Similarity Analysis of Ransomware based on Portable Executable (PE) File Metadata

Md. Ahsan Ayub and Ambareen Siraj
Department of Computer Science, Tennessee Tech University, Cookeville, USA
Emails: mayub42@tntech.edu, asiraj@tntech.edu

***Abstract*—Threats, posed by ransomware, are rapidly increasing, and its cost on both national and global scales is becoming significantly high as evidenced by the recent events. Ransomware carries out an irreversible process, where it encrypts victims' digital assets to seek financial compensations. Adversaries utilize different means to gain initial access to the target machines, such as phishing emails, vulnerable public-facing software, Remote Desktop Protocol (RDP), brute-force attacks, and stolen accounts. To combat these threats of ransomware, this paper aims to help researchers gain a better understanding of ransomware application profiles through static analysis, where we identify a list of suspicious indicators and similarities among 727 active ransomware samples. We start with generating portable executable (PE) metadata for all the studied samples. With our domain knowledge and exploratory data analysis tasks, we introduce some of the suspicious indicators of the structure of ransomware files. We reduce the dimensionality of the generated dataset by using the Principal Component Analysis (PCA) technique and discover clusters by applying the KMeans algorithm. This motivates us to utilize the one-class classification algorithms on the generated dataset. As a result, the algorithms learn the common data boundary in the structure of our studied ransomware samples, and thereby, we achieve the data-driven similarities. We use the findings to evaluate the trained classifiers with the test samples and observe that the Local Outlier Factor (LoF) performs better on all the selected feature spaces compared to the One-Class SVM and the Isolation Forest algorithms.**

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

I. Introduction

Ransomware, a special type of malware, is currently one of the major cyber threats. It is crippling small to large organizations by primarily encrypting and withholding important and sensitive digital assets to demand a ransom to release them. The adversaries are launching ransomware attacks on the computer systems of government bodies, healthcare, banking sector, airports, U.S. school districts, etc. to cause alarming damage to their enterprise resources. A recent (May 2021) gruesome example of such an attack on the critical infrastructure system is the DarkSide ransomware attack on the colonial pipeline network, a company that supplies about half of the U.S. East Coast's gasoline. 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 states to tackle the significant fuel shortages. It is to note that the company has a 5,500-mile pipeline system with a capacity of carrying 2.5 million barrels of fuel per day. After several days of investigations on this largest-ever cyber-attack on an American energy system, as well as paying US$ 4.4 million worth of bitcoin, the company resumed its operation after five days of national panic.

The average financial loss for an organization to recover from ransomware attacks is enormous. The Sophos Labs reported that the average recovery cost has jumped to US$ 1.85 million in 2021 considering downtime, people time, device cost, network cost, lost opportunity, ransom paid, etc. while it was US$ 761,106 in 2020. Additionally, it has been observed that paying the hefty ransom allows the victim organizations to restore only 65% (approx.) of the encrypted data [17]. However, the damage caused by such attacks does not stop in financial loss. Modern ransomware campaigns (*e.g.*, Ryuk, Sodinokibi, Nefilim, etc.) involve double extortion – forcing the victims to meet the criminals' demands as they threaten to publicize the stolen sensitive information [13]. Also, the reputation of affected organizations becomes at stake. Therefore, it is significantly important to empower researchers with more knowledge that they can use to build effective defenses against ransomware. In this respect, the two important research questions addressed in this paper are:

RQ1. Can we identify suspicious indicators from ransomware samples' structural information?

RQ2. Is there any PE file metadata-based similarities among the studied ransomware samples as well as their families?

The potential answers to these questions can help researchers identify ransomware at the beginning of their life cycle on a victim's Windows machine before the encryption process starts. Based on the experimental investigations on 727 active ransomware samples, belonging to 50 families, our major contributions in this study are as follows:

- We identify suspicious indicators on the generated PE metadata of ransomware based on the exploratory data analysis tasks and domain knowledge (see Section 4).
- We leverage the powerful one-class classification algorithms to capture the similarities among all the studied ransomware samples (see Section 5).

Paper Organization. Some of the background topics are covered in Section 2. We describe the test dataset and the methods used in the design of the experimental setting in section 3. Section 4 and 5 present the details of the empirical findings on suspicious indicators and the similarities among the studied ransomware samples. Related research work in the field of ransomware detection using static analysis techniques

978-1-7281-9048-8/21/$31.00 ©2021 IEEE

is discussed in section 6. A description of the limitations of our study, along with the future work, is presented in section 7 with a summary in section 8.

II. Background

A. Portable Executable (PE) File

The portable executable (or image) file is a common object file on the Windows Operating System with extensions include *.exe* (executable file), *.dll* (dynamic link library), *.sys* (system file), etc. The files are not architecture-specific and consist of several containers within their structure, such as file headers, section tables, import library, etc [22]. The file header holds important set of information, *e.g.*, 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 file, etc. In addition, from the Optional Header, we can read the magic number of the file (that tells us the image file's state), the size of code, initialized data, image, the subsystem required to run the image, DLL characteristics (a flag value to specify some of the security features for the PE file), and the address of the entry point (as the executable file is loaded into the memory). Next, the PE section-related information appears in the section header, which includes each section's virtual address, virtual size, and size of raw data. A few most common section names are *.text* (executable code), *.data* (read/write data), *.idata* (import address table), *.edata* (export information), etc. The import address table contains information about both the libraries and the imports used by the PE file. For example, for one of the studied samples from *Petya* ransomware family, we find out that it uses *wininet.dll* library, Windows Internet (WinINet) application programming interface (API), that interacts with 12 imports, for example, *HttpOpenRequest*, *HttpSendRequest*, etc. Similarly, we learn the exports of the PE file from the export address table.

B. Static Analysis

Static Analysis is a technique that involves examining malware (*e.g.*, ransomware, in our case) without executing it. This is an important step to gather any structural details to extract useful insights of the studied malware file to understand its capabilities. Our executed tasks include inspecting the information of the PE files' structure, extracting strings, functions, cryptographic libraries used, and metadata associated with the samples, scanning the ransomware files with anti-virus engines through VirusTotal [21], and checking whether the files are packed used to thwart analysis. We investigate all the studied ransomware samples inside a VMWare Virtual Box[1], where the host machine runs on macOS while the target machine operates on Microsoft Windows 10 operating system (64 bit).

C. One-Class Classification

The powerful One-Class Classification technique entails working with one target / positive class. The goal of the algorithms that carry out this classification task is to learn the behavior of a certain set of samples from the same domain. By building the learned class boundary from the training examples, we can allow the models to perform outlier detection, novelty detection, and concept learning or single class classification [7]. In this study, we aim to investigate the effectiveness of identifying clusters/class boundary of all the studied ransomware samples. Afterwards, we determine whether we can map any new observations (also known as, the test set) within the decision boundary. Thus, we explore finding similarities on a wide range of ransomware variants.

III. Experimental Methodology

Fig. 1 illustrates the design of our experimental settings that starts with collection of the dataset and then the generation of feature set to perform intelligence analysis to derive answers to the defined research questions.

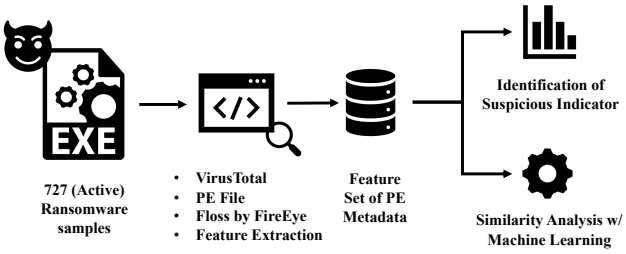

Fig. 1. Framework of our approach to identify similarities among the studied ransomware samples based on PE file metadata.

A. Dataset

As mentioned before, we base our study on 727 active crypto-ransomware samples, primarily collected from Continella *et al.* [4], along with other online resources, *e.g.*, VirusTotal[2] and VirusShare[3]. It is to note that each sample belongs to a particular ransomware family, and we assign them as per the steps described in Ayub *et al.* [2].

First, we use the VirusTotal API engine [21] to access the scan report via MD5 hashes. The report consists of the scan results of more than 50 Anti-Virus engines on average, and each engine labels one probable family. Therefore, as a generalized approach, we utilize the AVClass [15], a malware labeling tool, to determine one ransomware family for each sample. As a result, we obtain 50 ransomware families, among which 23 families have more than one sample represented.

B. Experimental Setup

In this section, we describe the steps involved in the experimentation with the collection of ransomware sample.

Generation of Feature Set of PE Metadata. We execute three separate tasks to generate the feature sets. First, we gather the numeric details of how many Anti-Virus (AV) engines, *e.g.*, Kaspersky, Symantec, Microsoft, etc., are able to identify the ransomware sample file as malicious or safe through VirusTotal API [21]. Then, we utilize the PEFile library, available as a Python module, to parse through the Portable Executable (PE) files' information [3]. To generate the metadata feature

[1]https://www.vmware.com/products/fusion.html

[2]https://www.virustotal.com

[3]https://virusshare.com

set with that library, the magic value from the DOS Header is used to identify whether or not the sample is an executable file. From the File Header, we access the type of the file (32 or 64 bit), the number of sections, and the characteristics flag. Then, we collect the size of code, the initialized data, the size of the image, the subsystem required to run this image file, and the DLL characteristics flags. From the section header, we extract the section names, along with each section's virtual size and size of raw data. We take note that if the virtual size is more than the size of raw data, then the section will allocate more memory space than it has data written to disk. Additionally, we read and store the Import Table and Export Table information from each sample. We use a couple of Yara-based scripts inside our feature extraction built tool to identify if the sample is packed or uses crypto libraries. At last, we leverage the FireEye Labs Obfuscated String Solver (FLOSS) to potentially extract obfuscated strings from the used samples [1]. With the FLOSS library, we extract the strings from each sample in three categories: static ASCII and UTF16LE strings, obfuscated strings, and stack strings. This concludes the generation of feature sets for all the studied samples. We develop our feature extraction tool using Python 3.

Analysis of the generated feature set. After the successful generation of the feature set, we carry out two unique tasks: (1) identification of any suspicious behavior (described in section 4), and (2) finding similarity based on the one-class classification algorithms (reported in section 5). We explore all the parts of generated feature set to locate data-driven suspicious behavior using our domain knowledge. We incorporate the statistical analysis to identify the highlights of our key findings. Then, we inspect the effectiveness of obtaining common clusters to group all the samples. To successfully perform this task, we leverage three efficient algorithms, such as One-Class Support Vector Machine (SVM), Isolation Forest, and Local Outlier Factor (LoF).

IV. Observations of Exploratory Data Analysis

At first, we examine all 727 ransomware samples to understand their PE structure based on the static analysis. We report some of the highlights of the analysis as follows:

- All the studied samples are Non-Executable types of files that target 32-bit Microsoft Windows machines.
- 16% of the studied samples provide information regarding when the files were created as well as the type of systems they require to run on. The majority of such samples were created in 2006 (the median value) while the latest one was in 2018. Conversely, we notice that all but one samples' images run in the Windows Graphical User Interface (GUI) subsystem while the other one runs in the Windows Character User Interface (CUI) subsystem.
- 87% of the studied samples allocate more memory space in their PE sections than they have data written to disk. It indicates that the loader can provide a respective section a chunk of memory space to store variables into it.
- We find that the total unique number of libraries and imports used by all the studied ransomware samples are 106 and 3,345 respectively. However, we do not find any sample that has any presence in the export tables.
- We utilize the VirusTotal API Engine [21] to scan all the ransomware samples. In general, it is observed that 20% of the engines report the samples as safe (the median value). While 6% of the engines labeled one sample as safe (the lowest value), 44% of the engines labeled another sample as safe (the highest value).
- We leverage the Yara tool [23], which was designed to identify and classify malware samples, to find out if a given ransomware sample uses packer and crypto libraries. With that, we notice that 4% and 11% of the studied samples show the usage of the packer and crypto libraries respectively. Additionally. we observe that the total unique number of packer and crypto libraries used by all the studied samples are 11 and 37 respectively.

Addressing RQ1. As mentioned earlier, we generate a list of unique imports, libraries, and Strings that are used by all 727 ransomware samples. With our domain knowledge, we aim to analyze each item in the list to derive a set of suspicious indicators that are frequent in our studied ransomware samples.

Imports. We start with describing the suspicious indicators obtained from the imports. We observe that several ransomware samples access the mouse or cursor movement Win32 API libraries, *e.g.*, *TrackMouseEvent*, *GetCursorPos*, etc. We fear that such libraries could be used to monitor whether the user is active or not. It is reported that the ransomware takes much longer time to perform its damage on the infected victim machines [8]. Thus, we suspect that such libraries might be used to to identify if the user is inactive. In addition to monitoring user's activeness, we include the process-based imports, *e.g.*, *TerminateProcess*, into our consideration. Due to the fact that ransomware needs to communicate with its command and control (C&C) server, we scan the list to gather a set of imports that are used to make network calls, such as *http*, *ftp*, *url*, and *icmp*. We notice that a good number of ransomware samples check if the process is being debugged by a user-mode debugger, *e.g.*, *IsDebuggerPresent*. This approach is used to determine whether the debugger is present so that ransomware can prevent itself from execution. It has been also reported that ransomware uses PowerShell to download its code, invoke command, create a backdoor, and propagate its infection to other machines [8]. Therefore, we add shell execution based imports, *e.g.*, *ShellExecuteA*, in our list of suspicious indicators. Since the goal of the ransomware is to encrypt the files, we include all the file-based imports, *e.g.*, *LockFile*, *EncryptFileA*, *UnlockFile*, *DecryptFileA*, etc., in our indicators' list. To summarize, we isolate 670 unique imports (out of 3,345) based on the following categories:

- Cursor and/or Mouse. We find it present in 62% samples.
- Network Calls (30% samples). Imports based on *http*, *ftp*, *url*, and *icmp* are present in 14%, 12%, 15%, and 17% samples respectively.
- Shell Execution (13% samples).
- Debugger Presence Checker (30% samples).

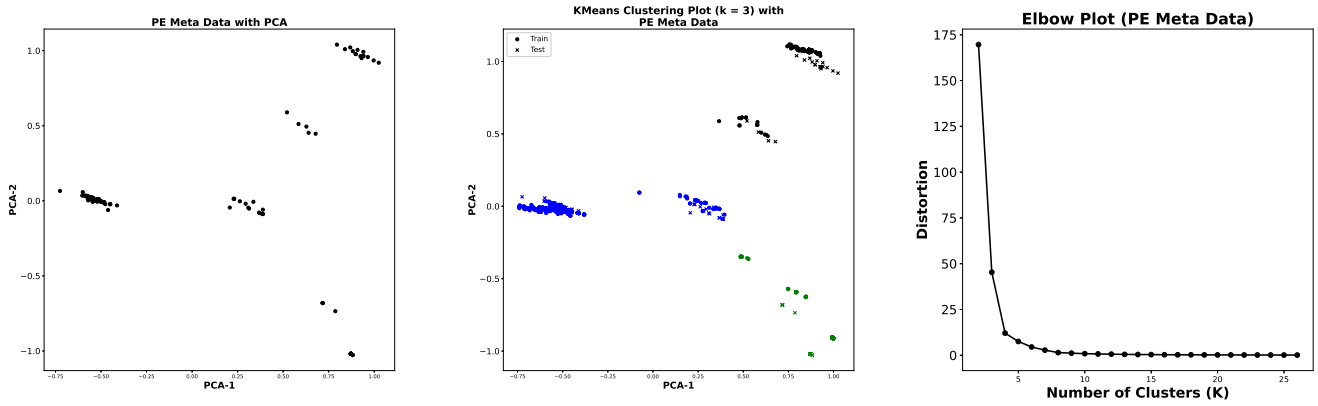

Fig. 2. Visualization of PE Meta Data with PCA feature space (left) and its KMeans clustering plot (middle) for k=3 – selected from the elbow plot (right).

- Process-based Imports (76% samples). The median count per sample is 2 while the max count is 12 (out of 39).
- File-based Imports (87% samples). The median count per sample is 16 while the max count is 60 (out of 223).

Libraries. One of the most common attack vectors of ransomware infection to the victim machine is through the remote desktop protocol (RDP). While examining the libraries used by the studied ransomware samples, we find out that 19% samples call *wtsapi32.dll*. With this library, the application can leverage the remote desktop service environment during the run-time. To communicate with C&C server, ransomware needs to access the Internet. We observe that 15% samples use *wininet.dll* - Internet Extensions for Win32, by which the application interacts with the http and ftp protocols to access online resources. Ransomware attempts to become aware of these processes running in the victim machine for different reasons, such as to terminate the Anti-Virus applications, to launch its attack while the machine is active, etc. [8]. Furthermore, we notice that 19% samples use *psapi.dll* - Process Status Helper API, that help them gain information about the running processes and device drivers.

Strings. We obtain a list of Strings from the FireEye Labs Obfuscated Strings Solver (FLOSS) in three categories: Static Strings, Decoded Strings – with the help of WinDbg debugger routine [19], and Stack Strings – recovered from the stack with the help of IDA Pro plugin [5]. We scan each item in the list to check if it belongs to an English word, and if it does, we perform further search queries on it. After we complete the search query tasks, we gather the following notable insights:

- Encryption-based keywords. The case-insensitive search results show us that "encrypt", "decrypt", "RSA", and "AES" keywords are present in 16.3%, 25.27%, 48.1%, and 22.1% samples respectively.
- Ransom-based notice. Similar to above, "payment" or "pay", "bitcoin" or "btc", and "usd" keywords are present in 14.09%, 7.74%, and 10.5% samples respectively.
- File path. Similarly, "C://" and "/windows" keywords are present in 6.35% and 7.18% samples respectively.

V. Empirical Findings: Similarity Analysis

We base our approach to identify similarities among the studied ransomware samples with several feature spaces, such as PE Metadata, Imports, Libraries, and PE Sections. The PE metadata feature space includes the statistic of each PE file structure, *e.g.*, characteristics, file size, size of code, size of initialized data, size of image, dll characteristics, and the number of PE sections, imports, libraries, packer libraries used, and crypto libraries used.

We begin our analysis by examining the possibility of finding clusters on this feature set (with a size of 727×11). To perform this task, we select the KMeans clustering technique [14] with the Min-Max scaling of the feature values to lie between 0 and 1. Then, to reduce the dimensionality of the feature set, we apply the Principal Component Analysis (PCA) to capture 38.43% and 26.05% of information with the first and second principal components respectively. As our goal is to include at least one sample of a given ransomware family in both the training and testing process, we remove the observations for families that have got only one sample – this reduces the size of the entire feature set as 703×2 for 27 ransomware family. From there, we take 80% and 20% of observations as the train and test instances respectively. We present Fig. 2 to show the feature distribution and the elbow plot to select the value of k for the KMeans algorithm. From the plot, we select $k = 3$ as this is the point before we notice a roughly linear decrease in the inertia. We additionally include the visualization of the algorithm's performance in Fig. 2.

With 3 clusters, we are able to isolate the PE metadata of 27 ransomware families' samples. This results further motivates us to carry out the One-Class classification tasks in order to find the similarities among all 727 ransomware samples.

Addressing RQ2. We select One-Class classification algorithms, *e.g.*, One-Class SVM, Isolation Forest, and Local Outlier Factor, to identify similarities among the ransomware files. We explore Imports, Libraries, and PE Sections feature

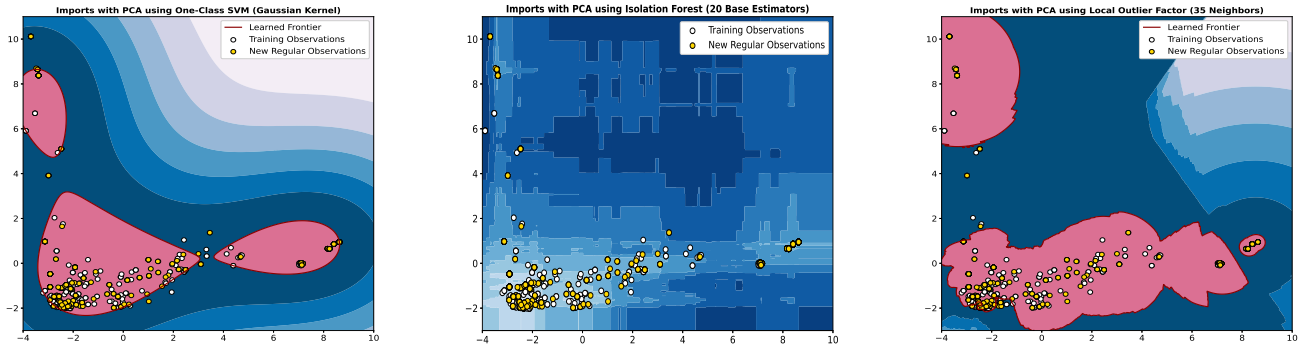

Fig. 3. Visualization of the learned cluster region of One-Class SVM (left), Isolation Forest (middle), and Local Outlier Factor (right) classifiers for the Imports with PCA feature space. White points are the training instances while yellow points are the testing instances from one of the 5-fold cross-validations.

spaces to assess each algorithm's performance in predicting new variants of ransomware files. That being said, once we train each model and obtain the learned cluster based on the training instances, we use two performance metrics to examine their effectiveness: Error Train – the percentage of training instances not falling within the cluster and Error Novel – the percentage of testing instances not falling within the cluster. For both cases, the lower the scores are, the better. Similar to the processes described above, we apply the PCA algorithm after scaling the dataset before the training process. For each experimental setting, we perform 5-fold cross-validation, and the reported scores for both metrics as mean values.

TABLE I
PERFORMANCE OF ONE-CLASS CLASSIFICATION ALGORITHMS IN DIFFERENT EXPERIMENTAL SETTINGS

Algorithm	Feature	Error Train	Error Novel
One-Class SVM	Imports	8.15%	18.52%
	Imports, Libraries	8.63%	18.11%
	Imports, PE Sections	7.88%	18.51%
	Imports, Libraries, PE Sections	8.77%	18.53%
Isolation Forest	Imports	7.50%	26.90%
	Imports, Libraries	6.95%	25.38%
	Imports, PE Sections	7.50%	26.90%
	Imports, Libraries, PE Sections	7.50%	26.90%
Local Outlier Factor (LOF)	Imports	6.57%	10.04%
	Imports, Libraries	6.91%	12.10%
	Imports, PE Sections	6.57%	10.04%
	Imports, Libraries, PE Sections	6.57%	10.04%

We present Fig. 3 to show an example of the learned clusters derived by each one of the one-class classification algorithms. The figure illustrates both the training and the testing instances. Additionally, we can locate whether the points that fall outside the cluster. We select the Gaussian Kernel for the One-Class SVM algorithm with 0.3 as an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. For the Isolation Forest algorithm, we assign 20 base estimators in the ensemble learning process and tune individual trees that can fit on the random subsets of the training data sampled with replacement. At last, we choose 35 number of neighbors for the Local Outlier Factor algorithm. Table I presents the performances for these hyper-meter settings of all the selected models. From the table, we observe that the Local Outlier Factor outperforms other two algorithms by a small margin for all the combination of the feature spaces. For instance, with the Local Outlier Factor and Imports feature space, we achieve 6.57% as Error Train and 10.04% as Error Novel.

VI. Related Work

Static analysis in the field of ransomware is prevalent as an initial assessment process to examine the potential threats a type of malware can pose. In this section, we discuss the prior work in this area to fight against the ransomware attacks.

The metadata of ransomware samples' PE file structure is one of the most popular mechanisms to propose innovative detection schemes. Security researchers have leveraged the details of import address table to execute the statistical analysis tasks based on the frequency of appeared items [6], [18]. Notably, the detection techniques include Association Rule [12] and Cosine Similarity on DLLs used [10]. Additionally, we have noticed that a few researchers focused on a set of selective imports in their study, *e.g.*, interesting DLLs/function calls [11], encryption-based calls [24], 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. [9], [16].

In our study, we aggregate the PE metadata of 727 active ransomware samples, belonging to 50 families, to isolate a list of suspicious indicators and find similarities among the generated datasets to detect new variants in the future. We are motivated that this acquired knowledge will help the dynamic analysis as per other research work in this field [10], [20]. This will further strengthen the capabilities of the ransomware detection schemes on Windows machines.

VII. Discussion and Future Work

This work has inspired us to include the OpCode and the HexCode of ransomware samples in order to carry out a

similar experimental study. Due to the exceptional capabilities, we feel encouraged to apply deep learning techniques to learn the underlying patterns of such feature spaces' complex structure. Inclusion of the benign software that are capable of encrypting files on the storage, *i.e.*, WinZip, Winrar, etc., is also left as one of the future work to investigate the data-driven dissimilarities with respect to the studied ransomware files. Additionally, we intend to add more recent ransomware variants (released after 2019) in our study.

VIII. Conclusion

This static analysis-based research study investigates 727 ransomware samples, belonging to 50 ransomware families, to derive usable intelligence from their portable executable (PE) format's structure. We extract the PE metadata to perform our analysis and primarily select imports, libraries, PE sections, and strings feature spaces to explore potential answers to the research questions. Based on domain knowledge, we highlight a list of suspicious indicators derived from exploratory data analysis tasks. Furthermore, we find out similarities among the studied ransomware samples from such feature spaces. We leverage the One-Class SVM, the Isolation Forest, and the Local Outlier Factor (LoF) algorithms to execute one-class classification tasks. We notice that the LoF, obtaining 6.57 in Error Train and 10.04 in Error Novel, achieves better results.

The aim of our study is to gather the knowledge required to detect ransomware from its executable file's structure so that we can capture its presence before execution. That being said, failure to successfully accomplish this task will lead to compromising the digital assets of the victim machine. Thus, the security researchers and defenders encourage the organizations to use the 3-2-1 rule, that is to keep 3 back-ups of their data: 2 on different storage types while 1 on offsite. To contribute to the cyber defense community, we have published our implementation, along with the generated feature sets, on GitHub[4] under the MIT license.

Acknowledgement

We would like to extend our gratitude to Dr. Stacy Prowell from the Oak Ridge National Lab in Tennessee, USA for his expert opinions and insights at the initial stage of this research project. The research work reported in this paper has been entirely supported by Cybersecurity Education, Research & Outreach Center (CEROC) at Tennessee Tech University.

References

[1] Fireeye labs obfuscated string solver (floss), 2021. URL: https://github.com/fireeye/flare-floss.

[2] Md Ahsan Ayub, Andrea Continella, and Ambareen Siraj. 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)*, pages 319–324. IEEE, 2020.

[3] Ero Carrera. Pefile, 2021. URL: https://github.com/erocarrera/pefile.

[4] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347, 2016.

[5] Jay Smith from FireEye. Flare ida pro script series: Automatic recovery of constructed strings in malware, 2014. URL: https://www.fireeye.com/blog/threat-research/2014/08/flare-ida-pro-script-series-automatic-recovery-of-constructed-strings-in/-malware.html.

[6] 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.

[7] Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[8] Ryan Maglaque Magno Logan, Erika Mendoza and Nikko Tamaña. Trendmicro research: The state of ransomware, 2020. URL: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/the-state-of-ransomware-2020-s-catch-22/?utm_source=trendmicroresearch&utm_medium=smk&utm_campaign=0203_StateRansomware.

[9] May Medhat, Samir Gaber, and Nashwa Abdelbaki. A new static-based framework for ransomware detection. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 710–715. IEEE, 2018.

[10] Subash Poudyal and Dipankar Dasgupta. Ai-powered ransomware detection framework. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1154–1161. IEEE, 2020.

[11] Subash Poudyal, Dipankar Dasgupta, Zahid Akhtar, and K Gupta. A multi-level ransomware detection framework using natural language processing and machine learning. In *14th International Conference on Malicious and Unwanted Software" MALCON*, 2019.

[12] Subash Poudyal, Kul Prasad Subedi, and Dipankar Dasgupta. A framework for analyzing ransomware using machine learning. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1692–1699. IEEE, 2018.

[13] TrendMicro Research. Ransomware, 2021. URL: https://www.trendmicro.com/vinfo/us/security/definition/ransomware.

[14] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[15] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, 2016.

[16] 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.

[17] Sophos. The state of ransomware 2021, April, 2021. URL: https://secure2.sophos.com/en-us/content/state-of-ransomware.aspx.

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

[19] FireEye Tyler Dean. Flare script series: Automating obfuscated string decoding, 2015. URL: https://www.fireeye.com/blog/threat-research/2015/12/flare_script_series.html.

[20] Deepti Vidyarthi, CRS Kumar, Subrata Rakshit, and Shailesh Chansarkar. Static malware analysis to identify ransomware properties. *International Journal of Computer Science Issues (IJCSI)*, 16(3):10–17, 2019.

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

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

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

[24] 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.

[4] https://github.com/TnTech-CEROC/static_ransomware_analysis