

Crypto-Ransomware and Their Defenses: In-Depth Behavioral Characterization, Discussion of Deployability, and New Insights

WENJIA SONG, Virginia Tech, Blacksburg, United States

SANJULA KARANAM, Virginia Tech, Blacksburg, United States

YA XIAO, Computer Science, Virginia Tech, Blacksburg, United States

JINGYUAN QI, Virginia Tech, Blacksburg, United States

NATHAN DAUTENHAHN, Rice University, Houston, United States

NA MENG, Virginia Tech, Blacksburg, United States

ELENA FERRARI, University of Insubria, Varese, Italy

DANFENG YAO, Virginia Tech, Blacksburg, United States

Crypto-ransomware has caused an unprecedented scope of impact in recent years, with an evolving level of sophistication. An extensive range of studies have been conducted on defending against ransomware and reviewing the efficacy of various protections. However, for practical defenses, deployability holds equal significance as detection accuracy. Therefore, in this study, we review 117 published ransomware defense works, categorize them by the level they are implemented, and discuss the deployability. API-based solutions are easy to deploy, and most existing works focus on machine learning-based classification. To provide more insights, we quantitatively characterize the runtime behaviors of real-world ransomware samples. Based on our experimental findings, we present a possible future detection direction with our consistency analysis and API-contrast-based refinement. Moreover, we experimentally evaluate various commercial defenses and identify the security gaps. Our findings help the field understand the deployability of ransomware defenses and create more effective, practical solutions.

CCS Concepts: • **Security and privacy** → **Systems security; Malware and its mitigation**;

Additional Key Words and Phrases: Ransomware, data security

ACM Reference Format:

Wenjia Song, Sanjula Karanam, Ya Xiao, Jingyuan Qi, Nathan Dautenhahn, Na Meng, Elena Ferrari, and Danfeng Yao. 2026. Crypto-Ransomware and Their Defenses: In-Depth Behavioral Characterization, Discussion of Deployability, and New Insights. *ACM Comput. Surv.* 58, 10, Article 256 (April 2026), 40 pages. <https://doi.org/10.1145/3798092>

This work has been supported by the National Science Foundation under Grant number SaTC-TTP-2343742 and the Office of Naval Research under Grant number N00014-22-1-2057.

Authors' Contact Information: Wenjia Song (corresponding author), Virginia Tech, Blacksburg, Virginia, USA; e-mail: wenjia7@vt.edu; Sanjula Karanam, Virginia Tech, Blacksburg, Virginia, USA; e-mail: sanjula@vt.edu; Ya Xiao, Computer Science, Virginia Tech, Blacksburg, Virginia, USA; e-mail: yax99@vt.edu; Jingyuan Qi, Virginia Tech, Blacksburg, Virginia, USA; e-mail: jingyq1@vt.edu; Nathan Dautenhahn, Rice University, Houston, Texas, USA; e-mail: ndd@rice.edu; Na Meng, Virginia Tech, Blacksburg, Virginia, USA; e-mail: nm8247@vt.edu; Elena Ferrari, University of Insubria, Varese, Lombardy, Italy; e-mail: elena.ferrari@uninsubria.it; Danfeng Yao, Virginia Tech, Blacksburg, Virginia, USA; e-mail: danfeng@vt.edu.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 0360-0300/2026/04-ART256

<https://doi.org/10.1145/3798092>

1 Introduction

Crypto-ransomware extorts money from victims by encrypting their files. It first appeared in 1989 and has had a resurgence recently. In 2017, WannaCry hit around 230,000 computers across 150 countries, causing a loss of \$4 billion [85]. The notorious Colonial Pipeline hack affected nearly half of the U.S. east coast gas supply, and roughly \$5 million was paid for recovery in May 2021 [157]. 37% of organizations reported being attacked by ransomware in 2021, and the average cost of recovering was \$1.85 million [150]. In 2023, MGM Resorts, a Las Vegas casino, lost \$100 million as a result of a ransomware attack [54]. Even worse, on average, only 65% of the data was recovered after the ransom payment [150]. One possible reason for this low recovery rate is file corruption.

Many efforts have been made to defend against ransomware attacks. However, in successful ransomware attacks, 77% of victims are running up-to-date endpoint protection, implying inadequacy in current solutions in practice [151]. There have also been academic solutions proposed to detect ransomware threats. Monitoring the file system is a widely used approach [55, 71, 90, 116, 138, 155]. Hardware performance [29, 124], API call occurrence [18, 52, 80], and network activities [16, 17, 42] could also reveal the malicious purpose of a program. Survey works have also been published to summarize the knowledge and effectiveness of defenses in recent years [114, 122, 127].

Despite these research advances, there seems to be limited literature focusing on the deployability of existing ransomware defenses. While effectiveness is crucial for a defense system, the significant amount of deployment effort could be a reason that stops the state-of-the-art detection system from turning into prevalent endpoint protection solutions. Existing file system-based solutions (e.g., UNVEIL [90] and ShieldFS [55]) focus on low-level file system activities, requiring kernel-level modifications. API-based approaches could be a deployable alternative, taking advantage of fine-grained program behavior information. API-based solutions monitor the target's behaviors at the program level and require fewer system modifications. Current API-based detection works [18, 52] mostly rely on machine learning classification. In-depth analyses of ransomware-specific execution patterns could complement and strengthen the detection. To close this gap, we aim at answering the following important research questions:

RQ1: What are the existing ransomware defenses? What are the more deployable approaches? (Section 3)

RQ2: For highly deployable commercial defenses, which ransomware behaviors trigger detection? What are the security gaps? (Section 4)

RQ3: What are the quantitative ransomware API invocation behaviors? How do they systematically compare with benign software? How to quantify the unique ransomware API usage patterns for detection? (Sections 5 and 6)

The objective of this work is to comprehensively review various existing ransomware defenses and discuss their deployability, provide experimental insights on the unique ransomware API invocation patterns, and demonstrate a deployable detection direction by quantifying program API usage. Specifically, we review 117 published ransomware defense works, up to 2023, and categorize them by the level they are implemented. We discuss the deployability of each category. Furthermore, we experimentally evaluate three types of commercial tools, including 6 crypto decryptors, around 70 malware scanners, and 8 antivirus software. Additionally, we execute and analyze 2 sets of real-world ransomware, with over 400 samples in total.

We summarize our experimental findings below.

- **API-usage profiling.** We quantitatively analyze the behaviors of ransomware samples through two sets of experiments. We first manually inspect 54 real-world ransomware samples from 35 families, including the notorious WannaCry, Sodinokibi, Babuk, and the most active ransomware families in 2021, LockBit, Mespinoza, and Hive, with a focus on

encryption activities. We find that ransomware has a distinct file access behavior pattern during execution. We further collect the occurrence frequency of 288 Windows APIs from 348 ransomware samples. We report differences in API occurrences and invocation frequencies between benign and ransomware executions, which is beneficial for improving detection accuracy. We leverage these for computing the API contrast score. Our findings extend existing knowledge on ransomware's API behaviors by highlighting statistical behavioral distinctions.

- **API-based classification.** Based on our observation of API usage, we discuss a possible classification mechanism, consisting of consistency analysis and API usage contrast refinement operations. In consistency analysis, we use multiple mathematical methods to capture the unique ransomware execution features, focusing on the fundamental encryption nature. In refinement, we further examine the positive cases by an API contrast score to filter wrongly classified cases. We further provide an in-depth case analysis of API usage in contrast with the attack context. We carry out a feasibility assessment of our classification approach against 29 sets of execution traces. The results show that our consistency analysis effectively identifies all malicious executions. Our findings could help inspire future detection works in this field.
- **Evaluation of commercial tools.** We extensively evaluate three types of commercial defenses. Through our experiments, we find that the success rate of commercial decryptors is low (1 success out of 6), suffering from low generality across variants of ransomware. Anti-virus software detects only generic malicious behaviors, being insensitive to core ransomware encryption. Malware scanners use signature-based detection and miss unknown or obfuscated samples. The observations reveal that behavioral detection is necessary, and our new API findings could help strengthen ransomware-specific protection.

The rest of the article is organized as follows. **Section 2** reviews the related survey works. **Section 3** categorizes existing ransomware defenses and discusses their deployability. **Section 4** evaluates various commercial tools. **Section 5** reports our experimental findings on ransomware API-based behavioral characteristics, with systematic comparison to benign software. **Section 6** shows the results of the feasibility test of API-based classification. **Section 7** further discusses our insights and limitations of our work. **Section 8** concludes the work.

2 Related Work

Many survey works have been done to summarize the knowledge of ransomware and its defenses from different aspects. We review existing surveys in this section and discuss how the focus of our work differs.

Surveys of ransomware evolution. Ransomware first appeared in 1989 as PC Cyborg and has been evolving for several decades. Techniques utilized in each attack stage have exhibited new features and increased complexity. Research has been done to study the evolution [38, 60, 67, 109, 127, 133, 135, 153, 170]. Specifically, Oz et al. [127] summarized the evolution of ransomware families from 1989 to 2020. Zimba et al. [170] reviewed the transformation of ransomware attack structure in the **Internet-of-Things (IoT)** system. Maniath et al. [109] discussed the change of encryption key management strategies used by ransomware. Alwashali et al. [22] conducted a survey on **Ransomware-as-a-Service (RaaS)**, a newly emerged business selling malware to anyone who wants to launch an attack. A recent article also provides an overview of the evolving ecosystem of ransomware [110].

Surveys of ransomware on different platforms. Ransomware targets various systems and platforms. Comprehensive studies have been done on ransomware on different platforms, such as **personal computers (PCs)** [25, 38, 39, 58, 67, 70, 70, 87, 93, 94, 106, 109, 130, 130, 147], mobile devices [1, 39, 58, 87, 106, 147], and **cyber-physical systems (CPS)** [37, 58, 79, 87, 130]. Some less

common platform targeted by ransomware includes cloud [81] and wearable devices [12]. Besides, works have also been done analyzing ransomware infecting specific operating systems, such as Windows [27, 122, 123] and Android [25, 60, 118, 125].

Surveys of ransomware taxonomy. Ransomware could be categorized by various features. By attack methods, two main types are identified, namely locker-ransomware and crypto-ransomware [12, 127]. Besides these two types, Razaulla et al. [133] further included scareware and leakware as categories of ransomware. By targeted victims, they divide ransomware into consumer ransomware and organization ransomware [12, 127]. As mentioned above, by platform, ransomware could be divided into PC ransomware, mobile ransomware, cloud ransomware, IoT ransomware, and ransomwear. Bajpai et al. [34] also introduced a taxonomy based on the key management strategies used by ransomware.

Surveys of ransomware defenses. The detection and mitigation is also an important field of ransomware study. Extensive research has been done reviewing a wide range of defensive strategies [12, 15, 19, 25, 35, 36, 38, 39, 45, 60, 70, 79, 93, 94, 98, 106, 109, 114, 118, 122, 146]. While most of the works discuss all types of defensive strategies, some of them have a specific focus. Particularly, Cen et al. [45] reviewed detections targeting the early attack stage. Begovic et al. [36] and Larsen et al. [98] analyzed encryption stage detection methods. Machine learning and deep learning solutions have also been studied [1, 25, 98]. McIntosh et al. [114] analyzed various existing ransomware mitigation techniques and emphasized the need for unified evaluation metrics.

Our work. Our work's focus is on the deployability of ransomware defense. While effectiveness is one of the most important metrics in evaluating a defense strategy and the focus of the majority works in this field, the deployability gap hinders many solutions from being prevalent in the real world. Therefore, to understand the deployability of different methods, we analyze and categorize existing solutions by the layer where they are deployed in the system and discuss the challenges of deploying a solution at each level. Beyond reviewing existing work, we further provide experimental insights on the evaluation of commercial ransomware defenses and dynamic features of core crypto-ransomware behaviors, which could help the future development of deployable ransomware detection methods.

3 Existing Ransomware Defenses and Their Deployability

In this section, we discuss existing ransomware defenses concentrating on their deployability. A summary is shown in Table 1. Generally, the lower the level in the system a solution is implemented at, the more difficult it is to deploy it on a new system or device. One common reason is that lower-level solutions require more dependency and manual configurations.

3.1 Defenses at User Mode Level

Solutions implemented at the user mode level are easy to deploy as little system modifications are required. We discuss approaches that analyze or monitor various features in this section.

Static analysis. Static analysis extracts useful information from the target program without execution. This method is fast and efficient as no environment setup is required. A range of static information could be useful for making security decisions. Opcode sequences are found to be an effective feature for machine learning models to perform classification between ransomware and benign programs [66, 84, 166, 167]. Bytecode sequences [51, 73, 89, 134, 158] and APIs [20, 23, 26, 63, 66, 69, 72, 107, 115, 152] are some other useful static features. Besides code, other text and files could also reveal the purpose of a program. Specifically, Android permission file is widely used in Android ransomware detection [6, 21, 23, 24, 26, 69]. Detecting potential threatening and ransom notes in strings could also help identify malicious programs [6, 26, 69, 72, 144, 152]. Furthermore, Cuzzocrea et al. [57] and Yamany et al. [163] utilize the structural entropy and code

Table 1. Summary of Ransomware Defenses Based on the System Level They are Implemented at

	Methods	Features	Existing Solutions	Advantage
User Mode	Antivirus Software			
	Static Analysis	Opcode	[167], [166], [84], [66]	Fast and efficient. No execution is required.
		Bytecode	[51], [89], [134], [158], [73], [163]	
		API	[20], [23], [63], [107], [115], [152], [66], [69], [72], [26]	
		Text and Other Files	[23], [24], [57], [115], [152], [21], [26], [69], [72], [6], [86], [144], [158], [3]	
		Program Header	[86], [144], [158], [172], [28]	
		Domain Name	[131]	
	User File Configuration or Analysis	Honey Pot	[86], [144], [161], [172], [119], [71], [121], [99]	Only limited monitoring is required.
		File Entropy	[83], [161], [46], [100]	
		Data Decryption	[92], [82], [102]	
	Dynamic API Monitoring	API Occurrence or Sequence	[13], [66], [69], [72], [52], [96], [18], [80], [154], [49], [142], [7], [8], [76], [104], [65], [171], [11], [32], [103], [108], [139], [143], [10], [95], [159], [126], [169], [6], [28], [68]	Easy to deploy. Less or no system modification or dependency is required.
			File Access API	
			Crypto API	
			Registry-related API	
		Network API	[72], [86], [3], [47], [75]	
			Clipboard API	
			Process-related API	
	Other Dynamic Behavior Analysis	Query Sequences	[141]	Effectively captures program execution patterns.
		Foreground Analysis	[46]	
		Memory Dumps	[53], [6], [104]	
		System Logs	[137]	
	Network Packet Monitoring	Other	[171], [172], [66]	
		Number of Packets	[66], [158], [104], [17], [117]	
		Text and Packet Information	[26], [86], [161], [16], [17], [42], [56], [64], [117], [162], [4], [148]	
		HTTP Requests	[40], [41]	
Kernel Mode	File System I/O Analysis	IRP	[144], [138], [90], [155], [55], [116], [5], [132], [30], [74], [91], [128], [121], [31], [61]	Difficult to evade.
		Buffer Entropy	[138]	
	Process Analysis	Process Access Control	[112]	
	Hardware Performance Monitoring	Counter	[66], [124], [29], [156], [88], [132], [128], [14], [44], [113]	
Hardware	Firmware Modification		[33], [78], [136], [160]	
External Devices	External Backup		[43]	

Hybrid methods may appear in multiple categories. Generally, the lower the level a solution is implemented at, the more difficult to deploy it to a new system or device.

n-grams to determine if a sample is similar to a ransomware sample. Program header information, such as META-INF for Android applications and import table for executables, could also aid the detection [28, 86, 144, 158, 172]. YARA¹ is a widely used tool that could be utilized to define static rules for malware detection. Medhat et al. [115] used the combination of matched YARA rules of a sample to detect ransomware.

¹<https://yara.readthedocs.io/en/stable/index.html>

Dynamic analysis. Dynamic analysis collects traces during program execution and could reveal program behaviors that are obfuscated in static files. Dynamic analysis usually happens at the program level, e.g., program instrumentation, requiring relatively light effort to deploy. Programs exhibit a variety of behaviors during execution and many of them have been proven to be valuable for ransomware detection. One of the most popular approaches is to use API call occurrences or sequences as features and classify samples using machine learning models [6–8, 10, 11, 13, 18, 28, 32, 49, 52, 65, 66, 66, 68, 69, 72, 76, 80, 95, 96, 103, 104, 108, 126, 139, 142, 143, 154, 159, 169, 171]. Beyond leveraging recorded traces directly, Takeuchi et al. [154] used API call n-grams to detect ransomware execution. Chen et al. [49] constructed a call graph to capture the relationship between calls. Coglio et al. [52] selected the most informative APIs as input features. While most works leverage API traces at all execution stages, classification based on only early-stage (i.e., pre-encryption) calls has also been studied [52, 96]. Approaches that capture program behaviors with a focus on a single or a few API categories, such as file [3, 48, 75, 77, 111, 158], crypto [3, 9, 101, 129, 144, 158], network [48, 161], and registry [3, 47, 72, 75, 86], are also effective. Recovery strategies that try to restore the decryption key by hooking crypto-related APIs have been proposed as well [50, 97, 165]. API instrumentation (a.k.a., API hooking) could be easily done by utilizing tools such as Intel PIN.² While instrumentation may result in notable runtime overhead during deployment, strategies like selective monitoring can serve to alleviate this issue. We discuss more in Section 7. The balance between deployment effort and the amount of valuable data attainable from API monitoring makes API-based detection a promising direction.

Other behavioral features. Moreover, besides API monitoring, Sendner et al. [141] utilized query sequences to detect database ransomware. Chen et al. [46] studied the foreground behaviors of programs. Cohen et al. [53] analyzed the program memory dump. Roy et al. [137] used system logs to identify malicious ransomware behaviors. Some research works also define customized behavioral rules to detect ransomware [158, 171, 172].

3.2 Defenses at Kernel Mode Level

Kernel-level solutions monitor program behaviors at a lower level and could be more difficult for malware to evade, as obfuscation usually happens at a higher-level. However, system version dependency and modifications of the kernel drivers make the deployment of such solutions to new devices very challenging. Many detection researches have been done analyzing the **I/O Request Packets (IRP)** to detect malicious file access patterns [5, 30, 31, 55, 61, 74, 90, 91, 116, 121, 128, 132, 138, 144, 155]. Encrypted data usually has a higher entropy. Therefore, the entropy of the reading and writing buffer is also a useful feature [138]. McIntosh et al. [112] proposed Staged Event-Driven Access Control to limit processes' access during execution. Examining hardware counters to identify intensive computation for encryption behaviors [14, 29, 44, 66, 88, 113, 124, 128, 132, 156].

3.3 Defenses at Hardware Level

Solid State Drives (SSDs) have the property of out-of-place writing, which could help recover ransomware encrypted data. Solutions [33, 78, 136, 160] built into SSD firmware have been developed to recover data when a ransomware attack is detected. While adding an extra layer of mitigating the risk of data loss, such built-in solutions require acquiring certain hardware.

4 Evaluation of Commercial Defense

Commercial tools are in the format of download and run executables. Minimum effort is required to deploy such end-point protection. In this section, we report the findings from our experimental

²<https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html>

Table 2. List of Decryptors We Test

Ransomware family	SHA-256	File extension	Decryptor provider	Requirements	Success	Notes
Alcatraz	9185	.alcatraz	Avast	A pair of files	Yes	
Babuk	eb18	.doydo	Avast		No	
Jigsaw	9074	.v316	Trend Micro		No	
Ragnarok	db8b	.ragnarok_cry	Emsisoft	Ransom note	No	Ransom note file is not supported by decryptor
Sodinokibi	fd16	.031j2adrq8	BitDefender	Internet access	No	
Xorist	fb54	.locks	Trend Micro	A pair of files	No	Not able to proceed after adding file pair

SHA-256 column shows the first 4 digits of the ransomware sample that generates the encrypted files. File extension is the extension appended by ransomware after encryption. Notes explain the reason if we are not able to get the decryption step. The Trend Micro tool is designed for decrypting files infected by multiple families. Different families can be selected before decryption.

evaluation of decryptors, antivirus software, and malware scanners and answer **RQ2**. Our observations show that there exist security gaps.

4.1 Experimental Setup

We describe the experimental setup for evaluating various types of commercial defense in this section.

Decryptor. For the ransomware samples we successfully run, we found decryptors for 6 of them on the NO MORE RANSOM website.³ The 6 decryptors are specifically designed for 6 distinct families (Table 2). Different decrypting tools require different information, such as uploading the ransom note or providing file pairs (i.e., both unencrypted and encrypted versions) for key cracking.

Commercial Antivirus Software. We tested eight different commercial antivirus software that provides protection against ransomware, namely Antivirus A (anonymized), Antivirus B (anonymized), Bitdefender, Malwarebytes, Kaspersky, McAfee, Norton, and 360 Total Security (Table 16 in the appendix). We anonymized antivirus tools A and B following their user terms and conditions. All 8 antivirus software are generic for all types of malware with ransomware detection feature. We conducted three series of experiments. We first evaluated them with 20 real ransomware samples from 20 families active from 2015 to 2020 (marked in Table 17 in the appendix). Each sample was compressed with the password “infected” at the beginning. We uncompressed the sample and if the antivirus had no reaction, we then executed it. The machine was disconnected from the internet during all real ransomware execution to prevent spreading. Because 360 Total Security performed poorly in the offline setting, we added an online setting for it, in which we only uncompressed the samples without executing them. Second, we used two publicly available ransomware simulators to test more attack scenarios. RanSim⁴ simulates 23 ransomware attack scenarios. Quickbuck⁵ simulates several typical ransomware behaviors. Lastly, we also used our own script to further test the behavioral detection because most antivirus tools block public simulators as malware, giving them no chance to execute. Our script simulates behaviors such as iterating and encrypting files, appending random or known ransomware file extensions, deleting volume shadow copies, and dropping ransom notes. Those ransomware-like behaviors can be run separately or together. The script is in Python 3.8. To mimic real ransomware, we statically link the crypto libraries used (PyAesCrypt⁶ and Cryptography⁷) so that no external dependencies are needed. All tests were

³<https://www.nomoreransom.org/en/decryption-tools.html>

⁴<https://www.knowbe4.com/ransomware-simulator>

⁵<https://github.com/NexttronSystems/ransomware-simulator>

⁶<https://github.com/marcobellaccini/pyAesCrypt>

⁷<https://github.com/pyca/cryptography>

run on a Windows 7 VM with 2 CPUs and 4096 MB memory. The machine was reversed to the initial image after each test.

Commercial Malware Scanner. We test over 70 commercial malware scanners available on VirusTotal⁸ using the 54 samples (Table 17 in the appendix). All scanners are general purpose malware scanners in a black-box manner (i.e., the algorithm used is unknown). For each sample we scanned, the number of available scanners slightly varies (Table 18 in the appendix). We tested the scanners in three settings. First, we used them to scan the plain executable file (.exe). Second, we scanned the compressed file with a simple password “infected”. This password is used as a convention for sharing malware samples. Lastly, we made a compressed file with a random, complex password (i.e., DDfA3WFxcPMUrsA) to test the scanners’ ability to detect obfuscated samples. We chose these settings because password obfuscation is a very low-cost way for the attacker to distribute malware samples.

4.2 Crypto Decryptors

Decryption is a ransomware-specific recovery strategy that aims to recover files without payment. Generally, it works by inspecting the encryption algorithm and inferring the key. Strategies include finding implementation flaws of encryption functions, brute forcing the key in a certain scope, and monitoring the key generation.

Among the six decryptors we test, only the decryptor for Alcatraz successfully recovers the files. The decryptor requires a pair of original and encrypted files for cracking the password and finds the password in seconds. To minimize the effect of randomness and confirm the effectiveness, we run the attack three times. For each attack, the decryptor crack the password in 12119, 3724, and 4879 tries, respectively. Alcatraz first appeared in 2016 and uses AES-256 with Base64 encoding for encryption. It is computationally infeasible to brute force the AES-256 key as the key space is 2^{256} [140]. Therefore, we suspect that there might be a design flaw in Alcatraz’s encryption function that the decryptor uses as a shortcut to search for the correct key. This low success rate of recovery further necessitates early detection.

4.3 Commercial Antivirus Software

We test the antivirus software with real ransomware samples and simulations of ransomware behaviors. First, we report their effectiveness and analyze the features (static or behavioral) used for detection based on their reactions. The results are summarized in Table 3.

Detection before execution. All eight commercial tools detect and block the majority of threats before they start execution and make any modifications to the system. They do so by moving the executable file to quarantine or prohibiting it from execution. It is likely that they run a signature matching similar to the malware scanners. Antivirus tools A, B, and 360 (online) detect all malware executables immediately upon decompression, suggesting that they have up-to-date malware databases. Bitdefender, Malwarebytes, Kaspersky, McAfee, and Norton block 19, 17, 12, 16, and 18 samples, respectively, before execution (Table 3).

Detection during execution. We execute a sample to test behavioral detection if it is not blocked by scanning. Bitdefender and Malwarebytes catch all threats they missed before execution and completely roll back the malicious behaviors, with no signs of infection left in the system. Kaspersky successfully detects all seven attacks missed earlier and revokes five of them. In the other two cases, a few infected files are left in the system but all original data is accessible. This shows that antivirus software has real-time behavioral detection to catch ongoing threats.

⁸<https://www.virustotal.com/gui/>

Table 3. Evaluation Results of Eight Commercial Antivirus Software

Antivirus software	Real ransomware samples				Ransomware simulators		Our script			
	# detection before execution	# detection during execution	# completely rolled back	# failed	RanSim	Quick-Buck	File traverse	File encryption	Ransom note dropping	Volume shadow deletion
Antivirus A	20				Blocked	Blocked	–	–	–	–
Antivirus B	20				Blocked	Blocked	–	–	–	–
Bitdefender	19	1	1		Passed all scenarios	Blocked	–	–	–	Detected
Malwarebytes	17	3	3		Blocked	Blocked	–	–	–	–
Kaspersky	12	7	5	1	Blocked	Blocked	–	–	–	–
McAfee	16			4	Blocked	Blocked	–	–	–	–
Norton	18			2	Blocked	Detected marco simulation	–	–	–	–
360 Total Security	offline	3		17	Failed all scenarios	–	–	–	–	–
	online	20			Blocked	Blocked	–	–	–	Detected

detection before execution: number of samples that are blocked before the malware executes. # detection during execution: number of samples that execute but get terminated during execution. # completely rolled back: among the attacks that started, the number of those being completely rolled back to the state with no trace of infected files. # failed: number of attacks the tool failed to generate alerts or make a reaction. Empty means zero. Ransomware simulators are software that mimics ransomware behaviors for security evaluation purposes. Blocked and detected are reactions taken by antivirus software. – represents no reaction.

While all other attacks are terminated with no additional information, Bitdefender gives two warnings before the remediation of the Cerber ransomware attack starts. One of the firewall rules is triggered first, followed by the discovery of an infected file, which is the ransom note in this case. **Failed cases.** Kaspersky, McAfee, and Norton miss 1, 4, and 2 attacks, respectively. One special case is the Hive sample, which Kaspersky and McAfee miss. It runs in the background but no attack behaviors are observed in 10 minutes. In all other cases, a large number of files are encrypted, but no warning or action is triggered. In an offline setting, 360 only catches 3 out of 20 threats, implying cloud computation for detection.

Detection on ransomware-like behaviors. We further investigate the ability of antivirus software to detect various ransomware-like behaviors by running simulations. Most of them block public simulators as real malware. A few exceptions include Bitdefender, which passes all RanSim scenarios, Norton, which detects the macro simulation, and 360 (offline), which has no reaction at all. Emulating various ransomware behaviors, we find that antivirus software only reacts to the deletion of volume shadow copy, which is a backup copy of computer volume. Intensive encryption and file access do not invoke any warning.

4.4 Commercial Malware Scanners

We also evaluate the efficacy of general-purpose malware scanners. We find that the scanners' detection capability could be significantly weakened by simple obfuscation, i.e., compression with passwords. Therefore, dynamic behavioral detection is necessary to complement the protection.

Malware scanners provide pre-execution scanning, an early layer of protection against infection. Our experiments show that the majority of the scanners can effectively identify the threat in plain executable files. On average, 56 out of over 70 scanners raise an alert. However, the detection capability significantly reduces on password-protected samples. When compressed with the simple password "infected", WannaCry triggers three alarms, which is the most among the 54 samples. In the complex password setting, 37 out of 54 samples completely evade detection, i.e., marked as safe by all scanners. This result suggests that the scanners are likely signature-based and have very limited detection capability against unknown and obfuscated samples. Full results are shown in Table 18 in the appendix. Obfuscating a malicious sample with encryption requires little

effort but significantly increases the chance of escaping. However, forcing breaking advanced encryption algorithms is impossible and should not be the goal of a malware scanner. One possible approach to strengthen security is to warn the user that the file is encrypted and suggest further scanning.

5 Characterization of Ransomware API Usage

To develop new insights into building deployable behavioral detection against ransomware, we experimentally analyze the API utilization of real-world ransomware samples. In this section, we present our findings on the ransomware API usage pattern with detailed analyses. With a comparison to benign software behaviors, we find that API usage and frequency are informative in terms of revealing malicious behaviors. We present a possible classification approach based on these findings in Section 6. As API monitoring happens at the program level, the deployability of such approaches is also high. We discuss more about its deployability in Section 7.

5.1 Encryption and File Access Behaviors

To take a close look at the file encryption behaviors, we analyze cryptographic and file-related API usage and call frequency of ransomware samples.

5.1.1 Experimental Setup 1. Ransomware samples. We collect 262 ransomware samples from MalwareBazaar⁹ and VirusShare.¹⁰ Among these samples, we find 54 of the samples (from 35 distinct families) provide more meaningful traces for manual analysis. The SHA256 hashes of the 54 samples are shown in Table 17 in supplemental materials, which can help find the exact samples. **Analysis environment.** We set up an isolated environment for safely executing ransomware samples using Cuckoo Sandbox (v2.0.7),¹¹ with VirtualBox (v6.1)¹² as the hypervisor and Windows 7 as the guest system. We use 4 CPUs and 4096 MB memory for the VM. We install several applications, including Chrome, Adobe PDF Reader, NotePad++, and LibreOffice, to make the environment more realistic. We also put random files under several directories, such as disk C, Documents, and Downloads for ransomware to encrypt. Each execution starts with a clean system image. To avoid VM escape, we use a different host system (Ubuntu 21.04). We also set up a fake internet service using INetSim.¹³ Lastly, to minimize the risk of spreading, we disconnect the machine from the internet.

Report and API analyses. For each successful execution, the Cuckoo sandbox generates a report in JSON format, from which we extract data for analysis. We conduct API call frequency analyses based on the reported API calls.

Comparison with benign software samples. We download the top popular Windows applications from a range of different categories from software.informer¹⁴ or the software's official website. We use 38 samples from 12 categories for manual analysis (Table 18 in supplemental materials). We run the benign samples in the Cuckoo sandbox the same way we run ransomware samples and collect the execution reports for analysis. Among those samples, we use the ones with intensive file access behaviors in the evaluation of our classification (Section 6.2).

5.1.2 Characterization Findings on Encryption Behaviors. Our characterization study identifies interesting encryption behaviors.

⁹<https://bazaar.abuse.ch/browse/>

¹⁰<https://virusshare.com/>

¹¹<https://cuckoosandbox.org/>

¹²<https://www.virtualbox.org/>

¹³<https://www.inetsim.org/>

¹⁴<https://software.informer.com/System-Tools/>

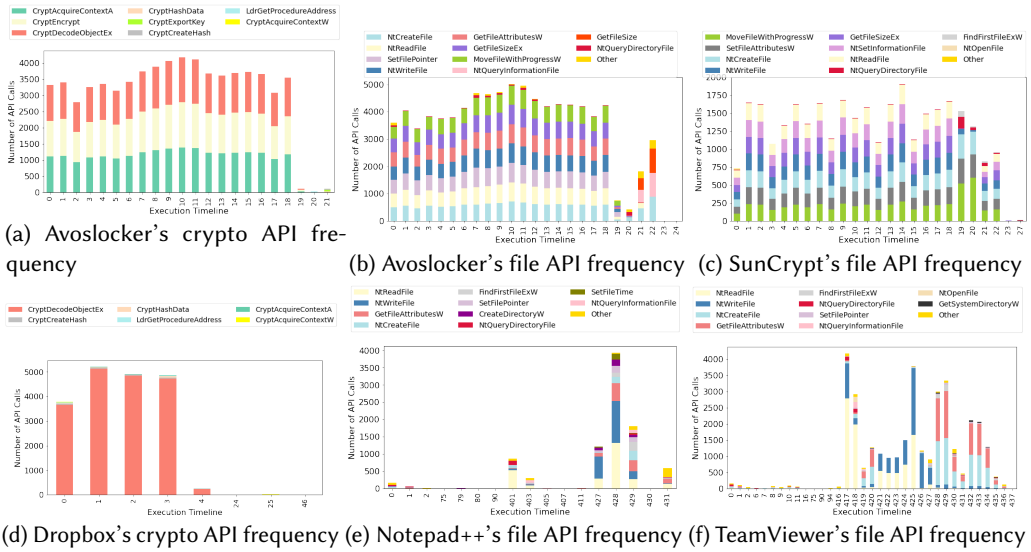


Fig. 1. API call statistics of executing ransomware samples (top) and benign software (bottom). The x-axis is the timestamp during execution and the y-axis shows the number of calls. If no call is made during a second, then it is not shown in the figures. Ransomware uses intensive crypto and file API calls during execution with a distinctive pattern. The same colors represent the same APIs across subfigures.

CryptoAPI frequency. Some ransomware samples make intensive CryptoAPI calls, in the order of thousands, during execution. We show AvosLocker as an example in Figure 1(a). It uses the API sequence `CryptAcquireContextA`, `CryptDecodeObjectEx`, `CryptEncrypt` for encryption. The number of calls to each of the three APIs is similar, adding up to around 4,000 calls per second.

Ransomware file access. Similar to crypto-related calls, file access frequency is also in the order of thousands per second. The peaks of calls are around 5000 and 1900 calls per second for AvosLocker and SunCrypt (Figures 1(b) and 1(c)), respectively. By examining the directory paths touched through calls to `NtCreateFile`, `NtWriteFile`, and `NtOpenFile`, we notice that these samples start traversing the directory at `C://` and then go into child directories. MountLocker even searches disks through `a://` to `z://`. AvosLocker accesses 780 unique directories during the execution, SunCrypt accesses 180, and MountLocker accesses 2,236 in our testbed. The three families use a similar combination of 6 to 7 file-related APIs repetitively during the encryption process. The combination includes APIs to create files, read files, write files, query file size or information, and set file pointers. The number of each API being called every second is somewhat evenly distributed, with no absolute dominance.

Comparison with benign samples. To contrast with ransomware behavior, we also analyze the execution of 38 benign software samples. The top API categories of the majority of benign samples include system, registry, and miscellaneous, varying from ransomware behavior. However, there are a few exceptions. Dropbox (Figure 1(d)) makes over 3,500 crypto-related calls per second in the first four seconds of execution, with a peak of over 5,000 calls per second. The difference is in the composition of the API calls made. While AvosLocker uses a combination of three CryptoAPIs with relatively even distribution, the vast majority of calls made by Dropbox are to a single CryptoAPI, i.e., `CryptDecodeObjectEx`.

Moreover, some benign samples also make a notable amount of file-related calls, as shown in Figures 1(e) (Notepad++) and 1(f) (TeamViewer). By examining the composition and call patterns, one can easily tell the benign usage pattern is rather random, varying from program to program.

Table 4. List of APIs that Significantly More Prevalent in Ransomware When Compared with Benign

	API	% RW	% Benign
1	NtOpenDirectoryObject	65%	32%
2	CoInitializeSecurity	59%	26%
3	MoveFileWithProgressW	57%	29%
4	WriteConsoleW	50%	5%
5	Process32NextW	49%	11%
6	CreateToolhelp32Snapshot	49%	12%
7	Process32FirstW	48%	11%
8	CryptEncrypt	18%	0%
9	CryptExportKey	15%	3%
10	CryptGenKey	10%	0%

The percentage is calculated by (number of ransomware that calls this API / 348) for ransomware and (number of benign programs call this API / 330) for benign programs.

In summary, ransomware shows a distinguishable repetitive file-access API pattern throughout the execution. We further analyze the distinction between ransomware behavior and intensive benign file access in-depth. The results show that this unique pattern can be quantified to help classify malicious from benign execution. Our classification method is presented in Section 6.

5.2 API Occurrence Contrast Analysis

In another characterization study, we conduct a contrast analysis of 288 APIs, comparing their occurrences and usage frequencies in ransomware and benign programs.

5.2.1 Experimental Setup 2. We collect 348 ransomware samples from MalwareBarzaar. The samples are from 37 families. For benign samples, we collect 330 of them from software.informer. We use the same sandbox setting in this experiment and the JSON report for API analysis. The virtual machine has 4 CPUs and 8192 MB of memory. We refer to this setting as setup 2.

5.2.2 Characterization Findings on Contrast Analysis. We observe some APIs are used more commonly by ransomware than benign software. For instance, WriteConsoleW is used by 50% of ransomware samples we measure, while only occurs in the execution traces of 5% of benign samples. We show a list of such APIs in Table 4. Moreover, even if some APIs occur in a similar number of ransomware and benign software, the call frequency could vary substantially. For example, NtWriteFile is used by a comparable number of ransomware (333 samples) and benign programs (328 samples). However, ransomware samples make, on average, 40554 calls during execution, which is around 8 times compared with benign (5031 on average). More examples of such APIs are shown in Table 5. Comparably, there are also a set of APIs that are more commonly observed during benign executions. A list is presented in Table 6.

The invocation patterns of specific APIs vary between ransomware and benign programs. This API usage contrast is useful for building new detection. Later, we show how they aid detection in Section 6.1.

5.3 Dynamic Resolution of API and Windows Modules

From reverse engineering, we found some samples obfuscate strings and Windows library names, including crypto-related modules, and resolve them at runtime. Dynamic resolution is to load a

```

uStack960 = 0x7b;
uVar11 = 0;
abStack956[0] = 8;
abStack956[1] = 0x13;
abStack956[2] = 0x1e;
abStack956[3] = 0x17;
abStack956[4] = 0x17;
abStack956[5] = 0x48;
abStack956[6] = 0x49;
abStack956[7] = 0x55;
abStack956[8] = 0x1f;
abStack956[9] = 0x17;
abStack956[10] = 0x17;
uStack945 = 0;
do {
  abStack956[uVar11] = abStack956[uVar11] ^ 0x7b;
  uVar11 = uVar11 + 1;
} while (uVar11 < 0xb);

uStack945 = 0;
if (DAT_004f081c == 0) {
  pbStack1180 = (byte *)0x4c01c9;
  DAT_004f081c = FUN_00411a10();
}
if (DAT_004f079c == (code *)0x0) {
  pbStack1180 = (byte *)0x4c01df;
  DAT_004f079c = (code *)FUN_004122e0(DAT_004f081c);
}
pbStack1180 = abStack956;
pbStack1184 = (byte *)0x4c01f1;
(*DAT_004f079c)();

```

Stack string (obfuscated DLL name)

Decoding by XORing each byte with 0x7b. The decoded DLL is shell32.dll in this case.

Decoded DLL name

Load it dynamically

Fig. 2. Example of LockBit code that dynamically resolves libraries.

Table 5. List of APIs that are Comparably Prevalent in Ransomware and Benign but have a Much Higher Frequency in Ransomware Execution

	API	RW freq mean	Benign freq mean	# RW / # Benign
1	CryptCreateHash	159568.5	12.6	48 / 63
2	NtWriteFile	40554.4	5031.2	333 / 328
3	NtReadFile	33883	11532.1	324 / 329
4	SetFilePointerEx	7472	179.8	240 / 247
5	NtAllocateVirtualMemory	4839.2	1602.5	342 / 330
6	NtFreeVirtualMemory	4599.4	579.9	331 / 330
7	NtCreateFile	4564.3	1292.2	340 / 329
8	FindFirstFileExW	2666.1	963.6	281 / 308
9	CryptAcquireContextA	2170.6	16.1	50 / 60
10	GetFileType	1149.9	226.7	240 / 257
11	SetFileAttributesW	1105.6	103.1	152 / 115
12	NtDeviceIoControlFile	682.1	71.9	243 / 193
13	RegDeleteValueW	245.3	13.8	137 / 150
14	OpenSCManagerW	90.8	4.5	191 / 180
15	GetUserNameExW	34.3	6.5	133 / 166
16	OpenServiceW	23.6	7.6	174 / 140
17	NtOpenThread	14.3	4	170 / 164
18	CoCreateInstanceEx	6	2	126 / 96

RW stands for ransomware. RW freq mean is the average call frequency based on all ransomware samples that use this API. # RW is the number of ransomware samples that used this API and # benign is the number of benign programs that used this API. The call frequency is collected during a 300-second execution period for each sample.

module or an API using its address during execution. In this way, the usage of a module remains unknown until being executed. For instance, LockBit obfuscates strings into random text and only statically imports five modules, preventing critical information from being extracted easily. Each string and library name is encoded by XORing or subtracting a unique value, and is decoded individually during execution. Figure 2 shows how LockBit decodes and loads shell32.dll. 28 such strings were found in the entry function of the LockBit. 21 of them turn out to be Windows 32 libraries, which are a much longer list than the imports. Four of them are crypto-related, namely advapi32.dll, bcrypt.dll, crypt32.dll, and cryptbase.dll.

Table 6. List of APIs that Significantly More Prevalent in Benign Software When Compared with Ransomware

	API	% RW	% Benign
1	RemoveDirectoryA	0%	53%
2	NtDeleteKey	1%	94%
3	GetSystemDirectoryA	3%	57%
4	FindResourceA	5%	63%
5	NtReadVirtualMemory	9%	74%
6	SendNotifyMessageW	10%	96%
7	FindWindowW	12%	90%
8	SetEndOfFile	12%	81%
9	GetKeyState	13%	97%
10	GetTempPathW	13%	76%
11	GetFileVersionInfoW	14%	88%
12	GetFileVersionInfoSizeW	15%	88%
13	RegCreateKeyExA	15%	65%
14	FindResourceW	16%	98%
15	GetFileInformationByHandle	16%	82%
16	DrawTextExW	17%	100%
17	GetCursorPos	17%	98%
18	SearchPathW	17%	98%
19	SetFileTime	18%	97%
20	GetVolumePathNameW	18%	80%
21	SizeofResource	20%	99%
22	NtCreateKey	20%	97%
23	FindResourceExW	21%	100%
24	OleInitialize	23%	87%
25	GetForegroundWindow	24%	100%
26	EnumWindows	24%	96%

The percentage is calculated by (number of ransomware that calls this API / 348) for ransomware and (number of benign programs that call this API / 330) for benign programs.

To further inspect this feature, we analyzed the execution report and found that the samples use `LdrLoadDll` and `LdrGetProcedureAddress` to dynamically load DLLs and APIs. By looking at the loaded modules, we found 7 crypto-related modules, namely `advapi32`, `cryptsp`, `cryptbase`, `bcryptimitives`, `crypt32`, `rsaenh`, and `bcrypt`. In the import table of the 54 ransomware samples (experimental setup 1), only `advapi32` and `crypt32` show, in 38 and 13 samples, respectively. However, all 7 crypto-related modules are dynamically loaded through `LdrLoadDll`, by 48, 33, 22, 19, 15, 8, and 3 samples, respectively. Other non-crypto modules that are frequently loaded by ransomware samples include `kernel32` (48 samples), `ole32` (42 samples), `rpcrt4` (39 samples), and `user32` (36 samples).

A short import table with few crypto-related APIs may put detection off guard and let the malware through. Encryption is one of the most distinctive features of ransomware. Hiding it increases their chance of escaping detection. One step further, although not observed in our analysis, the hacker could write dead code where the execution would never reach and put irrelevant APIs in the import table for distraction, hindering the efficiency of static analysis.

Table 7. API Examples in Crypto-Related Modules Dynamically Resolved by Ransomware and Benign Samples

	Top modules	API examples	Total count	# New / # Old	Change%
Ransomware	cryptsp	CryptReleaseContext	42	29 / 13	-4.93%
		CryptAcquireContextW	27	22 / 5	26.64%
		CryptGenRandom	24	20 / 4	27.63%
	advapi32	CryptDestroyKey	10	7 / 3	-0.33%
		CryptGenKey	10	7 / 3	-0.33%
		CryptImportKey	10	7 / 3	-0.33%
	rsaenh	CPGenRandom	19	16 / 3	23.36%
		CPGenKey	19	16 / 3	23.36%
		CPEncrypt	19	16 / 3	23.36%
	cryptbase	RtlGenRandom (SystemFunction036)	19	12 / 7	-12.17%
Benign	bcrypt-primitives	GetHashInterface	12	12 / 0	31.58%
		GetCipherInterface	7	7 / 0	18.42%
	advapi32	EventUnregister	26	-	-
		EventRegister	26	-	-
		EventWrite	26	-	-
	cryptsp	CryptReleaseContext	23	-	-

Total number: number of samples that dynamically load this API. # New / # Old: numbers of new samples (2020 or after) and old samples (2019 or earlier) that load this API. Change %: the difference between the newer and older sample group, (# newer sample / 38 - # older sample / 16). Positive values suggest an increase in usage in recent years.

Comparison between newer and older ransomware samples. To find out if there exist any trends of evolution, we analyzed the usage of top dynamic resolved APIs in the newer sample group (samples from 2020 and after) and the older sample group (2019 and before). The numbers and percentage of changes are summarized in Table 7. Overall, the usage of dynamic resolutions has increased in recent years. The rsaenh module becomes 23.36% more prevalent in newer samples. The usage of CryptAcquireContentW and CryptGenRandom, which are APIs from cryptsp, increased by 26.67% and 27.63%, respectively. None of the samples before 2019 dynamically loaded the bcryptprimitives module but its APIs are being used by up to 31.58% of recent samples.

Comparison with benign samples. This comparison between ransomware and benign samples help identify the distinctions in run-time API usages, possibly. Benign samples also dynamically resolve Windows libraries, including crypto-related modules. Top modules include advapi32 (33 samples out of 38 dynamically load it) and cryptsp (23 samples). While the same crypto-related modules are used, the most frequent APIs loaded by benign samples are different. Most top benign loaded APIs are from advapi32, which stands for Advanced Windows 32 Base API. This module also contains basic Windows API other than crypto functions, such as EventRegister (Table 7).

5.4 Other Behavioral Signatures

The previous sections focus on the encryption and file access API usage during ransomware execution. In this section, we examine other ransomware behaviors. We summarized 16 common behavioral signatures in four categories (Table 8). We derived the severity of each based on its

Table 8. Non-Encryption Behavioral Signatures of Ransomware Samples

Category	Signature	Severity	# Samples (54) / # Families (35)	# New (38) / # Old (16)	Change %	# Benign (38)	Diff %
Anti-detection and anti-analysis	Attempts to detect sandbox through the presence of a file or process	High	24 / 17	18 / 6	9.87%	1	41.8%
	Attempts to delay analysis (a process tried to sleep)	Med	17 / 12	13 / 4	9.21%	11	2.5%
	Checks foreground human activities	Med	16 / 10	13 / 3	15.46%	21	-25.6%
	Creates a process named as a common system process	Med	4 / 3	3 / 1	1.64%	3	-0.5%
	Checks amount of memory or disk size in system to detect virtual machine	Med	31 / 20	24 / 7	19.41%	27	-13.6%
Modification of system and security setting	Operates on local firewall's policies and settings	High	7 / 4	7 / 0	18.42%	0	13.0%
	Attempts to modify desktop wallpaper	High	6 / 3	5 / 1	6.91%	0	11.1%
	Disables Windows Security features (e.g., Windows defender)	High	3 / 2	1 / 2	-9.87%	0	5.6%
	Clears Windows event logs	High	1 / 1	1 / 0	2.63%	0	1.9%
	Installs itself for autorun at Windows startup	Low	22 / 13	14 / 8	-13.16%	13	6.5%
Prevention of recovery	Deletes the Shadow Copy	High	16 / 12	12 / 4	6.58%	0	29.6%
	Modifies boot configuration settings	High	11 / 7	10 / 1	20.07%	0	20.4%
	Uses wbadmin utility to delete backups	High	5 / 2	5 / 0	13.16%	0	9.3%
Modification of file	Appends a new file extension to files	High	51 / 33	36 / 15	0.99%	5	81.3%
	Moves a number of files	High	13 / 10	10 / 3	7.57%	2	18.8%
	Deletes a number of files	High	6 / 5	1 / 5	-28.62%	6	-4.7%

Samples / # Families: numbers of ransomware samples and families have this signature. # New / # Old: numbers of new samples (2020 or after) and old samples (2019 or earlier) that have this signature. Change %: the difference between the newer and older sample group, (# newer sample / 38 - # older sample / 16). Positive values suggest an increase in usage in recent years. # Benign: number of benign samples also have this signature. Severity level is based on its severity score defined by the Cuckoo community. Diff %: difference in this signature's prevalence between ransomware and benign samples (# Samples / 54 - # Benign Samples / 38). The numbers in () are the number of samples in this group. The behavioral signatures more prevalent in ransomware are highlighted in bold.

severity score defined by the Cuckoo community. Ransomware samples are further divided into newer and older groups based on the year for analysis of evolution. We also calculate the difference between the prevalence of each signature in ransomware and benign software. The higher the value, the more indicative of ransomware the signature is. The ones that are more prevalent in ransomware are marked in bold in Table 8.

Prevention of recovery. The nature of ransomware is to hold the victim's data hostage. Therefore, they use multiple techniques to avoid data or system recovery. To reduce the possibility of recovery from the volume shadow copy, which is a snapshot that duplicates data stored in a volume, many ransomware attempts to delete it. 16 samples use commands such as `vssadmin delete shadows /all /quiet` and `wmic shadowcopy delete` to do so. We also observed that 11 samples modify the boot configuration of the system to prevent the Windows system from going into the recovery environment (Windows RE). This is often done by using `bcdedit` commands (e.g., `bcdedit /set default recoveryenabled no` and `bcdedit /set default bootstatus-policy ignoreallfailures`). Yet another technique to prevent the usage of backups is to delete the global catalog that contains system backup information. Five samples do it through the command `wbadmin delete catalog -quiet`.

Modification of file. File type changing is often a signal of data being encrypted. We notice that 51 out of 54 ransomware samples append a new extension to the files in the system, some are the same as the ransomware family name (e.g., `.alcatraz` by Alcatraz) and some look random (e.g., `.36Q62M` by Sodinokibi). Moving and deleting a large number of files (observed in 9 and 3 samples, respectively) are also signs of file modification.

Modification of system and security setting. Before launching the attack, ransomware often changes Windows system setting to their convenience, especially security settings. Seven samples modify local firewall rules and three samples disable several Windows Defender features. After the attack, one sample (i.e., Hive) also clears the Windows event log to hide its trace. 22 samples install themselves to run at system startup, possibly launching the attack again every time the system

Table 9. Top Processes Used by Ransomware Samples

Process	#Samples\$54)	#Families\$35)	#New (38)	#Old (16)	Change %	#Benign\$38)	Diff %
lsass	54	35	38	16	0.00%	38	0.0%
cmd	36	23	24	12	-11.84%	1	95.0%
vssadmin	20	15	15	5	8.22%	0	55.6%
explorer	15	9	13	2	21.71%	22	-68.3%
reg	13	7	7	6	-19.08%	1	31.1%
WMIC	12	8	11	1	22.70%	0	33.3%
bcdedit	10	6	9	1	17.43%	0	27.8%
netsh	7	4	7	0	18.42%	0	19.4%
PING	6	4	4	2	-1.97%	0	16.7%
taskkill	6	4	4	2	-1.97%	1	11.7%
net	5	4	5	0	13.16%	1	8.9%
net1	5	4	5	0	13.16%	1	8.9%
wbadmin	5	2	5	0	13.16%	0	13.9%
notepad	4	3	3	1	1.64%	0	11.1%
chrome	4	4	2	2	-7.24%	1	6.1%
powershell	4	3	3	1	1.64%	0	11.1%
icacls	3	2	2	1	-0.99%	0	8.3%
schtasks	3	3	1	2	-9.87%	1	3.3%
cscript	3	2	0	3	-18.75%	0	8.3%

Samples / # Families: numbers of ransomware samples and families initiate this process. # New / # Old: numbers of new samples (2020 or after) and old samples (2019 or earlier) that use this process. Change %: the difference between the newer and older sample group. Positive values suggest an increase in usage in recent years. # Benign: number of benign samples. Diff %: difference in this process' prevalence between ransomware and benign samples. The numbers in () are the number of samples in this group. The processes more prevalent in ransomware are in bold. All processes have a .exe suffix.

reboots in case of recovery. In addition, 6 samples attempt to modify the desktop wallpaper to make the message more attention-calling and threatening.

Anti-analysis and anti-detection. The majority of ransomware samples attempt to hinder analysis and detection in order to increase the success rate of attacks. Because of the extensive use of VMs and sandboxes in malware analysis, many ransomware samples try to distinguish a VM from a physical machine [145, 164]. Out of the 54 samples examined, 31 check the amount of system memory or disk size, 24 try to detect the presence of a sandbox, and 16 check foreground human activities. Some dynamic detection or analysis run a program for a few minutes to evaluate the risk. To evade detection, 17 samples delay the execution. In the case of finding clues of being analyzed in a VM, the malware would likely end execution or hang indefinitely, leaving no trace for analysis.

Processes. Additionally, we summarize the processes initiated by the 54 samples during execution (Table 9). Consistent with the usage of prevention-of-recovery commands, we find 20, 12, 10, and 5 samples utilize vssadmin, WMIC, bcdedit, and wbadmin, respectively. Frequent usage of cmd (36 samples) and powershell (4 samples) are also observed, likely for executing the commands.

Comparison between newer and older ransomware samples. We do the compare between newer and older samples to identify the trends of evolution (Table 8). One notable trend is the increased usage of recovery prevention. 20.07% more newer samples modify the boot configurations, 13.16% more delete backups using wbadmin, and 6.58% more deletes the volume shadow copy. Moreover, the usage of evasive techniques has a trend of growing as well, with up to a 19.41% increase.

Comparison with benign samples. We further inspected the benign sample to see the similarities and differences with ransomware. Shared behavioral signatures include checking foreground human activities (observed in 21 out of 38 benign samples) and checking system memory and disk size (27 samples). Although these features look suspicious, they would not separate malicious and benign well. Modification of security settings and prevention system recovery are rarely seen in regular applications. On the other hand, except for lsass and explorer, the processes used by ransomware are not common in regular software execution, including command line-related processes (e.g., cmd and powershell) and system backup-related processes (e.g., vssadmin, WMIC, bcdedit, and wbadmin).

6 New API-Profilng Based Classification Method

This section presents our new API-based classification algorithm for identifying ransomware threats (RQ3).

6.1 Algorithms

Our method consists of two main operations: (i) **consistency-based classification** and (ii) **refinement using API contrast score**. In consistency-based classification, we quantify the ransomware execution patterns according to their API repetition patterns. We build several consistency-based mathematical models for classification. Then, in the refinement operation, we compute API contrast scores to further improve the detection accuracy.

6.1.1 Consistency-Based Classification. We present multiple computational methods, with varying complexity, for summarizing ransomware's API invocation behavioral patterns. These methods are for the first stage of our detection.

Consistency-based detection. We design four consistency metrics to quantify the variation during the execution. We treat the API composition for a short execution time period as a vector and compare the vector with a previous period. Small variations between vectors suggest more consistency. Specifically, the metrics are Cosine-based consistency (Equation (1)), Manhattan-based consistency (Equation (2)), frequency-weighted consistency (Equation (3)), and Euclidean-based consistency (Equation (4)):

$$1 - \frac{c \cdot p}{\|c\| \|p\|} \quad (1)$$

$$\sum_{i=1}^n |c_i - p_i| \quad (2)$$

$$\sum_{i=1}^n f_i |c_i - p_i| \quad (3)$$

$$\sqrt{\sum_{i=1}^n (c_i - p_i)^2} \quad (4)$$

c is the vector representing the current execution window, p is the vector representing the previous execution window, and f is the vector representing the frequency of top APIs. When calculating the scores, we consider the top 10 file API and use 3-second and 1-second windows for the previous and current, respectively. All of those parameters can be adjusted at the time of application.

As the names suggest, Manhattan-based and Euclidean-based consistency use Manhattan and Euclidean distance to compute the difference between execution periods. The smaller the score, the more consistent the execution. Frequency-weighted consistency is a variation of the Manhattan distance that each element is weighted by the frequency. That is, the top frequent API takes a larger part in the score. Lastly, cosine-based consistency is calculated based on the cosine similarity between the two vectors representing previous and current execution. Because the more similar the vectors are, the closer to 1 the cosine similarity is, we use 1 minus cosine similarity here to

be consistent with other metrics (i.e., smaller values represent more consistency). For all four consistency algorithms, the smaller the value, the more malicious the program is.

Evenness-based detection. In this method, we analyze the frequency distribution of multiple top APIs together. This method aims to capture the ransomware feature that the API usage composition of each epoch is relatively evenly distributed during its execution. We develop two evenness-related metrics to quantify the execution pattern, namely normalized evenness (Equation (5)) and squared evenness (Equation (6)):

$$\sum_{i=1}^n \frac{|API_i - avg|}{avg} \quad (5) \quad \sum_{i=1}^n (API_i - avg)^2 \quad (6)$$

avg is the average of top API usage, n is the number of top APIs used. The evenness is first calculated for each small execution period and then averaged to present the whole execution.

Changepoint-based detection. In this method, we count notable changes using **Bayesian Online Changepoint Detection (BOCD)** [2] to separate malicious and benign traces. BOCD is designed to identify abrupt changes in sequential data. Ideally, ransomware execution should have fewer changepoints due to the constant pattern.

As a baseline, we also implement a single API distribution approach, whose detection is based on the frequency distributions of top file APIs, such as `NtWriteFile`, `NtReadFile`, and `NtCreateFile`, using the Poisson distribution, Wilcoxon rank sum test, and **Jensen-Shannon (JS)** divergence. The equation for computing JS divergence is as follows:

$$JS(P \parallel Q) = \frac{1}{2} * KL(P \parallel M) + \frac{1}{2} * KL(M \parallel P), \quad (7)$$

where

$$M = \frac{1}{2} * (P + Q), \quad (8)$$

and KL is the KL divergence that

$$KL(P \parallel Q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right). \quad (9)$$

When computing a divergence score for a ransomware sample, P represents the API distribution of the specific sample and Q represents the overall benign distribution. Vice versa for benign.

6.1.2 Refinement Using API Contrast Score. The refinement operation – the second stage of our detection—builds on leveraging the API occurrence variation between ransomware and benign programs. In refinement, our screening is centered on API contrast scores, explained next. **First**, with the labeled dataset, we perform a comparative counting analysis to compute a contrast score for each distinct API. This training process also organizes the APIs into three distinct sets, based on their occurrences and invocation frequencies in ransomware and benign execution. The three sets are (i) likely ransomware API set R , which contains APIs that often occur in ransomware, but rarely in benign programs, (ii) likely benign API set B , which contains APIs that commonly appear in benign, but less often in ransomware, and (iii) co-occurring API set O , which consists of APIs used by both types of samples, but with a much lower call frequency in benign execution. This assignment is based on relative or pre-defined thresholds, as shown in Equation (10). **Second**, we compute the single contrast score C_i for each API_i following Equation (10). **Finally**, during the testing phase, given the profile of an unknown execution, we compute the total contrast score for all n occurrences of APIs in the profile, i.e., $\sum_{i=1}^n C_i$, with deduplication (each distinct API will only be counted once).

Table 10. List of Top Ransomware APIs Used in API Contrast Score (set R)

API			
CoInitializeSecurity	Process32FirstW	WriteConsoleW	CryptEncrypt
CreateToolhelp32Snapshot	Process32NextW	CryptGenKey	CryptExportKey

Table 11. List of Co-Occurring APIs Used in Calculating the API Contrast Score (set O)

API	Threshold	API	Threshold
NtAllocateVirtualMemory	6412	RegDeleteValueW	56
NtFreeVirtualMemory	2320	GetUserNameExW	28
OpenSCManagerW	20	CoCreateInstanceEx	8
OpenServiceW	32	CryptAcquireContextA	64
NtOpenThread	16	CryptCreateHash	52

The frequency limits are calculated based on the observed frequency and the duration of execution. We set them to be double the average benign frequency. Because the execution time is 600s for setup 1 samples and 300s for setup 2 samples, we further double the value when applied on setup 1 samples.

Table 12. List of Top Benign APIs Used in Calculating the API Contrast Score (Set B)

API			
NtDeleteKey	GetCursorPos	GetForegroundWindow	FindResourceExW
EnumWindows	SizeofResource	GetFileVersionInfoSizeW	FindResourceA
GetKeyState	OleInitialize	GetFileVersionInfoW	RegCreateKeyExA
DrawTextExW	FindWindowW	SendNotifyMessageW	
FindResourceW	NtCreateKey	NtReadVirtualMemory	

$$C_i = \begin{cases} 1, \text{API}_i \rightarrow \text{R}, & \text{if } \frac{\text{occ}_i^{\text{R}}}{\text{occ}_i^{\text{B}}} \geq \tau_1 \\ -1, \text{API}_i \rightarrow \text{B}, & \text{if } \frac{\text{occ}_i^{\text{R}}}{\text{occ}_i^{\text{B}}} \leq \tau_2 \\ 1, \text{API}_i \rightarrow \text{O}, & \text{if } \frac{\text{occ}_i^{\text{R}}}{\text{occ}_i^{\text{B}}} \in (\tau_2, \tau_1) \wedge \frac{\text{freq}_i^{\text{R}}}{\text{freq}_i^{\text{B}}} \geq \tau_3 \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

In Equation (10), set R contains APIs occurring highly frequently in ransomware, but rarely in benign, B contains APIs associated with benign software, but not with ransomware, and set O contains APIs that occur in both, occ_i^{R} is the count of ransomware samples in which the execution traces include the occurrence of API_i , and occ_i^{B} is the count of benign samples whose traces include the occurrence of API_i .

freq_i^{R} is the average call frequency of API_i in O's ransomware execution, freq_i^{B} is the average call frequency of API_i in O's benign samples. In our implementation, τ_1 is set to 2, τ_2 is set to 3, and τ_3 is set to 2 (i.e., the call frequency exceeds the average benign call frequency by at least two times). The frequency limit used for each API in set O can be found in Table 11 in the appendix. Because we already consider file-related APIs in the previous classification stage, we only include non-file APIs in the refinement. The lists of APIs in each set are in Tables 10, 11, and 12, respectively.

6.2 Classification Results

In this section, we present the evaluation results of our two-stage approach, first on consistency-based classification, then on API contrast-based refinement. The test is conducted on real-world

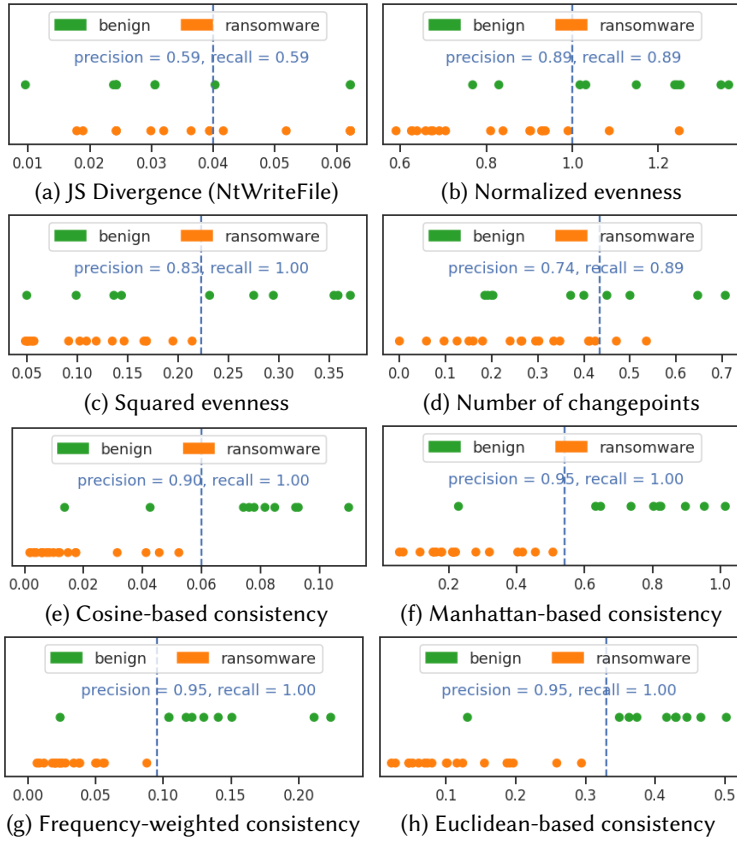


Fig. 3. Classification results using different methods and models. Each green dot (top row) represents a benign software sample and each orange dot (bottom row) represents a ransomware sample. The dotted blue line is a boundary for helping understand how well the separation is. The precision and recall shown in the figures are based on the chosen boundary and are in terms of the ransomware class.

samples from 15 distinct ransomware families. Then, we provide an in-depth analysis of the top important APIs with attack context as case studies. Finally, we demonstrate the performance of the commercial defense, identifying the security gap.

6.3 Results of Various Classification Models

We first present the classification results based on various mathematical models. Consistency-based algorithms show the best performance. The results of classifying 19 ransomware samples and 10 benign programs with notable amounts of file activities are shown in Figure 3.

Single API distribution. Using only the distribution of a single API (i.e., baseline), we observe the scores for ransomware and benign samples are highly overlapping, implying the inadequacy of this model. An example of JS divergence of NtWriteFile's distribution is shown in Figure 3(a).

Evenness and the number of changepoints. As shown in Figures 3(b), 3(c), and 3(d), the evenness and changepoint metrics worked better than relying on single API distribution. We can see the trend that most ransomware samples are on the left side in the figure while benign samples have relatively larger values in both cases. However, there are still no clear boundaries between them and there exists room for improvement.

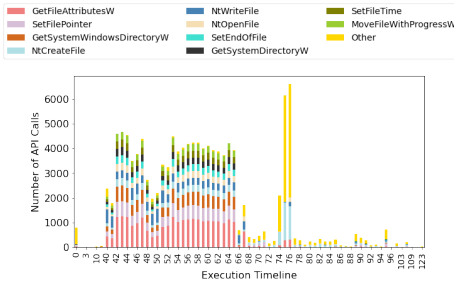


Fig. 4. Git's file API frequency (false positive)

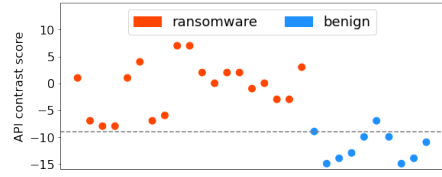


Fig. 5. API contrast scores of ransomware and benign samples.

Consistency. When considering the consistent execution pattern, there is a clear separation between ransomware and benign samples. All ransomware samples have a relatively small value, gathering on the left side of the figure (Figures 3(e), 3(f), 3(g), and 3(h)). Specifically, Manhattan-based consistency, frequency-weighted consistency, and Euclidean-based consistency show optimal performance. With a proper threshold, they accurately catch all ransomware cases with only one false positive. Cosine-based consistency has slightly a lower precision of 0.9. The false positive case that appeared in all four settings is Git, which has a period of execution with high consistency, shown in Figure 4. In summary, the consistency in ransomware file-access API usage helps to identify the threats from benign intensive file accesses.

6.4 Results of API Contrast-Based Refinement

We evaluate the effectiveness of API-contrast refinement, specifically how much it help the classification. The API patterns are distilled from samples from setup 2 and tested on setup 1. When considering only the top ransomware API score (i.e., only using APIs in sets R and O, as described in Section 6.1.2), all benign samples have a lower score, ranging from 0 to 4. Git, the false positive case generated by all four consistency algorithms, only has a score of 2, helping reduce the possibility of being malicious. Another false positive case produced by the cosine-based consistency is PaintNet, which hits none of these rules and gets a 0. On the other hand, ransomware samples have relatively high scores with a maximum of 11. The scores for all samples are listed in Table 13.

Comparably, for the top benign API score (i.e., only using APIs in set B), the benign samples tend to cluster within the range of -7 to -17 (Table 13). The false positive sample, Git, has a score of -17, which is the lowest among all samples, implying benignness. However, the scores of ransomware are rather spread out, with a lower bound of -15, overlapping with the benign range.

The API contrast score, which takes advantage of both scores discussed above, demonstrates the most promising performance. All benign samples have a score of at most -7, gathering at the lower area (in blue) in Figure 5. Git has a score of -15, falling in the range of benign. Filtering all positive predictions from Manhattan-based with a threshold of -10, the benign case Git is separated out while the decision on all ransomware cases remains unchanged, helping further boost the precision. The refinement outcome is the same for frequency-weighted and Euclidean-based consistency metrics.

On top of the consistency-based classification, the API contrast score helps further evaluate the risk and reduce false positives. However, this stage of detection could be evaded by sophisticated attacks. We acknowledge and discuss the limitations of it in Section 7.

6.5 Case Study of Most Informative APIs

To further investigate the most informative APIs for revealing ransomware behaviors, we conduct a feature importance analysis based on a random forest model (using setup 2). The trained model

Table 13. API Scores Calculated Based on the Usage of Top Ransomware APIs and Top Benign APIs

	Sample	RW API Score	Benign API Score	API Contrast Score
R	Hive (47db)	8	-7	1
	LockBit (a2ad)	8	-15	-7
	LockBit (dec4)	7	-15	-8
	LockFile (2a23)	3	-11	-8
	Ryuk (9eb7)	1	0	1
	Ryuk (40b8)	7	-3	4
	Sodinokibi (9b11)	5	-12	-7
	Sodinokibi (fd16)	6	-12	-6
	VirLock (7a92)	9	-2	7
	VirLock (f4b1)	11	-4	7
	MountLocker (5eae)	3	-1	2
	Karma (6c98)	0	0	0
	AvosLocker (7188)	6	-4	2
	AvosLocker (f810)	6	-4	2
	Dharma (dc5b)	9	-10	-1
	DoejoCrypt (e044)	0	0	0
	Mydoom (dd28)	0	-3	-3
	Sage (ac27)	2	-5	-3
	SunCrypt (759f)	3	0	3
B	Chrome	1	-10	-9
	Git	2	-17	-15
	Notepad++	0	-14	-14
	TeamViewer	4	-17	-13
	Bitdefender	2	-12	-10
	PaintNet	0	-7	-7
	iCloud	0	-10	-10
	OneDrive	1	-16	-15
	Skype	2	-16	-14
	ScreenSplit	3	-14	-11

The calculation process is described in Section 6.1. RW stands for ransomware. Ransomware API score is calculated based on APIs in sets R and O. Benign API score is calculated based on APIs in set B. API contrast score is the final score, shown in Figure 5, from adding benign score to ransomware score. The lower the final score, the more likely the sample is benign.

achieves 0.99 ransomware class recall and precision using API call frequencies during the whole execution. We then manually analyze the top 29 APIs (Table 15 in the appendix) with their usage context and compare with benign application statistics. Next, we present a few detailed case studies of the APIs, differentiating the usage between ransomware and benign samples.

Persistence (NtDeleteKey). We observe that the lack of registry key deletion is a feature of ransomware maintaining persistence after the attack. Windows registry is a database that keeps important information related to the operation of the system and services running in the system.

- **Ransomware:** Ransomware creates a registry value in the “Run” subkey for auto-launching after the system reboots. In our experiments, we observed that 343 ransomware samples called `NtOpenKey` and 71 called `NtCreateKey`, whereas only 4 called `NtDeleteKey`.
- **Benign:** On the opposite, most benign software calls `NtDeleteKey` at the end of execution to delete any registry keys they opened (via `NtOpenKey`) or created (via `NtCreateKey`). Among 330 benign applications that called `NtCreateKey`, 319 of them called `NtOpenKey`, and 310 called `NtDeleteKey`. A few exceptions of benign applications also exist, such as Norton and Viber, which launch at Windows startup and thus keep their registry keys.

Kernel security driver access (`DeviceIoControl`). Our experiments show that ransomware often uses control code “3735560”, which is related to the kernel security driver, when calling `DeviceIoControl` API. `DeviceIoControl` is used by programs to interact with device drivers in the system. Sending a specific control code will cause the corresponding driver to perform corresponding actions.

- **Ransomware:** 149 out of 348 ransomware samples invoked `DeviceIoControl` during execution. Among those samples, 79% samples used 3735560 (0x390008 in hexadecimal) for control code, in which 0x39 corresponds to the macros “`IOCTL_KSEC_RANDOM_FILL_BUFFER`”, or “`IOCTL_KSEC_RNG_REKEY`”. “KSEC” stands for Kernal SECurity. The driver contains security and crypto-related functions, which are potentially used by ransomware samples for encryption key generation.
- **Benign:** 321 out of 330 benign software invoked `DeviceIoControl`. Only 40% of them used control code 3735560. The majority of benign samples invoked the API with code 589916 (0x9005C) and 590016 (0x900C0), with 0x9 referring to the file system (`FILE_DEVICE_FILE_SYSTEM`).

Besides, foreground-related API invocation differences may also serve as useful features for classification. For example, ransomware checks currently active program less frequently than benign ones, mainly for anti-analysis purposes. Only 82 ransomware samples (24%) used the `GetForegroundWindow` API, among which around 70% of samples belong to only a few families (i.e., LockBit, Stop, Ryuk, and Venus). In benign scenarios, We observe 329 benign applications invoked `GetForegroundWindow`. Another helpful API is `DrawTextExW`, which is for front-end formatting. Only 59 of 348 ransomware samples (17%) call `DrawTextExW` with low frequency. On contrast, 329 out of 330 benign samples make use of it, with an average call of 2256 times and a high of 49000.

6.6 Comparison with Benign File Operations

In this section, we compare several intensive file operations for benign purposes with ransomware. Benign software can also be designed to handle a significant amount of file operations, such as backup. To investigate how to differentiate such benign behaviors from ransomware, we manually run a set of file operations using the 7zip file manager. The operations we perform include compressing, copying, moving, encrypting, extracting, and deleting. Each operation is performed on 1,000 to 3,000 files in at least 3 distinct directories. The encryption algorithm used is AES-256.

While some file operations have a repetitive execution period, they can be distinguished from malicious behaviors in a few ways (Figure 6 in the appendix). Copying (Figure 6(b)) and moving (Figure 6(c)) a large number of files show a few repetitive execution windows. However, the API composition is simpler during these periods, consisting of only 3 APIs, while typical ransomware execution uses 6 to 10. For extracting files (Figure 6(e)), the vast majority of calls were made to a single API, namely `NtWriteFile`. The execution process of compression (Figure 6(a)) and encryption (Figure 6(d)) has a 2-stage pattern, calling `NtOpenFile` and `NtQueryDirectoryFile` first for preparation and then using `NtReadFile`, `NtCreateFile`, and `GetFileInformationByHandle`

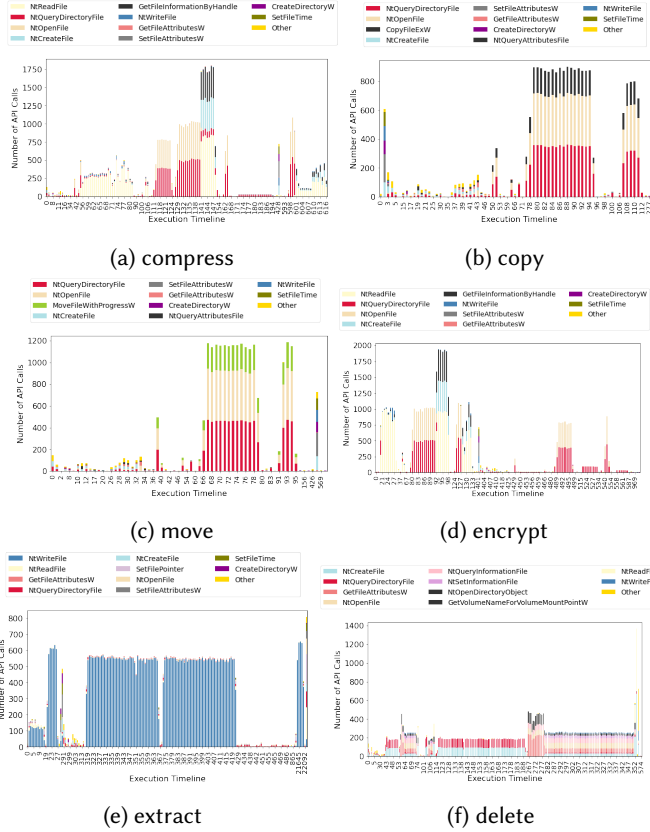


Fig. 6. File-related API call statistics of benign file operations (performed using the 7zip file manager). The same colors represent the same APIs across subfigures.

for the operations. Deleting also has this 2-stage feature with the usage of a different set of APIs (Figure 6(f)). The repetitive period lasts longer in the case of deletion, but the call frequency is also lower for each second.

On the other hand, when looking at the top ransomware and benign API usage scores (described in Sections 6.1 and 6.4), we can also distinguish the file operations from malicious behaviors. The highest ransomware API score among all 6 operations is 1 (out of 18), suggesting the benignness. None of the operations used any of the top prevalent ransomware APIs. Only 2 of them exceed the frequency threshold of `RegDeleteValueW`. Furthermore, when looking at the top benign APIs, the scores range from -13 to -14, falling in the cluster of benign programs (benign scores are shown in Table 13).

In summary, with further inspection, it is possible to separate benign file accesses from malicious ransomware behaviors. However, execution patterns depend on the implementation of specific programs and ransomware could evolve to mimic benign behaviors. Therefore, while being helpful, those observations might not generalize to all cases.

We summarize our experimental findings as follows:

- Ransomware has extremely high crypto API and file API usage frequency during execution, up to 5,000 times a second, with a unique high repetition pattern throughout the execution process, which we leverage for detection.

- The usage patterns of specific APIs exhibit disparities between ransomware and benign programs. This variation helps our classification further evaluate the risk.
- Our feasibility study shows that our two-stage API-based classification achieves perfect precision and recall in differentiating ransomware execution from benign. Our consistency-based detection, capturing the unique ransomware file API usage features, effectively recognizes all malicious execution during evaluation, with only 1 false alarm. The API contrast score, as a second step, successfully filters the false positives.
- For commercial defense, the efficacy and comprehensiveness of decryptors are very limited, with only 1 out of 6 tested decryptors successfully recovering infected files. Malware scanners use static signature-based detection. 69% complex password-protected ransomware samples evade the detection of all 70 malware scanners tested. Anti-virus software detects generic malware behaviors, overlooking the most essential ransomware encryption actions.

7 Discussion and Limitations

Deployability of API-based solutions. While API-based solutions do not require low-level system modifications, one challenge is the runtime overhead caused by API instrumentation. Previous work [105] reported that instrumenting over 300 APIs causes an average of 115% runtime overhead compared with uninstrumented baseline. This notable slowdown might demotivate users from applying such protection. Possible workarounds include selective API instrumentation and selective monitoring. Selective API instrumentation refers to only instrument a small set of the most important APIs. Research has shown that some APIs take more significant roles in the classification task [52]. We also discuss some top informative APIs in Section 6.5. Selective monitoring refers to only monitor a program exclusively during predetermined time intervals, such as only in the evenings. The length and frequency of monitoring time could also be determined based on the program's trust level.

Defense evasion. Newly emerged evasion strategies [59, 62] render many existing ransomware detections less effective. They split the malicious program into several subprocesses and make each process only do the small portion of the tasks. In this way, the traces of each process will contain less information about the attack and thus less likely to be detected. However, splitting and creating too many process could also be a flag. Monitoring the process hierarchy and analyzing subprocesses created by the same parent could also help identify malicious patterns. Imitation-based attack against kernel-level I/O analysis has also been crafted in recent years [168]. Multi-layer protection, i.e., incorporating API monitoring into I/O based detection, was purposed by the authors as a possible counter-measurement.

Detecting core ransomware-specific features. Static signature-based detection, while effective in catching known malware samples, falls short in detecting any variants. For behavioral detection, although there are many prevalent features in ransomware executions, such as prevention of recovery, none of them is required for launching an attack and keeping the data hostage, i.e., not the core behaviors of ransomware and can evolve at any time. However, in our experiments, none of the antivirus software reacts to intensive encryption or file modification. SonicWall reports that [149], on average, 9.7 ransomware attacks are attempted every business day for a company running in the US. Although generic detection catches many threats, with such a huge base number, even a small escape rate could cause big problems.

Highly deployable commercial antivirus products commonly rely on static signatures on known samples or specific sequences of behaviors. As shown in our evaluation, while a majority of antivirus tools detect plain versions of ransomware samples, the detection success rate drops sharply for even mildly obfuscated variants. This illustrates a critical security gap that current defense needs more resilience against statically modified but behaviorally similar malware.

Our classification focuses on identifying the unique execution pattern during the rapid encryption of data. This feature holds for the majority of current ransomware, as encryption is the core of ransomware and slower procedures increase the likelihood of detection and termination. Our method may miss newly emerged samples that exhibit significantly different functionalities, which is a limitation shared by feature-based detection approaches. However, because encryption is the fundamental nature of crypto-ransomware and is not easily discardable, we believe that it will remain a key feature of ransomware attacks.

Moreover, it is worth noting that different techniques and implementations can be used to achieve a single function, resulting in different traces for analysis. Strategies built using specific traces may not generalize well to other malicious samples or families. In our consistency analysis, we address this issue by distilling the core repetitive patterns from the execution. This enables the identification of the encryption process utilizing diverse API combinations.

While prior API-based approaches often train on raw occurrences or sequences with limited interpretability, our classification targets the fundamental behavioral signature of the encryption process, enabling better generalization to previously unseen ransomware samples. This also offers insights into why a sample is flagged as malicious. Although continuous API monitoring may introduce overhead, our study also reveals a subset of APIs that are more indicative of ransomware behavior, suggesting open opportunities for selective monitoring to mitigate performance cost.

Limitations. Our work has several limitations due to the usage of sandbox and VM. The sandbox cannot catch the dynamically resolved APIs and I/O through memory mapping, resulting in several reports with only a few records, even when we observe numerous infected files in the system. We were also unable to retrieve the memory access behaviors, which will be our future work. In many reports, hundreds of records have exactly the same timestamp, making it impossible to calculate the inter-arrival time of some API sequences. Thus, we only report the call frequencies per second. In some cases, the time interval between the first and last API call in the report exceeds the total execution time. In these cases, we omit the exceeding part.

Our second limitation involves the possibility of new attack strategies evading our rule- and statistic-based detection. Our second stage API contrast score, despite being helpful, considers only non-file APIs and should not be used alone. The reason is that some behaviors, while common, are not essential for launching a ransomware attack. The hackers can selectively discard or incorporate specific functions to bypass this detection stage. Like other detection works relying on specific API usage, the effect of system updates and environment changes needs to be carefully handled.

Additionally, in our implementation, we manually set the thresholds for selecting APIs. This poses another challenge to maintain detection accuracy against evolving attack patterns. Similar to various machine learning models that require periodic retraining to address concept drift, our decision thresholds will require future adjustments to adapt to emerging attacks. Alternative thresholds or additional selection criteria should also be explored in the future to enhance the effectiveness.

8 Conclusion

Development of deployable ransomware protection is crucial for future research. Our work focuses on discussing the deployability of existing solutions and providing insights on API detections based on ransomware behaviors profiling. We also report new insights from our in-depth API case studies and the evaluation of commercial defenses, which previously have not been reported in the literature. Ongoing work is focused on addressing overhead-related deployment challenges.

References

- [1] Jitti Annie Abraham and Susan M. George. 2019. A survey on preventing crypto ransomware using machine learning. In *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICT)*, Vol. 1. IEEE, 259–263.

- [2] Ryan Prescott Adams and David J. C. MacKay. 2007. Bayesian Online Changepoint Detection. Retrieved from <https://arxiv.org/abs/0710.3742>
- [3] Mohammad Mehdi Ahmadian and Hamid Reza Shahriari. 2016. ZentFOX: A framework for high survivable ransomwares detection. In *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. IEEE, 79–84.
- [4] Mohammad Mehdi Ahmadian, Hamid Reza Shahriari, and Seyed Mohammad Ghaffarian. 2015. Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares. In *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. IEEE, 79–84.
- [5] Muhammad Ejaz Ahmed, Hyoungshick Kim, Seyit Camtepe, and Surya Nepal. 2021. Peeler: Profiling kernel-level events to detect ransomware. In *Computer Security – ESORICS 2021*, Elisa Bertino, Haya Shulman, and Michael Waidner (Eds.). Cham, 240–260.
- [6] Usman Ahmed, Jerry Chun-Wei Lin, and Gautam Srivastava. 2022. Mitigating adversarial evasion attacks of ransomware using ensemble learning. *Computers and Electrical Engineering* 100 (2022), 107903.
- [7] Yahye Abukar Ahmed, Baris Kocer, and Bander Ali Saleh Al-rimy. 2020. Automated analysis approach for the detection of high survivable ransomware. *KSII Transactions on Internet and Information Systems (TIIS)* 14, 5 (2020), 2236–2257.
- [8] Yahye Abukar Ahmed, Barış Koçer, Shamsul Huda, Bander Ali Saleh Al-rimy, and Mohammad Mehdi Hassan. 2020. A system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection. *Journal of Network and Computer Applications* 167 (2020), 102753.
- [9] Bander Ali Saleh Al-Rimy, Mohd Aizaini Maarof, Mamoun Alazab, Fawaz Alsolami, Syed Zainudeen Mohd Shaid, Fuad A Ghaleb, Tawfik Al-Hadhrani, and Abdullah Marish Ali. 2020. A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. *IEEE Access* 8 (2020), 140586–140598.
- [10] Bander Ali Saleh Al-Rimy, Mohd Aizaini Maarof, Mamoun Alazab, Syed Zainudeen Mohd Shaid, Fuad A. Ghaleb, Abdulmohsen Almalawi, Abdullah Marish Ali, and Tawfik Al-Hadhrani. 2021. Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection. *Future Generation Computer Systems* 115 (2021), 641–658.
- [11] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, Yuli Adam Prasetyo, Syed Zainudeen Mohd Shaid, and Asmawi Fadillah Mohd Ariffin. 2018. Zero-day aware decision fusion-based model for crypto-ransomware early detection. *International Journal of Integrated Engineering* 10, 6 (2018), 82–88.
- [12] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. 2018. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers and Security* 74 (5 2018), 144–166.
- [13] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. 2019. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Generation Computer Systems* 101 (2019), 476–491. tt DOI:<https://doi.org/10.1016/j.future.2019.06.005>
- [14] Manaar Alam, Sayan Sinha, Sarani Bhattacharya, Swastika Dutta, Debdeep Mukhopadhyay, and Anupam Chattopadhyay. 2020. Rapper: Ransomware prevention via performance counters. arXiv:2004.01712. Retrieved from <https://arxiv.org/abs/2004.01712>
- [15] Fatimah Aldauji, Omar Batarfi, and Manal Bayousef. 2022. Utilizing cyber threat hunting techniques to find ransomware attacks: A survey of the state of the art. *IEEE Access* 10 (2022), 61695–61706.
- [16] Omar M. K. Alhawi, James Baldwin, and Ali Dehghantanha. 2018. Leveraging machine learning techniques for windows ransomware network traffic detection. *Advances in Information Security* 70 (2018), 93–106.
- [17] Ahmad O. Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O’Kane. 2019. A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE Access* 7 (2019), 47053–47067.
- [18] May Almousa, Sai Basavaraju, and Mohd Anwar. 2021. API-based ransomware detection using machine learning-based threat detection models. In *2021 18th International Conference on Privacy, Security and Trust (PST)*. 1–7.
- [19] Abdullah Alqahtani and Frederick T. Sheldon. 2022. A survey of crypto ransomware attack detection methodologies: an evolving outlook. *Sensors* 22, 5 (2022), 1837.
- [20] Samah Alsoghyer and Iman Almomani. 2019. Ransomware detection system for android applications. *Electronics* 8, 8 (2019). tt DOI:<https://doi.org/10.3390/electronics8080868>
- [21] Samah Alsoghyer and Iman Almomani. 2020. On the effectiveness of application permissions for Android ransomware detection. In *2020 6th Conference on Data Science and Machine Learning Applications (CDMA)*. IEEE, 94–99.
- [22] Ali Ahmed Mohammed Ali Alwashali, Nor Azlina Abd Rahman, and Noris Ismail. 2021. A survey of ransomware as a service (RaaS) and methods to mitigate the attack. In *Proceedings of the International Conference on Developments in eSystems Engineering, DeSE*, Vol. 2021-December. 92–96.
- [23] Abdulrahman Alzahrani, Hani Alshahrani, Ali Alshehri, and Huirong Fu. 2019. An intelligent behavior-based ransomware detection system for android platform. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. 28–35. tt DOI:<https://doi.org/10.1109/TPS-ISA48467.2019.00013>

- [24] Abdulrahman Alzahrani, Ali Alshehri, Hani Alshahrani, Raed Alharthi, Huirong Fu, Anyi Liu, and Ye Zhu. 2018. RanDroid: Structural similarity approach for detecting ransomware applications in Android platform. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 0892–0897.
- [25] Nisreen Alzahrani and Daniyal Alghazzawi. 2019. A review on android ransomware detection using deep learning techniques. In *Proceedings of the 11th International Conference on Management of Digital EcoSystems*. 330–335.
- [26] Nicolás Andronio, Stefano Zanero, and Federico Maggi. 2015. Heldroid: Dissecting and detecting mobile ransomware. In *Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan, November 2-4, 2015. Proceedings 18*. Springer, 382–404.
- [27] Sana Aurangzeb, Muhammad Aleem, Muhammad Azhar Iqbal, Muhammad Arshad Islam, et al. 2017. Ransomware: A survey and trends. *Journal of Information Assurance & Security* 6, 2 (2017), 48–58.
- [28] Sana Aurangzeb, Haris Anwar, Muhammad Asif Naeem, and Muhammad Aleem. 2022. BigRC-EML: Big-data based ransomware classification using ensemble machine learning. *Cluster Computing* 25, 5 (2022), 3405–3422.
- [29] Sana Aurangzeb, Rao Naveed Bin Rais, Muhammad Aleem, Muhammad Arshad Islam, and Muhammad Azhar Iqbal. 2021. On the classification of Microsoft-Windows ransomware using hardware profile. *PeerJ Computer Science* 7 (2 2021).
- [30] Md Ahsan Ayub, Andrea Continella, and Ambareen Siraj. 2020. An I/O request packet (IRP) driven effective ransomware detection scheme using artificial neural network. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 319–324.
- [31] Md. Ahsan Ayub, Ambareen Siraj, Bobby Filar, and Maanak Gupta. 2024. RWArmor: A static-informed dynamic analysis approach for early detection of cryptographic windows ransomware. *International Journal of Information Security* 23, 1 (2024), 533–556. DOI: <https://doi.org/10.1007/s10207-023-00758-z>
- [32] Seong Il Bae, Gyu Bin Lee, and Eul Gyu Im. 2020. Ransomware detection using machine learning algorithms. *Concurrency and Computation: Practice and Experience* 32, 18 (2020), e5422.
- [33] Sungha Baek, Youngdon Jung, Aziz Mohaisen, Sungjin Lee, and Daehun Nyang. 2018. SSD-Insider: Internal defense of solid-state drive against ransomware with perfect data recovery. In *Proceedings of International Conference on Distributed Computing Systems* (Vienna, Austria). 875–884.
- [34] Pranshu Bajpai, Aditya K. Sood, and Richard Enbody. 2018. A key-management-based taxonomy for ransomware. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–12.
- [35] Craig Beaman, Ashley Barkworth, Toluwalope David Akande, Saqib Hakak, and Muhammad Khurram Khan. 2021. Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & Security* 111 (12 2021).
- [36] Kenan Begovic, Abdulaziz Al-Ali, and Qutaibah Malluhi. 2023. Cryptographic ransomware encryption detection: Survey. *Computers & Security* 132 (2023), 103349.
- [37] Mourad Benmalek. 2024. Ransomware on cyber-physical systems: Taxonomies, case studies, security gaps, and open challenges. *Internet of Things and Cyber-Physical Systems* 4 (2024), 186–202.
- [38] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, and Mikel Izal. 2019. A survey on detection techniques for cryptographic ransomware. *IEEE Access* 7 (2019), 144925–144944.
- [39] CV Bijitha, Rohit Sukumaran, and Hiran V. Nath. 2020. A survey on ransomware detection techniques. In *Secure Knowledge Management In Artificial Intelligence Era: 8th International Conference, SKM 2019, Goa, India, December 21–22, 2019, Proceedings 8*. Springer, 55–68.
- [40] Riccardo Bortolameotti, Thijs van Ede, Marco Caselli, Maarten H. Everts, Pieter Hartel, Rick Hofstede, Willem Jonker, and Andreas Peter. 2017. Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. In *Proceedings of the 33rd Annual computer security applications Conference*. 373–386.
- [41] Krzysztof Cabaj, Marcin Gregorczyk, and Wojciech Mazurczyk. 2018. Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. *Computers & Electrical Engineering* 66 (2018), 353–368.
- [42] Krzysztof Cabaj and Wojciech Mazurczyk. 2016. Using software-defined networking for ransomware mitigation: The case of cryptowall. *IEEE Network* 30, 6 (11 2016), 14–20.
- [43] Jason Castiglione and Dusko Pavlovic. 2019. Dynamic distributed secure storage against ransomware. *IEEE Transactions on Computational Social Systems* 7, 6 (2019), 1469–1475.
- [44] Alberto Huertas Celdrán, Pedro Miguel Sánchez Sánchez, Jan von der Assen, Dennis Shushack, Ángel Luis Perales Gómez, Jérôme Bovet, Gregorio Martínez Pérez, and Burkhard Stiller. 2023. Behavioral fingerprinting to detect ransomware in resource-constrained devices. *Computers & Security* 135 (2023), 103510.
- [45] Mingcan Cen, Frank Jiang, Xingsheng Qin, Qinghong Jiang, and Robin Doss. 2024. Ransomware early detection: A survey. *Computer Networks* 239 (2024), 110138.
- [46] Jing Chen, Chiheng Wang, Ziming Zhao, Kai Chen, Ruiying Du, and Gail-Joon Ahn. 2017. Uncovering the face of android ransomware: Characterization and real-time detection. *IEEE Transactions on Information Forensics and Security* 13, 5 (2017), 1286–1300.

- [47] Qian Chen and Robert A. Bridges. 2017. Automated behavioral analysis of malware: A case study of wannacry ransomware. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 454–460.
- [48] Qian Chen, Sheikh Rabiul Islam, Henry Haswell, and Robert A. Bridges. 2019. Automated ransomware behavior analysis: Pattern extraction and early detection. In *Science of Cyber Security: Second International Conference, SciSec 2019, Nanjing, China, August 9–11, 2019, Revised Selected Papers 2*. Springer, 199–214.
- [49] Zhi-Guo Chen, Ho-Seok Kang, Shang-Nan Yin, and Sung-Ryul Kim. 2017. Automatic ransomware detection and analysis based on dynamic API calls flow graph. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems (RACS'17)*. Association for Computing Machinery, Krakow, Poland, 196–201.
- [50] Guang Cheng, Chunsheng Guo, and Yongning Tang. 2019. dptCry: An approach to decrypting ransomware WannaCry based on API hooking. *CCF Transactions on Networking* 2, 3 (12 2019), 207–216.
- [51] Aniello Cimitile, Francesco Mercaldo, Vittoria Nardone, Antonella Santone, and Corrado Aaron Visaggio. 2018. Talos: No more ransomware victims with formal methods. *International Journal of Information Security* 17, 6 (2018), 719–738.
- [52] Filippo Coglio, Ahmed Lekssays, Barbara Carminati, and Elena Ferrari. 2023. Early-stage ransomware detection based on pre-attack internal API calls. In *Advanced Information Networking and Applications*, Leonard Barolli (Ed.). Springer International Publishing, Cham, 417–429.
- [53] Aviad Cohen and Nir Nissim. 2018. Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. *Expert Systems with Applications* 102 (2018), 158–178.
- [54] Kevin Collier. [n.d.]. Cyberattack cost MGM Resorts about \$100 million, Las Vegas company says. Retrieved from <https://www.nbcnews.com/business/business-news/cyberattack-cost-mgm-resorts-100-million-las-vegas-company-says-rcna119138>
- [55] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barengi, Stefano Zanero, and Federico Maggi. 2016. ShieldFS: A self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications (Los Angeles, California, USA)*. 336–347.
- [56] Greg Cusack, Oliver Michel, and Eric Keller. 2018. Machine learning-based detection of ransomware using SDN. In *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. 1–6.
- [57] Alfredo Cuzzocrea, Fabio Martinelli, and Francesco Mercaldo. 2018. A novel structural-entropy-based classification technique for supporting android ransomware detection and analysis. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 1–7.
- [58] Tooska Dargahi, Ali Dehghantanha, Pooneh Nikkhah Bahrami, Mauro Conti, Giuseppe Bianchi, and Loris Benedetto. 2019. A cyber-kill-chain based taxonomy of crypto-ransomware features. *Journal of Computer Virology and Hacking Techniques* 15, 4 (2019), 277–305.
- [59] Fabio De Gaspari, Dorjan Hitaj, Giulio Pagnotta, Lorenzo De Carli, and Luigi V. Mancini. 2022. Evading behavioral classifiers: A comprehensive analysis on evading ransomware detection techniques. *Neural Computing and Applications* 34, 14 (2022), 12077–12096.
- [60] Usama Desai. 2019. A survey on android ransomware and its detection methods. *International Research Journal of Engineering and Technology* 6, 03 (2019), 3081–3087.
- [61] Abdulrahman Abu Elkhail, Nada Lachtar, Duha Ibdah, Rustam Aslam, Hamza Khan, Anys Bacha, and Hafiz Malik. 2023. Seamlessly safeguarding data against ransomware attacks. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 1–16.
- [62] Fenil Fadadu, Anand Handa, Nitesh Kumar, and Sandeep Kumar Shukla. 2020. Evading API call sequence based malware classifiers. In *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers 21*. Springer, 18–33.
- [63] Hossam Faris, Maria Habib, Iman Almomani, Mohammed Eshtay, and Ibrahim Aljarah. 2020. Optimizing extreme learning machines using chains of salps for efficient Android ransomware detection. *Applied Sciences* 10, 11 (2020), 3706.
- [64] Lorenzo Fernandez Maimo, Alberto Huertas Celdran, Angel L. Perales Gomez, Felix J. Garcia Clemente, James Weimer, and Insup Lee. 2019. Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments. *Sensors* 19, 5 (2019), 1114.
- [65] Damien Warren Fernando and Nikos Komninos. 2024. FeSAD ransomware detection framework with machine learning using adaption to concept drift. *Computers & Security* 137 (2024), 103629.
- [66] Alberto Ferrante, Mirosław Malek, Fabio Martinelli, Francesco Mercaldo, and Jelena Milosevic. 2018. Extinguishing ransomware—a hybrid approach to android ransomware detection. In *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23–25, 2017, Revised Selected Papers 10*. Springer, 242–258.
- [67] Dhruv Garg, Abha Thakral, Tarun Nalwa, and Tanupriya Choudhury. 2018. A past examination and future expectation: Ransomware. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*. IEEE, 243–247.

- [68] Mazen Gazzan and Frederick T. Sheldon. 2023. An enhanced minimax loss function technique in generative adversarial network for ransomware behavior prediction. *Future Internet* 15, 10 (2023), 318.
- [69] Amirhossein Gharib and Ali Ghorbani. 2017. Dna-droid: A real-time android ransomware detection framework. In *Network and System Security: 11th International Conference, NSS 2017, Helsinki, Finland, August 21–23, 2017, Proceedings 11*. Springer, 184–198.
- [70] Daniel Gonzalez and Thaier Hayajneh. 2017. Detection and prevention of crypto-ransomware. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE, 472–478.
- [71] J. A. Gómez-Hernández, L. Álvarez González, and P. García-Teodoro. 2018. R-Locker: Thwarting ransomware action through a honeypole-based approach. *Computers & Security* 73 (3 2018), 389–398.
- [72] Md Mahbub Hasan and Md Mahbubur Rahman. 2017. RansHunt: A support vector machines based ransomware analysis framework with integrated feature set. In *2017 20th International Conference of Computer and Information Technology (ICIT)*. IEEE, 1–7.
- [73] Gregory Hill and Xavier Bellekens. 2018. CryptoKnight: Generating and modelling compiled cryptographic primitives. *Information* 9, 9 (2018), 231.
- [74] Manabu Hirano and Ryotaro Kobayashi. 2019. Machine learning based ransomware detection using storage access patterns obtained from live-forensic hypervisor. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 1–6.
- [75] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, and Raouf Khayami. 2017. Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. *IEEE Transactions on Emerging Topics in Computing* 8, 2 (2017), 341–351.
- [76] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, Raouf Khayami, Kim-Kwang Raymond Choo, and David Ellis Newton. 2019. DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer. *Future Generation Computer Systems* 90 (2019), 94–104.
- [77] Toshiaki Honda, Kohei Mukaiyama, Takeharu Shirai, Tetsushi Ohki, and Masakatsu Nishigaki. 2018. Ransomware detection considering user's document editing. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 907–914.
- [78] Jian Huang, Jun Xu, Xinyu Xing, Peng Liu, and Moinuddin K. Qureshi. 2017. FlashGuard: Leveraging Intrinsic Flash Properties to Defend Against Encryption Ransomware. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA). 2231–2244.
- [79] Mamoon Humayun, NZ Jhanjhi, Ahmed Alsayat, and Vasaki Ponnusamy. 2021. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal* 22, 1 (2021), 105–117.
- [80] Jinsoo Hwang, Jeankyung Kim, Seunghwan Lee, and Kichang Kim. 2020. Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications* 112, 4 (6 2020), 2597–2609.
- [81] Jaime Ibarra, Usman Javed Butt, Anh Do, Hamid Jahankhani, and Arshad Jamal. 2019. Ransomware impact to SCADA systems and its scope to critical infrastructure. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*. IEEE, 1–12.
- [82] CrowdStrike Intelligence. 2022. How to Decrypt the PartyTicket Ransomware Targeting Ukraine. Retrieved from <https://www.crowdstrike.com/blog/how-to-decrypt-the-partyticket-ransomware-targeting-ukraine/>
- [83] Sangmoon Jung and Yoojae Won. 2018. Ransomware detection method based on context-aware entropy analysis. *Soft Computing* 22, 20 (2018), 6731–6740.
- [84] Alireza Karimi and Mohammad Hosein Moattar. 2017. Android ransomware detection using reduced opcode sequence and image similarity. In *2017 7th International Conference on Computer and Knowledge Engineering (ICCCKE)*. IEEE, 229–234.
- [85] Kaspersky. [n. d.]. Ransomware WannaCry: All you need to know. Retrieved from <https://usa.kaspersky.com/resource-center/threats/ransomware-wannacry>
- [86] Chee Keong Ng, Sutharshan Rajasegarar, Lei Pan, Frank Jiang, and Leo Yu Zhang. 2020. VoterChoice: A ransomware detection honeypot with multiple voting framework. *Concurrency and Computation: Practice and Experience* 32, 14 (2020), e5726.
- [87] Masoudeh Keshavarzi and Hamid Reza Ghaffary. 2020. I2CE3: A dedicated and separated attack chain for ransomware offenses as the most infamous cyber extortion. *Computer Science Review* 36 (2020), 100233.
- [88] Eleni Ketzaki, Petros Toupas, Konstantinos M. Giannoutakis, Anastasios Drosou, and Dimitrios Tzovaras. 2020. A behaviour based ransomware detection using neural network models. In *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*. IEEE, 747–750.
- [89] Firoz Khan, Cornelius Ncube, Lakshmana Kumar Ramasamy, Seifedine Kadry, and Yunyoung Nam. 2020. A digital DNA sequencing engine for ransomware detection using machine learning. *IEEE Access* 8 (2020), 119710–119719.

- [90] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. 2016. UNVEIL: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*. 757–772.
- [91] Dae-Youb Kim, Geun-Yeong Choi, and Ji-Hoon Lee. 2018. White list-based ransomware real-time detection and prevention for user device protection. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 1–5.
- [92] Giyoon Kim, Soram Kim, Soojin Kang, and Jongsung Kim. 2022. A Method for Decrypting Data Infected with Hive Ransomware. Retrieved from <https://arxiv.org/abs/2202.08477v1>
- [93] Muhammad Ubale Kiru and Aman B. Jantan. 2019. The age of ransomware: Understanding ransomware and its countermeasures. In *Artificial Intelligence and Security Challenges in Emerging Networks*. IGI Global, 1–37.
- [94] S. Kok, Azween Abdullah, N. Jhanjhi, and Mahadevan Supramaniam. 2019. Ransomware, threat and detection techniques: A review. *Int. J. Comput. Sci. Netw. Secur* 19, 2 (2019), 136.
- [95] S. H. Kok, A. Azween, and N. Z. Jhanjhi. 2020. Evaluation metric for crypto-ransomware detection using machine learning. *Journal of Information Security and Applications* 55 (2020), 102646.
- [96] S. H. Kok, Azween Abdullah, and N. Z. Jhanjhi. 2022. Early detection of crypto-ransomware using pre-encryption detection algorithm. *Journal of King Saud University—Computer and Information Sciences* 34, 5 (5 2022), 1984–1999.
- [97] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. 2017. PayBreak: Defense Against Cryptographic Ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (Abu Dhabi, United Arab Emirates) (ASIA CCS'17)*. 599–611.
- [98] Erik Larsen, David Noever, and Korey MacVittie. 2021. A Survey of Machine Learning Algorithms for Detecting Ransomware Encryption Activity. Retrieved from <https://arxiv.org/abs/2110.07636v1>
- [99] Jeonghwan Lee, Jinwoo Lee, and Jiman Hong. 2017. How to make efficient decoy files for ransomware detection?. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*. 208–212.
- [100] Kyungroul Lee, Sun-Young Lee, and Kangbin Yim. 2019. Machine learning based file entropy analysis for ransomware detection in backup systems. *IEEE Access* 7 (2019), 110205–110215.
- [101] Kyungroul Lee, Kangbin Yim, and Jung Taek Seo. 2018. Ransomware prevention technique using key backup. *Concurrency and Computation: Practice and Experience* 30, 3 (2018), e4337.
- [102] Sehoon Lee, Myungseo Park, and Jongsung Kim. 2020. Magniber v2 ransomware decryption: Exploiting the vulnerability of a self-developed pseudo random number generator. *Electronics* 2021, Vol. 10, Page 16 10, 1 (12 2020), 16.
- [103] Tianliang Lu, Yanhui Du, Jing Wu, and Yuxuan Bao. 2020. Ransomware detection based on an improved double-layer negative selection algorithm. In *Testbeds and Research Infrastructures for the Development of Networks and Communications: 14th EAI International Conference, TridentCom 2019, Changsha, China, December 7–8, 2019, Proceedings 14*. Springer, 46–61.
- [104] Tianliang Lu, Lu Zhang, Shunye Wang, and Qi Gong. 2017. Ransomware detection based on V-detector negative selection algorithm. In *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*. IEEE, 531–536.
- [105] Lorenzo Maffia, Dario Nisi, Platon Kotzias, Giovanni Lagorio, Simone Aonzo, and Davide Balzarotti. 2021. Longitudinal Study of the Prevalence of Malware Evasive Techniques. arXiv:2112.11289. Retrieved from <https://arxiv.org/abs/2112.11289>
- [106] Abdullahi Mohammed Maigida, Shafi'i Muhammad Abdulhamid, Morufu Olalere, John K. Alhassan, Haruna Chiroma, and Emmanuel Gbenga Dada. 2019. Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms. *Journal of Reliable Intelligent Environments* 5, 2 (2019), 67–89.
- [107] Davide Maiorca, Francesco Mercaldo, Giorgio Giacinto, Corrado Aaron Visaggio, and Fabio Martinelli. 2017. R-PackDroid: API package-based characterization and detection of mobile ransomware. In *Proceedings of the Symposium on Applied Computing*. 1718–1723.
- [108] Sumith Maniath, Aravind Ashok, Prabakaran Poornachandran, VG Sujadevi, Prem Sankar AU, and Srinath Jan. 2017. Deep learning LSTM based ransomware detection. In *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*. IEEE, 442–446.
- [109] Sumith Maniath, Prabakaran Poornachandran, and VG Sujadevi. 2019. Survey on prevention, mitigation and containment of ransomware attacks. In *Security in Computing and Communications: 6th International Symposium, SSCC 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 6*. Springer, 39–52.
- [110] Jean Yves Marion. 2025. Ransomware: Extortion is my business. *Commun. ACM* 68, 5 (5 2025), 36–47. tt DOI:<https://doi.org/10.1145/3697829>
- [111] Michael J. May and Etamar Laron. 2019. Combating ransomware using content analysis and complex file events. In *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 1–5.
- [112] Timothy McIntosh, A. S. M. Kayes, Yi-Ping Phoebe Chen, Alex Ng, and Paul Watters. 2023. Applying staged event-driven access control to combat ransomware. *Computers & Security* 128 (2023).

- [113] Timothy McIntosh, ASM Kayes, Yi-Ping Phoebe Chen, Alex Ng, and Paul Watters. 2023. Applying staged event-driven access control to combat ransomware. *Computers & Security* 128 (2023), 103160.
- [114] Timothy McIntosh, A. S. M. Kayes, Yi-Ping Phoebe Chen, Alex Ng, and Paul Watters. 2021. Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions. *ACM Comput. Surv.* 54, 9, Article 197 (oct 2021), 36 pages.
- [115] May Medhat, Samir Gaber, and Nashwa Abdelbaki. 2018. A new static-based framework for ransomware detection. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 710–715.
- [116] Shagufta Mehnaz, Anand Mudgerikar, and Elisa Bertino. 2018. RWGuard: A real-time detection system against cryptographic ransomware. In *Research in Attacks, Intrusions, and Defenses*. 114–136.
- [117] Jaimin Modi, Issa Traore, Asem Ghaleb, Karim Ganame, and Sherif Ahmed. 2020. Detecting ransomware in encrypted web traffic. In *Foundations and Practice of Security: 12th International Symposium, FPS 2019, Toulouse, France, November 5–7, 2019, Revised Selected Papers 12*. Springer, 345–353.
- [118] Jithin Chandra Mohan and Renuka Kumar. 2017. On the efficacy of android ransomware detection techniques: A survey. *International Journal of Pure and Applied Mathematics* 115, 8 (2017), 115–120.
- [119] Chris Moore. 2016. Detecting Ransomware with Honeypot Techniques. In *2016 Cybersecurity and Cyberforensics Conference (CCC)*. 77–81. tt DOI:<https://doi.org/10.1109/CCC.2016.14>
- [120] Daniel Morato, Eduardo Berrueta, Eduardo Magaña, and Mikel Izal. 2018. Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications* 124 (2018), 14–32.
- [121] Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Boudier, Nora Cuppens, and Jean-Louis Lanet. 2018. Ransomware’s early mitigation mechanisms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*. 1–10.
- [122] Routa Moussaileb, Nora Cuppens, Jean-Louis Lanet, and Hélène Le Boudier. 2021. A survey on windows-based ransomware taxonomy and detection mechanisms. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–36.
- [123] Ayesha Naseer, Riffat Mir, Azmat Mir, and Muhammad Aleem. 2020. Windows-based ransomware: A survey. *Journal of Information Assurance & Security* 15, 3 (2020), 107–125.
- [124] David Noever and Samantha Miller Noever. 2021. POSSE: Patterns of Systems During Software Encryption. *arXiv:2109.12162*. Retrieved from <https://arxiv.org/abs/2109.12162>
- [125] Mohammad N. Olaimat, Mohd Aizaini Maarof, and Bander Ali S. Al-Rimy. 2021. Ransomware anti-analysis and evasion techniques: A survey and research directions. *2021 3rd International Cyber Resilience Conference, CRC 2021 (1 2021)*.
- [126] Harun Oz, Ahmet Aris, Abbas Acar, Güliz Seray Tuncay, Leonardo Babun, and Selcuk Uluagac. 2023. RøB: Ransomware over modern web browsers. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 7073–7090.
- [127] Harun Oz, Ahmet Aris, Albert Levi, and A Selcuk Uluagac. 2022. A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.
- [128] Aurélien Palisse, Antoine Durand, Hélène Le Boudier, Colas Le Guernic, and Jean-Louis Lanet. 2017. Data aware defense (DaD): Towards a generic and practical ransomware countermeasure. In *Secure IT Systems: 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8–10, 2017, Proceedings 22*. Springer, 192–208.
- [129] Aurélien Palisse, Hélène Le Boudier, Jean-Louis Lanet, Colas Le Guernic, and Axel Legay. 2017. Ransomware and the legacy crypto API. In *Risks and Security of Internet and Systems: 11th International Conference, CRiSIS 2016, Roscoff, France, September 5–7, 2016, Revised Selected Papers 11*. Springer, 11–28.
- [130] Segun I. Popoola, Ujioghosa B. Iyekepolo, Samuel O. Ojewande, Faith O. Sweetwilliams, Samuel N. John, and Aderemi A. Atayero. 2017. Ransomware: Current trend, challenges, and research directions. In *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 1. 169–174.
- [131] Florian Quinkert, Thorsten Holz, KSM Tozammel Hossain, Emilio Ferrara, and Kristina Lerman. 2018. RAPTOR: Ransomware Attack PredicTOR. *arXiv:1803.01598*. Retrieved from <https://arxiv.org/abs/1803.01598>
- [132] Gowtham Ramesh and Anjali Menen. 2020. Automated dynamic approach for detecting ransomware using finite-state machine. *Decision Support Systems* 138 (2020), 113400.
- [133] Salwa Razaulla, Claude Fachkha, Christine Markarian, Amjad Gawanmeh, Wathiq Mansoor, Benjamin C. M. Fung, and Chadi Assi. 2023. The Age of Ransomware: A Survey on the Evolution, Taxonomy, and Research Directions. *IEEE Access* 11 (2023), 40698–40723. tt DOI:<https://doi.org/10.1109/ACCESS.2023.3268535>
- [134] Bheemidi Vikram Reddy, Gutha Jaya Krishna, Vadamani Ravi, and Dipankar Dasgupta. 2021. Machine learning and feature selection based ransomware detection using hexacodes. In *Evolution in Computational Intelligence: Frontiers in Intelligent Computing: Theory and Applications (FICTA 2020), Volume 1*. Springer, 583–597.

- [135] Habib ur Rehman, Eiad Yafi, Mohammed Nazir, and Khurram Mustafa. 2019. Security assurance against cybercrime ransomware. In *Intelligent Computing & Optimization 1*. Springer, 21–34.
- [136] Benjamin Reidys, Peng Liu, and Jian Huang. [n. d.]. 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 (Lausanne, Switzerland) (ASPLOS'22)*. New York, NY, USA, 726–739.
- [137] Krishna Chandra Roy and Qian Chen. 2021. DeepRan: Attention-based BiLSTM and CRF for ransomware early detection and classification. *Information Systems Frontiers* 23, 2 (2021), 299–315.
- [138] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin R. B. Butler. 2016. CryptoLock (and drop it): Stopping ransomware attacks on user data. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 303–312.
- [139] Michele Scalas, Davide Maiorca, Francesco Mercaldo, Corrado Aaron Visaggio, Fabio Martinelli, and Giorgio Giacinto. 2019. On the effectiveness of system API-related information for Android ransomware detection. *Computers & Security* 86 (2019), 168–182.
- [140] ScramBox. 2016. How long would it take to brute force AES-256? Retrieved from <https://scrambox.com/article/brute-force-aes/>
- [141] Christoph Sendner, Lukas Iffländer, Sebastian Schindler, Michael Jobst, Alexandra Dmitrienko, and Samuel Kounev. 2022. Ransomware detection in databases through dynamic analysis of query sequences. In *2022 IEEE Conference on Communications and Network Security (CNS)*. 326–334. tt DOI:<https://doi.org/10.1109/CNS56114.2022.9947244>
- [142] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C. Lupu. 2016. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. arXiv:1609.03020. Retrieved from <https://arxiv.org/abs/1609.03020>
- [143] Shaila Sharmeen, Yahye Abukar Ahmed, Shamsul Huda, Bari Ş Koçer, and Mohammad Mehedi Hassan. 2020. Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches. *IEEE Access* 8 (2020), 24522–24534.
- [144] Saiyed Kashif Shaukat and Vinay J. Ribeiro. 2018. RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 356–363.
- [145] Hao Shi, Jelena Mirkovic, and Abdulla Alwabel. 2017. Handling anti-virtual machine techniques in malicious software. *ACM Transactions on Privacy and Security (TOPS)* 21, 1 (12 2017), 31.
- [146] Rhythima Shinde, Pieter Van der Veeke, Stijn Van Schooten, and Jan van den Berg. 2016. Ransomware: Studying transfer and mitigation. In *2016 International Conference on Computing, Analytics and Security Trends (CAST)*. IEEE, 90–95.
- [147] Juan A. Herrera Silva, Lorena Isabel Barona López, Ángel Leonardo Valdivieso Caraguay, and Myriam Hernández-Álvarez. 2019. A survey on situational awareness of ransomware attacks–detection and prevention parameters. *Remote Sensing* 11, 10 (2019), 1168.
- [148] Jaskaran Singh, Keshav Sharma, Mohammad Wazid, and Ashok Kumar Das. 2023. SINN-RD: Spline interpolation-envisioned neural network-based ransomware detection scheme. *Computers and Electrical Engineering* 106 (2023), 108601.
- [149] SonicWall. 2021. ‘The Year of Ransomware’ Continues with Unprecedented Late-Summer Surge. Retrieved from <https://www.sonicwall.com/news/sonicwall-the-year-of-ransomware-continues-with-unprecedented-late-summer-surge/>
- [150] Sophos. [n. d.]. The State of Ransomware 2021. Retrieved from <https://secure2.sophos.com/en-us/medialibrary/pdfs/whitepaper/sophos-state-of-ransomware-2021-wp.pdf>
- [151] Sophos. 2018. Businesses Impacted by Repeated Ransomware Attacks and Failing to Close the Gap on Exploits, According to Sophos Global Survey. Retrieved from <https://www.sophos.com/en-us/press-office/press-releases/2018/01/businesses-impacted-by-repeated-ransomware-attacks-according-to-sophos-global-survey>
- [152] Dan Su, Jiqiang Liu, Xiaoyang Wang, and Wei Wang. 2018. Detecting android locker-ransomware on Chinese social networks. *IEEE Access* 7 (2018), 20381–20393.
- [153] Hirra Sultan, Aqeel Khaliq, Safdar Tanweer, and Shah Imran Alam. 2018. A survey on ransomware: Evolution, growth, and impact. *International Journal of Advanced Research in Computer Science* 9, 2 (4 2018), 802–810.
- [154] Yuki Takeuchi, Kazuya Sakai, and Satoshi Fukumoto. 2018. Detecting ransomware using support vector machines. In *Workshop Proceedings of the 47th International Conference on Parallel Processing (Eugene, OR, USA) (ICPP Workshops'18)*. Article 1, 6 pages.
- [155] Fei Tang, Boyang Ma, Jinku Li, Fengwei Zhang, Jipeng Su, and Jianfeng Ma. 2020. RansomSpector: An introspection-based approach to detect crypto ransomware. *Computers & Security* 97 (2020), 101997.
- [156] Kumar Thummapudi, Palden Lama, and Rajendra V. Boppana. 2023. Detection of ransomware attacks using processor and disk usage data. *IEEE Access* 11 (2023), 51395–51407. tt DOI:<https://doi.org/10.1109/ACCESS.2023.3279819>

- [157] The New York Times. 2021. Colonial Pipeline Paid Roughly \$5 Million in Bitcoin to Hackers. Retrieved from <https://www.nytimes.com/2021/05/13/technology/colonial-pipeline-ransom.html>
- [158] Mayank Verma, Ponnurangam Kumarguru, Shuva Brata Deb, and Anuradha Gupta. 2018. Analysing indicator of compromises for ransomware: Leveraging IOCs with machine learning techniques. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 154–159.
- [159] R. Vinayakumar, K. P. Soman, K. K. Senthil Velan, and Shaunak Ganorkar. 2017. Evaluating shallow and deep networks for ransomware detection and classification. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 259–265.
- [160] Peiying Wang, Shijie Jia, Bo Chen, Luning Xia, and Peng Liu. 2019. MimosafTL: Adding secure and practical ransomware defense strategy to flash translation layer. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy* (Richardson, Texas, USA) (CODASPY'19). New York, NY, USA, 327–338.
- [161] Zihan Wang, ChaoGe Liu, Jing Qiu, ZhiHong Tian, Xiang Cui, and Shen Su. 2018. Automatically traceback RDP-based targeted ransomware attacks. *Wireless Communications and Mobile Computing* 2018, 1 (2018), 1–13.
- [162] Azka Wani and S. Revathi. 2020. Ransomware protection in IoT using software defined networking. *Int. J. Electr. Comput. Eng* 10, 3 (2020), 3166–3175.
- [163] Bahaa Yamany, Mahmoud Said Elsayed, Anca D. Jurcut, Nashwa Abdelbaki, and Marianne A. Azer. 2022. A new scheme for ransomware classification and clustering using static features. *Electronics* 11, 20 (2022), 3307.
- [164] Miuyin Yong Wong, Matthew Landen, Manos Antonakakis, Douglas M. Blough, Elissa M. Redmiles, and Mustaque Ahamad. 2021. An inside look into the practice of malware analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea). 3053–3069.
- [165] Javier Yuste and Sergio Pastrana. 2021. Avaddon ransomware: An in-depth analysis and decryption of infected systems. *Computers & Security* 109 (2021).
- [166] Bin Zhang, Wentao Xiao, Xi Xiao, Arun Kumar Sangaiah, Weizhe Zhang, and Jiajia Zhang. 2020. Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes. *Future Generation Computer Systems* 110 (2020), 708–720.
- [167] Hanqi Zhang, Xi Xiao, Francesco Mercaldo, Shiguang Ni, Fabio Martinelli, and Arun Kumar Sangaiah. 2019. Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Generation Computer Systems* 90 (2019), 211–221.
- [168] Chijin Zhou, Lihua Guo, Yiwei Hou, Zhenya Ma, Quan Zhang, Mingzhe Wang, Zhe Liu, and Yu Jiang. 2023. Limits of I/O based ransomware detection: An imitation based attack. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2584–2601.
- [169] Jiaying Zhou, Miyuki Hirose, Yoshio Kakizaki, and Atsuo Inomata. 2020. Evaluation to classify ransomware variants based on correlations between APIs. In *ICISSP*. 465–472.
- [170] Aaron Zimba and Mumbi Chishimba. 2019. Understanding the evolution of ransomware: paradigm shifts in attack structures. *International Journal of Computer Network and Information Security* 11, 1 (2019), 26.
- [171] Hiba Zuhair and Ali Selamat. 2019. RANDS: A machine learning-based anti-ransomware tool for windows platforms. In *Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques*. IOS Press, 573–587.
- [172] Hiba Zuhair, Ali Selamat, and Ondrej Krejcar. 2020. A multi-tier streaming analytics model of 0-day ransomware detection using machine learning. *Applied Sciences* 10, 9 (2020), 3210.

Appendix

Table 14. Evaluation Results of Commercial Malware Scanners

Ransomware Family	SHA256	Plain File	Simple Password	Complex Password
7ev3n	000e	56/71	0/62	0/62
	0047	62/71	0/64	0/61
	0084	57/70	0/62	0/62
Alcatraz	9185	52/70	1/61	0/61
AvosLocker	7188	48/64	2/59	1/59
	f810	51/66	1/57	0/58
Babuk	eb18	54/67	1/58	0/59
BlackBasta	9a55	52/70	1/62	1/62
	7883	60/70	1/62	1/62
Cerber	0cd2	49/69	1/57	0/58
Dharma	dc5b	52/68	0/62	0/62
DoejoCrypt	e044	56/72	0/62	0/64
HelloKitty	fa72	59/69	1/57	0/60
Hive	47db	49/69	1/58	0/59
Jigsaw	9c74	48/67	2/58	1/61
	3ae9	59/68	1/62	1/62
	df04	56/72	1/62	1/62
Karma	6c98	55/68	2/58	1/58
LockBit	a2ad	55/70	1/58	0/61
	dec4	59/69	1/58	0/57
LockFile	2a23	53/68	1/58	0/57
Lorenz	1264	49/68	2/57	1/58
	a0cc	53/69	2/59	1/61
	edc2	55/69	1/57	0/60
MarraCrypt	be88	60/71	0/62	0/62
MedusaLocker	0abb	49/67	1/59	0/59
	f5fb	51/68	2/59	1/59
Mespinoza	4dc8	61/69	1/58	0/59
	44f1	57/69	1/58	0/61
	af99	58/69	1/57	0/58
MountLocker	5eae	56/69	1/59	0/59
Phobos	9dde	52/68	1/58	1/59
	265d	58/67	2/58	1/58
	8710	62/70	2/59	1/61
Ragnarok	db8b	52/69	1/61	0/59
Ryuk	9eb7	61/68	1/60	0/59
	40b8	58/68	1/58	0/60
Sage	ac27	65/71	0/62	1/62
SatanCryptor	dd28	56/71	0/64	0/62
Snatch	edad	53/69	1/57	0/61
	9b11	61/68	1/58	0/58

(Continued)

Table 14. Continued

Ransomware Family	SHA256	Plain File	Simple Password	Complex Password
Sodinokibi	fd16	60/70	1/57	0/61
Sugar	1d4f	52/67	1/57	0/60
SunCrypt	759f	58/72	1/62	1/62
TellYouThePass	7697	47/68	2/61	1/59
TeslaCrypt	4de6	61/71	0/61	0/62
Venus	d609	61/72	0/62	0/62
	ee03	61/72	0/62	0/64
	59b0	58/72	0/61	0/64
VirLock	7a92	60/67	1/59	0/50
	f4b1	63/71	1/59	0/61
VoidCrypt	4b78	53/71	0/62	0/60
WannaCry	ed01	60/67	3/59	2/59
Xorist	fb54	62/71	1/59	0/59
Average		56	0.96	0.33

The plain file column shows the results of scanning plain .exe ransomware samples. Simple password and complex password columns show the scanning results for simple password- and complex password-compressed files (in .zip format), respectively. The number before the slash is the number of successful detections. The number after is the number of total scanners available for this sample. This number varies slightly for each sample. Last row shows the average number of detection on 54 samples. SHA256 is the first 4 digits of the sample's SHA256 hash. The samples are uploaded for scanning in May 2022 and December 2022.

Table 15. Top 29 Important APIs for Identifying Ransomware Execution from Feature Importance Analysis Based on a Random Forest Model

API		API	
1	RegEnumKeyExW	16	NtCreateKey
2	CreateDirectoryW	17	LoadResource
3	DrawTextExW	18	GetDiskFreeSpaceExW
4	CoInitializeEx	19	EnumWindows
5	NtDeleteKey	20	RegOpenKeyExW
6	SHGetFolderPathW	21	NtQueryKey
7	GetFileInformationByHandleEx	22	NtQueryValueKey
8	GetForegroundWindow	23	NtSetValueKey
9	NtQueryAttributesFile	24	CreateActCtxW
10	DeviceIoControl	25	GetSystemTimeAsFileTime
11	SearchPathW	26	GetSystemWindowsDirectoryW
12	SetFileTime	27	SetErrorMode
13	SendNotifyMessageW	28	GetFileVersionInfoSizeW
14	GetSystemMetrics	29	NtOpenMutant
15	GetKeyState		

Table 16. Version Information of Antivirus Software Tested

	Version/Build	License
Antivirus A	N/A	N/A
Antivirus B	N/A	N/A
Bitdefender	26.0.18.75	30-day trail
Malwarebytes	4.5.10.200	14-day trail
Kaspersky	21.3.10.391 (h)	30-day trail
McAfee	16.0 R31	30-day trail
Norton	22.22.4.11	30-day trail
360	10.8.0.1465	free

Table 17. The Full SHA-256 Hashes of the Ransomware Samples We Measure

Ransomware Family	Compiled Year	SHA256	Entropy	if packed	Used for evaluation
7ev3n	2016	000ec059ab4eae4f2591449c6581b34748d3f90ef1688b9ec6daf5ab58d5da73	6.40 (80%)		
	2016	0047aed5ba539ab2e56e78d47b0ae8673d4f221bf5106987f66437e6eb0978ba	6.38 (79%)		
	2016	0084af770e99180fcdc6778c513d36384cf4b3ff24d0f8bc62ecaa76651be616	6.40 (80%)		
Alcatraz	2016	918504ede26bb9a3aa315319da4d3549d64531afba593bfad71a653292899fec	6.48 (81%)		
AvosLocker	2021	718810b8eeb682fc70df602d952c0c83e028c5a5bfa44c506756980caf2edebb	6.63 (82%)	yes	yes
	2021	f810deb1ba171cea5b595c6d3f816127fb182833f7a08a98de93226d4f6a336f	6.63 (82%)	yes	
Babuk	2021	eb180fcc4380b15013d9fe42e658fc6f6c32cf23426ef10b89bc6548d40523b	5.73 (71%)		yes
BlackBasta	2022	9a55f55886285eef7ffabdd55c0232d1458175b1d868c03d3e304ce7d98980bc	6.62 (82%)	yes	
	2022	7883f01096db9b0cf90c2317749b6873036c27ba92451b212b8645770e1f0b8a	6.62 (82%)		
Cerber	2015	0cd28b912cf4d989a6f03c4edfd73d1d90faf971ad84b28c6c25440ad87630f	7.86 (98%)	yes	yes
Dharma	2017	dc5ba8e45e7cf8d8dfcb8fb2de6f842786428fc46c34d8a3e02c8119bbd9f7584	7.23 (90%)	yes	
DoejoCrypt	2021	0e44d9f2d0f1260c3f4a543a1e67f33fcac265be114a1b135df575b860d2b8c6	6.99 (87%)	yes	
HelloKitty	2020	fa722d06ea7418d68e4935e1461010a8f730f021fa1595ee68bd0768fd5d1bf61e7	5.98 (26%)		yes
Hive	1969	47dbb2594cd5eb7015ef08b7fb803cd5adc1a1fbe4849dc847c0940f1ccace35	6.06 (75%)		yes
	2020	9c748a69c48b79e6422b3bea1766e415de5532cb7ba2b9673d5a511636ec1df2	7.98 (99%)	yes	yes
Jigsaw	2016	3ae96f73d805e1d3995253db4d910300d8442ea603737a1428b613061e7f61e7	7.68 (95%)	yes	
	2020	df049efbfa7ac0b76c8daff5d792c550c7a7a24f6e9e887d01a01013c9aa763	7.61 (95%)	yes	
Karma	2021	6c98d424ab1b9bfb8a3eda340fef6540ffe4ec4634fb95cf9c70fe4ab2de90	5.87 (73%)		yes
	2021	a2ad5c5045a16450f7da7eab14ba13eb69ab7286204f61ba6a4226bfad7f17	6.88 (93%)	yes	yes
LockBit	2021	dec4ca3a0863919f85c2a1a4a7e607e68063a9be1719ccb395353fe4a2d087e5	6.68 (83%)	yes	
	2021	2a23fac40f6a97cc738d633ec00f3f8e93ba22d2498f14dea08983026df128a	7.92 (98%)	yes	yes
LockFile	2021	1264b40fea845c4ed119d002a7569be6c99a9bba732d0057364da9350f9d3659b	6.26 (78%)		yes
	2021	a0ccb9019b90716c8ee1bc0829e0e04cf7166be2f25987abbc8987e65cef2e6f	6.31 (78%)		
Lorenz	2021	edc2070fd8116f1dfc8d419189331ec606d10062818c5f3de865cd0f7d6db84	6.27 (78%)		
	2020	be88512fc9250a558a3524e1c3bbd0299517cb0d6c3fb749c22df32033bf081e8	7.40 (92%)	yes	
MarraCrypt	2021	f5fb7fa5231c18f0951c755c4cb0ec07b0889b5e320f42213cbf6bbbe499ad31	5.57 (69%)		
	2021	0abb4a302819cdca6c9f56893ca2b52856b55a0aa68a3cb8bdcd55dec1fad9ad	5.57 (69%)		
MedusaLocker	2020	4dc802894c45ec4d119d002a7569be6c99a9bba732d0057364da9350f9d3659b	6.65 (83%)	yes	yes
	2021	44f1def68aef34687bfac3668e56873f9d603fc6741d5da1209cc55bdc6f1f9	6.65 (83%)	yes	
Mespinoza	2020	af99b482eb0b3ff976fa719bf0079da15f62a6c203911655ed93c52ae05c4ac8	6.65 (83%)	yes	
	2020	5eae13527dc99059025c3e56dad966cf6747f6e7830090e40c14d0a4046adf0	4.00 (50%)		yes
MountLocker	2020	9dde984b21a00bc3307c28bd81f229500b795ce4e908b6f8cb5fbd338b22b8e1	3.38 (42%)		yes
	2020	265d1ae339e939776d9328b2c84aca61a7cb6c0bca9f2f8dc213678e2b2ad86	6.97 (87%)	yes	
Phobos	2020	8710ad8fb293832665335455987aa17961b2496a345a7ed9f4bbfcb278212bc	6.70 (83%)	yes	
	2020	db8b499d613b604a439bca37c3be2f578bdfcd1b2271ecbcbf22db85996e785	6.73 (84%)	yes	
Ragnarok	2021	9eb7abf2228ad28db7f571e0495d4a35da40607f04355307077975e271553b8	6.42 (80%)	yes	yes
	2020	40b865d1c3ab1b8544bcf57c88edd30679870d40b27d62feb237a19f0c5f9cd1	5.11 (63%)		
Sage	2017	ac2736be4501b8c6823ebcf7241ceda38c3071418fb43c08b30f54f1a45d07e0	6.54 (81%)	yes	
SatanCryptor	1969	dd286a4d79d0f4c2b906073cf7f46680252ca09c1c39b0dc12c92097c56662876	7.91 (98%)	yes	
Snatch	1969	edade6616334f3d313ac3ea7c3e432d8d9461cddad8e2ec3a94ffdc6e336a94e	7.89 (98%)	yes	yes
	2021	9b11711efed24b3c6723521a7d7eb4a52e4914db7420e278aa36e727459d59dd	6.14 (76%)		yes
Sodinokibi	2021	fd164c4c121371f94cf3a034ad8cf8edc7c0f7141a8f4c9da1683d41b212a87	6.76 (84%)	yes	
	2021	1d4f0f02e613cbbbc47e32967371aa00f8d3dcf388c39f0c55a911b8256f654	7.89 (98%)	yes	yes
Sugar	2021	1d4f0f02e613cbbbc47e32967371aa00f8d3dcf388c39f0c55a911b8256f654	7.89 (98%)	yes	
SunCrypt	1969	759f2b24be12e208903b00f9719db71a332dd8f252986c26afbcbda9f32623bc4	6.88 (86%)	yes	
TellYouThePass	1969	76960749ed11d97582923e31d59115910ae74d8753c8e92f918f604ca8ad26d	6.04 (75%)		yes
TeslaCrypt	2005	4de6675c089aad8a52993b1a21afd06dc7086f4ea948755c09a7a8471e4fddbd	7.57 (94%)	yes	
	2022	d6098f0d579273528b28b0b49c8b72b6f9908aef9e1ba0ec5da0874fa8c92266	7.04 (88%)	yes	
Venus	2022	ee036f333a0c4a24d9aa09848e635639e481695a9209474900eb71c9e453256b	7.04 (88%)	yes	
	2022	59b05789e5ac3d747c0a3d0f3e4ccac2667cb7367e42adb9a3cbb108a538fc77	7.04 (88%)	yes	
VirLock	2015	7a92e23a6842bc51c9959892b83aa3be633d56ff50994e251b4fe82be1f2354c	7.96 (99%)	yes	yes
	2015	f4b11885a3056fc56efd8b0cdd71fae152368e4c2e96a3481c6ddf21e9d75aa	7.94 (99%)	yes	
VoidCrypt	2021	4b78968928cfa5437ffdd56a39a5ea8c10a7b6dc5d3f342d003260088876b3cf	7.96 (99%)	yes	
WannaCry	2010	ed01ebfbc9eb5bbee545af4d01bf5f1071661840480439c6e5ab8e080e41aa	8.00 (99%)	yes	yes
Xorist	2012	fb54a1b85ab37cdec346e06cf716cbe0b071f4833020823595f3b69614c5446e	7.20 (90%)	yes	

The exact samples can be found using the hashes. Year is the (possible) compilation time from the executable file. 1969 and 2010 might be intentional for anti-analysis reasons. Entropy is the file entropy calculated by Detect it Easy (DiE). A sample is packed if the entropy is over 6.5. Samples marked with yes in the used for evaluation column are used for testing antivirus software.

Table 18. The List of Benign Samples Used

Software	File Name	SHA-256	# Alters from Malware Scanners
7 Zip	7z2200-x64.exe	0b01c258a2e9857de86bd845deef59953cff2836ed030dba3da529262484b00	1/68
Atom	atomsetup.exe	ca69560bbc0f868301b1797580ce0d5dfe9a7822b0917897c2f3542393dde358	0/63
Chorme	ChromeSetup.exe	72222838e052e5151eeca0427eb0502b7a9395403b8be89f9a177aa8e9b43a5d	0/69
Discord	DiscordSetup.exe	ee9f94706055735af63117f1e6c80ca50c7244d6a44f751ae2e33934910b58	0/66
Ditto	dittosetup_64bit_3_21_134_0.exe	dbd049b9dde36b45659d97d88cbe35f2fbb3f31b8fd8bbe682f1b700aa8e7e	0/59
Dropbox	dropbox_99.501_offline_installer.exe	f8b83cc9b7172002f2767c53696ec8e1a84af21d4d19b6dd9151c03d4e2521ea	0/66
Firefox	Firefox Installer.exe	9b14ca825c3ce54440a32217e97f6c33e4d2ab9492deb558943406023ef8c68	1/69
Git	git-2.37.1-32-bit.exe	714069fe4291c4ca7a51f7e7e81b0c94038590294f3b9e0981456a664c92966b	0/67
LibreOffice	LibreOffice_7.3.4_Win_x64.msi	509c70c1c8136805480146b55e4bad5dc73b11ee47b4682b43cf07670109e176	0/40
NotePad++	npp.8.4.3.installer.exe	367893ed67fb585446bef6128774e5f35eff9c2f89e9e89c006dce8f61d8128	0/69
Python 3.8	python-3.8.5.exe	f5fe57acaa90ff4c5afed629b51880b53e4cabd0ebcadb33f56ca56fa1654de8	0/69
Adobe PDF Reader DC	readerdc64_en_1_cra_install.exe	f492b470a1b60a5075cd4ebd5b52f1a274f4292a2c8dbd571af208c5b8690b7c	0/64
Spotify	SpotifySetup.exe	a8e15459a613063f3fc47ca1d77239615834831a2df3fd4c7a270ff70a298d8	0/69
Steam	SteamSetup.exe	874788b45dfc043289ba05387e83f27b4a046004a88a4c5ee7c073187ff65b9d	0/69
TeamViewer	teamviewer_setup.exe	e463f3a11c4eafc698906876d610702fc9227a65183a30104579d8912ecdef4	1/68
Tree Size	TreeSizeFreeSetup.exe	4de19445d877ef4dF981fhead9440cf4a8832a284eae753ff1c7dd41dc10fa	0/69
WhatsApp	whatsappsetup.exe	8aefba89a391331d8d3ad08f988c2a5ba0d69d04069f03a121a01573da7be6c	0/66
WinRAR	winnrar-x32-611.exe	59276c49519ebd5194b95622c1c814db2c45d14eb6b07eacd9f2b37c9c7bbf93	1/68
WinZIP	winzip26-home.exe	a9ed6c5db282c4d42f4fd232627dab25b27f77b561a0065998a82b9e668d9f70	1/69
Zoom meeting	zoominstallerfull.exe	cd83a3b20d65db7e51e345aa32075bc37b799dc8d8e86c5df950409bd56168da53	5/66
NetBean IDE	apache-netbeans-13-bin-windows-x64.exe	a06ea580a2bfe50bdc8e9791fed5c6032ce8330b1e0c6c5dbf6c9e1c931dc9e	0/61
Bitdefender	bitdefender_avplus_v10.exe	b97314fe1f3bb9de06abd2f615d2f47c52810e09d17255a6ba3c0a65eb801	0/65
Dev-C++	dev-cpp_5.11_tdm-gcc_4.9.2_setup.exe	faad96bbcc51f115c9edd691785d1309e7663b67dcfcf7c115153d28c9c0f1f	1/67
Evernote	Evernote-10.48.4-win-dll-ga-3760-5f4dcc5719-setup.exe	a81ec8d119abaaef31cb46125f0c0089f054d085ca3b1f6927b48f3be40e9de	0/66
Google Drive	GoogleDriveSetup.exe	2cb39f4b8e64094c83e7eee34f0f886b58df23fa4141c225714cd6646b96575	0/67
Grammarly	grammarlysetup.exe	368e252f2e066bd482b8524c4ac939e5728154a334b991153a4fcb3ca2320f14	0/69
iCloud	iCloudSetup.exe	4cfd20d13cdce2b5c435f2ddaf4ee4c81d976461846bf3b954e8af6cbcd9f7	0/67
KeePass password safe	keepass-2.52-setup.exe	da403bc2e91132d1c1e0c49f585441e4cd430c8195ca8a38adce2a300de52cb	0/71
MalwareBytes	mbsetup.exe	057ac0f95e80abc5c73da9aefbc4e5e1bb778c2c154bf65c35435a34cdfa3da94	0/72
Microsoft Teams	microsoft-teams-1-5-00-28567.exe	dcca2a974c673e21f3b5b11cee955fb20b14903c3218cef3b9d2b061cc8a0c30	0/61
OneDrive	OneDriveSetup.exe	83d429a8568ee4ea0ed002c0897560c6b0a3e0b2a66f72a4149a521d461c6e7	0/71
Paint.Net	paint.net.4.0.21.install.exe	088a02864e8daf807584fdd14ba3ced191979db0af301a318e7c1e8fc4c03dcdb	3/68
QuickTime	quicktimeinstaller.exe	56eff77b029b5f56c47d11fe58878627065d8beacbc3108d50d98a83420152c2b	0/66
Skype	Skype-8.90.0.405.exe	d073e31487c5584f12b263d0372288c049a1d316a7f5151801bb7e6ebb39766b1	0/69
Slack	SlackSetup.exe	0682a25eae6bfc3bc42e949aa4a0274c983690b726922217befb02d8e2f5306	0/69
Screen Split	SS_Setup_6-57.exe	78a9ba1748686eccc181e97151c813acc92ecdcd35c0a6b73f24823ae0bb2a8cc	0/66
Mozilla Thunderbird	Thunderbird Setup 91.3.1.exe	39a502318a8b10bc25d9547b9c48fb646f7083d7161a4da7cee48ceabe77c65e	0/53
WordWeb	wordweb10.exe	27142582b89e0fa2ca6a9d5036ec3bde140109aa9632f9b5eac30933a082080	1/70

SHA256 is the hash of the sample we run. The # alter column shows the number of malware scanners that mark the samples as malicious (false positives). The number after the slash is the number of scanners available for scanning this sample. The samples are uploaded for scanning in July 2022 and December 2022.

Received 27 February 2024; revised 28 July 2025; accepted 15 February 2026