

Data-Centric Machine Learning Approach for Early Ransomware Detection and Attribution

A. Vehabovic¹, H. Zanddizari², N. Ghani¹, F. Shaikh¹, E. Bou-Harb², M. Safaei Pour³, J. Crichigno⁴
¹Univ. of South Florida, ²Univ. of Texas San Antonio, ³San Diego State Univ., ⁴Univ. of South Carolina

Abstract—Researchers have proposed a wide range of ransomware detection and analysis schemes. However, most of these efforts have focused on older families targeting Windows 7/8 systems. Hence there is a critical need to develop efficient solutions to tackle the latest threats, many of which may have relatively fewer samples to analyze. This paper presents a *machine learning* (ML) framework for early ransomware detection and attribution. The solution pursues a data-centric approach which uses a minimalist ransomware dataset and implements static analysis using *portable executable* (PE) files. Results for several ML classifiers confirm strong performance in terms of accuracy and zero-day threat detection.

Index Terms—Cybersecurity, malware analysis, ransomware detection and attribution

I. INTRODUCTION

Ransomware operates by encrypting files on a host computer and demanding some form of payment to release the keys. This malware has become the most lucrative revenue source for cybercriminals, and many ransomware “families” have impacted a wide range of users. Moreover, numerous cyber-criminal affiliates are also offering *ransomware-as-a-service* (RaaS), further reducing the barrier to such extortion [1].

Ransomware follows a multi-stage “kill-chain” comprising of reconnaissance, distribution, installation, communication, encryption, and extortion [2], [3]. To date, numerous designs have been evolved with increasing levels of secrecy, speed, and complexity. For example, various methods have been used to breach systems (e.g., remote access, drive-by, and privilege escalation) and encrypt data in collaboration with *command and control* (C&C) servers. Data exfiltration has also been used to extort users (double ransomware) [2]. As this threat continues to grow, surveys indicate that almost half of large corporations have experienced such attacks [4]. Windows ransomware is of particular concern as this *operating system* (OS) is still the most prevalent.

In light of the above, researchers have proposed a range of ransomware analysis solutions. Many of these schemes extract information from network traces or host files/logs to train advanced *machine learning* (ML) classifiers. However, most efforts have focused on a specific ransomware family or older families targeting

dated Windows 7/8 systems. As such, these methods may not be applicable to the latest threats facing Windows 10/11 users. Hence there is a pressing need to detect new ransomware designs *and* classify them for improved mitigation, i.e., attribution. Preferably, ransomware should be tackled early in the kill-chain to minimize damage [1]. Since new ransomware releases will likely have fewer available samples, solutions must also operate effectively with smaller “minimalist” datasets. This requirement is very much in line with current trends in *artificial intelligence* (AI) to develop more focused “data-centric” solutions [5].

Accordingly, this paper presents a novel ML solution for ransomware detection and attribution using static analysis. First, a unique malware repository is built by collecting samples of some of the latest ransomware families, i.e., Babuk/Babyk, BlackCat, Chaos, DJVu/STOP, Hive, LockBit, Netwalker, Sodinokibi/REvil, and WannaCry (after 2017). Next, feature extraction is done using Windows *portable executable* (PE) format file information. Finally, several supervised ML classifiers are trained and tested on these extracted features, including *support vector machines* (SVM), *random forest* (RF), *extreme gradient boost* (XGBoost), and *feed-forward neural networks* (FNN) [6]. Overall, this solution has very amenable run-times and can be integrated into network/host-based defenses to target ransomware early in the kill-chain (prevention).

This paper is organized as follows. Section II reviews some key studies on ransomware analysis. Next, Section III details the proposed ML-based framework, including dataset collection and feature extraction. Performance results are then presented in Section IV, followed by future work directions in Section V.

II. LITERATURE REVIEW

A range of ransomware analysis schemes have been proposed, and survey articles have detailed various (overlapping) taxonomies to classify these methods, e.g., static or dynamic analysis, network- or host-based, etc [1]- [3]. These efforts are further reviewed here.

Static analysis examines executable files to detect artifacts of maliciousness, e.g., via author attribution,

code/segment identification (de-anonymization), etc [1]. Some common methods used here include *binary code analysis* (BCA), source code analysis via reverse engineering, and C&C server domain prediction [2]. For example, [7] specifies a multi-level framework to detect ransomware from raw binaries, assembly code, and libraries. ML classifiers are then trained with the extracted data, yielding detection rates around 90%. Meanwhile, [8] transforms code sequences into N-grams and extracts frequency-based features for classification. Results show detection rates around 91% for several ML classifiers (decision tree, RF, etc). However, code-based analysis is very labor-intensive [9] and represents a more latent “post-infection” forensics approach.

Recent efforts have also used other static features to analyze ransomware. For example, [10] leverages image processing techniques to convert ransomware binary files into grayscale images and then performs texture analysis for feature extraction. Results for several ML classifiers show high accuracy (97%) for a small dataset with a mix of old and new ransoms (379 samples). However, this scheme imposes added computational burdens and does not consider benign applications. Meanwhile, [11] details another static analysis scheme which extracts entropy and image-based features to train a specialized Siamese NN classifier. Tests with a small dataset (about 1,000 samples and 10 families) show accuracy values in the mid-90% range but notably lower precision and recall rates (upper 70% range). Also, most of the ransomware families used here are older (mid-2010s) and benign applications are not considered.

Studies have also used static PE header file data for broader malware detection (not just ransomware). However, these efforts focus on detection and not attribution. For example, [12] collects many samples (over 100,000) from a repository called VX Heaven (now inactive) and trains ML classifiers using 7-10 extracted PE file features. Results show detection rates in the upper 90% range. Also, [13] extracts PE features from about 5,500 malware samples and 1,200 benign applications (early 2010s). Detection is done using a set of heuristics, achieving 95% accuracy. Finally, [14] extracts 9 PE file features (on sections, data directories, and entropy) from a dataset with 1,200 malicious and benign samples each. Results for several classifiers show 95% detection rates. However, these studies present no details on their malware datasets, most of which are over a decade old.

By contrast, dynamic analysis scans run-time actions and event sequences for ransomware activity. Specifically, dynamic network-based schemes examine packet traces for C&C communications, *domain name service* (DNS) queries, network storage access, etc. For example, [15] presents a detection system for Locky ransomware which uses traffic features to train classi-

fiers and yields over 95% detection rates. Meanwhile, [16] analyzes *server message block* (SMB) protocol patterns to detect older ransomware (2015-2017). The NetConverse scheme [17] also uses ML methods to analyze host traffic for earlier threats and achieves high detection rates (over 95%). Finally, [18] uses deep learning to analyze network activity and classify abnormal operation in Windows 7. Results show high detection rates for several families (over 97%).

Meanwhile, dynamic host-based schemes monitor local system activity to detect ransomware, e.g., memory and file operations, *application programmer interface* (API) function calls, *dynamic link library* (DLL) calls, etc. For example, [19] uses a sandbox to track file encryption/deletion, persistent messages, etc. Results show 96% detection rates for older ransomware types (mid-2010s). Also, [20] presents a scheme to monitor and store encryption keys for ransomware detection and file recovery. Results show successful mitigation of 12 out of 20 families. Similarly, [21] scans input/output requests for ransomware activity and flags affected files. Studies have also proposed ransomware “paranoia” schemes that try to detect environments and avoid fingerprinting/detection, e.g., [22] tracks API calls.

Although the above works present some notable contributions, key concerns still remain. Foremost, studies have largely focused on older ransomware targeting Windows 7/8 systems (mid-2010s). Given the expanding nature of this threat, it is imperative to study newer families targeting Windows 10/11. However, there are few datasets here, and new malwares may have smaller sample sizes to analyze (a challenge for ML schemes). Hence effective “data-centric” [5] schemes are required for minimalist datasets. Finally, ransomware detection and attribution schemes must have amenable run-times and preferably target ransomware earlier the distribution/delivery stages to minimize damage [2]. It is here that static analysis offers an expedient approach for tackling malicious payloads prior to infection. By contrast, dynamic analysis requires more indepth examination of network or host activities over longer intervals in virtual environments. As a result, a static analysis solution is presented using PE format file analysis.

III. DATA-CENTRIC STATIC ANALYSIS USING ML

The static analysis framework for ransomware detection and attribution (classification) is shown in Fig. 1 and comprises of several stages. The first stage (Empirical Data Collection) builds an up-to-date repository of some of the latest Windows 10/11 ransomware threats (since 2017). Regular benign Windows-based applications are also added here to improve classifier performance. The second stage (Feature Selection/Extraction) processes raw executables to extract key features. An

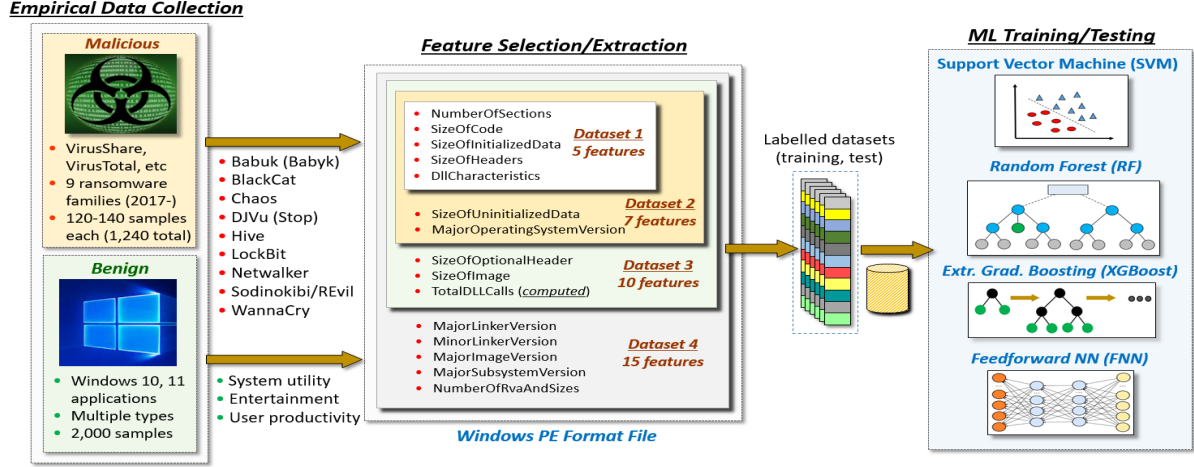


Fig. 1. Overview of static analysis ML framework for ransomware detection and attribution

efficient static analysis approach is proposed here using Windows PE format files. Finally, the last stage (ML Training/Testing) uses the feature datasets to train ML classifiers to detect and attribute ransomware. On a high level, this setup follows a well-defined ML approach, similar to that used in other studies. However, the novel contributions here include the collection of new ransomware datasets and extraction of lightweight static feature sets. Further details are now presented.

TABLE I
EMPIRICAL DATASET

Family	Samples	Avg. Size	Avg. PE File
Babuk (Babyk)	140	0.19 MB	32.68 KB
BlackCat	120	3.91 MB	1,147 KB
Chaos	140	0.49 MB	35.2 KB
DJVu (STOP)	140	0.71 MB	66.2 KB
Hive	140	3.51 MB	403.9 KB
LockBit	140	1.30 MB	171.5 KB
Netwalker	140	0.26 MB	35.72 KB
Sodinokibi	140	0.30 MB	50.89 KB
WannaCry	140	7.62 MB	21.83 KB
Benign	2,000	26.86 MB	155.88 KB

A. Empirical Data Collection

As per Section II, existing studies on PE file analysis provide little/no details on their datasets, e.g., type of malwares, executable file sizes, collection time frames, percentage of ransomware, etc. Many of these malwares are old and related repositories are inactive [12]. Hence a new repository is curated for the latest ransomware families. Now given the rapidly changing nature of the ransomware threat, it may be difficult to get sufficient samples of each. Hence realistic “data-centric” ML frameworks must achieve good detection and attribution with minimalist datasets (perhaps only hundreds of

samples). However, limited dataset size/diversity can also have a negative impact on classifier performance.

Now many active repositories host malware executables, e.g., MalwareBazar, Triage, VirusShare, and VirusTotal, etc. These sites provide varying degrees of access and usability, e.g., VirusTotal and VirusShare require registration to access uploads. Detailed cross-checking and analysis also shows notable duplication across portals, e.g., many Sodinokibi samples on MalwareBazar match those on Triage. There are also discrepancies between the number of samples for each family, e.g., DJVu is abundant whereas Babuk/Babyk and BlackCat are more scarce. Finally, some repositories (VirusShare and VirusTotal) do not organize or label their data, further complicating collection. Hence unlabeled data dumps have to be tediously analyzed using hashing and cross-checked with labelled samples. Hence there is potential for a lack of diversity, even scarcity, of new ransomware.

In light of the above, a smaller “minimalist” data repository is curated with 9 active ransomware families, i.e., Babuk/Babyk, BlackCat, Chaos, DJVu/STOP, Hive, LockBit, Netwalker, Sodinokibi/REvil, and WannaCry (Table I). These families are amongst the most prevalent ransomware threats in 2022, as per the IBM X-Force Threat Intelligence Index, i.e., LockBit (17%) followed by WannaCry (11%) and BlackCat (9%). A total of 140 unique executables are collected for each family, except for BlackCat which only yielded 120 samples due to scarcity, i.e., total of 1,240 malicious samples. Many Windows 10/11 applications are also added to construct a benign class (2,000 samples). These programs are collected from a range of websites and include system utility, entertainment, and productivity tools (Fig. 1). Overall, having a large set of non-malicious training data is very beneficial since regular applications down-

loads will exceed (unintended) ransomware downloads, This addition contrasts with work in [10], [11].

B. Feature Selection/Extraction

ML classifier performance is heavily dependent upon input training data. Hence feature extraction (engineering) plays a vital role in transforming raw executables to generate meaningful information for classifiers [6]. As per Section II, static analysis is more expedient for tracking ransomware early in its kill-chain. Hence this strategy is applied to Windows PE format files which contain data structures to support program execution in 32-bit and 64-bit Windows OS environments. Namely, these files use the *common object file format* (COFF) and contain information for the OS loader to setup/run wrapped executable code (including memory mapping and permissions). For example, a PE format file has several initial lead-in headers along with multiple sections. Here each section specifies file content (i.e., code or data) and also contains its own section header.

As per Section II, studies on PE format files have considered a range of malwares for Windows 7/8 [12]- [14] (mostly unspecified and not necessarily ransomware). Hence there is a further need to extend such analysis to Windows 10/11 ransomware threats. Now PE files contain a wealth of information, and programs can have unique non-overlapping parameters (depending upon functionality). Hence when extracting PE format data, it is important to select a subset of parameters which exist across all sample files and also exhibit good variability.

In light of the above, PE files are generated for all executables, with the resultant sizes shown in Table I. A total of 4 datasets are built by extracting feature vectors with 5, 7, 10, and 15 parameters, labeled as Datasets 1-4, respectively (Fig. 1). Each successive vector expands upon its predecessor by adding new parameters. Now the exact parameters are chosen using careful experimentation with the *Image_File_Header*, *Image_Optional_Header*, and *Image_Section_Header* sections. Some key features include *NumberOfSections*, *SizeOfCode*, *SizeOfHeaders*, etc. Note that PE files also contain information on *dynamic-link library* (DLL) calls which are indicative of functionality. For example, ransomware typically calls encryption, socket communications, and registry-modification functions. Hence the total number of DLL calls is also added to the 10 and 15 feature vectors (*TotalDLLCalls*, Fig. 1). Note that this is a *computed* feature and not an extracted parameter.

IV. PERFORMANCE EVALUATION

The static analysis framework is now evaluated using the data repository from Section III-A. Namely, feature vectors extracted from the PE files are labelled to generate input datasets. These datasets are then used

to train/test supervised ML classifiers, i.e., SVM, RF, XGBoost, and FNN [6] (Fig. 1). All evaluation is done using the Keras and TensorFlow toolkits, as well as Pandas and Sklearn. As per Section III-A, a total of 9 malicious ransomware families are evaluated along with a set of benign applications, i.e., 10 classes. As noted earlier, there are a total of 1,240 malicious samples (140 samples for each family except BlackCat which has 120 samples). The samples for each class are further partitioned to generate separate training and testing pools. Namely, 20 random samples of each class are selected for testing and the remainder are used for training, i.e., 120 training samples for all classes except BlackCat which only has 100 samples. Furthermore, 1,700 benign samples are selected for training and the remaining 300 samples are used for testing. This partitioning reflects an approximate 85/15 training/testing split. All results are averaged over 100 trial runs, with each using a different randomized 85/15 partitioning of the datasets. Detailed findings are now presented.

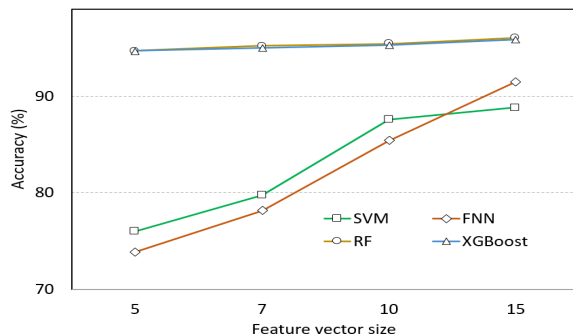


Fig. 2. Average multi-class accuracy (100 trials)

The average accuracy values (over all runs) are plotted for different feature vector sizes in Fig. 2, i.e., multi-class attribution. Results show improved performance for all schemes with increasing feature vector sizes. In particular, the SVM and FNN classifiers give the best improvement, with accuracy gains of 15-20%. Conversely, the RF and XGBoost classifiers have much lower gains as feature vector sizes increase from 5 to 15 parameters, i.e., 0.5-1.5% range. These two classifiers also give the best accuracy (94-96% range). However, the FNN scheme approaches these methods with 15 features, i.e., 91% accuracy. These findings are very encouraging given the relatively small-sized training datasets and feature vectors used. The results also match those for other schemes using much heavier feature extraction and ML algorithms, e.g., image and entropy-based features, deep NN algorithms, etc [10], [11].

Next, consider attribution errors in more detail. Indeed, mis-classifying ransomware as benign is much more harmful than mis-classifying it as the wrong type

of ransomware, i.e., since such errors can allow malware to bypass network or host defenses and infect host machines. Hence to quantify this behavior, a modified *ransomware detection rate* (RDR) is defined as:

$$RDR = \frac{T_{rs}}{T_{rs} + F_{rs}} \quad (1)$$

where T_{rs} is the total number of ransomware samples classified as (any class of) ransomware, and F_{rs} is the total number of ransomware samples mis-classified as benign, i.e., total number of ransomware test samples is $(T_{rs} + F_{rs})$. This metric essentially captures the *binary* detection capability of a multi-class classifier and is similar to the recall formula, i.e., tracks false negatives. A *benign detection rate* (BDR) is also defined as:

$$BDR = \frac{T_{bn}}{T_{bn} + F_{bn}} \quad (2)$$

where T_{bn} is the total number of benign samples classified as benign, and F_{bn} is the total number of benign samples mis-classified as ransomware. In general, though, false negative attribution of benign executables is less of a security concern.

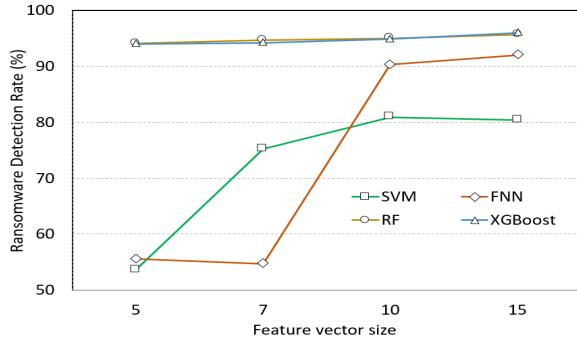


Fig. 3. Average ransomware detection rate (100 trials)

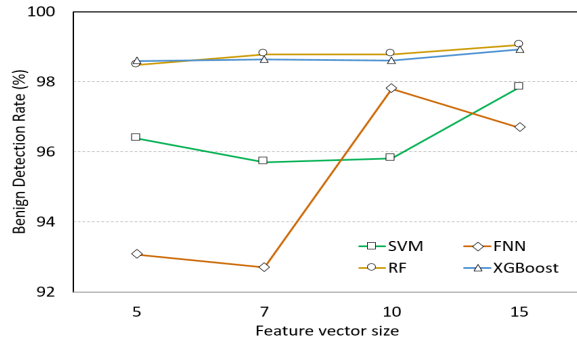


Fig. 4. Average benign detection rate (100 trials)

Accordingly, Fig. 3 plots the binary RDR results averaged over 100 trials. Akin to the multi-class case, the RF and XGBoost schemes give the highest ransomware detection rates, close to 95%. Again, larger

feature vector sizes also give smaller improvements with these classifiers, i.e., 2% range. By contrast, the SVM and FNN schemes give very poor results for small feature vectors, with ransomware mis-classification rates around 50% ($1-RDR$). These classifiers are also very sensitive to feature vector size. Nevertheless, the FNN scheme still approaches the performance of the RF and XGBoost schemes with larger feature vectors, i.e., 92% RDR. The BDR results are also plotted in Fig. 4. As expected, these values are higher than the RDR values since a larger amount of benign data is used for training. Again, the RF and XGBoost schemes give the lowest benign program mis-classification rates, close to 99%. Although the other methods (SVM, RF) give slightly lower BDR rates, they are still over 92% (less than 1 error in 12). Note that these binary detection rates closely match those from other malware detection studies which make use of much larger datasets and more elaborate feature extraction schemes (Section II).

Meanwhile, Fig. 5 shows an average confusion matrix for the XGBoost classifier (classes 0-8 represent the 9 ransomware families and class 9 represents the benign class). Here, the numbers in row 9 are larger as there are more benign test samples. These results confirm that most samples are classified correctly, i.e., diagonal numbers dominate. Moreover, even when ransomware samples are mis-classified, they are mostly flagged as another ransomware (mirroring RDR results in Fig. 3).

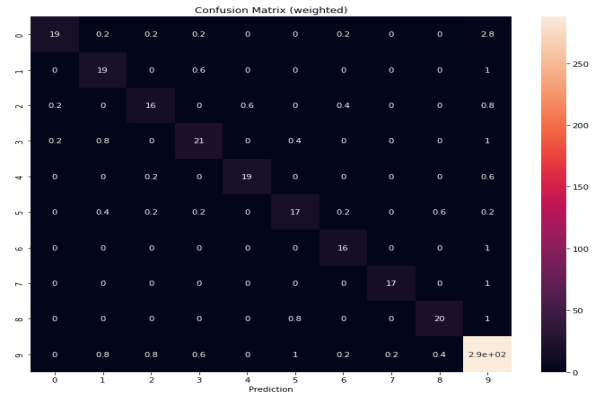


Fig. 5. Confusion matrix (XGBoost, 15 features)

Tests are also done to gauge zero-day attack detection. Namely, 8 out of the 9 ransomware families are aggregated into a single malicious class and used to train a binary classifier (versus benign class). The remaining family is then tested as a zero-day threat, i.e., to see if the binary classifier can flag it as ransomware. Hence all samples are either used for training or testing. The associated detection rates are shown in Table II for all possible zero-day attack scenarios. These results show very good performances for several

classifiers. For example, the RF (XGBoost) scheme gives approximately 80-99% detection rates for 8 (6) out of the 9 ransomwares tested, i.e., at least 4 out of 5 zero-day attacks detected. However, the WannaCry malware is very effective at evading all schemes and has low detection rates in 14-20% range, i.e., only 1 in 7 detected. Hence additional PE file features (parameters) or other static parameters may need to be incorporated.

TABLE II
ZERO-DAY ATTACKS (DETECTION ACCURACY)

Zero-Day	SVM	RF	XGB	FNN
Babuk/Babyk	81.43%	94.86%	93.57%	74.38%
BlackCat	36.67%	78.17%	35.83%	65.70%
Chaos	93.57%	87.00%	87.14%	83.95%
DJVu/STOP	82.86%	98.64%	90.71%	79.40%
Hive	15.71%	82.71%	82.88%	72.30%
LockBit	57.14%	84.79%	62.86%	63.06%
NetWalker	95.00%	97.36%	97.14%	98.21%
Sodinokibi/REvil	95.00%	91.00%	90.05%	89.86%
WannaCry	13.57%	14.86%	14.29%	19.81%

Finally, run-times are measured by averaging PE format file generation, feature extraction, and ML attribution times. Tests are done on a Windows 11 server with a 3.60 GHz Intel Core i9 processor and 64 Gb of *random access memory* (RAM). Only *trained* ML models are timed to reflect operational settings, and PE file generation is done using a Github package (<https://github.com/erocarrera/pefile>). Results show that PE file generation times are directly correlated to executable file sizes (Table I). For example, benign files average 1.46 sec, whereas ransomware files range from 50-300 ms (larger for LockBit, DJVu/STOP, and Hive). Meanwhile, classification times vary between 2-8 ms for the SVM, RF, and XGboost classifiers, but are higher for the FNN scheme at 47 ms. Overall, many operators are willing to accept these delays for scanning incoming downloads/attachments in network/host-based defenses.

V. CONCLUSIONS

It is imperative to track the latest ransomware releases and develop effective solutions for mitigating these threats. This paper presents a static analysis scheme for ransomware detection and attribution. First, a new dataset is curated with the latest Windows 10/11 ransomware families. Windows *portable executable* (PE) format files are then used to extract feature vectors and train *machine learning* (ML) classifiers. Overall findings show very good performance in terms of ransomware detection, attribution, and zero-day threat detection. These results are achieved using minimalist datasets with about 100-120 training samples per class and relatively compact feature vectors. The solution also gives very amenable run-times for realistic settings. This work presents a strong basis from which to expand into

future work. Specifically, more ransomware families can be added to the repository and refined feature extraction and ML methods can also be studied.

REFERENCES

- [1] R. Moussaileb, N. Cuppens, J.-L. Lanet, and Boudier, "A survey on windows-based ransomware taxonomy and detection mechanisms: Case closed?" *ACM Computing Surveys*, vol. 54, no. 6, July 2022.
- [2] A. Vehabovic, N. Ghani, E. Bou-Harb, J. Crichigno, and A. Yayingimli, "Ransomware detection and classification strategies," in *IEEE Black Sea Comm 2022*, Sofia, Bulgaria, June 2022.
- [3] E. Berrueta, D. Morato, E. Magaña, and M. Izal, "A survey on detection techniques for cryptographic ransomware," *IEEE Access*, vol. 7, pp. 144 925–144 944, October 2019.
- [4] A. Kapoor, "Ransomware detection, avoidance, and mitigation scheme: A review and future directions," *Sustainability*, vol. 14, no. 1, December 2021.
- [5] A. Ng, "Ai minimalist," *IEEE Spectrum*, vol. 59, no. 4, pp. 23–25, April 2022.
- [6] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Second Edition*. O'Reilly Media, 2019.
- [7] S. Poudyal, K. P. Subedi, and D. Dasgupta, "A framework for analyzing ransomware using machine learning," *IEEE 2018 SSCI*, Nov. 2018.
- [8] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. Sangaiah, "Classification of ransomware families with machine learning based onn-gram of opcodes," *Future Generation Computer Systems*, vol. 90, pp. 211–221, 2019.
- [9] D. Mulders, "Network based ransomware detection on the samba protocol," *MS Thesis, Department of Mathematics, TU Eindhoven*, 2017.
- [10] B. Wang, H. Liu, X. Han, and D. Xuan, "Image-based ransomware classification with classifier combination," in *ACM Advanced Information Science and System (ACM AISS) 2021*, Sanya, China, November 2021.
- [11] J. Zhu, J. Jaccard, A. Singh, I. Welch, and H. A.-S. amd S. Camtepe, "A few-shot meta-learning based siamese neural network using entropy features for ransomware classification," *Computers & Security*, vol. 117, pp. 1–11, June 2022.
- [12] D. Kim, S. Woo, D. Lee, and T. Chung, "Static detection of malware and benign executable using machine learning," in *Internet 2016*, Barcelona, Spain, November 2016.
- [13] Y. Liao, "Pe-header-based malware study and detection," in *Semantic Scholar*, 2021.
- [14] T. Rezaei and A. Hamze, "An efficient approach for malware detection using pe header specifications," in *6th International Conference on Web Research (ICWR)*, Tehran, Iran, April 2020.
- [15] A. Almarshadani, M. Kaiiali, S. Sezer, and P. O'Kane, "A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware," *IEEE Access*, vol. 7, no. 1, pp. 47 053–47 067, 2019.
- [16] D. Morato, E. Berrueta, E. Magaña, and M. Izal, "Ransomware early detection by the analysis of file sharing traffic," *Journal of Network and Computer App.*, vol. 124, no. 1, pp. 14–32, 2018.
- [17] O. Alhawi, J. Baldwin, and A. Dehghantanha, "Leveraging machine learning techniques for windows ransomware network traffic detection," *Adv. in Info. Security*, p. 93–106, July 2018.
- [18] K. C. Roy and Q. Chen, "Deeptran: Attention-based bilstm and crf for ransomware early detection and classification. information systems frontiers," *Information Systems Frontiers*, vol. 0, pp. 1–17, 2021.
- [19] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "Unveil: A large-scale, automated approach to detecting ransomware," in *USENIX Security 2016*, Austin, TX, August 2016.
- [20] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "Pay-break: Defense against cryptographic ransomware," in *ACM Asia CCS 2017*, Abu Dhabi, UAE, April 2017.

- [21] A. Kharraz and E. Kirda, "Redemption: Real-time protection against ransomware at end-hosts," in *RAID 2017*, Atlanta, GA, October 2017.
- [22] A. AlSabeh, H. Safa, E. Bou-Harb, and J. Crichigno, "Exploiting ransomware paranoia for execution prevention," in *IEEE ICC 2020*, Dublin, Ireland, June 2020.