International Journal of Information Security
https://doi.org/10.1007/s10207-023-00758-z

REGULAR CONTRIBUTION

RWArmor: a static-informed dynamic analysis approach for early detection of cryptographic windows ransomware

Md. Ahsan Ayub[1] · Ambareen Siraj[1] · Bobby Filar[2] · Maanak Gupta[1]

Abstract
Ransomware attacks have captured news headlines worldwide for the last few years due to their criticality and intensity. Ransomware-as-a-service (RaaS) kits are aiding adversaries to launch such powerful attacks with little to no technical knowledge. Eventually, with the successful progression of ransomware attacks, organizations suffer financial loss, and their proprietary-based sensitive digital assets end up on the dark web for sale. Due to the severity of this situation, security researchers are seen to conduct static and dynamic analysis research for ransomware research. Both analyses have advantages and disadvantages, and prompt ransomware detection is expected to stop the irreversible encryption process. This research proposes a novel static-informed dynamic analysis approach, RWArmor, which includes the knowledge of the already-trained machine learning models based on static features to improve the ransomware detection capabilities during dynamic analysis. The effectiveness of our approach is evaluated by predicting a novel/unknown ransomware between 30 and 120 seconds of its execution. The random forest algorithm is utilized to accomplish this task and tested against 215 active cryptographic Windows ransomware collected between 2014 and 2022. Based on our empirical findings, our method achieves 97.67%, 92.38%, and 86.42% accuracy within 120, 60, and 30 seconds of behavioral logs, respectively.

Keywords Dynamic Analysis · Machine Learning · Ransomware · Static Analysis

1 Introduction

Ransomware, a special type of malware, has continued to be a significant threat for small to large organizations over the past several years. In general, adversaries launch ransomware attacks to cripple a company's daily operations or hold their digital assets hostage because of financial gains. The common ways attackers enter the victim's digital environment include exploitation of remote services, stolen/compromised credentials via brute-force attacks, software vulnerabilities, drive-by downloads, phishing, network configuration, etc. Based on the key findings from the 2022 incident response report from Palo Alto Networks, the adversaries spend 28 days on average inside the organizational environment before they are detected. Once the ransomware attack begins, the threat actor(s) encrypts the organization's important and sensitive digital assets and then demands a hefty ransom through cryptocurrency (e.g., Bitcoin) primarily in exchange for a key by which the assets are likely to be decrypted. It is important to mention that organizations not only suffer financial loss due to their critical data/resource being inaccessible, leading to the inability to perform normal business operations, but they also become subject to defamation as adversaries often post confidential organizational information on the dark web as a means of double extortion. It severely affects the company's reputation and hampers the public's trust. The Unit 42 team at Palo Alto Network outlined different sectors where ransomware attacks are observed at a higher rate, such as financial, real estate, wholesale and retail, high tech, construction, manufacturing, transportation and logistics, hospitality, healthcare, education, and professional and legal services. A recent (May 2021) well-known example of such an attack is the DarkSide ransomware attack on the Colonial Pipeline network, a critical infrastructure system supplying about half of the US East Coast's gasoline. The company has a 5,500-mile pipeline system that carries 2.5 million barrels of fuel per day. Because of this ransomware-as-a-service (RaaS) affiliate program's attack, the Federal

✉ Md. Ahsan Ayub
mayub42@tntech.edu

[1] Department of Computer Science, Tennessee Tech University, Cookeville, TN, USA

[2] Sublime Security, Inc., Washington, DC, USA

Published online: 23 September 2023

Motor Carrier Safety Administration (FMCSA) announced a state of emergency in 18 US States to tackle the significant fuel shortages. After five days of investigations on this largest-ever cyber-attack on an American energy system, the company resumed its operation after payment of US$ 4.4 million worth of bitcoin.

Ransomware is not a novel threat. In 1989, a 20k floppy drive was first infected with ransomware at an AIDS conference, and later, it was named AIDS Trojan. Decades later, the world noticed the first Windows-based ransomware, named "Archiveus Trojan," in 2006 that uses the RSA encryption algorithm. However, security researchers and practitioners started noticing a rise in ransomware attacks in 2012, which has grown significantly since then. Figure 1 is presented to depict some notable ransomware events in a brief historical timeline. Ransomware can be primarily grouped into two categories: (1) locker ransomware—locking down the victim machine to prevent regaining control of the affected system and (2) cryptographic ransomware—encrypting the files of the victim machine to demand ransom in return for the key needed for encryption. It is reported that cryptographic ransomware attacks are carried out more frequently than locker ransomware attacks [1, 2]. Although ransomware attacks are seen to be targeted at Windows, Linux, Mobile, and Internet of Things (IoT) ecosystems, this research is primarily focused on cryptographic Windows ransomware.

Due to the capabilities of posing severe damage, ransomware (or malware in general) researchers work relentlessly to detect them and carry out two main types of experiments: (1) static (or code) analysis and (2) dynamic (or behavior) analysis. Static analysis primarily undergoes inspection of the structural information to gain insights into how it is written. However, ransomware authors can evade the detection ability of static analyzers by obfuscating, encrypting, packing, or recompiling [3]. Although dynamic analysis helps mitigate this limitation by running a sample in a safe environment, performing environmental checks allows the ransomware authors to sense that it is being diagnosed and thus may not show its behavior [3].

Problem Statement and Vision. Performing static and dynamic analysis for ransomware detection has advantages and disadvantages and hence becomes a difficult job. It is not only much desired to correctly predict a malicious or untrusted application as ransomware before it starts its infection. Identifying ransomware process as early as possible is vital to prevent further damage. This research addresses this critical problem with a vision that incorporates both analyses and detects novel/unknown ransomware between 30 and 120 seconds of its execution.

Proposed Approach: RWArmor. In this paper, we propose RWArmor, a static-informed dynamic analysis approach that uses the output of the static analysis-based machine learning models and later includes it as a feature for the dynamic analysis for improved detection abilities. An Application Programming Interface (API) is developed to query the trained models built in the Static-RWArmor project with the static features of ransomware and benign applications. It returns the highest probability score predicting how malicious a given sample is. We perform our experiments in Any Run,[1] an interactive malware hunting service, providing a detailed dynamic analysis report. This sandbox-generated report (in JSON format[2]) contains several pieces of behavioral information, including the changes in the file system, the changes in the registry, network activities, windows API calls made by each process, and much more. All the events contain a timestamp which helps us devise a timely detection approach. To evaluate the effectiveness of our approach, we aim to address the following two important research questions:

RQ1. How early can we detect cryptographic windows ransomware based on its behavioral events?

RQ2. Does static-informed dynamic analysis improve the detection capabilities of dynamic analysis?

Major Contributions. The major contributions of our study are broadly summarized as follows:

- The behavioral similarities among 383 ransomware samples and their behavioral dissimilarities compared with 2,245 different benign applications against execution time are investigated based on I/O Request Packet (IRP) logs (Sect. 3).
- An Application Programming Interface (API) is developed to leverage the extensive knowledge of static analysis-based machine learning models, which were trained by 2,436 ransomware samples and 3,014 benign applications (Sect. 4).
- A novel static-informed dynamic analysis approach named "RWArmor" is proposed for cryptographic windows ransomware detection between 30 and 120 seconds of their execution life cycle (Sects. 5, 6, and 7).

Paper Organization. The rest of the paper is structured as follows: Sect. 2 describes the background details with related work. In Sect. 3, we discuss the dynamic analysis of ransomware based on IRP, which motivates our work. Our prior work in static analysis is shared in Sect. 4. Later, our proposed approach, RWArmor, is highlighted in Sect. 5. The methodology, including the dataset by which the experiments were conducted, and the evaluation of RWArmor are presented in Sects. 6 and 7, respectively. The limitations of our study, as well as future work, are listed in Sect. 8. Finally, we conclude this research in Sect. 9.

[1] https://any.run.

[2] https://www.json.org/json-en.html.

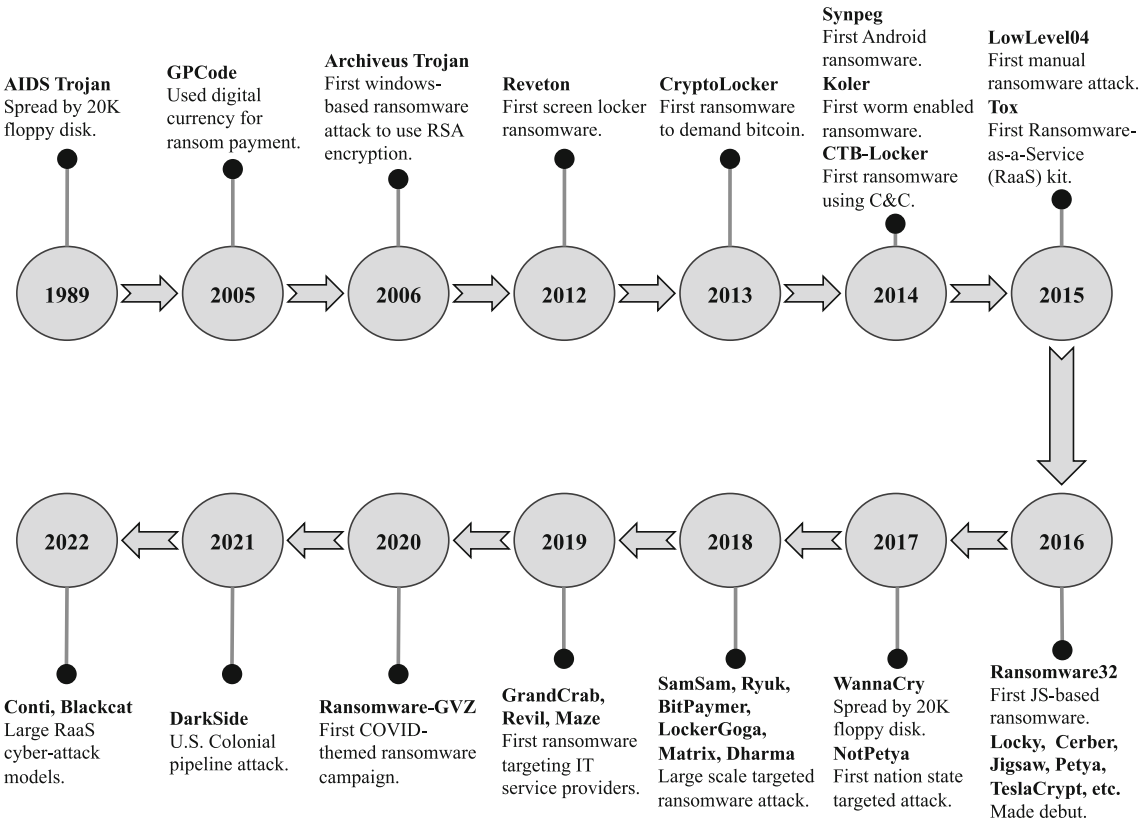

Fig. 1 A brief historical timeline of ransomware attack

2 Background and related work

This section describes the background details with related work to better understand our research.

2.1 Static analysis

Static analysis is an important step for security researchers, who analyze a built software (e.g., a ransomware exe file) to understand an attacker's intent. In general, the goal is to gather the structure details of the examining samples to locate their malicious nature without executing it. Using static analysis to detect and prevent a ransomware attack is popular as it does not require kernel-level privilege or a virtual machine for investigation [4]. Researchers leverage byte sequences, opcodes, header information of Portable Executable files, functions, and imports to propose unique mechanisms for ransomware detection [5–7]. Additionally, using Machine Learning (ML) after extracting such information is widely common to strengthen the capabilities of detection schemes [8]. ML-based classifiers are built for the feature spaces, including PE headers, PE section names, functions, imports, and String metadata of a sample [9–13]. On the flip side, Medhat et al. [14] constructed a deep learning (DL) network for OpCode feature space to learn the underlying ransomware pattern. Several other studies utilized OpCode (machine instruction code of executable files) for ransomware detection [15–17].

However, with the help of code obfuscation, encryption, and packing techniques, adversaries aim to evade and defeat powerful tools devised based on static analysis [18–20]. It is important to mention that legitimate users use such techniques to avoid stealing their intellectual property via reverse engineering techniques. However, security researchers and practitioners perform dynamic analysis to overcome this limitation. We share its description in the following section.

2.2 Dynamic analysis

Dynamic analysis is a crucial step for security researchers as they begin to understand the behavior of a malicious sample (e.g., ransomware, in our case) during its execution in a safe environment. It allows us to analyze the footprints an examining sample leaves behind in terms of windows API calls, registry events, file system operations, network calls, etc. The investigation of such event logs enables the researchers to propose defensive strategies to detect a malicious process as it is executed on a compromised machine to cause damage (i.e., encrypting files in the case of ransomware).

As the nature of ransomware samples is to encrypt files in the file systems, several research works have been centered around the I/O Request Packet (IRP), a low-level I/O log. It is a common mechanism for requesting I/O operations between the user and the kernel mode. Kharraz et al. [21] were the first to analyze 1,359 ransomware samples to describe the workings and effects of ransomware and the usefulness of monitoring the file system through IRP logs in users' machines for successful ransomware detection. Other researchers later adapted their work in this domain as a viable ransomware detection technique. Continella et al. [22] used IRP logs as a focal point in their research to propose a real-time self-healing virtual file system approach, which was resilient to malicious encryption to prevent the effects of ransomware attacks. Then, Kharraz et al. [23] utilized a mini-filter driver to collect IRP logs to monitor system-wide file system change and access a substantial number of objects of the Windows-based Operating System. The authors experimented with their proposed approach with 13,637 ransomware samples that cover both crypto and locker type of ransomware. Mehnaz et al. [24] leveraged IRP logs to offer a ransomware surveillance system with the utilization of process monitoring (upon receipt of IRP open, close, read, write, and create operation) and file change monitoring (upon receipt of IRP write operations). In addition to sharing research work utilizing IRP logs, using planting decoy or honey files for ransomware detection has been well adopted by security researchers in this field [25–33].

In addition to observing file systems operations, security researchers pay close attention to the traces of native functions' invocation and Windows API calls. They mainly monitored such trails in the *Process* module to propose impactful detection techniques [5, 22–24, 34–43]. On the other hand, the use of network calls made by ransomware during its execution is also studied in this domain [6, 44–46].

The aforementioned categories of information are gathered by running the ransomware samples in a variety of environments, such as virtual machines,[3] sandbox,[4] emulator,[5] and bare metal (physical device). As we focus on cryptographic ransomware targeting the Windows environment, we notice that researchers run their experiments on Windows XP, Windows 7, Windows 8, and Windows 8.1 versions. After the logs generation and collection phase, machine learning and deep learning techniques were seen to be utilized to learn the ransomware behavior and later detect the unseen samples. In our case, decision tree and random forest classifiers are developed to train them on the registry, files, and process-based event logs of both ransomware and benign processes. Additionally, such algorithms led several security researchers to success in the past [47–56].

It is also important to mention that dynamic analysis is not free from limitations. Virtualization-based sandbox environment requires significant computational resources to examine an untrusted application [57]. Additionally, malware authors utilize environmental checks to enable their built samples to sense whether it is being run on a virtual environment [58–61]. Thus, security researchers and analysts struggle to analyze them. Our study's collected ransomware samples, however, exhibit encryption-related behavior in the virtualized environment (e.g., Any Run), allowing us to investigate the event logs it left behind.

2.3 Distinction from existing related work

A tabular comparison chart is presented in Table 1 to highlight the key attributes of some state-of-the-art peer-reviewed academic research projects published between 2015 and 2022. All the cited papers in the table executed a dynamic analysis approach for ransomware detection. The extracted event logs during ransomware execution include changes in the "registry", changes in the "files" system, windows API calls made by each "process", and "network" calls. Only a few research projects focused on network logs for purpose solutions. However, we observe that 63% of ransomware samples in our study make network calls during their encryption. For this reason, we exclude this category of event logs from the analysis for generalization. However, we incorporate the reminder feature spaces into our research.

Researchers utilized either machine learning (ML) or deep learning (DL) techniques to train their built models on the collected event logs. We leverage decision tree and random forest classifiers to identify the difference between ransomware and benign applications' event logs, similar to [22, 24, 43, 62]. To our knowledge, no other research has monitored ransomware behavior in a Windows 10-based sandbox environment. Additionally, this research is the first to include the probabilistic output of static features-based trained ML model (Static-RWArmor) into the dynamic ransomware analysis and test the effectiveness of built ML models with as fewer events logs as possible. This static-informed dynamic analysis approach, called "RWArmor," aims to effectively flag a malicious running process *early* enough so that further damage can be prevented inside a victim machine. We are optimistic that our study, as a contribution, will help other security researchers and practitioners in the field.

[3] https://www.virtualbox.org.

[4] https://cuckoosandbox.org.

[5] https://github.com/mandiant/speakeasy.

Table 1 State-of-the-art peer-reviewed research projects for ransomware detection using dynamic analysis approach

Paper	Year	Technique		Early detection	Static feature	Dynamic Features				Ransomware samples	Benign logs	Analysis environment	Target OS
		ML	DL			Registry	Files	Process	Network				
Kharraz et al. [23]	2015	×	×	×	×	–	✓	✓	×	13,637	✓	VM	Win XP
Continella et al. [22]	2016	✓	×	✓	×	×	✓	✓	×	383	✓	Sandbox	Win 7
Scaife et al. [34]	2016	×	×	✓	×	×	✓	✓	×	492	✓	Sandbox	Win 7
Sgandurra et al. [35]	2016	✓	×	✓	×	✓	✓	✓	×	582	✓	Sandbox	Win XP
Kharraz and Kirda [36]	2017	✓	×	✓	×	×	✓	✓	×	1174	✓	–	Win 7
Palisse et al. [37]	2017	×	×	✓	×	×	✓	✓	×	798	–	–	Win 7
Chen et al. [38]	2017	✓	×	×	×	×	×	✓	×	83	✓	VM	Win 7
Hasan and Rahman [6]	2017	✓	×	×	✓	✓	✓	✓	✓	360	✓	Sandbox	Win 7
Mehnaz et al. [24]	2018	✓	×	✓	×	×	✓	✓	×	- (14 families)	✓	–	–
Kim et al. [63]	2018	×	×	✓	×	×	✓	×	×	–	–	–	–
Jung and Won [64]	2018	×	×	×	×	×	✓	×	×	- (5 families)	×	–	–
Daku et al. [39]	2018	✓	×	×	×	×	✓	✓	×	150	×	–	–
Shaukat and Ribeiro [5]	2018	✓	×	×	✓	✓	✓	✓	×	574	✓	Sandbox	Win 8.1
Chew and Kumar [65]	2019	×	×	×	×	×	✓	×	×	- (4 families)	×	Sandbox	Win 8.1
Lee et al. [62]	2019	✓	×	×	×	×	✓	×	×	–	×	–	–
May and Laron [66]	2019	×	×	×	×	×	✓	×	×	- (1 family)	×	VM	Win 8.1
Hirano and Kobayashi [67]	2019	✓	×	✓	×	×	✓	×	×	2	✓	Hypervisor	Win 7
Homayoun et al. [40]	2019	×	✓	×	×	–	–	✓	–	660	✓	–	–
Al-rimy et al. [41]	2019	✓	×	✓	×	×	–	✓	×	8,152	✓	Sandbox	–
Tang et al. [44]	2020	×	×	✓	×	×	✓	×	✓	771	✓	VM	Win 7
Roy and Chen [42]	2021	×	✓	✓	×	✓	✓	✓	×	17	✓	Bare Metal	Win 7
Kok et al. [43]	2022	✓	×	×	×	✓	✓	✓	×	205	✓	Sandbox	–
Trizna [7]	2022	×	✓	×	✓	✓	✓	✓	×	12,514	✓	Emulator	–
RWArmor (our work)	–	✓	×	✓	✓	✓	✓	✓	×	215	✓	Sandbox	Win 10

✓ annotates Yes, × annotates No, and – annotates Uncertain

3 Motivation: dynamic analysis of ransomware through IRP

3.1 I/O request packet (IRP)

The I/O Request Packet (IRP) is a common mechanism for requesting I/O operations between the user and the kernel mode. When a user executes a command to open a file residing in the file system, the I/O system service inside the kernel mode performs the triggered task by the user. At first, it searches the input file name(s) and then the user's access rights to check whether or not the user can access the file. Secondly, it finds the file inside the file system and allocates the required memory for the IRP request inside the I/O manager. Then, the I/O manager transfers the IRP information to the file system driver, from where the task to open the file gets completed. At last, after receiving an I/O status from the IRP, the I/O manager frees the memory and returns a handle to the user with a success or failure operation status [68]. The structure of IRP can be discussed in the following four categories:

Types of IRP Operations. Three types of I/O operations can be triggered from user space: IRP; FIO (Fast I/O)—designed to transfer data between user buffer and system cache directly; and FSF (File System Filter)—designed to support IRP operations on file system.[6]

Process-based Features. Each IRP log resembles the I/O operation of a process initiated from the user space and recorded in the kernel space. Several pieces of information regarding a process in the IRP include Process ID, Process Name, Thread ID, and Parent ID. It is worth mentioning that such IDs are only valid as long as the process is active. For example, the Operating System (OS) allocates a certain set of IDs to a process upon its starting execution. When the process completes its actions, such IDs are freed, and the OS can reallocate the same ID(s) to another process. All this to say is that the process ID (or any given ID) is not unique in any captured ransomware sample's IRP dataset. Each IRP log presents two additional pieces of information regarding the process: pre-operation time—the timestamp of a process that starts its operation for a given IRP request; and post-operation time—the timestamp of the process competes for its IRP request with either a failure or a success status.

Flag based Features. The IRP structure provides several flag-based features, or categorical variables in other words, to cover additional information about the type of task a process carries out. It includes IRP Flag, IRP Major Operation Type, IRP Minor Operation Type, Status, Inform, Transaction, and Argument 1-6. IRP Flag feature comprises four special kinds of flags, such as No Cache, Paging I/O, Synchronous API, and Synchronous Paging I/O, along with a 32-bit Hexadecimal value. IRP Major Operation Type feature indicates the type of the IRP operation executed by the process, e.g., read, write, close, etc. IRP Minor Operation Type feature also shows a process's IRP operation type, e.g., query directory, start/remove an I/O device, etc.; however, these two features depend on each other. Status is another feature with a 32-bit Hexadecimal value that is designed to map the IRP operation request to a human-readable format, e.g., success, abandoned, alerted, timeout, etc. Microsoft provides most of the possible flag values' definitions in its documentation.[7]

File System-based Features. The IRP structure has a set of features representing intrinsic pieces of information, such as File Object, Device Object, File Name, Buffer Length, and Entropy, as a process interacts with the machine's file system. While a process accesses one of the files in the file system, the IRP records the objects' locations created by the process with the file's name as string datatype. Buffer length (a float datatype) and Entropy (a float datatype between 0 and 1) features indicate the portion of the file is written on memory and modified from its previous state, respectively, from each IRP request.

3.2 Dataset acquisition

The IRP-based dataset is acquired from [22]. Their research proposed an intuitive ransomware detection framework, ShieldFS, in 2016 that successfully identified the signs of ransomware. To obtain the knowledge of benign users' behavior, the authors performed a large-scale IRP data collection generated by benign applications with the help of an IRPLogger. The researchers built this data-collection agent to capture the day-to-day tasks of 11 voluntary machines used by home, office, and developer type of users for several weeks. On the other side, by leveraging the same tool installed on a Windows 7 (64-bit) machine, the authors collected 383 active ransomware samples' IRP logs during its run-time execution. It is noted that each ransomware sample's captured IRP dataset covers some of the common utility applications' IRP logs, such as Adobe Reader, Microsoft Office, Web Browsers, and Media Player.

[6] IRPs Are Different From Fast I/O: https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/irps-are-different-from-fast-i-o.

[7] https://github.com/microsoft/Windows-driver-samples/blob/8fb512ac674df5ba129a69906d450f2a1361136d/filesys/miniFilter/minispy/user/mspyLog.h.

3.3 Behavioral dissimilarities of ransomware processes with benign processes and benign users' interaction over time

3.3.1 Identifying trends in ransomware behavior

To understand ransomware behavior, each ransomware sample's dataset is partitioned in a 5-min time frame. For every feature mentioned earlier, we conduct Time Series Trend analysis, which helps us visualize the infection rate of all the studied ransomware families over time. We group the discussion in feature spaces as per the following:

Types of IRP Operation. This feature space covers three types of features: IRP, FSF, and FIO. Every ransomware sample needs to carry out a significant number of IRP operations to encrypt the assets in a victim machine. We analyze the differences of the median values between ransomware and benign processes with all the studied ransomware samples and present the visualization in Fig 2. From the data distribution plots, we notice that the differences in every 5 min time frame are very high, with a maximum reported at the 20th min for all three features.

We compute the median counts of such features' ransomware processes for every studied ransomware sample and the highest counts for benign processes. The purpose behind this approach is to explore how much more IRP operations would require for ransomware processes during its encryption process compared to the idle state of a machine with frequently used software, e.g., web browser, windows tools, etc. Now, from Table 2, it is inferred that a ransomware process is likely to show more than 50,000 IRP, 10,000 FSF, and 5,000 FIO operations compared to the benign processes in any given 5-min time frame.

IRP Flags based Features. The statistical counts of this feature space, covering features like IRP Flags, IRP Major Operation Type, IRP Minor Operation Type, and Status, do not help us find distinguishable ransomware characteristics regarding unique counts compared to the benign processes. As per Table 2, the statistical data distribution differences between ransomware and benign processes are quite minimal.

File System based Features. This feature space, including File Object, Unique Files Accessed, Buffer Length, and Entropy, is important to learn the encryption patterns of all ransomware samples. We illustrate visualizations of each feature's data distribution trend in Fig 3 to portray their differences against benign processes. Ransomware samples tend to access the greatest number of unique files by creating file objects within 30 min of its execution. As a result, the median counts for Buffer Length (the size of the buffer for encrypted files) and Entropy (the change of files in the latter part) features are much higher.

Similar to the Type of IRP Operations feature, it is inferred from Table 2 that a ransomware process is likely to create 1,000 (approx.) file objects, access 2,000 (approx.) unique files, and obtain more than 10,000 and 0.07 buffer length and entropy, respectively, compared to the benign processes in any given 5-min time frame.

3.4 Comparison with collected IRP logs from users' machines (Benign)

As mentioned before, the collected dataset from [22] comes in with benign IRP logs of 11 volunteers' machines, ranging from Microsoft Windows 7 to Windows 10, to record their daily activities. These recorded logs were considered the ground truth dataset based on the assumption that none of these machines were compromised by cyber-attacks. Each machine's captured log is stored in sessions. We base our comparison analysis on such user sessions regarding the statistical counts of all the highlighted file system features.

The file system features, e.g., Unique File Accessed, Unique File Objects, Buffer Length, and Entropy, are considered to generate counts for the statistical measurements of minimum, median, mean, and maximum. To compare the computed values with the ransomware samples, we select the counts of 5 min IRP logs from 90 min of ransomware execution. The hypothesis behind this approach is that the number of files a ransomware process will access during its execution will be much higher than a regular user using his/her machine. Therefore, we present our findings in a tabular format in Table 3. We observe that the total number of unique files accessed and unique file objects created by the voluntary users in a standard session is much less than ransomware for all the mentioned statistical measurements. On the other hand, the mean values of the other two features, the highest buffer length and entropy from a process of users' machines, are significantly higher than ransomware processes. From these results, our remark is that in a 5-min time frame, the ransomware processes are likely to access notably large number of files. However, ransomware processes are less likely to perform much modifications on the accessed files compared to an average user based on entropy feature.

3.5 Concluding remarks

From our data-driven analysis of 383 ransomware samples' IRP logs, ransomware processes access many more unique files in a given 5-min window compared to both other benign processes during its encryption and standard user sessions. [69] involved in constructing an effective Artificial Neural Network to detect ransomware by incorporating all the collected IRP logs of every ransomware encryption process, as well as benign users' sessions. Performing this case study motivates us to carry out detecting ransomware with as less

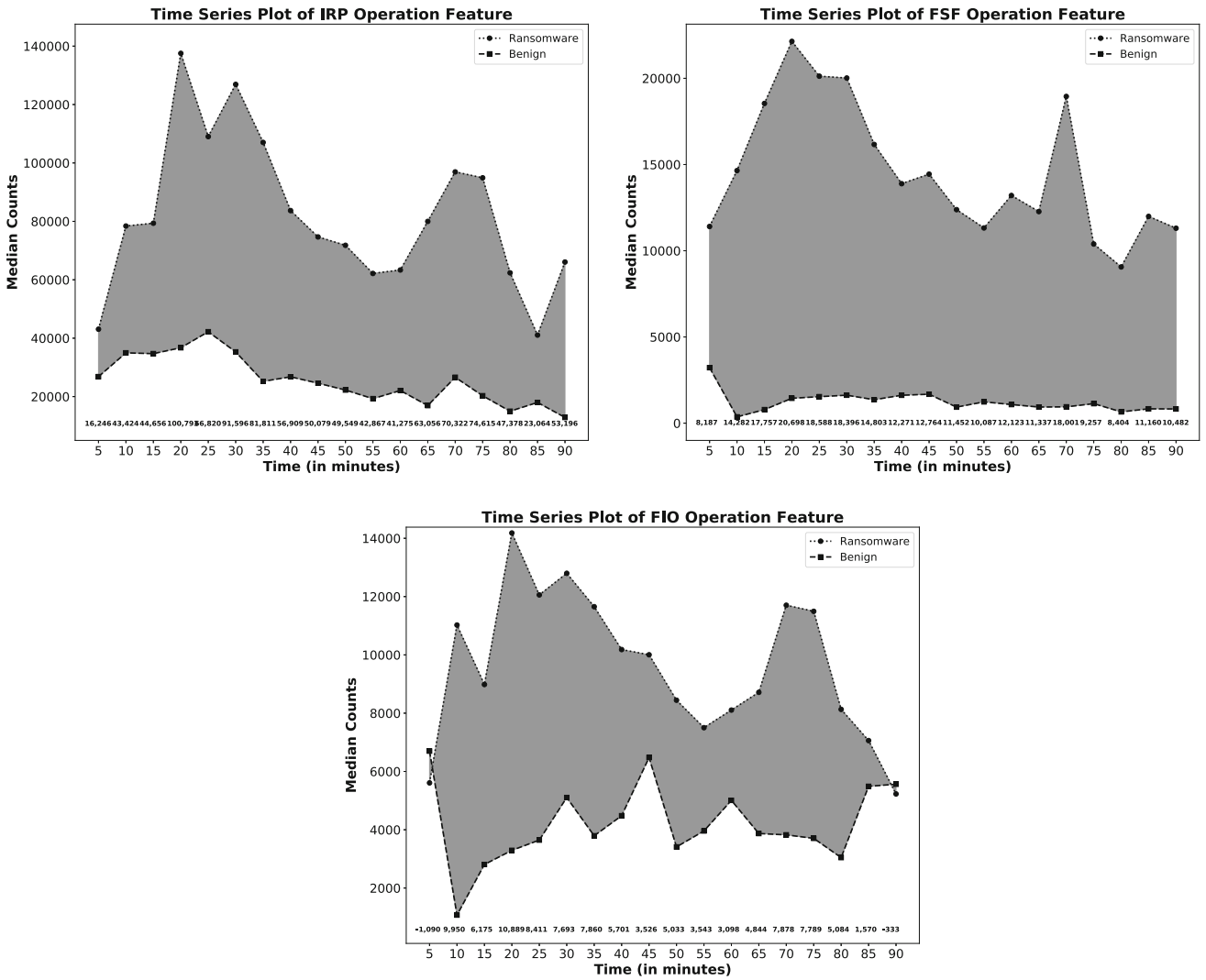

Fig. 2 Data distribution difference trend between ransomware and benign processes of IRP, FSF, and FIO features in the Type of IRP Operation feature space

Table 2 Statistical data distribution differences of notable feature spaces between the IRP logs of ransomware and benign processes over 5-min time frame (✓ indicates unique counts—denoted as U, while × does not)

Feature Space		U	Time Series Trend (Median Values)			
Group	Features		Min	Mean	Median	Max
Types of	IRP	×	16,264	56,536.6	51,637.8	100,793
IRP	FSF	×	8,187	13,336.3	12,197.5	20,698
Operations	FIO	×	333	5,581.8	5,393	10,889
IRP	IRP flags	✓	0	1.38	1	9
Flags	Major Opn	✓	0	1.22	1	3
Based	Minor Opn	✓	3	3.72	4	5
Features	Status	✓	3	5.33	5	8
File	File objects	✓	158	1,079	1,068	2,063
System	File accessed	✓	783	2,002	1,913	3,356
Based	Buffer length	×	820	10,553	11,684	21,744
Features	Entropy	×	0.03	0.074	0.071	0.11

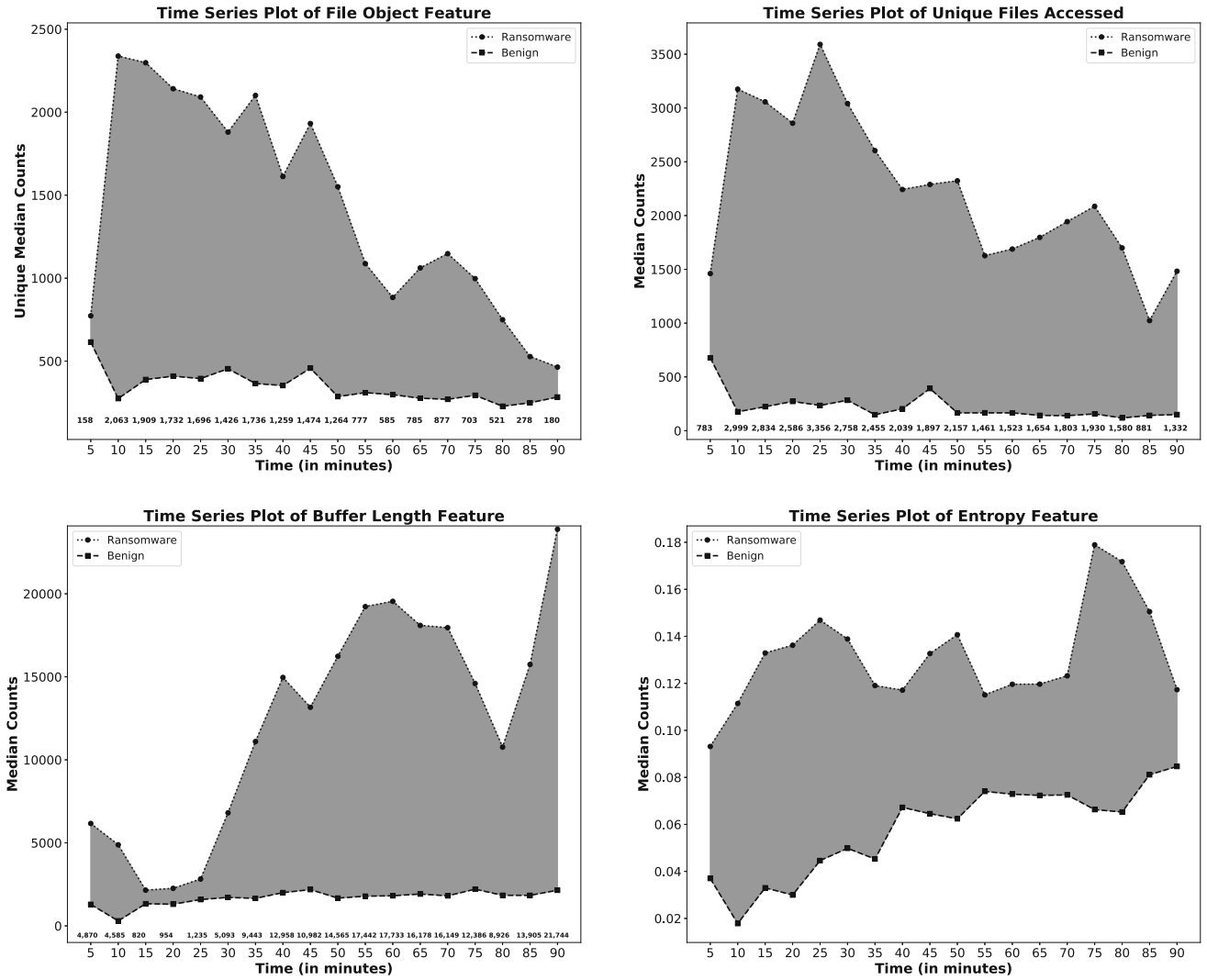

Fig. 3 Data distribution difference trend between ransomware and benign processes of File Object (top left), Unique Files Accessed (top right), Buffer Length (bottom left), and Entropy (bottom right) features in the File System-based feature space

Table 3 Comparison statistical counts of file system features between a complete session of 11 users' machines and over 5 min average time frame of ransomware execution (90 min session)

File system features	Min		Median		Mean		Max	
	Users	Ransomware	Users	Ransomware	Users	Ransomware	Users	Ransomware
File accessed (Unique)	30	1,667	519	3,065	1,583.64	2,851	9,277	3,715
File objects (Unique)	47	1,135	378	1,899	470.18	2,059	1,521	3,231
Buffer length (Mean)	8,192	5,870	32,768	21,125	42,891.8	20,734	141,626.25	37,435
Entropy (Mean)	0.066	0.077	0.549	0.125	0.502	0.12	0.79	0.16

data as possible because the more data we include, the easier it gets to identify them. Additionally, the longer we wait, the more damage it causes to a victim's machine. For this reason, we challenge ourselves to detect ransomware with less than 5 min of a behavioral log-based dataset to propose an early detection approach.

4 Our prior work: static-RWArmor

In addition to understanding ransomware's behavioral dissimilarities comparing with benign instances, the structural dissimilarities are identified between 2,436 ransomware samples and 3,014 benign applications to develop a static analysis-based ransomware detection approach called "Static-

RWArmor." The details of how it was implemented are shared in this section.

4.1 Portable executable (PE) metadata

The portable executable (PE) is an object file of Windows OS including .exe (executable file), .dll (dynamic link library), .sys (system file), etc., as extensions. The metadata of a PE file contain several pieces of information, such as file headers, section tables and import libraries. The file header includes the type of targeting machine, the size of the section table, the time and date that the file was created, the flags indicating different attributes of the file, etc. Additionally, it tells us the magic number of the file, the size of the code, initialized data, the image, the subsystem required to run the image, DLL characteristics, and the address of the entry point. In the section header, we can extract information on each section's virtual address, virtual size, and the size of raw data. We focus more on the import address table which contains information on both the libraries and the functions used by the PE file. For example, "47363b94cee907e2b8926c1be61150c7" is a ransomware sample from the Cryptowall family. It is bound to dbghelp.dll, KERNEL32.dll, COMDLG32.dll, ADVAPI32.dll, USER32.dll, and COMCTL32.dll import libraries. Additionally, we can also extract the "AppendMenuA", "CallWindowProcA", "CharLowerBuffA", "Char UpperA", etc. functions that are required from the "USER32.dll" import library [8].

4.2 Dataset and feature set construction

We gather ransomware PE data from Sophos ReversingLabs 20 Million (SOREL-20M) dataset [70]. The repository extracted various disarmed malware samples' features and metadata. The types of malware samples include adware, flooder, ransomware, cryptominer, file infector, installer, spyware, etc., from which we obtained several packed ransomware samples collected between 2018 and 2020. With that, we accumulate 2,436 ransomware samples' PE metadata in total. We double-check with VirusTotal by providing the hashes of each sample to confirm that they are ransomware samples. To accomplish this task, a Python[8] script utilizing VirusTotal API v3 Engine[9] is written, which lets us scan the hash of each sample. In return, it sends back a detailed report of the sample with over 70 antivirus scanners' evaluation on whether or not it is labeled as malicious or safe.

In addition to collecting the ransomware dataset from the SOREL-20M repository, we randomly picked benign applications' metadata for the binary classification tasks. We include a list of cloud-based backup and file-compressing software as such applications interact heavily with the file system. Overall, 3,014 benign applications' PE metadata are stored, and a PE metadata extraction engine is built using Python 3 programming language to process the dataset.

Data Storing Method. All the pieces of PE metadata information are grouped into the following categories, which can be treated as a tabular format of a relational database.

- Sample Info: MD5, Sample Size, Collected Year, and Is Malicious.
- File Generic Info: MD5, SHA1, SHA256, First Seen by Virus Total, Mime Type, File Type, PE File, and File Type Extension.
- Library Imports: MD5 and Library Names.
- Function Name Imports: MD5 and Function Names.
- Sections: MD5, Section Name, Raw Size, Virtual Size, and Entropy.
- PE Info: MD5, Subsystem, Subsystem Version, Machine Type, Time Stamp, Code Size, Initialized Data Size, Uninitialized Data Size, OS Version, Magic, and PE Entry Point.
- VirusTotal Info: MD5, Scan ID, Total Scan Engines, and Number of Positives.

Our hope is that other researchers in the field can benefit from our dataset's structured formation.

4.3 Binary classification

Binary classification tasks are performed to discover the structural dissimilarities between ransomware and benign application. We focus on imports and function names as feature spaces to investigate if supervised machine learning algorithms can learn the underlying patter. Support vector, decision tree, random forest, AdaBoost, and gradient boosting classifiers are selected to achieve this task. The following four experimental settings are administered to evaluate the mentioned algorithms.

- Imports. We select the "imports" feature space for our first experiment setting. We obtain 2,576 unique numbers of imports for all the ransomware samples and benign applications. Then, we create a sparse matrix of 5,450 rows (ransomware and benign applications) and 2,577 columns (imports and target class). We apply Principal Component Analysis (PCA) [71] to reduce the size of columns into two. It enables us to capture 23% of information.
- Function Names. Similar to the first experimental setting, we choose the "function names" feature space for our second one. For this case, we gather 105,546 unique numbers of function names. Similarly, after creating the

[8] https://www.python.org.

[9] https://developers.virustotal.com/docs/api.

sparse matrix, we utilize PCA while capturing 13% of information of the entire matrix dataset.
- Imports and Function Names Combined. For our third experimental setting, we combine both "import" and "function names" feature spaces. The size of the sparse matrix becomes 5,450 rows and 108,123 columns. Then, applying PCA similarly gives us 13% of information capture.
- Numeric Feature Set. The last experiment focuses on the numeric feature set that helps us detect ransomware. For every ransomware sample and benign application, we compute imports' count, function names' count, section names' count, sample size, Code Size, Initialized Data Size, Uninitialized Data Size, and Resource Languages w/ PCA. We have not applied PCA after processing the dataset.

Scikit Learn [72], a well-documented machine learning package, is utilized to apply principal component analysis (PCA) on the high-dimensional feature space of the processed dataset. Now, fig. 4 is presented to show the distribution of ransomware samples and benign applications' feature spaces for experiments 1, 2, and 3. The visualization motivates us to select tree-based supervised learning algorithms because linearly separating two classes' data points does not appear feasible.

4.4 Empirical findings

As mentioned in the previous section, four experimental settings are designed based on the selected feature spaces to discover the structural dissimilarities between our studied ransomware samples and benign applications. We accomplish this task by choosing two different kernels of support vector classifier: RBF and polynomial, decision tree, random forest, AdaBoost, and gradient boosting. Scikit Learn [72] was used to apply the mentioned algorithms. The evaluation of such mentioned classifiers is performed in terms of:

- Accuracy. This metric describes the correct prediction of a given classifier.
- Precision. It is the ratio of the true positive records to all positively labeled instances.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- Recall. It is the ratio of the true positive instances to all instances that should have been labeled positive.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- F1 Score. This metric is the harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The performance of all the classifiers for the mentioned experiments is reported in a tabular format (see Table 4). It is noted that we perform 5-fold cross-validation for all cases, and the documented scores for all the metrics are their mean values. We observe that the random forest classifier has outperformed others for every designed experiment. Among experiments 1, 2, and 3, random forest achieves the best performance for experiment 3, that is 88.39% of accuracy, 88.28% of precision, 87.77% of recall, and 87.93% of F1 score. However, we notice that it performs even better for experiment 4, where accuracy, precision, recall, and F1 scores are 91.75%, 91.99%, 90.47%, and 91.05%, respectively.

It is important to mention that all the ensemble learning methods and decision tree have produced satisfactory results. In other words, accuracy, precision, recall, and F1 scores are in the high 80s for experiment 3 while in the low 90s for experiment 4.

4.5 Concluding remarks

The aforementioned static analysis approach, named Static-RWArmor, allows us to automate the task of predicting ransomware based on its structural information (PE Metadata) without execution. The intensive training process accounts for many ransomware and benign applications. Our aim is to reuse the built random forest classifiers as an offline API (application programming interface) for novel/unknown ransomware samples, as well as benign processes, to generate a probabilistic score of it being malicious. For example, the trained models can be queried by providing the above feature information. In return, a score ranging between 0 and 1 should be received to signify how confident the model is that the sample is malicious. We can potentially integrate "Static-RWArmor" prediction scores into the "RWArmor" approach as a feature for both ransomware and benign applications to achieve a static-informed dynamic analysis approach for ransomware detection.

5 Overview of our approach: RWArmor

This section describes our proposed static-informed dynamic analysis approach, named "RWArmor."

Figure 5 depicts the overview of our static-informed dynamic analysis approach for cryptographic windows ransomware detection. The approach integrates the knowl-

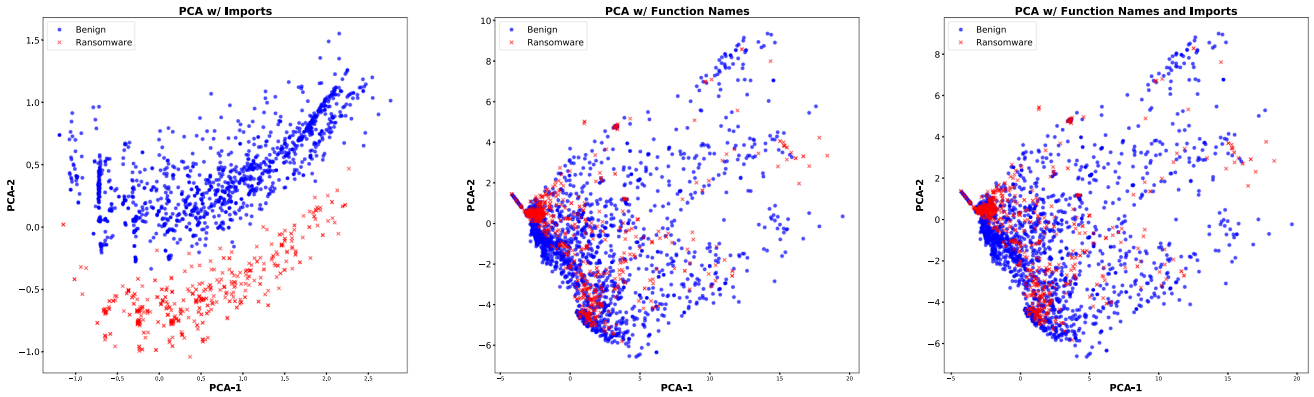

Fig. 4 Visualization of different feature spaces after applying principal component analysis (PCA)

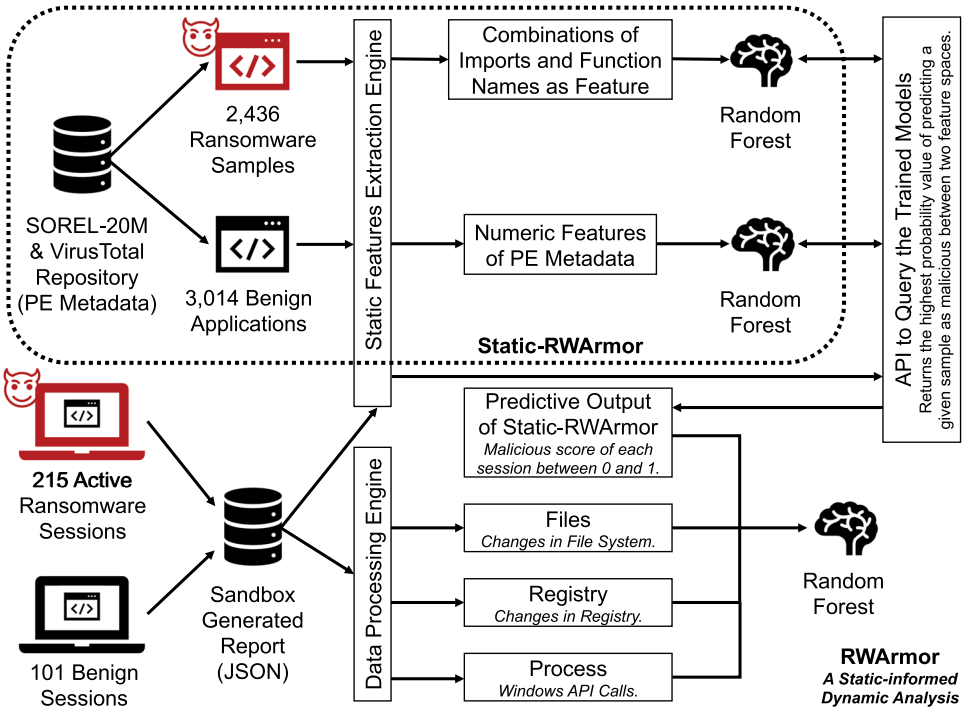

Fig. 5 Overview of RWArmor, our proposed static-informed dynamic analysis, for cryptographic windows ransomware detection with as fewer behavioral event logs as possible

edge acquired in the Static-Armor project, mentioned earlier, as a predictive pipeline. We send the static features (e.g., Combination of Imports & Function Names and Numeric Features) of the ransomware under examination and applications executed in the benign sessions to query the trained models. The developed API is used to query the models to receive a prediction score for two feature space groups instantly. However, we consider the highest probability value between the models. For example, `e3b7d39be5e821b59636d0fe7c2944cc` is a Petya ransomware family's sample.[10] With the developed static features extraction engine, we extract the above-mentioned feature groups and then query the models via API. As a result, we obtain 0.81 as a malicious score from the combination of imports and function names based random forest classifier, while 0.06 as a malicious score from the numeric features-based Random Forest classifier. We select the highest value (0.81, in this case) between the two probabilistic scores. In other words, it signifies that the static analyzer is 81% sure that the given sample is malicious. The rationale behind this approach is that we base our analysis on a sensitive setting. On another instance, `d76c47211551f7c1f1427b4bad8e6aa9`, a Windows backup software named EaseUS Todo Backup,[11] was used in one of the benign sessions. By following the same steps, the combination of imports and function names based model predicted it being a malicious sample as 0.68,

[10] https://www.virustotal.com/gui/file/e3b7d39be5e821b59636d0fe7c2944cc.

[11] https://www.easeus.com.

Table 4 Performance of a suite of machine learning algorithms for binary classification task on different experimental settings

Model	Imports (Experiment 1)				Function names (Experiment 2)				Imports & Function names (Experiment 3)				Numeric features (Experiment 4)			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
SVC (rbf)	0.6917	0.6953	0.7009	0.6902	0.7022	0.7372	0.6556	0.6509	0.7063	0.7346	0.662	0.6596	0.6207	0.3104	0.5	0.383
SVC (poly)	0.6416	0.7112	0.6813	0.6369	0.5861	0.293	0.5	0.3695	0.6323	0.7703	0.5574	0.4877	0.6207	0.3104	0.5	0.383
Decision tree	0.8179	0.8136	0.8171	0.8136	0.8712	0.8682	0.8669	0.8669	0.8667	0.8631	0.8632	0.8625	0.8911	0.8846	0.8845	0.8839
Random forest	0.8286	0.824	0.8261	0.824	0.8839	0.8825	0.8781	0.8795	0.8839	0.8828	0.8777	0.8793	0.9175	0.9199	0.9047	0.9105
AdaBoost	0.7721	0.7662	0.7608	0.7628	0.8337	0.8343	0.8215	0.8249	0.8281	0.8333	0.8111	0.8165	0.8601	0.8561	0.8457	0.8489
Gradient boosting	0.8179	0.8125	0.8151	0.8132	0.8622	0.8595	0.856	0.8571	0.8644	0.862	0.858	0.8592	0.9132	0.9107	0.9049	0.9069

while it was 0.33 for the other one. Although we know that the sample is benign in nature, we choose the value 0.68 to maintain a generalized approach. Our goal is to design a resilient setting for the experiment, and thus, we query two models instead of one. To further describe, we noticed that there are several samples where the Resources Directory table could not be extracted. As this information was included in the Numeric Feature set, we could not query the model; however, we achieved predictive results from the other model. In this way, we assign the output of Static-RWArmor to 215 ransomware samples, where the median value is 0.71, and 101 benign session applications, where the median value is 0.15.

An online virtualization-based sandbox environment named Any Run is utilized to conduct dynamic analysis while Windows 10 Professional (build: 19044, 32-bit) is selected as the target operating system inside the platform. For every analyzed session, we obtain a detailed report that outlines the behavior of the examined sample.[12] A generic format of such reports is illustrated in Fig. 6. From a set of aggregated pieces of information encapsulated in the JSON file, we focus on "incidents", "processes", "network", and "modified". Multiple processes can be involved during the ransomware process, and each of them is assigned with a *pid* (Process ID). Therefore, we locate ransomware processes from the incident segment of the report and record the timestamp at which they start execution. Knowing *pid* helps us build the following feature sets for our data-driven analysis.

File System Changes. The prime focal point of our analysis is to monitor the interaction of ransomware processes with the files system. Therefore, we parse through the modified section of the report to identify the changes in file system events caused by them throughout their execution.

Registry Changes. The registry changes reveal insightful information in ransomware detection [35]. Malware authors leverage registry in Windows environment for persistence [3]. We discover registry keys based on events created, deleted, and modified for benign applications and ransomware processes. To capture them, we similarly browse the modified section of the report.

Processes-Windows API Calls. The report documents all the events invoked by every process in the process section. It contains all the modules/DLLs imported by every process, which provides an understanding of the Windows API Calls made by them.

Network Logs. The generated reports store the network logs regarding DNS requests, HTTP requests, Connections, and Any Run analyzed Threats for each process. As mentioned in the beginning, about 63% of ransomware in our study exhibit network-based events. Therefore, this category of events are

[12] https://app.any.run/tasks/39375aa1-7bd6-470f-b6de-42b92f471253.

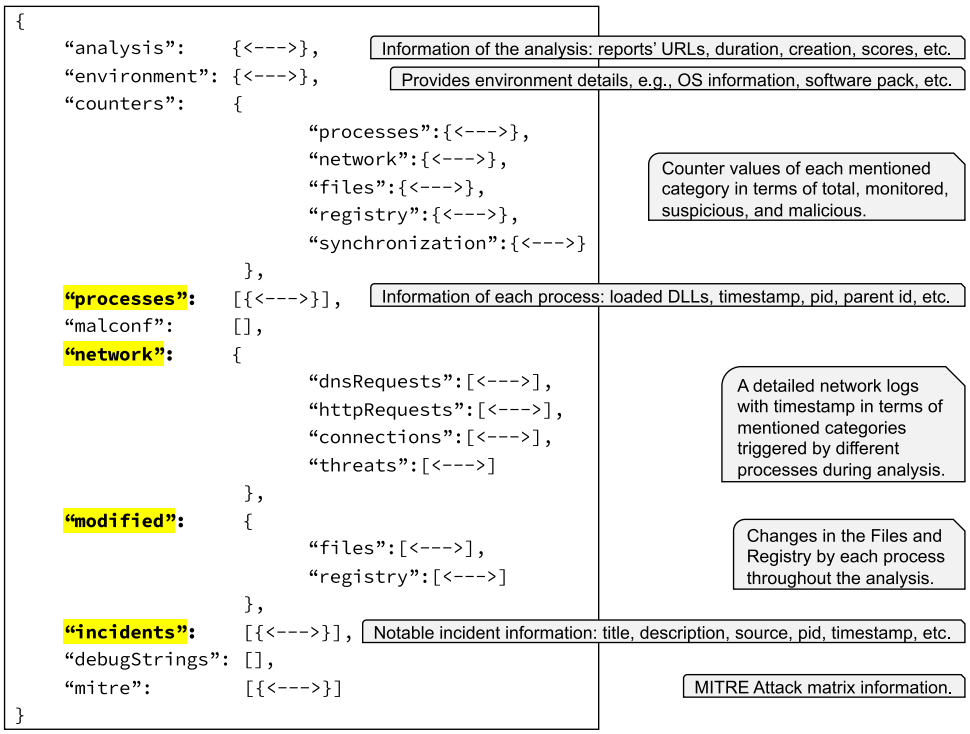

Fig. 6 A generic example for Any Run sandbox report in JSON format

excluded from our research for the sake of generalization purposes.

6 Methodology

6.1 Dataset construction

This section discusses the dataset collection phase to conduct the experiments. As mentioned, we utilize the Any Run platform to capture the behavioral Windows event logs for ransomware and benign applications. We choose a 32-bit Windows 10 Professional operating system to run the experiments. The virtual environment provides a wide array of installed software packs, such as Opera, Microsoft Office Professional 2019, VLC Media Player, WinRAR, Google Chrome, Java 8, Skype, Adobe Acrobat Reader DC, Adobe Flash Player, CCleaner, FileZilla Client, Microsoft Edge, Mozilla Firefox, Notepad++, etc. In addition to the examining sample(s), the sandbox-generated report contains the event logs of such benign applications.

Ransomware Dataset. 215 active cryptographic Windows ransomware are supplied to the sandbox to investigate their behavior. Based on the VirusTotal API service,[13] we perform a couple of checks by passing the hash value of each sample: (1) which ransomware family it belongs to and (2) when it was found in the wild. We discover that the samples belonged to 34 ransomware families and were collected between 2014 and 2022 through this process.

We continue running the experiments until any form of ransom note (e.g., as a pop-up window, a desktop background, an HTML file, a text file, etc.) is observed. We excluded the samples for which we did not notice any ransom note; however, the final count is 215. A visualized example of such types of notes is depicted in Fig. 7. On average, it took approximately 10 min from the start of execution to exhibit this type of note by a ransomware.

The studied ransomware families are Babuk, Bart, Cerber, Conti, Critoni, Critroni, Cryakl, Cryptowall, Cuba, Dalexis, Darkside, Dharma, Gandcrab, Hanta, Jigsaw, Lockergaga, Locky, Matrix, Ouroboros, Petya, Phobos, Ragnarok, Rapid, Rokku, Ryuk, Satana, Sepsis, Sodinokibi, Spora, Stop, Teslacrypt, Unlock26, Wannacry, and Yakes.

Ground Truth (Benign) Dataset. The collection of benign logs is similarly achieved. In addition to capturing logs from the benign applications mentioned in the beginning, 101 benign sessions are recorded by interacting with 163.19 MB of files in the file systems on average. To highlight, we incorporate 24 Windows-based file compressing and cloud-based backup software To mimic the ransomware-like behavior. To further describe, we compress and then decompress all the files in the file systems for each file-compressing application. Additionally, we backed up all the files residing on the disc when we operated the cloud-based backup applications. Each process and its activities during the benign sessions are labeled as the ground truth dataset.

[13] https://developers.virustotal.com/reference/overview.

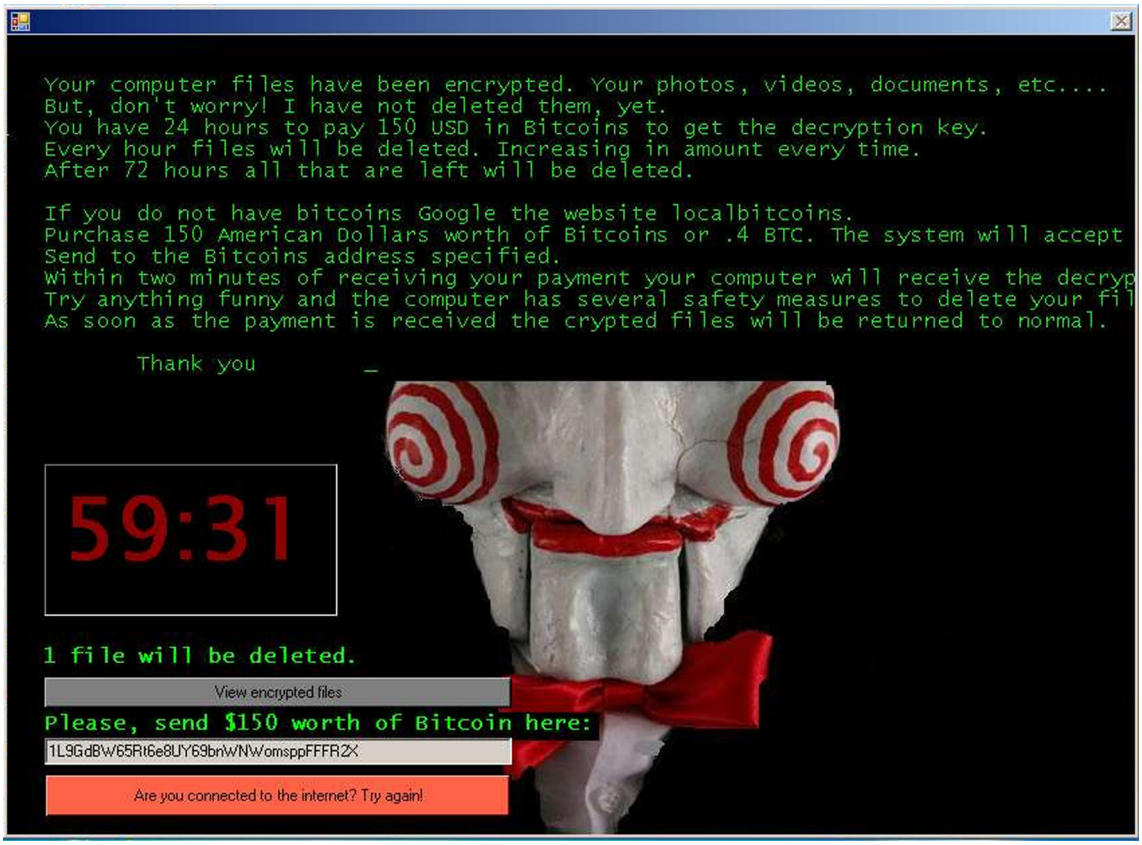

Fig. 7 A pop-up window to show the ransom note for a Jigsaw ransomware family's sample (MD5 hash: 2773e3dc59472296cb0024ba7715a64e)

The construction of the two aforementioned categories of the dataset using the Any Run platform's premium subscription has taken close to 2 months.

6.2 Dataset processing

We begin processing the generated reports by parsing through the JSON files. As described the previous section, we first identify ransomware processes to label them as malicious, while the rest of the processes, including benign sessions, are labeled as safe. We create the feature spaces corresponding to each process. In other words, the records in the "file" system changes, "registry" changes, and "process" feature groups reflect the behavior of a certain process during its execution life cycle. We document the start and the end time of such processes so that we can potentially focus on ransomware-based processes' events up to a certain time from its initiation.

Based on each process's activities, the following features are constructed in the respective feature groups:

- Files. The features included in this group are MD5 (key-value), File Name, File Type, Mime Type, size, pid, Timestamp, and Class. To further describe, MD5 represents the hash value of a benign application or a ransomware sample. Each process is assigned with an ID, *pid*. We save the name of the file a process interacts with, the type of the file, along with its mime type, and the size of the file. The class is a binary valued feature, where 0 is safe, and 1 is malicious. To highlight a key findings in our research, we compute in a 2-min time frame that a ransomware process accesses 148,206 unique files (median value), while a benign process interacts with 34,738 unique number of files (max value).
- Registry. We list the features for this case as MD5 (key-value), Key, Name, Operation Type, pid, Timestamp, and Class. MD5 plays the same role as per the above discussion. Each record indicates what type of operation a process makes to the registry (signifying with registry name and registry key value). The unique registry key operations include "write", "delete value", and "delete key" for both benign and ransomware processes. However, we calculate that all the ransomware processes modify 980 unique registry keys, while the number is 2,070 for the benign processes. We suspect it is the case because we installed several applications during the benign sessions.
- Process. This feature group contains MD5 (key-value), Command Line, File Name, File Type, Main Process,

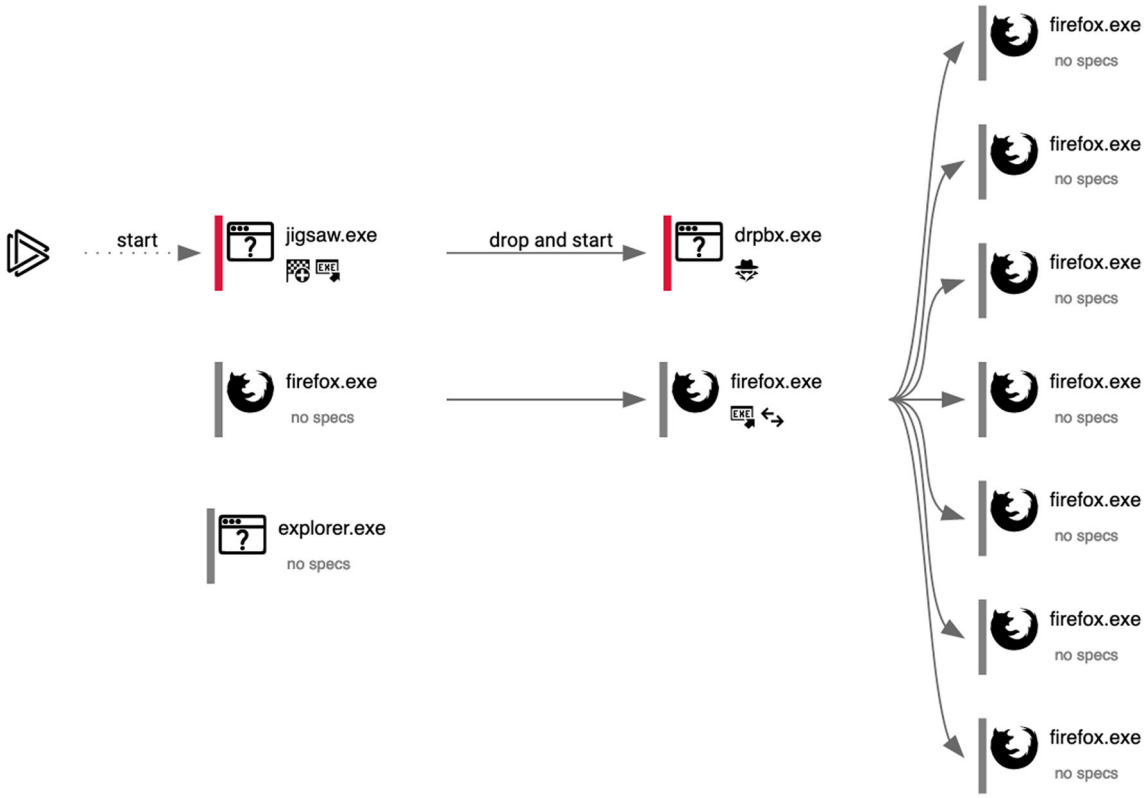

Fig. 8 Visualization of the generated process graph for a Jigsaw ransomware family's sample (MD5 hash: 2773e3dc59472296cb0024ba7715a64e)

API Name, pid, Parent pid, Start Timestamp, End Timestamp, and Class as features. To share the details of the aforementioned features, we observe that the main ransomware benign process causes the start of other child processes to encrypt the files. By mapping the records with MD5 values, we locate the main process involved in the operation with the help of the "Main Process" feature (a binary flag) and then locate its subprocesses to label them as malicious. The features *pid* and Parent *pid* are used for this approach. The other processes are then labeled as benign. The additional pieces of information per process include the name and the type of the file from which the process was triggered (i.e., an exe file named "jigsaw.exe" as per Fig. 8). It is inferred from the process graph visualization that two processes carry out the damage posed by the Jigsaw ransomware in this case. Additionally, it helps us store the actual start and end time of each ransomware sample's execution. The loaded DLLs for each process are recorded in the API Name feature. The command line feature demonstrates the argument passed to the Windows command to execute a certain process. MD5 values reflect the hash values for ransomware samples and the benign applications executed during the experiments.

6.3 Experimental setup

The experimental setup used to evaluate our proposed method is described in this section. The description concerning the dataset processing illustrates a vivid picture on how we isolate the ransomware and benign processes. To learn the underlying pattern between them, we perform further feature processing to combine the process-based information. We join the records based on MD5 and *pid* to obtain features per ransomware sample/benign application for all the feature groups. Additionally, we compute the numeric features for the Files and Registry feature group. However, we create a large sparse matrix for the Process feature group to capture the information of Windows API calls. To share the structure of a sparse matrix, the columns include unique file type items, unique command line items, and unique API names. Then, principal component analysis (PCA) is applied to reduce its dimension. A visualization in Fig. 9 is shared to showcase the distribution of 2-dimensional data plots between ransomware and benign applications. Finally, we conclude the processed feature set as follows.

- Files.
  - Number of unique files accessed ($x[0]$),
  - Number of unique mime type ($x[1]$),

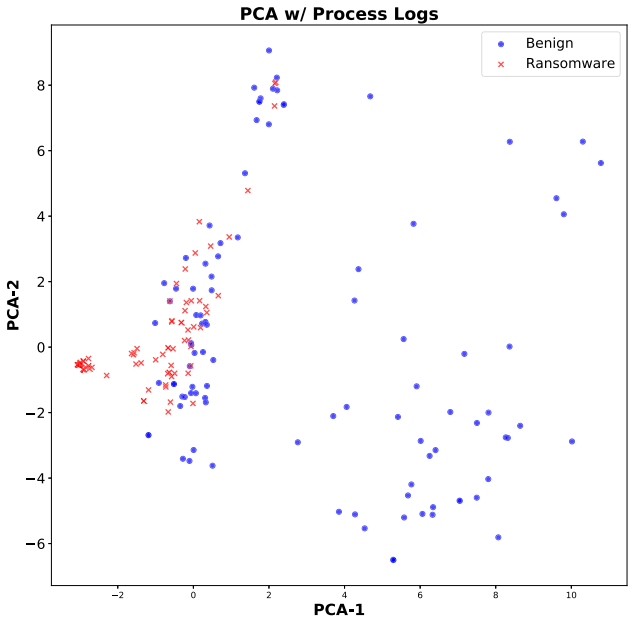

Fig. 9 Visualization of the process feature space after applying principal component analysis (PCA)

  - Number of unique processes ($x[2]$),
  - Minimum size of file ($x[3]$),
  - Q1 size of file ($x[4]$),
  - Mean size of file ($x[5]$),
  - Median size of file ($x[6]$),
  - Q3 size of file ($x[7]$), and
  - Maximum size of file ($x[8]$).
- Process.
  - 1st Principal Component ($x[9]$), and
  - 2nd Principal Component ($x[10]$).
- Registry.
  - Number of unique registry key ($x[11]$).

A Python 3 programming language-based "Data Processing Engine" is developed that scans all the reports and then performs all the tasks mentioned above for the generation of finalized feature set. The engine is equipped to create the feature set of ransomware for different time frames based on the timestamp. We begin our analysis with 120 seconds of event logs starting from the execution time for all the respective studied ransomware samples. We obtain the actual timestamp for every ransomware sample. As it is in milliseconds (t), we focus on 120 seconds ($t + 120,000$), 60 seconds ($t + 60,000$), and 30 seconds ($t + 30,000$)-based time windows. We do not proceed any further because only 55 ransomware samples (25%) in our study leave their behavioral footprints on Files, Registry, and Process features within 15 seconds from their execution. Due to the loss of most ransomware samples, we stop producing as low as 30 seconds of event logs. Then, decision tree and random forest classifiers are utilized to learn the behavior of ransomware and benign applications and distinguish them based on the processed dataset. To avoid overfitting the classifiers, we apply stratified 5-fold cross-validation for every experiment during the training process. Such experiments will help us address RQ1 as we explore training and testing the detection capabilities of machine learning models with as fewer ransomware behavioral event logs as possible.

In order to address the next research question (RQ2), one more feature is included in the processed feature set. As described the section 5, the API returns a predictive output of Static-RWArmor for all the benign applications and ransomware samples. As a result, we rerun the training and testing processes of the mentioned machine learning models to check if the detection capabilities are improved for the different time frames of execution event logs.

7 Evaluation

This section evaluates the effectiveness of our approach by addressing the research questions. The performances of the built random forest classifier are reported for different experimental settings based on accuracy, precision, recall, and F1 scores.

7.1 Addressing RQ1: dynamic analysis results

We begin sharing the empirical findings of decision tree and random forest classifiers for the processed feature set described in section 6.3. Our aim is to document the machine learning models' performances for ransomware detection. We start by describing how the decision tree algorithm produced the rules to perform prediction based on all the features. A visualization of the decision tree structure is shared in Fig. 10, where it incorporates Files ($x[0] - x[8]$), Process ($x[9]$ and $x[10]$), and Registry ($x[11]$) features.

As per the figure, the decision tree is first split based on a process feature—1st principal component, $x[9]$. Then, further splits are taken place based on the last process feature, 2nd principal component ($x[10]$), and a file-based feature (number of unique files accessed, $x[0]$). The rest are primarily dependent on file-based features. We note that the registry feature ($x[11]$) does not play an intrinsic role in this case, as the decision rule is observed at the tree's bottom. The visualization is generated with 120 seconds of ransomware events and the entire sessions of benign applications. The Files and Process feature groups are essential for our study to predict ransomware for dynamic analysis.

We are motivated that the random forest classifier will achieve better prediction performance as it is configured with

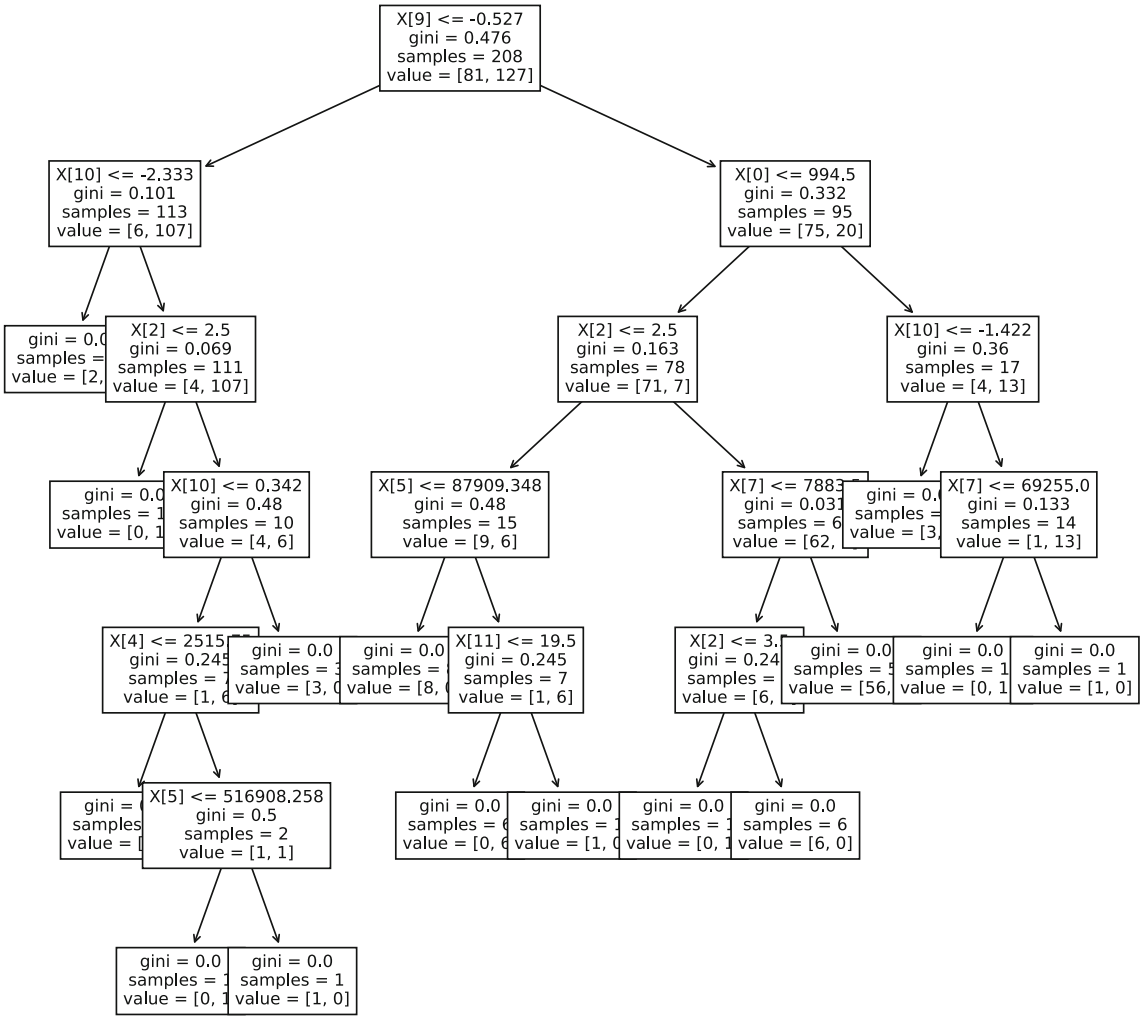

Fig. 10 Visualization of the built decision tree structure based on files, registry, and process feature spaces to showcase the rules used for prediction without static-RWArmor

100 estimators with "gini" criterion. The built models' effectiveness is checked with four different experimental settings: (1) Only files feature ($x[0] - x[8]$), (2) Files and Registry features ($x[0] - x[8]$ and $x[11]$), (3) Files and Process features ($x[0] - x[10]$), and (4) Files, Registry, and Process features ($x[0] - x[11]$). The performances for the mentioned experimental settings are reported in Tables 5, 6, 7, and 8, respectively.

As highlighted in the dynamic analysis part of Table 5, the performance of decision tree and random forest classifiers is documented with 120, 60, and 30 seconds of execution event logs of ransomware. The accuracy, precision, recall, and F1 scores drop as we shrink the ransomware dataset based on execution time. For example, the random forest classifier's precision scores for 120, 60, and 30 seconds execution time are 93.38%, 88.68% (4.7 point drop from 120-second), and 82.51% (10.87 point drop from 120-second), respectively. A similar trend can be noticed for other metrics, as well. For all the experiment settings, it is seen that random forest outperforms the decision tree classifier.

As illustrated in Table 6's dynamic analysis section, a similar trend is observed where the performance drops as we reduce the event logs for the testing process. We compare Table 5 (Files features) and Table 6 (Files and Registry features) and find up to 2% increase for random forest classifier in terms of accuracy, precision, recall, and F1 scores for every experimental setting. For example, the recall scores for the random forest classifier with 120, 60, and 30 seconds of execution time are 93.51% for Files features and 95.11% for Files and Registry features, 88.39% for Files features and 89.39% for Files and Registry features, 82.52% for Files features and 83.03% for Files and Registry features, respectively.

To describe the models' results for the Files and Process features, we take a look at the dynamic analysis part of Table 7. A similar drop in performances corresponding to the execution time is noticed. However, by comparing this table with the other two (Table 5 and 6), we achieve better empirical per-

Table 5 Binary classification performance of built machine learning models with only file features

ML models	Execution time	Dynamic analysis				Static-informed dynamic analysis			
		Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Decision tree	120	0.8751	0.8749	0.8707	0.8708	0.8843	0.8904	0.8715	0.8753
Random forest	seconds	0.9336	0.9338	0.9351	0.9313	0.9442	0.9409	0.9479	0.948
Decision tree	60	0.8405	0.8446	0.8491	0.8499	0.851	0.8538	0.8594	0.8502
Random forest	seconds	0.8838	0.8868	0.8839	0.8837	0.9095	0.9177	0.9078	0.9075
Decision tree	30	0.7864	0.7883	0.7807	0.7861	0.7947	0.7952	0.8085	0.7817
Random forest	seconds	0.824	0.8251	0.8252	0.8272	0.8342	0.8311	0.8334	0.8312

Table 6 Binary classification performance of built machine learning models with file and registry features

ML models	Execution time	Dynamic analysis				Static-informed dynamic analysis			
		Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Decision tree	120	0.8535	0.8552	0.858	0.8583	0.8687	0.8757	0.8569	0.8586
Random forest	seconds	0.9486	0.9533	0.9511	0.9512	0.9504	0.9598	0.951	0.9538
Decision tree	60	0.83	0.8372	0.8391	0.8392	0.8467	0.8402	0.8435	0.8445
Random forest	seconds	0.8938	0.8968	0.8939	0.8937	0.9043	0.9068	0.9026	0.9018
Decision tree	30	0.7893	0.7831	0.7888	0.7739	0.7883	0.7913	0.7994	0.7746
Random forest	seconds	0.8276	0.8373	0.8303	0.8305	0.8341	0.839	0.8331	0.8337

Table 7 Binary classification performance of built machine learning models with file and process features

ML models	Execution time	Dynamic analysis				Static-informed dynamic analysis			
		Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Decision tree	120	0.8836	0.8885	0.8809	0.8866	0.8959	0.8993	0.88195	0.8868
Random forest	seconds	0.9613	0.9627	0.95897	0.9592	0.9767	0.9761	0.9712	0.9757
Decision tree	60	0.8543	0.8583	0.8537	0.8537	0.8638	0.8654	0.8614	0.8626
Random forest	seconds	0.9019	0.902	0.9021	0.9018	0.9162	0.9184	0.9152	0.9161
Decision tree	30	0.8034	0.8091	0.8088	0.8083	0.8169	0.8103	0.8138	0.8114
Random forest	seconds	0.8276	0.8373	0.8303	0.8305	0.8341	0.839	0.8331	0.8337

formance. For example, the accuracy for the random forest classifier with 120, 60, and 30 seconds of execution time are 96.13% for Files and Process features (while 93.36% and 94.86% for the other two), 90.19% for Files and Process features (while 88.38% and 89.38% for the other two), and 82.76% for Files and Process features (while 82.40% and 82.76% for the other two), respectively. The improvement is expected due to the decision tree structure we depicted in Fig. 10.

Lastly, all the features are combined to report our findings in the dynamic analysis section of Table 8. This feature formation allows us to achieve the best results for 30 seconds of execution time. To further highlight, the Random Forest classifier achieves 85.88% accuracy, 86.43% precision, 84.57% recall, and 85.21% F1 scores with 30 seconds of event logs. Overall, we summarize that we obtain the ransomware prediction performance in the mid-90s for 120 seconds of execution time, the low-90s for 60 seconds, and the mid-80s for 30 seconds. The following section shares how including the Static-RWArmor predictive score improves our results for dynamic analysis (which addresses RQ2).

7.2 Addressing RQ2: RWArmor results

As described in the previous section, we produce a feature based on the probabilistic output received from the Static-RWArmor. The feature is injected after the Files features. Thus, the modified labels of the feature are $x[0] - x[8]$ for files, $x[9]$ for Static-RWArmor, $x[10]$ and $x[11]$ for process, and $x[12]$ for registry. A similar visualization of the decision tree structure is illustrated in Fig. 11 to showcase that the inclusion of the Static-RWArmor feature as a decision split is added in the 2nd level of the tree. The first decision split is based on the 1st Principal Component of the processes

Table 8 Binary classification performance of built machine learning models with file, registry, and process features

ML models	Execution time	Dynamic analysis				Static-informed dynamic analysis			
		Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Decision tree	120	0.8836	0.8887	0.8807	0.8857	0.8997	0.9049	0.8869	0.89095
Random forest	seconds	0.9536	0.9546	0.951	0.9512	0.9612	0.9626	0.9608	0.9596
Decision tree	60	0.8595	0.8557	0.8583	0.8588	0.8638	0.8654	0.8614	0.8626
Random forest	seconds	0.9095	0.9157	0.9083	0.9088	0.9238	0.9354	0.9214	0.9226
Decision tree	30	0.8071	0.8073	0.8182	0.8023	0.8132	0.8087	0.8187	0.8082
Random Forest	seconds	0.8588	0.8643	0.8457	0.8521	0.8642	0.8668	0.8577	0.8611

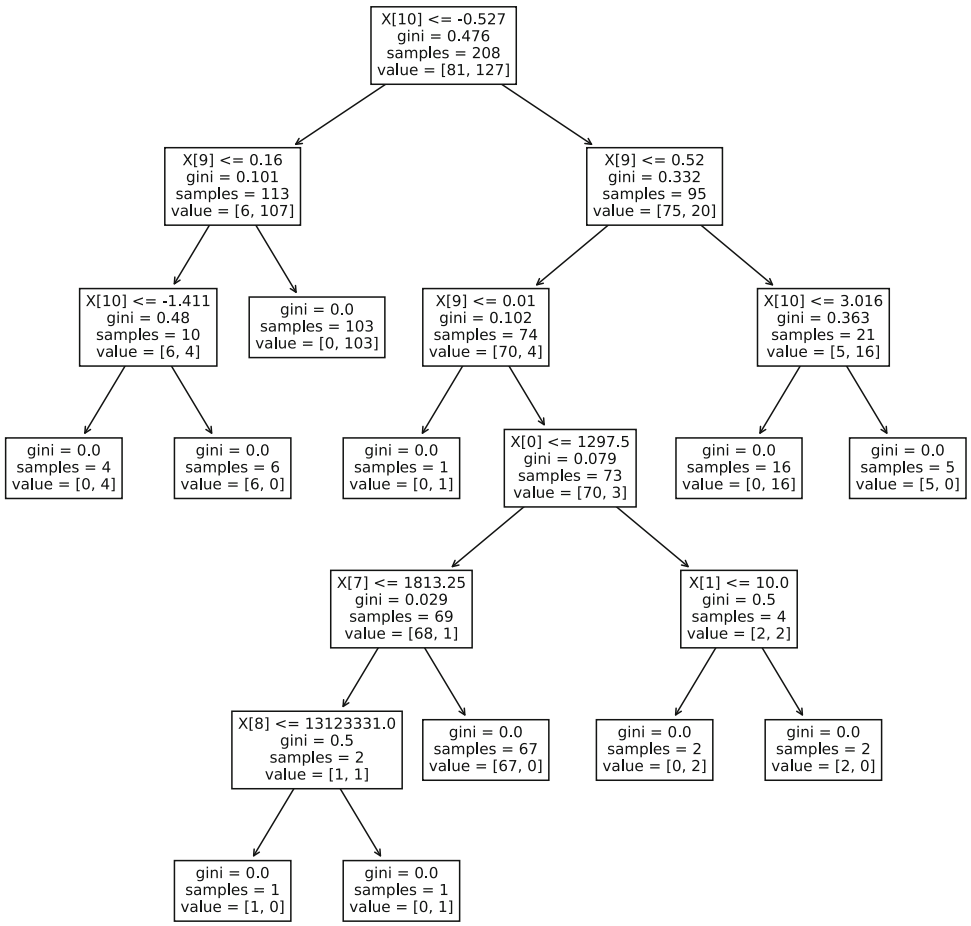

Fig. 11 Visualization of the built decision tree structure based on files, registry, and process feature spaces to showcase the rules used for prediction with static-RWArmor

feature ($x[10]$) as before, proving that the model does not become too biased with the Static-RWArmor as a feature. However, as per the figure, it plays a vital role in the decision process as it is present in the second decision split. The further splits are based on the features processed from files. It indicates that the decision tree and random forest classifier should perform better results in predicting ransomware than the dynamic analysis approach.

Next, our focus is turned into the static-informed dynamic analysis parts of Table 5, 6, 7, and 8. Like the dynamic analysis, the random forest classifier outperforms the decision tree classifier. It is evident from the tables that we achieve up to 2% of performance gain at every experimental setting when Static-RWArmor as a feature is included during the training and testing phase of the classifiers. Our empirical findings confirm that our proposed method, RWArmor (a static-informed dynamic analysis approach), improves the detection capabilities with 120, 60, and 30 seconds of execution events of ransomware. We present Fig. 12 to plot line graphs of the documented metrics of Table 8 as a comparison between dynamic analysis and static-informed dynamic analysis. As an illustration, it further proves our claim on RWArmor's effectiveness in accuracy, precision, recall, and F1 scores.

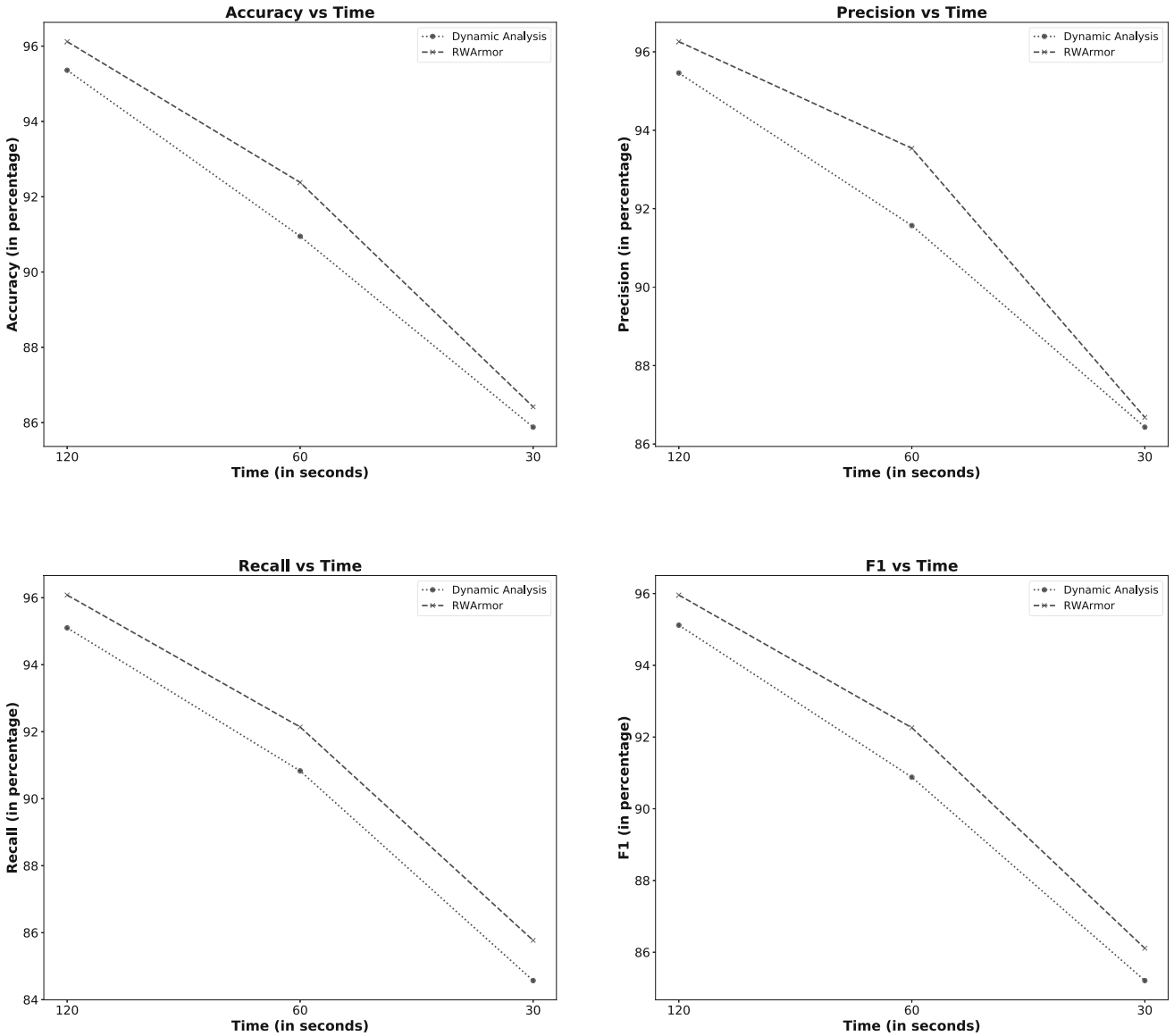

Fig. 12 A comparison visualization of the random forest classifier's performances between dynamic analysis and static-informed dynamic analysis (RWArmor) based on Files, Registry, and Process features (as per Table 8)

8 Discussion, limitations, and future work

This section provides a discussion of our research, including its limitations and potential areas for extensions that will lead to future work.

Detection Time. Our proposed method, RWArmor, aims to effectively detect cryptographic windows ransomware as early as possible during its execution cycle. We leverage the Random Forest algorithm to accomplish this goal with Windows events captured from the start of every ransomware sample's execution to 30, 60, and 120 seconds. The precision and recall scores are around 97% for 120 seconds, around 93% for 60 seconds, and around 86% for 30 seconds. In other words, the model predicts a sample is ransomware, it is correct 86% of the time given 30-second event logs. A promising future work is whether we can further improve the detection capabilities reported in our work.

Robustness. The basis of our analysis on the accurate Windows event logs collected from a sandbox environment (e.g., Any Run). Based on our domain knowledge, the dataset processing tasks take place to let the built machine learning models learn the underlying encryption patterns of ransomware. However, inspecting how well the models behave for other sandbox-generated and emulator-based reports will be interesting. This type of comparison study is left as our future work. On a side note, we conducted extensive feature engineering activities. One of the future works in this space is to construct a Deep Learning network that takes in such ran-

somware behavioral reports and identifies the best features for ransomware detection.

File Recovery. The prime reason behind devising a prompt detection approach is to flag a running ransomware process early enough to prevent further damage to the file system. Choosing 30 seconds of event logs for ransomware detection will still yield encryption of some files. Thus, an important extension of this research is to explore unique ways (e.g., decoy/canary files [24]) to recover the files which are already corrupted due to the ongoing ransomware attack.

9 Conclusion

In this paper, a static-informed dynamic analysis approach is proposed for cryptographic ransomware detection in a Windows environment, and we call it "RWArmor." Our method incorporates the predictive output of the trained machine learning models based on static features. Additionally, it is operated on the essential Windows events captured during the dynamic analysis of ransomware and benign applications. The uniqueness of our approach is that we combine all of them as features in order to propose a prompt detection mechanism. To achieve this goal, the random forest classifier achieves around 97% accuracy, precision, recall, and F1 scores within 120 seconds of time frame from the start of execution. The time frame is further minimized to evaluate the effectiveness of our approach. We find out that the number drops to 93% and 86% for the mentioned metrics with 60 seconds and 30 seconds of execution time, respectively. Additionally, our static-informed dynamic analysis approach gains ∼2% performance gain compared with the dynamic analysis-based features only for every experimental setting. We are hopeful that the data-driven concepts proposed in this research will be useful for prompt end-point protections.

Acknowledgements We thank the anonymous reviewers to review this manuscript and share valuable feedback to further improve its quality. We are grateful to VirusTotal and Any Run for generously giving us Academic Premium Subscription without any cost to use their platforms. We are thankful to Dr. Andrea Continella for providing us with the I/O Request Packet (IRP)-based dataset, published in [22]. We extend our sincere gratitude to Cybersecurity Education, Research & Outreach Center (CEROC) at Tennessee Tech University for supporting this research since its inception.

Research Data Policy and Data Availability Statements The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Compliance with Ethical Standards All authors confirm that accepted principles of ethical and professional conduct have been followed and declare that they have no conflict of interest. Additionally, this article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Connolly, L.Y., Wall, D.S.: The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures. Comput. Secur. **87**, 101568 (2019)
2. Pont, J., Abu Oun, O., Brierley, C., Arief, B., Hernandez-Castro, J.: A roadmap for improving the impact of anti-ransomware research. In: Secure IT Systems: 24th Nordic Conference, NordSec 2019, Aalborg, Denmark, November 18–20, 2019, Proceedings, pp. 137–154. Springer (2019)
3. Jindal, C., Salls, C., Aghakhani, H., Long, K., Kruegel, C., Vigna, G.: Neurlux: dynamic malware analysis without feature engineering. In: Proceedings of the 35th Annual Computer Security Applications Conference, pp. 444–455 (2019)
4. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. **44**(2), 1–42 (2008)
5. Shaukat, S.K., Ribeiro, V.J.: RansomWall: a layered defense system against cryptographic ransomware attacks using machine learning. In: 2018 10th international conference on communication systems & networks (COMSNETS), pp. 356–363. IEEE (2018)
6. Hasan, M.M., Rahman, M.M.: RansHunt: a support vector machines based ransomware analysis framework with integrated feature set. In: 2017 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–7. IEEE (2017)
7. Trizna, D.: Quo Vadis: hybrid machine learning meta-model based on contextual and behavioral malware representations. In: Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security, pp. 127–136 (2022)
8. Ayub, M.A., Sirai, A.: Similarity analysis of ransomware based on portable executable (PE) file metadata. In: 2021 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–6. IEEE (2021)
9. Subedi, K.P., Budhathoki, D.R., Dasgupta, D.: Forensic analysis of ransomware families using static and dynamic analysis. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 180–185. IEEE (2018)
10. Poudyal, S., Subedi, K.P., Dasgupta, D.: A framework for analyzing ransomware using machine learning. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1692–1699. IEEE (2018)
11. Poudyal, S., Dasgupta, D.: AI-powered ransomware detection framework. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1154–1161. IEEE (2020)
12. Poudyal, S., Dasgupta, D., Akhtar, Z., Gupta, K.: A multi-level ransomware detection framework using natural language processing and machine learning. In: 14th International Conference on Malicious and Unwanted Software" MALCON (2019)
13. Zhang, B., Xiao, W., Xiao, X., Sangaiah, A.K., Zhang, W., Zhang, J.: Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes. Future Gener. Comput. Syst. **110**, 708–720 (2020)
14. Medhat, M., Gaber, S., Abdelbaki, N.: A new static-based framework for ransomware detection. In: 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology

Congress (DASC/PiCom/DataCom/CyberSciTech), pp. 710–715. IEEE (2018)
15. Ahmed, Y.A., Koçer, B., Huda, S., Al-rimy, B.A.S., Hassan, M.M.: A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. J. Netw. Comput. Appl. **167**, 102753 (2020)
16. Al-rimy, B.A.S., Maarof, M.A., Prasetyo, Y.A., Shaid, S.Z.M., Ariffin, A.F.M.: Zero-day aware decision fusion-based model for crypto-ransomware early detection. Int. J. Integr. Eng. (2018). https://doi.org/10.30880/ijie.2018.10.06.011
17. Lu, T., Du, Y., Wu, J., Bao, Y.: 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, pp. 46–61. Springer (2020)
18. Sharif, M.I., Lanzi, A., Giffin, J.T., Lee, W.: Impeding malware analysis using conditional code obfuscation. In: NDSS (2008)
19. Collberg, C., Thomborson, C., Low, D.: A taxonomy of obfuscating transformations. Department of Computer Science, The University of Auckland, New Zealand, Tech. Rep (1997)
20. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 184–196 (1998)
21. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E.: Cutting the gordian knot: a look under the hood of ransomware attacks. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 3–24. Springer (2015)
22. Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., Maggi, F.: Shieldfs: a self-healing, ransomware-aware filesystem. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 336–347 (2016)
23. Kharraz, A., Arshad, S., Mulliner, C., Robertson, W.K., Kirda, E.: Unveil: a large-scale, automated approach to detecting ransomware. In: USENIX Security Symposium, vol. 25. Austin, Texas (2016)
24. Mehnaz, S., Mudgerikar, A., Bertino, E.: Rwguard: a real-time detection system against cryptographic ransomware. In: Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10–12, 2018, Proceedings, pp. 114–136. Springer (2018)
25. Gómez-Hernández, J.A., Álvarez-González, L., García-Teodoro, P.: R-Locker: thwarting ransomware action through a honeyfile-based approach. Comput. Secur. **73**, 389–398 (2018)
26. Moore, C.: Detecting ransomware with honeypot techniques. In: 2016 Cybersecurity and Cyberforensics Conference (CCC), pp. 77–81. IEEE (2016)
27. Yuill, J., Zappe, M., Denning, D., Feer, F.: Honeyfiles: deceptive files for intrusion detection. In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004, pp. 116–122. IEEE (2004)
28. Bowen, B.M., Hershkop, S., Keromytis, A.D., Stolfo, S.J.: Baiting inside attackers using decoy documents. In: International Conference on Security and Privacy in Communication Systems, pp. 51–70. Springer (2009)
29. Abdelsalam, M., Gupta, M., Mittal, S.: Artificial intelligence assisted malware analysis. In: Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, pp. 75–77 (2021)
30. McDole, A., Abdelsalam, M., Gupta, M., Mittal, S.: Analyzing CNN based behavioural malware detection techniques on cloud IaaS. In: CLOUD 2020 (2020)
31. McDole, A., Gupta, M., Abdelsalam, M., Mittal, S., Alazab, M.: Deep learning techniques for behavioural malware analysis in cloud IaaS. In Malware Analysis Using Artificial Intelligence and Deep Learning. Springer (2021)
32. Kimmell, J.C., Abdelsalam, M., Gupta, M.: Analyzing machine learning approaches for online malware detection in cloud. In: IEEE conference on smart computing (SMARTCOMP) 2021 (2021)
33. Kimmell, J.C., McDole, A.D., Abdelsalam, M., Gupta, M., Sandhu, R.: Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure. IEEE Access **9**, 68066–68080 (2021)
34. Scaife, N., Carter, H., Traynor, P., Butler, K.R.: Cryptolock (and drop it): stopping ransomware attacks on user data. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 303–312. IEEE (2016)
35. Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C.: Automated dynamic analysis of ransomware: benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020 (2016)
36. Kharraz, A., Kirda, E.: Redemption: real-time protection against ransomware at end-hosts. In: Research in Attacks, Intrusions, and Defenses: 20th International Symposium, RAID 2017, Atlanta, GA, USA, September 18–20, 2017, Proceedings, pp. 98–119. Springer (2017)
37. Palisse, A., Durand, A., Le Bouder, H., Le Guernic, C., Lanet, J.L.: 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, pp. 192–208. Springer (2017)
38. Chen, Z.G., Kang, H.S., Yin, S.N., Kim, S.R.: 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, pp. 196–201 (2017)
39. Daku, H., Zavarsky, P., Malik, Y.: Behavioral-based classification and identification of ransomware variants using machine learning. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 1560–1564. IEEE (2018)
40. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R., Choo, K.K.R., Newton, D.E.: DRTHIS: deep ransomware threat hunting and intelligence system at the fog layer. Future Gener. Comput. Syst. **90**, 94–104 (2019)
41. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. Future Gener. Comput. Syst. **101**, 476–491 (2019)
42. Roy, K.C., Chen, Q.: DeepRan: attention-based BiLSTM and CRF for ransomware early detection and classification. Inf. Syst. Front. **23**, 299–315 (2021)
43. Kok, S., Abdullah, A., Jhanjhi, N.: Early detection of crypto-ransomware using pre-encryption detection algorithm. J. King Saud Univ. Comput. Inf. Sci. **34**(5), 1984–1999 (2022)
44. Tang, F., Ma, B., Li, J., Zhang, F., Su, J., Ma, J.: RansomSpector: an introspection-based approach to detect crypto ransomware. Comput. Secur. **97**, 101997 (2020)
45. Alhawi, O.M., Baldwin, J., Dehghantanha, A.: Leveraging machine learning techniques for windows ransomware network traffic detection. In: Cyber Threat Intelligence, pp. 93–106 (2018)
46. Moussaileb, R., Cuppens, N., Lanet, J.L., Le Bouder, H.: Ransomware network traffic analysis for pre-encryption alert. In: Foundations and Practice of Security: 12th International Symposium, FPS 2019, Toulouse, France, November 5–7, 2019, Revised Selected Papers 12, pp. 20–38. Springer (2020)
47. Khammas, B.M.: Ransomware detection using random forest technique. ICT Express **6**(4), 325–331 (2020)

48. Kok, S., Abdullah, A., Jhanjhi, N., Supramaniam, M.: Prevention of crypto-ransomware using a pre-encryption detection algorithm. Computers **8**(4), 79 (2019)
49. Shijo, P., Salim, A.: Integrated static and dynamic analysis for malware detection. Procedia Comput. Sci. **46**, 804–811 (2015)
50. Walker, A., Sengupta, S.: Insights into malware detection via behavioral frequency analysis using machine learning. In: MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM), pp. 1–6. IEEE (2019)
51. Al-Rimy, B.A.S., Maarof, M.A., Alazab, M., Alsolami, F., Shaid, S.Z.M., Ghaleb, F.A., Al-Hadhrami, T., Ali, A.M.: A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. IEEE Access **8**, 140586 (2020)
52. Javaheri, D., Hosseinzadeh, M., Rahmani, A.M.: Detection and elimination of spyware and ransomware by intercepting kernel-level system routines. IEEE Access **6**, 78321–78332 (2018)
53. Cohen, A., Nissim, N.: Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. Expert Syst. Appl. **102**, 158–178 (2018)
54. Bekerman, D., Shapira, B., Rokach, L., Bar, A.: Unknown malware detection using network traffic classification. In: 2015 IEEE Conference on Communications and Network Security (CNS), pp. 134–142. IEEE (2015)
55. Azmoodeh, A., Dehghantanha, A., Conti, M., Choo, K.K.R.: Detecting crypto-ransomware in IoT networks based on energy consumption footprint. J. Ambient Intell. Hum. Comput. **9**, 1141–1152 (2018)
56. Cusack, G., Michel, O., Keller, E.: 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, pp. 1–6 (2018)
57. Aghakhani, H., Gritti, F., Mecca, F., Lindorfer, M., Ortolani, S., Balzarotti, D., Vigna, G., Kruegel, C.: When malware is packin'heat; limits of machine learning classifiers based on static analysis features. In: Network and Distributed Systems Security (NDSS) Symposium 2020 (2020)
58. Garfinkel, T., Adams, K., Warfield, A., Franklin, J.: Compatibility is not transparency: VMM detection myths and realities. In: HotOS (2007)
59. Lindorfer, M., Kolbitsch, C., Milani Comparetti, P.: Detecting environment-sensitive malware. In: Recent Advances in Intrusion Detection: 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20–21, 2011. Proceedings 14, pp. 338–357. Springer (2011)
60. Raffetseder, T., Kruegel, C., Kirda, E.: Detecting system emulators. In: Information Security: 10th International Conference, ISC 2007, Valparaíso, Chile, October 9–12, 2007. Proceedings 10, pp. 1–18. Springer (2007)
61. Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., Van Steen, M.: Prudent practices for designing malware experiments: status quo and outlook. In: 2012 IEEE Symposium on Security and Privacy, pp. 65–79. IEEE (2012)
62. Lee, K., Lee, S.Y., Yim, K.: Machine learning based file entropy analysis for ransomware detection in backup systems. IEEE Access **7**, 110205 (2019)
63. Kim, D.Y., Choi, G.Y., Lee, J.H.: White list-based ransomware real-time detection and prevention for user device protection. In: 2018 IEEE International Conference on Consumer Electronics (ICCE), pp. 1–5. IEEE (2018)
64. Jung, S., Won, Y.: Ransomware detection method based on context-aware entropy analysis. Soft Comput. **22**, 6731–6740 (2018)
65. Chew, C.J., Kumar, V.: Behaviour Based Ransomware Detection. EasyChair (2019)
66. May, M.J., Laron, E.: Combating ransomware using content analysis and complex file events. In: 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5. IEEE (2019)
67. Hirano, M., Kobayashi, R.: 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), pp. 1–6. IEEE (2019)
68. Microsoft Docs: Example I/O Request—An Overview—Windows drivers (2017). https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/example-i-o-request---an-overview
69. Ayub, M.A., Continella, A., Siraj, A.: 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), pp. 319–324. IEEE (2020)
70. Harang, R., Rudd, E.M.: SOREL-20M: a large scale benchmark dataset for malicious PE detection. arXiv preprint arXiv:2012.07634 (2020)
71. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdiscip. Rev. Comput. Stat. **2**(4), 433–459 (2010)
72. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)