Static-RWArmor: A Static Analysis Approach for Prevention of Cryptographic Windows Ransomware

Md. Ahsan Ayub*, Ambareen Siraj*, Bobby Filar†, and Maanak Gupta*
*Department of Computer Science, Tennessee Tech University, Cookeville, USA
†Sublime Security, Inc., Washington, District of Columbia, USA
Emails: mayub42@tntech.edu, asiraj@tntech.edu, bobby@sublimesecurity.com, and mgupta@tntech.edu

***Abstract*—The everlasting fight between security researchers and ransomware authors, including cyber criminals who leverage ransomware to cripple organizations worldwide, has continued to evolve as novel techniques are used to evade ransomware detection. The victim not only endures paramount financial loss from business downtime for several days and/or paying ransom to regain control of their environment but also becomes at risk of being exposed to the stolen digital assets out on the Internet. To tackle these threats against ransomware, our research project aims to identify (1) structural similarities among 2,436 cryptographic Windows ransomware samples per calendar year between 2017 and 2021 and (2) structural dissimilarities against 3,014 benign applications using machine learning classifiers. We base our analysis on PE metadata for similarity analysis and binary classification tasks. With the Cosine Index, we capture 71% – 87.80% and 66% – 82.30% of similarities based on imports and function names feature spaces, respectively. On the other hand, after designing four experimental settings, Random Forest outperforms other applied classifiers by achieving 91.75%, 91.99%, 90.47%, and 91.05% at best for accuracy, precision, recall, and F1 scores, respectively, for ransomware detection.**

***Index Terms*—Machine Learning, Ransomware, Similarity Analysis, Static Analysis**

I. Introduction

Ransomware is a special type of malicious software that encrypts or locks critical infrastructure or victim machines, in general, to demand a hefty ransom from them in exchange for giving back control of the environment. That said, we can group ransomware into two categories: "cryptographic" ransomware, encrypting important and sensitive digital assets, and "locker" ransomware, blocking user access from their device. In this study, our focus is on cryptographic ransomware, a.k.a. crypto-ransomware, that attacks especially Microsoft's Windows operating system and uses strong cryptographic libraries for file encryption. It is reported that cryptographic ransomware attacks are carried out more frequently than locker ransomware attacks [13], [46].

Although ransomware is not a novel malicious actor, it has been on the news as headlines since 2016 for the massive financial damages it causes for both organizations and individuals [42]. The magnitude of ransomware attacks has significantly grown over the years due to geopolitical unrest and the rise of ransomware activist groups globally. For example, the DarkSide ransomware attack on the colonial pipeline network (a critical infrastructure system), a company that supplies about half of the U.S. East Coast's gasoline, took place in May 2021. The company has a 5,500-mile pipeline system with a capacity of carrying 2.5 million barrels of fuel per day. Because of this ransomware-as-a-service (RaaS) affiliate program's attack, the Federal Motor Carrier Safety Administration (FMCSA) announced a state of emergency in 18 U.S. 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 by paying US$ 4.4 million worth of bitcoin [7].

Due to the gruesome political agenda and/or financial gain, small to large organizations are being targeted worldwide, especially in the USA, and adversaries use innovative ways to deploy malicious software. For example, the SophosLabs team reported in the "Sophos 2023 Threat Report" that the Darkside ransomware abused a clean antivirus utility program while it was a Google Updater application for Exx ransomware. Additionally, it is shared that ransomware authors are adapting new programming languages (*e.g.*, Rust, GoLang, etc.) to build ransomware to make it easier to compile and run in the victim machine while developing its capability of executing cross-platform [55]. For the severity of this frightful situation and to combat such threat actors, our goal in this paper is to investigate the prospect of detecting ransomware in terms of preventive measurements. In other words, we aim to examine the structural similarities among ransomware and dissimilarities between ransomware and benign applications so that we can prevent a ransomware attack before it even happens. It is a static analysis-based approach to detect ransomware: we call it "Static-RWArmor."

As an extension of [7] (our prior work), we ask the following research questions (RQ) in this paper:

RQ1. Do ransomware samples caught in the wild in a calendar year share similar structural information?

RQ2. Can we discover PE file metadata-based dissimilarities between ransomware samples and benign applications?

Our proposed answers to the aforementioned research questions will lead to identifying an unknown or new ransomware sample without running it in a safe environment. We hope our work will help security researchers locate ransomware at the beginning of their life cycle on a victim's Windows machine before the irreversible encryption process begins. Aligned with the research questions mentioned above, we list the main contributions of our paper as follows:

- We investigate how similar ransomware samples collected

in the same calendar year are based on the Cosine Index from 2017 to 2021.
- We build machine learning classifiers to effectively identify the structural dissimilarities between 2,436 ransomware samples and 3,014 benign applications.

Paper Organization. The rest of the paper is organized as follows: Section II describes the background. We discuss the related academic work in Section III. The methodology, including the dataset by which the experiments were conducted, and the empirical results are presented in Sections IV and V, respectively. We list the limitations of our study, as well as future work, in Section VI. Finally, the conclusion of this work is shared in Section VII.

II. Background

This section describes a few terminologies to understand our work better.

A. Portable Executable (PE) File

In Microsoft's Windows Operating System, a portable executable (PE) file is an object file that contains .exe (executable), .dll (dynamic link library), and .sys (system file) as extensions. It consists of several pieces of information in categories [60]. The highlight of some of them is as follows:

- File Header. It is the first container of a PE file. It contains several pieces of important information, such as the type of target machine, the number of PE sections, the date time stamp of the file's creation, an unsigned integer to identify the state of the file (*e.g.*, 0x10B is for an executable file), the size of the code section, the address of entry point, the size of the initialized and uninitialized data section, the subsystem value to indicate the Windows subsystem is needed, etc.
- Section Table. A section table entry consists of "name" (which is 8-byte long), "virtual size" (a value indicating the section's size when loaded into memory), "virtual address" (an address value of the section's first byte when loaded into memory), "size of row data" (a value indicating the size of initialized data on disk), etc. The most common section names include but are not limited to "text" - the executable code of the program; "data" - the initialized data; "bss" - the uninitialized data; "rdata" - read-only initialized data, "edata" - the export tables; "idata" - the import tables, and "reloc" - relocation information.
- Import Address Table. It provides two important pieces of information: "import libraries" that the PE file will be using in terms of "dll", and for each import library, it also lists the "function names." 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", "CharUpperA", etc. functions that are required from the "USER32.dll" import library.
- Export Address Table. It contains the public names of functions and values that other PE files can import.
- Resources Directory Table. The structure of the resource is tree-like. It informs us of its type, *e.g.*, manifest, icon, etc. The vital information we gather from this table is the language used. For example, we can extract such information during the inspection of a "Petya" ransomware family's sample and check whether it has any Russian language presence.

B. Static Analysis

Static analysis is an important technique that enables security researchers to investigate ransomware, or malware in general, without execution. To understand the capabilities of a target file, we extract its structural information for analysis. For example, examining PE metadata as features of all the collected ransomware samples is crucial for malware researchers to know how it was written by malware author(s). Performing static analysis of a PE file does not require kernel-level privilege or a virtual machine as opposed to dynamic analysis [17]. Additionally, we take note that ransomware authors write their samples in a way that checks the environment; for example, it is common to notice that a piece of malware will not be executing itself if it senses being run in a virtual environment [23], [39], [50]. Therefore, we are motivated to carry out static analysis tasks.

C. Binary Classification

In the field of Machine Learning, binary classification involves learning from instances of two distinct classes and then, given an instance, predicting the class it belongs to. With this learning approach, we attempt to distinguish two different entities. For example, the algorithms equipped to execute binary classification tasks help us identify a spam email [32], a specific disease of a patient [15], a malicious software [19], etc. We discuss the following algorithms to understand their details better as we have used them in our research project.

1) Support Vector Classification: Support Vector Machine (SVM), a memory-efficient algorithm, can be used for classification, regression, and outlier detection. It is effective when the number of instances is less than the number of features [45]. The algorithm can be configured with several kernel functions for the decision process, such as Linear, Polynomial, Radial Basis Function (RBF), and Sigmoid [1]. Our experiments find satisfactory results for "RBF" and "Polynomial" decision functions.

2) Decision Tree: Decision tree, a robust algorithm, is heavily used for classification and regression [8], [61]. It scans the data features to formulate decision rules to predict the class by incorporating Iterative Dichotomies 3 (ID3) algorithm, Successor of ID3 (C4.5) algorithm, Classification and Regression Tree (CART) algorithm, etc [10]. We used an optimized version of the CART algorithm implemented in the scikit-learn software [44]. Additionally, we select the Information Gain metric for tree segmentation [11].

3) Random Forest: A random forest classifier, an ensemble learning method, is a meta estimator that combines a forest of randomized decision tree estimators to improve the robustness [9]. It is tailored to apply Decision Tree algorithms on the subset of the dataset to increase the decision-making accuracy while avoiding overfitting. Similar to the decision tree algorithm, we can configure the classifier based on the Gini Impurity or Information Gain [24]. In our case, we choose the Gini method to evaluate the tree segmentation's quality.

4) AdaBoost and Gradient Boosting: Similar to the Random Forest classifier, we employ AdaBoost and Gradient Boosting classifiers, also examples of ensemble learning methods. AdaBoost, a powerful boosting algorithm, operates on a weighted voting mechanism of a sequence of weak learners for the prediction purpose [20]. It is used for classification and regression tasks [16], [27]. Gradient Boosting classifier operates on gradient-boosted decision tree algorithms, which offer a generalization of boosting to differentiable loss functions [21], [22]. We select the learning rate of the classifier as 1 to shrink the contribution of each tree.

III. Related Work

This section discusses the work reported by other researchers in this field.

A. State-of-the-Art Research Work

Static analysis is widely used to detect malicious software [36], [52]. Additionally, using off-the-shelf machine learning techniques led to exceptional success for academic and industry-based security researchers [5], [6], [28]–[30], [33], [35], [38], [41]. Investigation of static features to detect ransomware has been well studied for both mobile devices [4], [12], [14], [18], [31], [34], [56] (especially for Android OS) and Windows-based platforms [43], [51], [63], [64].

Similar to our work, which is focused on cryptographic ransomware built to target the Windows Operating System environment, all the cited research projects achieved satisfactory results from their approach to analyzing static features of ransomware. We note that the metadata of ransomware samples' PE file structure is a popular mechanism for proposing innovative detection schemes. We observe that security researchers utilized the information of the import address table to execute the statistical analysis tasks based on the frequency of appeared items [26], [57]. The detection techniques include Association Rule [49], and Cosine Similarity on DLLs used [47]. Additionally, we have noticed that a few researchers focused on a set of selective imports in their study, *e.g.*, interesting DLLs/function calls [48], encryption-based calls [63], etc. Furthermore, the Strings metadata has been similarly used to devise impactful schemes. For example, the researchers explored the presence of interesting strings in the ransomware samples, such as ransom, encrypt, bitcoin, crypto, IP addresses, etc. [43], [54].

B. Distinction from Existing Related Work

We present Table I to showcase each research paper's necessary details. While we focus on PE metadata feature space for this work, a few chose OpCode and Hexcode-based structural information to equip their machine learning models for training and testing purposes. The reason why three papers (*e.g.*, [26], [49], and [54]) are listed which collected dynamic analysis feature sets in their study, hence hybrid analysis, is because they reported the performance of machine learning classifiers for the extracted static features. Although we do not construct a deep learning network to learn the underlying patterns between ransomware and benign application, unlike [63], we notice that random forest and ensemble learning methods, generally, were applied in other mentioned work like ours. That said, we reduced the dimension of our feature space with Principal Component Analysis (PCA), whereas others did not mention using such a technique in their work. On the other hand, Yara is a pattern-matching tool developed and maintained by VirusTotal and is heavily used by malware researchers [62]. In our previous work [7], we created a couple of Yara-based scripts to identify whether or not a ransomware sample is packed and/or uses crypto libraries. As an extension, in this paper, we contribute by proposing a ransomware detection technique through Static Analysis based on PE metadata. Notably, we evaluate the built machine learning classifiers' performance with many more ransomware variants than others. Additionally, we identify similarities among ransomware (our prior work) and explore how many similarities we can capture from the ransomware samples collected in a calendar year (our current work). This task aims to investigate the similarities of a novel ransomware variant from other reported samples to understand its capabilities.

IV. Methodology

This section provides a detailed description of our experimental methodology to address the research questions.

A. Dataset Construction

The initial phase of our experiments was to construct the dataset, including both ransomware and benign applications. In [7], we collected 727 active ransomware samples. Including them, we gather additional ransomware PE data from Sophos-ReversingLabs 20 Million (SOREL-20M) dataset [25]. The repository extracted various disarmed malware samples' features and metadata. The types of malware samples include adware, flooder, ransomware, crypto-miner, 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, we write a Python script to utilize VirusTotal API v3 Engine [59], 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.

TABLE I
CLASSIFICATION OF RANSOMWARE DETECTION APPROACHES ON WINDOWS PLATFORM THROUGH STATIC ANALYSIS

Research Paper	Published Year	Hybrid Analysis	Samples' Count		Features			Technique		
			Ransomware	Benign	PE Metadata	OpCode	Hexcode	ML	DL	Yara
Medhat *et al.* [43]	2018	–	793	878	✓	–	–	–	–	✓
Zhang *et al.* [63]	2020	–	1,521	92	–	✓	–	–	✓	–
Zhang *et al.* [64]	2019	–	1,787	100	–	✓	–	✓	–	–
Reddy *et al.* [51]	2021	–	113	162	–	–	✓	✓	–	–
Hasan *et al.* [26]	2017	✓	360	460	✓	–	–	✓	–	–
Subedi *et al.* [57]	2018	–	211	239	✓	–	–	✓	–	–
Poudyal *et al.* [49]	2018	–	178	178	✓	–	–	✓	–	–
Poudyal *et al.* [47]	2018	✓	550	540	✓	✓	–	✓	–	–
Poudyal *et al.* [48]	2019	–	292	292	✓	✓	–	✓	–	–
Shaukat *et al.* [54]	2018	✓	579	442	✓	–	–	✓	–	–
Our Prior Work [7]	2021	–	727	–	✓	–	–	✓	–	✓
Our Current Work	2023	–	2,436	3,034	✓	–	–	✓	–	–

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, we store 3,014 benign applications' PE metadata. We build a PE metadata extraction engine to process the dataset using Python 3 programming language.

Data Storing Method. We group all the pieces of PE metadata information 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 and Uninitialized Data Size, OS Version, Magic, and PE Entry Point.
- VirusTotal Info: MD5, Scan ID, Total Scan Engines, and Number of Positives.

We hope other field researchers can benefit from our dataset's structured formation.

B. Experiment Setup

After constructing a dataset for ransomware samples and binary applications, we set up the experiments to address the research questions (as illustrated in fig. 1).

Methodology to Address RQ1: Similarity Analysis. To find the similarity among the ransomware samples from the same calendar year, we query our constructed dataset to generate a count. To further explain, we notice that we have the following number of ransomware samples per calendar year:

- 2017 (and past) – 14 DLLs and 820 EXEs (total: 834).

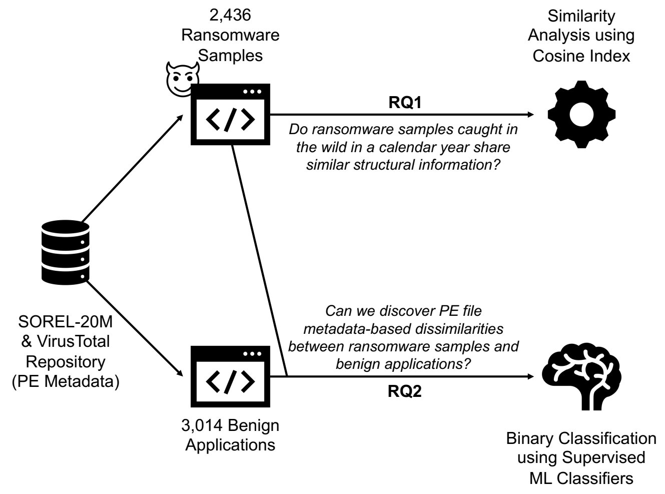
2,436 Ransomware Samples
Similarity Analysis using Cosine Index
RQ1
Do ransomware samples caught in the wild in a calendar year share similar structural information?
SOREL-20M & VirusTotal Repository (PE Metadata)
Can we discover PE file metadata-based dissimilarities between ransomware samples and benign applications?
RQ2
3,014 Benign Applications
Binary Classification using Supervised ML Classifiers

Fig. 1. Experimental methodology of detecting cryptographic ransomware through static analysis and identifying similarities among them.

- 2018 – 114 DLLs and 592 EXEs (total: 606).
- 2019 – 26 DLLs and 404 EXEs (total: 430).
- 2020 – 90 DLLs and 352 EXEs (total: 442).
- 2021 – 3 DLLs and 21 EXEs (total: 24).

With ransomware distribution between 2017 and 2021, we explore the feature sets and perform a frequency-based empirical case study. We highlight the following characteristics:

- The sample size has increased over the years. For example, the median value was 208.9 KB in 2017, but it has climbed up to 3764.224 KB because of presumably using RaaS (Ransomware-as-a-Service) kit.
- The uninitialized code size has been mostly 0 throughout the years.
- The maximum number of unique libraries used has decreased. Due to the use of packing libraries, the maximum number was 22 in 2017; however, it went down each year. In 2021, the number became 8.
- The maximum number of unique functions used has decreased. Again, because of being packed, it was 787

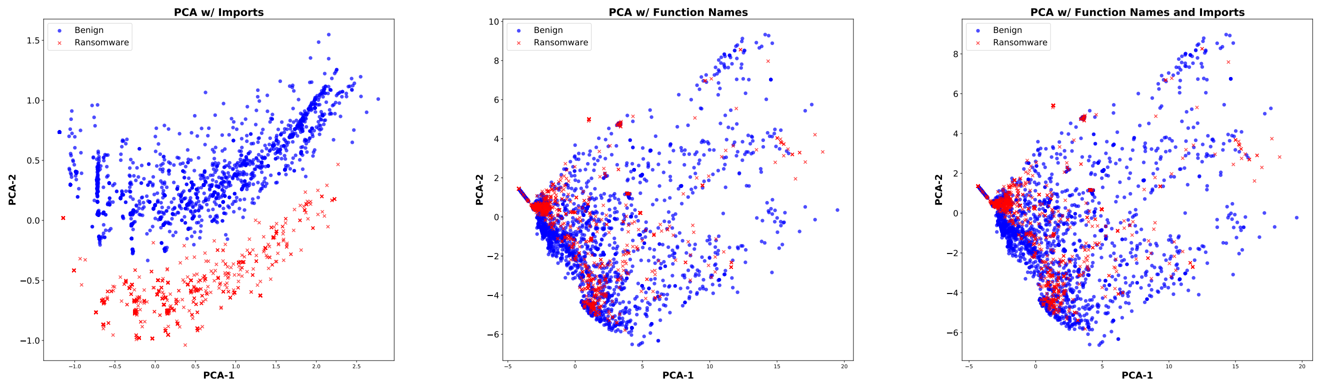

Fig. 2. Visualization of different feature spaces after applying Principal Component Analysis (PCA).

in 2017 but dropped to 165 in 2021.

- Although the median value is 80.28% for successful malware detection by VirusTotal, the minimum value over five years is alarming (28.79%).
- Ransomware samples tend to include a significantly fewer number of exports.

In [7], we obtained suspicious indicators from the "Imports" and "Function Names" feature spaces. Thus, we selected them to examine the similarity of all the ransomware samples yearly. To achieve this task, we choose the Cosine Index to generate the similarity index. Given x and y as row vectors, it computes their L2 normalized dot product based on the following equation [53].

$$\text{Cosine Index} = \frac{xy^T}{||x||||y||}$$

This mechanism is a popular choice when comparing the similarity between documents [37], [58]. In our case, we have a list of items (*e.g.*, import names and function names) for each ransomware sample, and then, based on that, we compute its cosine similarity with all other samples found in the same year.

Methodology to Address RQ2: Binary Classification. We perform binary classification tasks to discover the structural dissimilarities between ransomware and benign applications. As we explore similarities among ransomware samples based on imports and function names, we carry forward the same feature spaces to investigate if supervised machine learning algorithms can learn the underlying pattern. We select Support Vector, Decision Tree, Random Forest, AdaBoost, and Gradient Boosting classifiers to achieve this task. We administer four different experimental settings to evaluate the mentioned algorithms. We describe them as follows:

- 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) [2] to reduce the size of columns into two by capturing 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 sparse matrix, we utilize PCA while capturing 13% of information of the entire matrix dataset.
- Imports and Function Names Combined. We combine both "import" and "function names" feature spaces for our third experimental setting. 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.

We utilize Scikit Learn [44], a machine learning package, to apply Principal Component Analysis (PCA) on the high dimensional feature space of the processed dataset. Now, we present fig. 2 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.

V. Empirical Findings

This section reports the findings from all the experiments.

A. Answer to RQ1: Do ransomware samples caught in the wild in a calendar year share similar structural information?

As mentioned in the previous section, we utilize the "Cosine Index" to examine the similarity among the ransomware

TABLE II
PERFORMANCE OF A SUITE OF MACHINE LEARNING ALGORITHMS FOR BINARY CLASSIFICATION TASK ON DIFFERENT EXPERIMENTAL SETTINGS

	Imports (Experiment 1)				Function Names (Experiment 2)				Imports & Function Names (Experiment 3)				Numeric Features (Experiment 4)			
Model	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

samples collected yearly from 2017 to 2021. We compute the cosine index similarity, using Scikit Learn [44], of particular ransomware sample against all the samples caught in the same year. Then, we take the median value of the cosine index for that sample. For example, we possess 24 ransomware samples for 2021. We compare the similarity of each sample against the rest of the 23 samples. With this, we generate an array of cosine index values for each sample and evaluate its median result. This way, we compute all the samples' cosine index values yearly. Lastly, we calculate four statistical metrics from the list of samples' cosine values: minimum, median, mean, and maximum. We report the findings in Table III for two feature spaces: Imports and Function Names.

TABLE III
COSINE INDEX SIMILARITY OF RANSOMWARE SAMPLES PER CALENDAR YEAR BASED ON IMPORTS AND FUNCTION NAMES

	Imports				Function Names			
Year	min	median	mean	max	min	median	mean	max
2017	0.095	0.71	0.595	1.0	0.014	0.66	0.55	1.0
2018	0.12	0.76	0.63	1.0	0.019	0.75	0.59	1.0
2019	0.295	0.47	0.55	1.0	0.068	0.35	0.48	1.0
2020	0.34	0.77	0.73	1.0	0.34	0.823	0.77	1.0
2021	0.267	0.878	0.795	1.0	0.08	0.8	0.73	1.0

Apart from the samples collected in 2019, we observe that the cosine index similarity of ransomware samples per calendar year based on the median metric is between 71% and 87.80% for the imports feature space while 66% and 82.30% for the function names feature space. The median value for 2019 is below 50% for both feature spaces, indicating weak similarity among the samples we gathered in our study for the year mentioned.

B. Answer to RQ2: Can we discover PE file metadata-based dissimilarities between ransomware samples and benign applications?

As mentioned in the previous section, we design four experimental settings 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 [44] was used to apply and configure the algorithms based on the parameters discussed in Section 2. The evaluation of 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.

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

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

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

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

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

We report the performance of all the classifiers for the mentioned experiments in a tabular format (see Table II). 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.

VI. DISCUSSION AND FUTURE WORK

Our research aims to find the similarities among the studied samples collected per calendar year. However, it is noted that multiple samples can be part of a particular ransomware family. For example, the samples (MD5 hash) "db349b97c37d22f5ea1d1841e3c89eb4" and "41b5ba4bf74e65845fa8c9861ca34508" belong to the same ransomware family, named WannaCry. That said, we will venture to locate similarities of samples per ransomware family in a similar mechanism. The inclusion of Strings, OpCode, and Hexcode feature sets is also left for future work in this space.

It is essential to mention that both malicious and benign applications are found to be packed in the real world, which affects extracting the static features of such files [3], [40]. To address this concern, we include many packed ransomware samples in our study from the SOREL-20 repository. However, in future work, we intend to evaluate the efficiency of the applied machine learning classifiers only with the packed and encrypted ransomware samples. As the learning capabilities of deep learning networks are exceptional, constructing such a network will achieve better performance. We leave exploring this task as our future work.

However, we encourage organizations to use the 3-2-1 rule to stay safe regardless of the detection schemes they install in their environment. The rule involves keeping three backups of their data: 2 on different storage types and one offsite. Additionally, organizations can leverage cyber insurance to salvage the financial loss incurred by a ransomware attack. To contribute to the cyber defense community, we have published our implementation, along with the generated feature sets, on GitHub[1] under the MIT license.

VII. Conclusion

This static analysis approach, named "Static-RWArmor," is focused on detecting cryptographic Windows ransomware and is proposed based on machine learning. Additionally, we document the structural similarities observed among the ransomware samples collected between 2017 and 2021. In total, we obtain PE metadata from 2,436 ransomware samples (on which 247 DLLs and 2,189 EXEs) from SOREL-20M, VirusTotal, and [7]. We choose the cosine index to identify similarities in the Imports and Function Names feature space. We find 2021 ransomware samples to be 87.8% (median) similar for the Import feature space while 82.3% (median) similar for the Function Names feature space in 2020. In addition to reporting the Cosine Index's performance, we conduct a case study on all the ransomware samples to explore their characteristics (highlighted in Section IV(B)).

We employ several powerful machine learning algorithms to distinguish all the collected ransomware and 3,014 benign applications based on the PE metadata. The built classifiers, including Support Vector, Decision Tree, Random Forest, AdaBoost, and Gradient Boosting, achieve satisfactory results for all the designed experimental settings. We notice that Random Forest outperforms the other classifiers; however, other ensemble methods' performance, along with Decision Tree, is quite close. Nevertheless, regarding accuracy, precision, recall, and F1 scores, the Random Forest classifier is notably shown to reach the high 80s for Import and Function Names combined as feature space while low 90s for the numeric PE metadata features. Our research work is not free from limitations. We address them in Section VI and hope to carry out the listed tasks as future work for further improvements.

Acknowledgment

We thank the anonymous reviewers for reviewing this manuscript and sharing valuable feedback to improve its quality further. We extend our gratitude to Dr. Stacy Prowell from the Oak Ridge National Lab, Tennessee, USA, for his expert opinions at the initial stage of this research project. Additionally, we thank Mr. Joshua Saxe, currently working at Meta as a Senior Staff Research Scientist, for the valuable insights he shared to address the research questions in this paper. We are grateful to the Cybersecurity Education, Research & Outreach Center (CEROC) at Tennessee Tech University for supporting the research project. This work is partially supported by the NSF grants 2025682 and 2230609.

[1]https://github.com/AhsanAyub/deep_static_ransomware_analysis

References

[1] Support vector machines, 2023. URL: https://scikit-learn.org/stable/modules/svm.html.

[2] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[3] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. 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.

[4] Samah Alsoghyer and Iman Almomani. Ransomware detection system for android applications. *Electronics*, 8(8):868, 2019.

[5] Kshitiz Aryal, Maanak Gupta, and Mahmoud Abdelsalam. A survey on adversarial attacks for malware analysis. *arXiv preprint arXiv:2111.08223*, 2022.

[6] Kshitiz Aryal, Maanak Gupta, and Mahmoud Abdelsalam. Analysis of label-flip poisoning attack on machine learning based malware detector. In *In Proceedings of IEEE Big Data*, 2023.

[7] Md Ahsan Ayub and Ambareen Sirai. Similarity analysis of ransomware based on portable executable (pe) file metadata. In *IEEE Symposium Series on Computational Intelligence*, pages 1–6, 2021.

[8] Rodrigo Coelho Barros, Márcio Porto Basgalupp, Andre CPLF De Carvalho, and Alex A Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(3):291–312, 2011.

[9] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[10] Carla E Brodley and Paul E Utgoff. Multivariate decision trees. *Machine learning*, 19:45–77, 1995.

[11] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01):20–28, 2021.

[12] Aniello Cimitile et al. Talos: no more ransomware victims with formal methods. *International Journal of Information Security*, 17:719–738, 2018.

[13] Lena Y Connolly and David S Wall. The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures. *Computers & Security*, 87:101568, 2019.

[14] Alfredo Cuzzocrea, Fabio Martinelli, and Francesco Mercaldo. A novel structural-entropy-based classification technique for supporting android ransomware detection and analysis. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE, 2018.

[15] Dhiraj Dahiwade, Gajanan Patle, and Ektaa Meshram. Designing disease prediction model using machine learning approach. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1211–1215. IEEE, 2019.

[16] Harris Drucker. Improving regressors using boosting techniques. In *Icml*, volume 97, pages 107–115. Citeseer, 1997.

[17] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, 44(2):1–42, 2008.

[18] Hossam Faris et al. Optimizing extreme learning machines using chains of salps for efficient android ransomware detection. *Applied Sciences*, 2020.

[19] Yanick Fratantonio, Antonio Bianchi, William Robertson, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. TriggerScope: Towards Detecting Logic Bombs in Android Apps. In *Proceedings of the IEEE Symposium on Security and Privacy*, San Jose, CA, May 2016.

[20] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[21] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[22] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[23] Tal Garfinkel, Keith Adams, Andrew Warfield, and Jason Franklin. Compatibility is not transparency: Vmm detection myths and realities. In *HotOS*, 2007.

[24] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[25] Richard Harang and Ethan M Rudd. Sorel-20m: A large scale benchmark dataset for malicious pe detection. *arXiv preprint arXiv:2012.07634*, 2020.

[26] Md Mahbub Hasan and Md Mahbubur Rahman. Ranshunt: A support vector machines based ransomware analysis framework with integrated feature set. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–7. IEEE, 2017.

[27] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[28] Olivier Henchiri and Nathalie Japkowicz. A feature selection and evaluation scheme for computer virus detection. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 891–895. IEEE, 2006.

[29] Íñigo Íncer Romeo, Michael Theodorides, Sadia Afroz, and David Wagner. Adversarially robust malware detection using monotonic classification. In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*, pages 54–63, 2018.

[30] Grégoire Jacob, Paolo Milani Comparetti, Matthias Neugschwandtner, Christopher Kruegel, and Giovanni Vigna. A static, packer-agnostic filter to detect similar malware samples. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 9th International Conference, DIMVA 2012, Heraklion, Crete, Greece, July 26-27, 2012, Revised Selected Papers 9*, pages 102–122. Springer, 2013.

[31] Meet Kanwal and Sanjeev Thakur. An app based on static analysis for android ransomware. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 813–818. IEEE, 2017.

[32] Asif Karim, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, and Mamoun Alazab. A comprehensive survey for intelligent spam email detection. *IEEE Access*, 7:168261–168295, 2019.

[33] Md Enamul Karim, Andrew Walenstein, Arun Lakhotia, and Laxmi Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1-2):13–23, 2005.

[34] Alireza Karimi and Mohammad Hosein Moattar. Android ransomware detection using reduced opcode sequence and image similarity. In *IEEE Conference on Computer and Knowledge Engineering*, 2017.

[35] Jeffrey C Kimmell, Mahmoud Abdelsalam, and Maanak Gupta. Analyzing machine learning approaches for online malware detection in cloud. In *IEEE conference on smart computing (SMARTCOMP) 2021*, 2021.

[36] Dhilung Kirat, Lakshmanan Nataraj, Giovanni Vigna, and BS Manjunath. Sigmal: A static signal processing based malware triage. In *Annual Computer Security Applications Conference*, 2013.

[37] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International Conference on Cyber and IT Service Management*, pages 1–6. IEEE, 2016.

[38] Bo Li, Kevin Roundy, Chris Gates, and Yevgeniy Vorobeychik. Large-scale identification of malicious singleton files. In *Proceedings of the seventh ACM on conference on data and application security and privacy*, pages 227–238, 2017.

[39] Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. Detecting environment-sensitive malware. In *Recent Advances in Intrusion Detection: 14th International Symposium*. Springer, 2011.

[40] Robert Lyda and James Hamrock. Using entropy analysis to find encrypted and packed malware. *IEEE Security & Privacy*, 2007.

[41] Mohammad M Masud, Latifur Khan, and Bhavani Thuraisingham. A scalable multi-level feature extraction technique to detect malicious executables. *Information Systems Frontiers*, 10:33–45, 2008.

[42] Timothy McIntosh, ASM Kayes, Yi-Ping Phoebe Chen, Alex Ng, and Paul Watters. Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions. *ACM Computing Surveys (CSUR)*, 54(9):1–36, 2021.

[43] May Medhat, Samir Gaber, and Nashwa Abdelbaki. A new static-based framework for ransomware detection. In *IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing*, pages 710–715, 2018.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[45] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[46] Jamie Pont, Osama Abu Oun, Calvin Brierley, Budi Arief, and Julio Hernandez-Castro. 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*, pages 137–154. Springer, 2019.

[47] Subash Poudyal and Dipankar Dasgupta. Ai-powered ransomware detection framework. In *IEEE Symposium Series on Computational Intelligence*, 2020.

[48] Subash Poudyal, Dipankar Dasgupta, Zahid Akhtar, and K Gupta. A multi-level ransomware detection framework using natural language processing and machine learning. In *International Conference on Malicious and Unwanted Software" MALCON*, 2019.

[49] Subash Poudyal, Kul Prasad Subedi, and Dipankar Dasgupta. A framework for analyzing ransomware using machine learning. In *IEEE Symposium Series on Computational Intelligence*, 2018.

[50] Thomas Raffetseder, Christopher Kruegel, and Engin Kirda. Detecting system emulators. In *Information Security: 10th International Conference, ISC 2007, Valparaíso, Chile, October 9-12, 2007. Proceedings 10*, pages 1–18. Springer, 2007.

[51] Bheemidi Vikram Reddy, Gutha Jaya Krishna, Vadlamani Ravi, and Dipankar Dasgupta. Machine learning and feature selection based ransomware detection using hexacodes. In *Evolution in Computational Intelligence: Frontiers in Intelligent Computing: Theory and Applications (FICTA 2020), Volume 1*, pages 583–597. Springer, 2021.

[52] Nilo Redini, Ruoyu Wang, Aravind Machiry, Yan Shoshitaishvili, Giovanni Vigna, and Christopher Kruegel. B in t rimmer: Towards static binary debloating through abstract interpretation. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19–20, 2019, Proceedings 16*, pages 482–501. Springer, 2019.

[53] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[54] Saiyed Kashif Shaukat and Vinay J Ribeiro. Ransomwall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages 356–363. IEEE, 2018.

[55] Sophos. Sophos 2023 threat report - maturing criminal marketplaces present new challenges to defenders, Jan, 2023. URL: https://www.sophos.com/en-us/content/security-threat-report.

[56] Dan Su, Jiqiang Liu, Xiaoyang Wang, and Wei Wang. Detecting android locker-ransomware on chinese social networks. *IEEE Access*, 7:20381–20393, 2018.

[57] Kul Prasad Subedi, Daya Ram Budhathoki, and Dipankar Dasgupta. Forensic analysis of ransomware families using static and dynamic analysis. In *IEEE Security and Privacy Workshops*, 2018.

[58] Sandeep Tata and Jignesh M Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record*, 2007.

[59] VirusTotal. Public api v2.0, 2021. URL: https://developers.virustotal.com/reference.

[60] C+ Visual and Business Unit. Microsoft portable executable and common object file format specification, 1999.

[61] Min Xu, Pakorn Watanachaturaporn, Pramod K Varshney, and Manoj K Arora. Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, 97(3):322–336, 2005.

[62] Yara. The pattern matching swiss knife for malware researchers, 2021. URL: https://virustotal.github.io/yara/.

[63] Bin Zhang, Wentao Xiao, Xi Xiao, Arun Kumar Sangaiah, Weizhe Zhang, and Jiajia Zhang. Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes. *Future Generation Computer Systems*, 110:708–720, 2020.

[64] Hanqi Zhang, Xi Xiao, Francesco Mercaldo, Shiguang Ni, Fabio Martinelli, and Arun Kumar Sangaiah. Classification of ransomware families with machine learning based onn-gram of opcodes. *Future Generation Computer Systems*, 90:211–221, 2019.