilitate lateral movement.

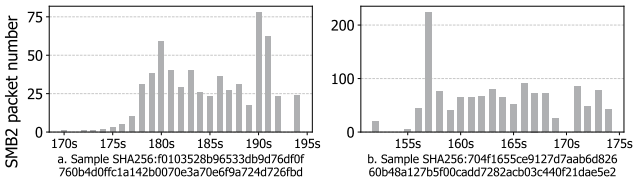


Figure 5: SMB2 frequency of two samples. Axis X refers to the execution time; Y refers to SMB2 requests per second.

As for the second method, 927 samples send an unusually high volume of TCP packets to port 5357 on other hosts, typically associated with Microsoft's WSDAPI. When displaying abnormal behavior, ransomware sends an average of 25 packets per second, with peak instances exceeding 175. Microsoft's WSDAPI is utilized to detect other hosts within the same network. This signifies ransomware's endeavor to facilitate lateral movement.

To conclude, these two methods help ransomware launch attacks on other hosts within the same network, facilitate targeted attacks against organizations, and cause widespread damage to all hosts within the internal network.

Finding 6: Ransomware tends to disrupt additional data within the same network, including damaging shared files and attempting intrusions. 3.36% of samples seek to exploit vulnerabilities in Microsoft's SMB service (port 445) for shared data disruption, and 12.66% attempt to discover other targets through WSDAPI (port 5357).

5 TOWARDS BETTER DEFENSE

Through our comprehensive examination of ransomware's data disruption phases, we have reached six critical findings about ransomware's procedures for targeting both user and system file spaces. These insights provide IT administrators and end-users valuable information to counter ransomware threats and ensure a clean and secure software application ecosystem. In this section, we propose ransomware detection strategies at different points during the three phases of data disruption. We also demonstrate the defense effect of the reconnaissance detector, tampering detector, and exfiltration detector. As shown in Figure 6, the ransomware survival rate (i.e., not being detected) gradually decreases to 0.88%, 0.61%, and 0% as each phase progresses.

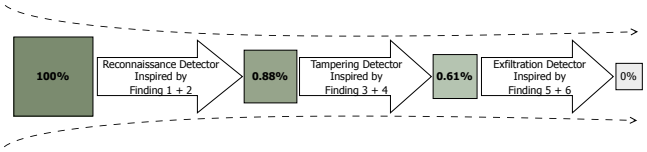


Figure 6: Survival rate after deploying proposed reconnaissance, tampering, and exfiltration detectors.

Reconnaissance Detector. Using libraries such as Windows Detours [54] to develop system-level sensitive behavior monitoring tools can help interrupt the encryption process at an early stage. According to Findings 1 and 2, process injection, registry modification, and file scanning are common procedures used by ransomware during the data reconnaissance phase. This suggests that we can develop a system-level monitoring tool capable of detecting ransomware intrusions by real-time monitoring sensitive behaviors in the system file space. By interrupting the subsequent file encryption process, we can safeguard the accessibility of critical data in the user file space. As shown in Table 7, we use process injection, registry modification, and user home folder scanning as examples of sensitive behaviors to demonstrate the defense effect achieved by their monitoring. "Encrypt-free rate" refers to the proportion of samples detected and blocked before the initiation of encryption, with no files encrypted. This demonstrates the effectiveness of the method. "Consumed time" refers to the average time required to detect ransomware, signifying the method's efficiency. We calculate the average values of all samples, and the detection of any of

these three behaviors is reported as ransomware. Different samples have distinct implementations, and to evade security checks, they may set sleep times, which introduce variations in the timing of sensitive behaviors, resulting in fluctuations in the defense effect. However, monitoring the combination of these behaviors can achieve an encrypt-free rate of 99.12% and a consumed time of only 2.15 seconds. These preliminary experimental results suggest that employing sensitive behavior monitoring tools to prevent file encryption is practical. By combining Findings 1 and 2, we can use sensitive behavior monitoring to detect potential ransomware attacks and protect data accessibility in the data reconnaissance phase. Only 0.88% of samples cannot be detected by this strategy.

Table 7: Defense results of sensitive behavior monitoring. Encrypt-free rate indicates the method effectiveness, and consumed time indicates the method efficiency.

Monitored Behavior	Avg. Encrypt-free Rate	Avg. Consumed Time
File Scanning	49.14%	47.00s
Registry Modification	57.62%	38.18s
Process Injection	98.13%	5.77s
Combination	**99.12%**	**2.15s**

Tampering Detector. Focusing on processes related to encryption operations and capabilities associated with system backup and recovery functions can help discover evidence of data manipulation. According to Findings 3 and 4, ransomware performs disruption actions against specific capabilities within the system file space, such as system backup and recovery, thereby rendering losses from encrypted files irreversible. On the other hand, the speed of encryption actions within the user file space is highly prioritized. Ransomware typically exhibits three encryption patterns, each with distinct speed and stealth characteristics. This suggests that by monitoring these ransomware encryption patterns, we can detect and terminate the encryption process, thereby reducing the number of files exposed to the risk of encryption and maximizing the preservation of data accessibility. As illustrated in Table 8, we present the defense effect of detecting encryption processes through three ransomware encryption patterns. "Time node" refers to the percentage of the total encryption process completed when ransomware is detected and blocked. Lower values imply fewer files have been encrypted, indicating the method's effectiveness. "Consumed time" refers to the average time it takes to detect ransomware, meaning the method's efficiency. Experimental results show that monitoring the three encryption patterns and then blocking corresponding processes keeps the proportion of encrypted files under 11%, and the time to detect and interrupt ransomware does not exceed 23 seconds. To conclude, combining Findings 3 and 4, even if ransomware has entered the highly damaging data tampering stage, this strategy can still rescue valuable data. After the data tampering phase, only 0.61% of samples cannot be detected.

Exfiltration Detector. Network traffic administration can help mitigate the impact of ransomware attacks within an organization's infrastructures. According to Findings 5 and 6, in the data exfiltration phase, anomalous DNS queries to C&C servers, unusual network connection requests, and employment of specific APIs or cloud-based file-sharing services may arise. By monitoring these

Table 8: Defense results of encryption process detecting. Time node indicates the method effectiveness, and consumed time indicates the method efficiency.

Encryption Pattern	Avg. Time Node	Avg. Consumed Time
Overwrite	15.97%	55.91s
Smash and Rewrite	12.14%	22.27s
Delete and Rewrite	10.44%	37.41s

indicators, we can prevent unauthorized data exfiltration from the user file space, as well as lateral movements to the system file space of other hosts. Specifically, when ransomware performs lateral movement, it tends to generate an unusually high volume of traffic, which targets specific service ports. This suggests that we can make preliminary judgments on attacks by implementing suitable network management practices, such as modifying default service ports and firewall rules, which are vulnerable to manipulation. As shown in Figure 7, we present the defense effect of managing network services. After changing the port of the SMB2 service from default 445 to a new one, i.e., 65511, traffic targeting at the host's SMB2 service is restricted and drops to zero. Experimental results show that this method can effectively prevent ransomware from propagating laterally by exploiting SMB2. To conclude, combining Findings 5 and 6, we can employ this method to stop ransomware from stealing users' data for double extortion and lateral movement for extensive impacts.

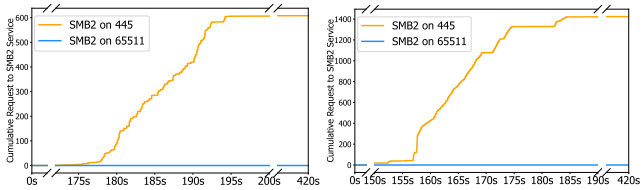


Figure 7: Defense result of changing SMB2 port from 445 to 65511. Axis X refers to the samples' execution time; Y refers to the cumulative count of requests to SMB2 service.

To further validate the usefulness of our findings, we implement Unveil* and Redemption* by adding the reconnaissance detector and the tampering detector to Unveil [23] and Redemption [55], respectively[2]. Considering the time and resources required to collect I/O request packets for Unveil and Redemption, we randomly select 100 samples from MarauderMap for this experiment. We also collect 100 benign executables to evaluate the false positive rate[3]. The benign applications chosen contain executables with similar behavior patterns to ransomware (e.g., archiver extractor) and other common office software (e.g., browser, text editor, video player). While a balanced dataset may not fully represent real-world scenarios, it is acceptable for assessing the detection rate and false positive rate in this experiment. As shown in Table 9, Unveil achieves a 59%

[2]We contacted the authors of both Unveil and Redemption, but neither of them provided tools or datasets, so we implemented tools from scratch following their papers.
[3]Resources for this enhancement experiment are available at https://github.com/m1-llie/MarauderMap-code.

detection rate, Redemption achieves a 31% detection rate, while Unveil* and Redemption* both achieve a 100% detection rate.

Table 9: Comparison results of the defense effect and false positive rates between Unveil, Redemption and their corresponding enhancements.

Comparison	Detection Rate	Increase	False Positive	Increase
	Ransomware Samples		Benign Executables	
Redemption	31%	69%	1%	0%
Redemption*	100%		1%	
Unveil	59%	41%	0%	0%
Unveil*	100%		0%	

This suggests that the capability of existing detection tools is greatly enhanced by applying our findings, with a 41%-69% increase in ransomware detection rate and no additional false positives. To further explain, Unveil monitors whether high-entropy buffers are written to files and whether specific I/O sequence patterns indicative of ransomware are present. Redemption assigns weights to several I/O related features and then calculates the final score to determine whether the program exceeds the threshold. These rule-based detection tools with limited inspection perspectives can significantly benefit from our findings, as including other critical observations, such as sensitive behaviors and encryption process patterns, can boost the defense performance of Unveil* and Redemption*. Moreover, our research opens up new avenues for ransomware defense. For example, our exploration into process injection motivates researchers to focus more on the security of benign processes and develop more behavior monitoring solutions; the information on how ransomware employs encryption algorithms can power future research on cryptographic detection.

6 DISCUSSION

Ethic Considerations. Ethics is a top priority when collecting ransomware samples and conducting experiments. Firstly, we chose legal and trustworthy sources for sample collection and transparently disclosed the sample sources in our paper. This ensures compliance with ethical norms and enhances the credibility of our research findings. Moreover, all experiments were conducted in the self-built testbed, isolated from external devices, ensuring no real-world computer equipment or end-users were affected. We believe the experiment setup poses no ethical issues and is sufficient to investigate data disruption procedures.

Threats to Validity. The main external threat to validity is the potential bias introduced by the distribution of ransomware families within our collected dataset, which may not accurately reflect the family distribution in the wild. This unequal distribution of samples among families may lead to biased analyses. We have sourced samples from various origins to mitigate this concern and persistently updated our dataset to ensure a more realistic representation. The top three most prevalent ransomware families in our dataset align with industry reports [56], suggesting that they reflect real-world attack patterns.

The first internal threat to validity lies in the anti-VM and anti-debug configuration of some ransomware. Due to our testbed establishment approach, we cannot run malicious code possessing such self-protection capabilities. This portion of samples, which was identified and excluded during the test running phase of dataset construction, constituted only 8.75% of the initial dataset. The second internal threat to validity is the running timeout settings during sample execution. The timeout could affect the completeness of the analysis. To mitigate this issue, we selected a 6-minute timeout setting based on the number of files within our testbed and the behavior of the samples. We found that 96.09% of our dataset samples completed their execution within the period.

Limitation. This paper does not cover the analysis of ransomware attacks on other operating systems, such as Linux, macOS, and mobile OSes. While the incidence and samples of these attacks are currently not too numerous (3.75% of the initial dataset we collected), they are escalating and can be further studied. Moreover, the readiness of defense strategies to counter the evolution of ransomware variants requires further investigation.

7 CONCLUSION

In this paper, we systematically analyze how ransomware disrupts the data accessibility of compromised hosts. We construct a novel real-world dataset MarauderMap and then gather runtime logs with a self-built testbed. We analyze procedures ransomware employs from the perspective of three data disruption phases and two kinds of file space, and further present insights and defense strategies. Finally, guided by our findings, we realize better detection and defense strategies to mitigate future ransomware attacks.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback and suggestions. This research is sponsored in part by the National Key Research and Development Project (No. 2022YFB3104000) and NSFC Program (No. 62022046, 92167101, 62021002).

REFERENCES

[1] BlackFog. The state of ransomware in 2023. https://www.blackfog.com/the-state-of-ransomware-in-2023/, 2023. (Online; accessed on Nov 26, 2023).
[2] The New York Times. Cyber attack suspected in german woman's death. https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomeware-death.html, 2020. (Online; accessed on Nov 26, 2023).
[3] NBC News. Baby died because of ransomware attack on hospital, suit says. https://www.nbcnews.com/news/baby-died-due-ransomware-attack-hospital-suit-claims-rcna2465, 2021. (Online; accessed on Nov 26, 2023).
[4] The Verge. Hospitals say cyberattacks increase death rates and delay patient care. https://www.theverge.com/2021/9/27/22696097/hospital-ransomware-cyberattack-death-rates-patients, 2021. (Online; accessed on Nov 26, 2023).
[5] Shou-Ching Hsiao and Da-Yu Kao. The static analysis of wannacry ransomware. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pages 153–158, 2018.
[6] Deepti Vidyarthi, CRS Kumar, Subrata Rakshit, and Shailesh Chansarkar. Static malware analysis to identify ransomware properties. *International Journal of Computer Science Issues (IJCSI)*, 16(3):10–17, 2019.
[7] Farnoush Manavi and Ali Hamzeh. Static detection of ransomware using lstm network and pe header. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–5. IEEE, 2021.
[8] Ahmad O Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O'Kane. A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE access*, 7:47053–47067, 2019.
[9] Rusydi Umar, Imam Riadi, and Ridho Surya Kusuma. Analysis of conti ransomware attack on computer network with live forensic method. *IJID (International Journal on Informatics for Development)*, 10(1):53–61, 2021.
[10] Emily Caroscio, Jack Paul, John Murray, and Suman Bhunia. Analyzing the ransomware attack on dc metropolitan police department by babuk. In *2022 IEEE International Systems Conference*, pages 1–8. IEEE, 2022.

[11] Ziya Alper Genç, Gabriele Lenzini, and Peter YA Ryan. No random, no ransom: a key to stop cryptographic ransomware. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 15th International Conference, DIMVA 2018, Saclay, France, June 28–29, 2018, Proceedings 15*, pages 234–255. Springer, 2018.

[12] Fei Tang, Boyang Ma, Jinku Li, Fengwei Zhang, Jipeng Su, and Jianfeng Ma. Ransomspector: An introspection-based approach to detect crypto ransomware. *Computers & Security*, 97:101997, 2020.

[13] Gaddisa Olani Ganfure, Chun-Feng Wu, Yuan-Hao Chang, and Wei-Kuan Shih. Rtrap: Trapping and containing ransomware with machine learning. *IEEE Transactions on Information Forensics and Security*, 18:1433–1448, 2023.

[14] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.

[15] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd annual conference on computer security applications*, pages 336–347, 2016.

[16] Firoz Khan, Cornelius Ncube, Lakshmana Kumar Ramasamy, Seifedine Kadry, and Yunyoung Nam. A digital dna sequencing engine for ransomware detection using machine learning. *IEEE Access*, 8:119710–119719, 2020.

[17] Manabu Hirano, Ryo Hodota, and Ryotaro Kobayashi. Ransap: An open dataset of ransomware storage access patterns for training machine learning models. *Forensic Science International: Digital Investigation*, 40:301314, 2022.

[18] Masoudeh Keshavarzi and Hamid Reza Ghaffary. I2ce3: A dedicated and separated attack chain for ransomware offenses as the most infamous cyber extortion. *Computer Science Review*, 36:100233, 2020.

[19] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.

[20] Harun Oz, Ahmet Aris, Albert Levi, and A Selcuk Uluagac. A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[21] Fabrizio Cicala and Elisa Bertino. Analysis of encryption key generation in modern crypto ransomware. *IEEE Transactions on Dependable and Secure Computing*, 19(2):1239–1253, 2020.

[22] Chijin Zhou, Lihua Guo, Yiwei Hou, Zhenya Ma, Quan Zhang, Mingzhe Wang, Zhe Liu, and Yu Jiang. Limits of i/o based ransomware detection: An imitation based attack. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2584–2601. IEEE Computer Society, 2023.

[23] Amin Kharraz, Sajjad Arshad, Collin Mulliner, William K. Robertson, and Engin Kirda. UNVEIL: A large-scale, automated approach to detecting ransomware. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 757–772. USENIX Association, 2016.

[24] Zhen Li and Qi Liao. Preventive portfolio against data-selling ransomware—a game theory of encryption and deception. *Computers & Security*, 116:102644, 2022.

[25] Yiwen Xu, Zijing Yin, Yiwei Hou, Jianzhong Liu, and Yu Jiang. Midas: safeguarding iot devices against malware via real-time behavior auditing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4373–4384, 2022.

[26] Kul Prasad Subedi, Daya Ram Budhathoki, and Dipankar Dasgupta. Forensic analysis of ransomware families using static and dynamic analysis. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 180–185. IEEE, 2018.

[27] M Satheesh Kumar, Jalel Ben-Othman, and KG Srinivasagan. An investigation on wannacry ransomware and its detection. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2018.

[28] Craig Beaman, Ashley Barkworth, Toluwalope David Akande, Saqib Hakak, and Muhammad Khurram Khan. Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & security*, 111:102490, 2021.

[29] Daniel Morato, Eduardo Berrueta, Eduardo Magaña, and Mikel Izal. Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications*, 124:14–32, 2018.

[30] SH Kok, Azween Abdullah, NZ Jhanjhi, and Mahadevan Supramaniam. Prevention of crypto-ransomware using a pre-encryption detection algorithm. *Computers*, 8(4):79, 2019.

[31] Seong Il Bae, Gyu Bin Lee, and Eul Gyu Im. Ransomware detection using machine learning algorithms. *Concurrency and Computation: Practice and Experience*, 32(18):e5422, 2020.

[32] Jian Huang, Jun Xu, Xinyu Xing, Peng Liu, and Moinuddin K Qureshi. Flashguard: Leveraging intrinsic flash properties to defend against encryption ransomware. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 2231–2244, 2017.

[33] Sungha Baek, Youngdon Jung, David Mohaisen, Sungjin Lee, and Daehun Nyang. Ssd-assisted ransomware detection and data recovery techniques. *IEEE Transactions on Computers*, 70(10):1762–1776, 2020.

[34] Jisung Park, Youngdon Jung, Jonghoon Won, Minji Kang, Sungjin Lee, and Jihong Kim. Ransomblocker: A low-overhead ransomware-proof ssd. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[35] Benjamin Reidys, Peng Liu, and Jian Huang. Rssd: Defend against ransomware with hardware-isolated network-storage codesign and post-attack analysis. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 726–739, 2022.

[36] Gaddisa Olani Ganfure, Chun-Feng Wu, Yuan-Hao Chang, and Wei-Kuan Shih. Deepware: Imaging performance counters with deep learning to detect ransomware. *IEEE Transactions on Computers*, 72(3):600–613, 2022.

[37] Boyang Ma, Yilin Yang, Jinku Li, Fengwei Zhang, Wenbo Shen, Yajin Zhou, and Jianfeng Ma. Travelling the hypervisor and ssd: A tag-based approach against crypto ransomware with fine-grained data recovery. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 341–355, 2023.

[38] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 618–631. IEEE, 2018.

[39] Alena Yuryna Connolly and Hervé Borrion. Reducing ransomware crime: analysis of victims' payment decisions. *Computers & Security*, 119:102760, 2022.

[40] Routa Moussaileb, Nora Cuppens, Jean-Louis Lanet, and Hélène Le Bouder. A survey on windows-based ransomware taxonomy and detection mechanisms. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[41] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security*, 74:144–166, 2018.

[42] Virustotal. https://www.virustotal.com/gui/contact-us/premium-services, 2023. (Online; accessed on Nov 26, 2023).

[43] Vx vault. http://vxvault.net/ViriList.php, 2023. (Online; accessed on Nov 26, 2023).

[44] tutorialjinni. https://www.tutorialjinni.com/download-free-malware-samples.htm, 2023. (Online; accessed on Nov 26, 2023).

[45] Inthewild. https://samples.vx-underground.org/samples/Blocks/InTheWild%20Collection/, 2023. (Online; accessed on Nov 26, 2023).

[46] Bazaar. https://bazaar.abuse.ch/browse/, 2023. (Online; accessed on Nov 26, 2023).

[47] thezoo. https://thezoo.morirt.com/, 2023. (Online; accessed on Nov 26, 2023).

[48] vx-underground. https://www.vx-underground.org/malware.html, 2023. (Online; accessed on Nov 26, 2023).

[49] Silvia Sebastián and Juan Caballero. Avclass2: Massive malware tag extraction from av labels. In *Annual Computer Security Applications Conference*, pages 42–53, 2020.

[50] Kevin O'Reilly. Malware configuration and payload extraction. https://github.com/kevoreilly/CAPEv2, 2023. (Online; accessed on Nov 26, 2023).

[51] Filezilla. https://filezilla-project.org/, 2023. (Online; accessed on Nov 26, 2023).

[52] WinSCP.net. Winscp 6.1 download. https://winscp.net/eng/download.php, 2023. (Online; accessed on Nov 26, 2023).

[53] Tor project. https://www.torproject.org/, 2023. (Online; accessed on Nov 26, 2023).

[54] Microsoft. https://github.com/microsoft/Detours, 2023. (Online; accessed on Nov 26, 2023).

[55] Amin Kharraz and Engin Kirda. Redemption: Real-time protection against ransomware at end-hosts. In *Research in Attacks, Intrusions, and Defenses: 20th International Symposium, RAID 2017, Atlanta, GA, USA, September 18–20, 2017, Proceedings*, pages 98–119. Springer, 2017.

[56] Diana Selck-Paulsson and Charl van der Walt. Crime time | rethinking ransomware and how to disrupt it. https://www.youtube.com/watch?v=gPJqtK-qP2E&list=PLtgaAEEmVe6AGQj2LhA4UnN0XolmeYw9_&index=3, 2022. (Online; accessed on Nov 26, 2023).