

Federated Learning Approach for Distributed Ransomware Analysis

Aldin Vehabovic¹, Hadi Zanddizari¹, Farook Shaikh¹, Nasir Ghani¹, Morteza Safaei Pour², Elias Bou-Harb³, and Jorge Crichigno⁴

¹ University of South Florida, Tampa, FL 33620, USA
{vehabovica,hadiz,nghani}@usf.edu

² San Diego State University, San Diego, CA 92182, USA
msafaeipour@sdsu.edu

³ University of Texas San Antonio, San Antonio, TX 78249, USA
elias.bouharb@utsa.edu

⁴ University of South Carolina, Columbia, SC 29208, USA
jcrichigno@cec.sc.edu

Abstract. Ransomware is a form of malware that uses encryption methods to prevent legitimate users from accessing their data files. To date, many ransomware families have been released, causing immense damage and financial losses for private users, corporations, and governments. As a result, researchers have proposed a range of ransomware detection schemes using various *machine learning* (ML) methods to analyze binary files and action sequences. However as this threat continues to proliferate, it is becoming increasingly difficult to collect and analyze massive amounts of ransomware executables and trace data at a common site (due to data privacy and scalability concerns). Hence this paper presents a novel *distributed ransomware analysis* (DRA) solution for detection and attribution using the decentralized *federated learning* (FL) framework. Detailed performance evaluation is then conducted for the case of static analysis with rapid/lightweight feature extraction using an up-to-date ransomware repository. Overall results confirm the effectiveness the FL-based solution.

Keywords: Ransomware · malware detection · federated learning

1 Introduction

Ransomware has evolved rapidly over the last decade and is now one of the most serious cyberthreats facing users and organizations today. This malware executes a multi-stage kill-chain to find and infect victim hosts, encrypt their data files, and extract ransom payments. Expectedly, ransomware represents one of the most lucrative revenue stream for cybercriminals today, many of who now offer *ransomware-as-a-service* (RaaS) [1] as well. Furthermore, many attackers are also targeting large organizations due to the potential of sizeable payouts. Overall, many different ransomware “families” have been developed and

weaponized, with most focusing on Windows users as this is still the most widely-used *operating systems* (OS) type in the enterprise today.

Given these challenges, researchers have been actively studying ransomware analysis schemes in recent years. The main objective in many of these efforts is to identify ransomware programs based on their static or dynamic characteristics and prevent harmful activities. For example, *static analysis* schemes [2] analyze the artifacts of malicious binary files. Hence these methods can be integrated into network-based defenses to analyze incoming files or attachments and detect ransomware earlier in the transmission stage of the kill-chain [1]. Meanwhile, *dynamic analysis* methods [2] track host system and/or network communication activities to detect ransomware after infection. These schemes can also be integrated into host/network-based defenses, but focus on later stages in the kill-chain (post-infection). Overall, many static and dynamic analysis designs also use *machine learning* (ML) algorithms to process large amounts of ransomware data and extract generalized behaviors, see surveys in [1],[2],[3].

Nevertheless, as ransomware threats continue to proliferate and diversify, users are being impacted across many domains and sectors. Therefore it is becoming increasingly difficult to collect and analyze massive amounts of ransomware executables and user/system/network trace data at a common (centralized) site. Clearly, this approach poses major privacy and scalability concerns. Foremost, off-site transmission and sharing of sensitive end-host data and network logs is problematic for many users and organizations [2]. Data pre-processing and ML training at a single location also imposes high computational burdens and bandwidth transfer overheads depending upon the type of data being shared (static, dynamic). Inevitably, these limitations complicate the real-world application of many ransomware solutions that use centralized ML training. In addition, most studies have used datasets containing a mixture of older ransomware families targeting Windows 7/8 systems (from mid-2010s time frame) [2].

In light of the above, there is a pressing need to develop new ransomware analysis schemes to detect *and* attribute the latest threats with improved scalability and privacy. Of key concern are threats targeting Windows 10/11, the most prevalent OS in enterprise environments. Hence this paper presents a novel ransomware solution which leverages *federated learning* (FL), a decentralized and collaborative ML framework designed to address scalability and privacy concerns [4],[5]. Indeed, FL has already been applied to range of problems in image/voice recognition, keystroke prediction, smart grids, fraud detection, etc [5],[6]. Recent studies have also proposed some FL-based cybersecurity schemes, e.g., for *Internet of Things* (IoT) intrusion detection, network anomaly detection, and traffic recognition [7],[8],[9],[10]. However the further application of this distributed ML paradigm for ransomware analysis (particularly detection and attribution) has not yet been considered. Hence this topic forms the key motivation for the work herein, and several key contributions are made:

1. Novel *distributed ransomware analysis* (DRA) architecture for ransomware detection and attribution based upon the FL framework. This solution em-

bodies a generic approach which can implement a wide range of static and dynamic ransomware analysis schemes.

2. Use of a new ransomware dataset repository with some of the latest families targeting Windows 10/11, i.e., including Babuk/Babyk, BlackCat, Chaos, DJVu/STOP, Hive, LockBit, Netwalker, Sodinokibi/REvil, and WannaCry. Since new malware can have much fewer available samples, this repository is limited to under 1,500 binary executables to model realistic scenarios.
3. Detailed evaluation of the DRA framework for the case of static ransomware analysis. Specifically, rapid/lightweight feature extraction is done using Windows *portable executable* (PE) files, and FL performance is then tested using decentralized *neural-network* (NN) based classifiers.

To the best of the authors' knowledge, this is the first known study on FL for ransomware detection and attribution. This paper is organized as follows. Section 2 presents a review of ransomware detection schemes as well as recent FL-based cybersecurity schemes. Subsequently, Section 3 details the new DRA architecture and its distributed FL algorithm. The ransomware dataset repository is then detailed in Section 4, including sample collection and feature selection/extraction. Detailed performance results are then presented in Section 5 for the case of static analysis, followed by conclusions and future work directions in Section 6.

2 Literature Review

Ransomware follows a well-established kill-chain sequence comprised of several key stages, i.e., reconnaissance, distribution/delivery, installation/infection, communication, encryption, and extortion/payment [2]. As a result, researchers have proposed a range of schemes for ransomware detection and mitigation targeting different stages in this sequence. A brief taxonomy and overview of some of these methods is presented along with recent FL-based security solutions.

2.1 Ransomware Detection

Ransomware analysis has received much attention in the past decade. Various survey articles have also appeared in this domain, detailing different (sometimes overlapping) taxonomies to classify the proposed solutions, e.g., including static or dynamic analysis schemes, network-based or host-based methods, forensic analysis techniques, etc [1]-[3]. Consider some details.

Static analysis methods analyze binary malware executables to detect artifacts of malicious behaviors [1]. Common techniques here include code analysis for malware author attribution, code/segment de-anonymization, reverse engineering, ransomware server address and domain prediction, etc [2]. For example, [11] specifies a multi-level framework to classify ransomware. This scheme analyzes raw binary files, assembly code, and libraries using Linux object-code dumps and portable executable parsers. ML classifiers are then trained using the extracted data, yielding detection rates around 90%. Meanwhile, [12] presents a

scheme to analyze operational code sequences. These sequences are transformed into N-grams, and *term frequency-inverse document frequencies* (TF-IDF) features are generated to train classifiers (decision tree, RF, etc). Results show detection rates around 90%. However, code-based analysis is a slow and labor-intensive approach [13] which is better suited for post-infection forensic analysis.

Recent efforts have also generated other features for static analysis. For example, [14] presents a unique ransomware classification scheme which uses image processing for feature extraction. Namely, ransomware binary files are first converted to grayscale images, and texture analysis is then used to compute features. Results for several classifiers show good accuracy (97%) for a small dataset containing a mix of older and newer ransoms (379 samples). However, this scheme imposes higher computational burdens for large datasets and does not consider benign applications. Meanwhile, [15] details another static ransomware analysis scheme which computes/extracts entropy and image-based features from binary files and uses them to train a Siamese NN classifier. Tests for a small dataset with about 1,000 samples and 10 ransomware 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 (from mid-2010s time frame) and benign applications are not considered.

Researchers have also used Windows PE format files for static malware analysis. These files contain metadata and other information for binary executables and enable very fast/lightweight feature extraction, i.e., versus more compute-intensive image or entropy-based methods above. However, most studies using PE files have focused on the generic malware detection problem (and not ransomware classification). For example, a 2016 study [16] uses malware samples from **VX Heaven** (now inactive) to train several ML-based detection classifiers using about 10 PE file features. Results show detection rates over 90%. Meanwhile, [17] extracts PE file features from a dataset with 5,500 malware and 1,200 benign samples (from the early 2010s) and uses customized rules to achieve 95% detection rates. Also, [18] extracts a small set of PE file features from a dataset with 1,200 malicious and benign samples. Several classifiers are then trained, yielding over 95% detection accuracy. However, these studies use malware datasets which are almost a decade old and provide few details on composition. To address this concern, [19] presents an updated study on ransomware detection *and* attribution using static PE file analysis. A new repository is curated comprising of 9 of the latest ransomware families (about 1,200 samples) and benign applications (2,000 samples). Results for several ML classifiers show impressive detection and attribution percentages in the mid-90% range with up to 15 features.

Meanwhile dynamic analysis examines run-time actions and events at the network and/or host levels. Namely, dynamic network-based schemes analyze packet traces for ransomware activity, e.g., server communications, *domain name service* (DNS) queries, networked storage access, etc. For example, [20] presents a detection scheme for the Locky ransomware (2016) which collects behavioral/non-behavioral traffic features in a testbed and then trains several classifiers. Results show a mean detection rate of 97%. Meanwhile, [21] presents a ransomware de-

tection scheme using advanced *neural network* (NN) classifiers which yields over 97% detection rates for older ransomware threats such as CryptoWall, TorrentLocker, and Sage. The NetConverse scheme [22] also uses ML methods to analyze Windows host traffic for several families from the 2010s time frame. Results show high detection rates, over 95%. Finally, [23] presents a deep learning approach to detect and classify abnormal traffic from Windows 7 hosts, and results show high detection rates for several families.

Dynamic host-based schemes monitor local system activities to detect ransomware operations and possibly recover encryption keys. These methods are more latent as they use virtual run-time environments to execute binary files and capture traces, i.e., *virtual machines* (VM) or sandboxes. A range of actions can be tracked here, including *application programmer interface* (API) calls, *dynamic link library* (DLL) calls, and memory and file operations. For example, [24] monitors for file encryption/deletion, persistent desktop messages, etc. Results show successful detection of about 96% of older ransomware. Meanwhile, [25] presents a scheme to monitor/store encryption keys and facilitate ransomware detection and recovery. This solution can mitigate about 12 out of the 20 families tested. Recent efforts have also targeted ransomware “paranoia” activities, i.e., pre-attack actions to detect environments and avoid fingerprinting. For example, [26] monitors pre-attack API calls and uses *natural language processing* (NLP) to extract features. Findings confirm that many ransomware families generate distinguishable API fingerprints. Also, [27] presents a host-based framework for API call monitoring. Some older ransomware families (such as Reveton, Locky, Teslacrypt, etc) are fingerprinted to extract features and build frequency pattern trees for real-time detection of API sequences.

Although the above works provide some key contributions, notable concerns still remain. Foremost, the majority of studies have focused on older ransomware families targeting dated Windows 7/8 systems (mid-2010s). Given the persistent nature of this malware, it is imperative to focus on newer threats to Windows 10/11 OS users. Furthermore, new releases will likely have fewer samples to analyze, posing added challenges. Hence “data-centric” ML solutions must achieve effective performance with constrained datasets. Finally, ransomware detection/attribution schemes must have amenable run-time complexity and ideally tackle threats in the earlier distribution/delivery stages to minimize damage [2]. It is here that static analysis is more expedient for examining suspicious executable payloads/downloads. By contrast, dynamic analysis requires more latent tracking and analysis of network or host activities and is better suited for later stages in the kill-chain. Finally, it is important to consider attribution, i.e., ransomware classification, as this represents a logical next step after detection.

2.2 Federated Learning (FL) in Cybersecurity

FL is a decentralized learning framework developed by Google in 2016. This solution uses multiple end-point clients to train ML models under the coordination of a central server [4]. Namely, “global” learning is done over multiple (synchronous or asynchronous) communication rounds. In each round, the server

selects a subset of clients and sends them its latest global ML model parameters. Clients then perform “local” training with their own data and only send back model parameter updates. The central server uses these updates to revise its global ML model parameters and then repeats the process.

Overall, FL provides some key benefits for large ML problems. Foremost, user privacy is greatly improved as sensitive data stays at local hosts. Computational scalability is also much better since many systems are involved in the distributed learning process. Finally, bandwidth scalability (efficiency) also increases since raw datasets do not need to be transferred to a central server. As a result, FL has seen strong traction in a diverse range of areas, e.g., image/voice recognition, keystroke prediction, smart grid operation, fraud detection etc [5],[6]. Many further variants of this approach have also been proposed, as surveyed in [28]. Although a detailed review of these works is out of scope herein, some key contributions include revised training and averaging methods for unbalanced datasets, compression/quantization methods to reduce FL communication overheads, client failure recovery techniques, detection and mitigation of malicious clients (adversarial FL), and fully distributed “peer-to-peer” FL schemes.

Now researchers have also applied FL to the security domain. For example, [7] presents a FL framework to train *deep NN* (DNN) models for multiple tasks such as anomaly detection, traffic recognition, and classification. Tests with older traffic profiles (2016) show good detection accuracies for various anomalous events, on par with centralized schemes. Meanwhile, [8] details a FL-based DeepFeed solution for intrusion detection in cyberphysical systems using *convolutional NN* (CNN) models. Results with real-world industrial network data show good detection accuracy, exceeding some centralized schemes. Meanwhile, [9] presents a FL scheme for anomaly detection of compromised IoT devices. Here, local access gateways monitor traffic and map packet streams to symbols. Language processing methods are then used in conjunction with *recurrent NN* (RNN) algorithms. Results for the Mirai malware show very good attack detection (over 95%) and low run-time complexity. Finally, [10] presents another FL approach for IoT malware detection using supervised and unsupervised ML algorithms for anomaly detection (with packet data). Again, FL gives very good performance, closely matching centralized classifiers. The authors also test several adversarial FL setups with model poisoning attacks. Results show a notable decline in accuracy when over 25% of clients are compromised. However, as noted earlier, the further application of FL for ransomware analysis has not been considered.

3 Distributed Ransomware Analysis (DRA)

The continued growth of ransomware is posing many operational challenges for threat detection and mitigation. Most notably, it is increasingly difficult (infeasible) to collect large amounts of raw data from massive user bases and transfer it to a common site for analysis. Indeed, privacy concerns will prevent many users and organizations from sharing their sensitive file/trace logs in the first place. Additionally, bandwidth transfer overheads can be very high, and a single cen-

tralized computing facility may not be able to process extreme amounts of data and train large models. Inevitably, these constraints will limit the effectiveness of ML-based solutions. Hence there is a pressing need to develop new ransomware analysis frameworks which address these critical concerns.

Now scalability and privacy demands in large ML problems are not new, and researchers have proposed various solutions such as privacy preserving computation and *federated learning* (FL) [4] (Section 2.2). Of these, the latter is the most scalable as it distributes and decentralizes training over multiple computational nodes. Hence a novel *distributed ransomware analysis* (DRA) solution is developed using the FL framework. This setup is shown in Fig. 1 and comprises of several *client sites* communicating with a *central server*. This architecture features a generic design that can be tailored for a full range of ransomware analysis schemes, both static and dynamic (Section 2.1). Furthermore, it is assumed that the whole setup operates in a trusted manner, i.e., careful vetting/pre-selection of client sites, authenticated and encrypted communications, encryption of training data, etc. The DRA architecture and its associated FL algorithm are now presented, followed by a detailed performance evaluation study in Section 5.

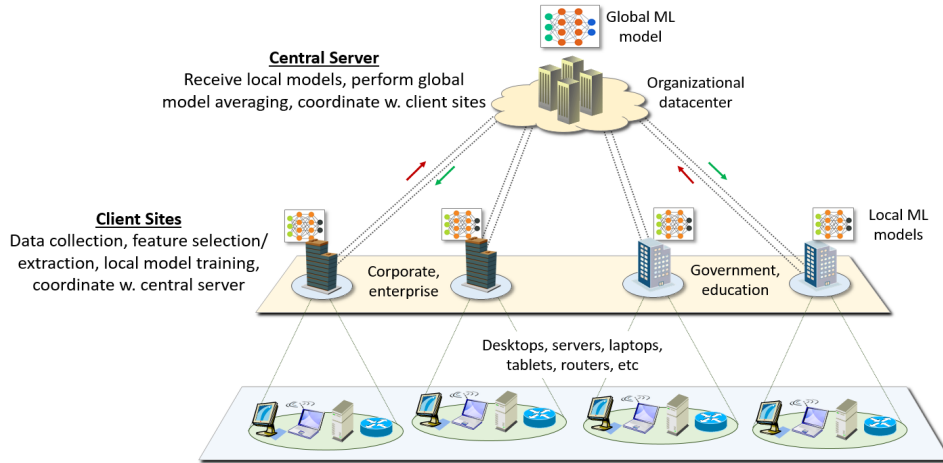


Fig. 1. Distributed ransomware analysis (DRA) framework using federated learning

3.1 Client Sites

Client sites in the DRA architecture mirror the roles of client nodes in the FL architecture [4]. These entities are located at carefully-selected, trusted organizations with large user bases, e.g., such as corporate campuses, government facilities, academic institutions, and even network service provider *points of presence*

(PoP). Note that it is not feasible for individual systems (such as servers, laptops, etc) to act as FL clients. The main reason here is that individual devices will not be able to collect a sufficient number of ransomware samples for effective ML training (data scarcity). Hence DRA client roles are placed at vetted institutional sites which have access to much larger volumes of raw data from many users and systems. As there are generally fewer client sites involved here, i.e., tens-hundreds, this approach embodies a cross-silo FL approach [10].

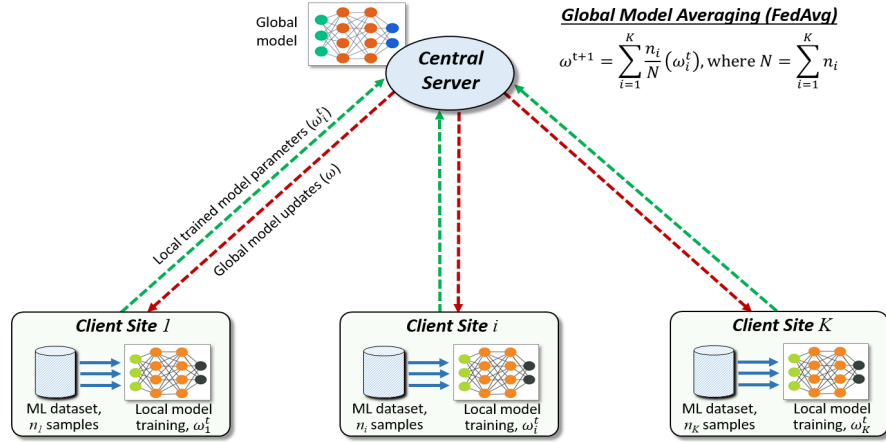


Fig. 2. Logical view of federated learning (FL) algorithm in DRA architecture

Overall, DRA client sites perform several key functions, including raw data collection, feature selection/extraction, local ML model training, and communication with the central server. Foremost, raw data samples are collected from the internal user base and other systems, e.g., desktops, laptops, tablets, mail and web servers, routers, and even cyberphysical devices. Depending upon the ransomware analysis approach being implemented (Section 2.1), data acquisition can entail a wide range of possibilities, e.g., such as extracting email files from mail servers, capturing webpage attachments from servers, logging router packet traces, extracting host/system call logs, etc. Client sites then perform further data pre-processing and feature selection/extraction on the raw data samples. Namely, feature selection identifies a subset of parameters of interest, whereas feature extraction selects parameters to generate ML insights. The extracted features are then collected to build a local ML training/testing dataset, Fig. 1.

Now client sites must also communicate with the central server to participate in the distributed FL training process. A logical overview of this strategy is shown in Fig. 2, along with sample client site pseudocode in Fig. 3. Namely, the training process is jumpstarted when a client site receives parameters for the initial global ML model from the central server, i.e., denoted by ω . Since NN-based algorithms are most amenable to FL implementation [5], these pa-

```

Input: Receive global ML model parameters from central server,  $\omega$ 
Partition local ML dataset ( $n_i$  samples) into  $B$  batches
/* Train local ML model over given epochs and batches */
for j=1 to  $E$  do
    for k=1 to  $B$  do
        Train local model over  $k$ -th batch of data
        Update local ML model parameters  $\omega \rightarrow \omega_i^t$ 
Output: Send updated ML model parameters,  $\omega_i^t$ , to central server
    
```

Fig. 3. Centralized FL local training algorithm at client site i

rameters will typically correspond to weight vectors for *feedforward NN* (FNN), *deep NN* (DNN), *convolutional NN* (CNN), or *long short-term memory* (LSTM) networks [29]. Client sites then train the received model using their local data. In particular, it is assumed that client site i has a local ML dataset with n_i samples, and this is further split into B batches for training over E epochs, see Fig. 3. Finally, the updated local model parameters at each round t , ω_i^t , are sent back to the central server for further processing and updating. Overall, localized model training allows client sites to retain their sensitive data.

3.2 Central Server

The central server in the DRA architecture manages the distributed, decentralized ML training process for ransomware analysis. Akin to its namesake in the FL setup [4], this entity communicates with the client sites to update the global ML model parameters. Sample pseudocode for the central server algorithm is also presented in Figure 4 based upon the **FedAvg** algorithm [4]. Namely, training is done over T communication rounds with the client sites. In each round, the central server selects a subset of K client sites to participate in the training and sends them its latest global model parameters, ω . It then waits for the local client training sessions to complete and receives/processes parameter updates, i.e., where ω_i^t denotes the updated ML model parameters from client site i in round t (Figs. 3, 4). Note that Fig. 4 shows a synchronous FL training approach where all clients respond in each iteration. However, this detail is not specific to the DRA architecture and can be easily modified for asynchronous operation, e.g., by specifying client site selection and update processing [4],[5].

Finally, client site updates are appropriately averaged to revise the global ML model parameters, ω . Although a wide range of options are possible here, without loss of generality, the **FedAvg** [4],[5] approach is used in Figs. 2 and 4. Namely, model averaging is done in a weighted manner based upon the size of each client site's dataset, i.e., proportional to $\frac{n_i}{N}$, where $N = \sum_i n_i$ is the aggregate amount of training data across all K client sites chosen. Overall, this

```

Input: Initial global ML model parameters,  $\omega_0$ 
/* Iterative FL training process over  $T$  rounds */
for  $t = 1$  to  $T$  do
    Select  $K$  client sites
    for  $i = 1$  to  $K$  do
        Send latest global ML model  $\omega$  to  $i$ -th client site
    Wait to receive local updates from all  $K$  client sites,  $\omega_i^t$ 
    Average to update global ML model parameters:  $\omega \leftarrow \omega^{t+1} = \sum_{i=1}^K \frac{n_i}{N} \omega_i^t$ 
Output: Final global ML model parameters,  $\omega$ 

```

Fig. 4. Centralized FL averaging algorithm at central server

method is well-suited for *independent identically distributed* (IID) datasets. However, researchers have also proposed modified FL server averaging schemes for heterogeneous (non-IID) local data [5], and these techniques can be further integrated into the central server algorithm (left for future study).

4 Ransomware Dataset Repository

Realistic datasets are critical for effective ML solutions. However, as noted in Section 2, most ransomware detection studies have at least in part, used older datasets with Windows 7 malwares. Indeed, many old ransomware control servers are no longer active as malactors have shifted to other families. In light of this, the new ransomware dataset repository curated in [19] is used. This dataset contains binary executables of some of the most prevalent ransomware threats today, as per the IBM X-Force Threat Intelligence Index, i.e., Babuk/Babyk, BlackCat, Chaos, DJVu/STOP, Hive, LockBit, Netwalker, Sodinokibi/REvil, and WannaCry (Table 1). Namely, 9 families are chosen, and a number of Windows application executables are also collected to build a benign class and improve classifier performance. As per Fig. 5, repository design involves two key steps, empirical data collection and feature extraction/selection, detailed next.

4.1 Empirical Data Collection

The work in [19] collects a realistic set of raw binary files, with the goal of mirroring data collection at client sites (Section 3.1). Now the size and diversity of input data will impact ML classifier performance. For example, most algorithms yield better generalization (class separation) with larger datasets and equal representation across classes. However, given the rapidly-changing nature of ransomware, it can be difficult to obtain a sufficient number of samples for each family under consideration. Hence effective solutions must achieve good detection and attribution performance with more constrained “minimalist” datasets,

i.e., only hundreds of samples per ransomware family. Note that other recent studies have also used smaller datasets with under 2,000 samples [14],[15]. This requirement is well-aligned with broader trends in *artificial intelligence* (AI) to develop more “data-centric” solutions for specialized problems [30]. Consider some further details.

Malware samples are extracted from various online sites to capture some of the latest ransomware threats. Now many portals allow users to upload/download malware executables, e.g., **MalwareBazar**, **Triage**, **VirusShare**, and **VirusTotal**, etc. However, these repositories provide varying access and usability. For example, **VirusTotal** and **VirusShare** require user registration to access private repositories. Detailed cross-checking and analysis of samples also reveals notable duplication across portals. For example, many Sodinokibi samples on **MalwareBazar** match those on **Triage**. There is also notable discrepancy between the number of samples for each family. For example DJVu is relatively abundant whereas Babuk/Babyk is more scarce and harder to find. Finally, repositories (such as **VirusShare**) do not label their raw data, further complicating collection. Hence samples from these large unlabeled data dumps have to be analyzed individually using hashing programs and then cross-checked with other labelled samples (a tedious and time-consuming process). While other repositories like **VirusTotal** may have labelled samples, they do not allow researchers to download them freely. Given these realities, there is potential for a lack of diversity (even scarcity) of unique samples for new ransomware families.

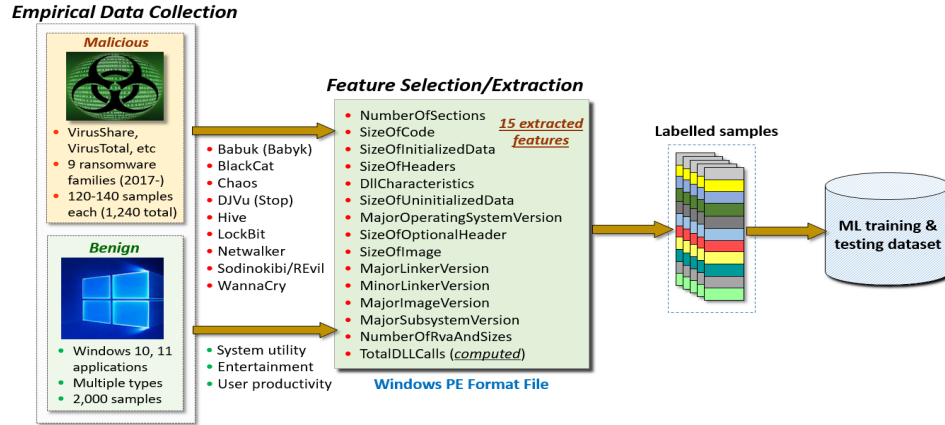


Fig. 5. Ransomware repository design and static PE file features (as per [19])

In light of the above, a “minimalist” raw dataset is collected for the 9 ransomware families [19]. Specifically, only 140 executable samples are gathered for each family, yielding a total of 1,260 malicious samples (under 1,500 samples). However, unlike some recent studies in ransomware classification [14],[15], a large

Table 1. Empirical ransomware dataset (as per [19])

Family	Samples	Avg. Size	Avg. PE File
Babuk (Babyk)	140	0.19 MB	32.68 KB
BlackCat	140	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

number of Windows 10/11 applications are also downloaded to build a benign class (2,000 in total). These programs are collected from a range of websites and include system utility, entertainment, and productivity tools (Fig. 5). Overall, the addition of a benign class is crucial for ML purposes as it can help establish a more clear delineation between malicious programs and reduce classification errors. Further details on the collected samples are also presented in Table 1.

4.2 Feature Selection/Extraction

ML performance is also very dependent upon the type of input training data. It is here that feature extraction (engineering) plays a vital role in transforming raw sample files (executables) and generating meaningful information for classifiers [29]. Now as noted in Section 2.1, static analysis is more effective for targeting the early stages of the kill-chain. Hence this strategy is chosen here. In particular a small set of static parameters are extracted from Windows PE format files to generate a very lightweight set of features (i.e., hundreds of bytes per sample, Fig. 5). This selection contrasts with other ransomware schemes which use much more compute-intensive methods to extract larger/heavier feature sets, e.g., using code extraction [12], image processing [14], and entropy [15] based methods.

Overall, PE format files are used to support executables running in 32-bit and 64-bit Windows OS settings [31] (and similar constructs also exist for other OS types). This file uses the *common object file format* (COFF) and contains key information for the OS loader to run wrapped executable code. Namely, a PE file specifies memory mapping and permissions and is comprised of initial lead-in headers and multiple sections. Each section contains actual file contents (such as code or data) and has its own header [31]. Overall, PE files contain a wealth of static information, and different executables can have unique non-overlapping parameters as per functionality. As a result, it is important to extract a proper subset of parameters here, i.e., feature vectors. Foremost, a chosen feature should exist in the PE format files for all executables in the repository. Additionally, selected parameter ranges should exhibit sufficient variability across classes.

In light of the above, PE files are generated for all raw executable files. Careful experimentation is then done to select a total of 15 PE file parameters and construct labelled feature vectors to build the aggregate ML training/testing dataset, Fig. 5 (i.e., 1,260 ransomware vectors and 2,000 benign vectors). Only entries from the *Image_File_Header*, *Image_Optional_Header*, and *Image_Section_Header* sections are chosen here, along the lines of earlier studies in [16]–[18]. Some selected PE parameters include *NumberOfSections*, *SizeOfCode*, *SizeOfHeaders*, etc. Note that PE format files also contain information on *dynamic-link library* (DLL) calls, and this can shed light on program functionality. For example, most ransomware programs use encryption, socket-based communication, and registry-modification functions. Hence the total number of DLL calls is also added to the feature vector, i.e., *TotalDLLCalls* (Fig. 5). Note that this is a *computed* feature and not a parameter read in from the PE file.

5 Performance Evaluation

As detailed in Section 3, the DRA architecture represents a generic approach which can implement a wide range of (NN-based) ransomware analysis schemes. Hence this framework is evaluated for the case of static analysis using the ML training/testing dataset curated in Section 4. All model development and testing is done using the *Keras* and *TensorFlow* libraries, along with other Python-based toolkits such as *Pandas*, *Numpy*, and *Sklearn*. Complete details on the testing setup and results from the performance evaluation study are now presented.

Table 2. Aggregate dataset partitioning (1,260 ransomware and 2,000 benign samples)

Clients	Ransomware				Benign		Malware %
	Training		Testing		Training	Testing	
	Per Family	Total	Per Family	Total			
$K=1$	120	1,080	20	180	1,700	300	38%
$K=2$	60	540	20	180	1,700	300	24%
$K=3$	40	360	20	180	1,700	300	17%
$K=4$	30	270	20	180	1,700	300	14%

FL performance is tested for a varying number of client sites ($K=2, 3, 4$) by partitioning the ML training/testing dataset, see Table 2. Recall that this aggregate dataset contains feature vectors (samples) for each raw binary file (where each feature vector contains 15 extracted PE file parameters, Fig. 5). Now there are 140 samples per family and 2,000 benign applications, i.e., total of 10 classes. Hence 120 samples from each ransomware family are randomly selected for training (1,080 total) and the remaining 20 are used for testing (180 total). This partitioning represents a 85/15% split between training and testing data. Next, the 120 ransomware samples (per class) are further partitioned between the client sites, yielding several local ML training datasets, i.e., full 120 samples per class

for $K=1$ (centralized ML), 60 samples per class for $K=2$ client sites (540 total), 40 samples per class for $K=3$ client sites (360 total), and 30 samples per class for $K=4$ client sites (270 total), see Table 2. Meanwhile the benign dataset is not partitioned between the client sites, thereby yielding a higher percentage of non-malicious training data. This choice mirrors realistic settings where regular applications downloads will exceed ransomware downloads. Moreover, there may be high commonality between application downloads across organizations, and hence it is plausible to use the same set of benign samples across client sites. Accordingly, the benign samples are split in a 85/15% manner, with 1,700 samples randomly selected for training and the remaining 300 for testing (per client site). Overall, the proportion of ransomware training data declines with the number of client sites, i.e., malware percentage, Table 2. For example, with 4 client sites there is more than 6 times less training data (i.e., 270 vs 1,700 samples). Carefully note that in practice, different regions (client sites) will experience varying types of ransomware attacks. These differences will yield unbalanced datasets with possibly different underlying distributions, and are left for future study.

Ransomware analysis is conducted using a range of supervised ML classifiers. These algorithms are trained and tested for the classification/attribution problem with 10 classes, i.e., 9 ransomware, 1 benign (Section 4). Foremost, the FL approach uses the supervised FNN algorithm (with gradient descent) to train local ML models at client sites. This network has 2 hidden layers and 32 nodes per layer, yielding a relatively small set of parameter weights (i.e., ω , Section 3). As noted earlier, FL can be used with many NN variants, but the FNN algorithm is chosen here given the smaller dataset sizes and feature vectors involved. Client sites train their local models over E epochs and send model parameter updates to the central server. Meanwhile, global training is done over T rounds. Several centralized ML algorithms are also used for comparison purposes, including centralized FNN ($K=1$), *support vector machines* (SVM), and *random forest* (RF) [29]. These classifiers are evaluated using the complete dataset with a 85/15% partitioning between training/testing samples. All results (FL, centralized) are averaged over 50 independent trails, with each using different randomized selections of the requisite dataset partitions. The control parameters for different ML algorithms are also fine-tuned to achieve the best classification rates.

Several metrics are used to gauge the ML classifiers, including accuracy, precision, recall, and F1 score. For example, precision is a measure of correctness whereas recall is a measure of relevance [29]. Two additional metrics are also defined to capture the *binary* detection capabilities of multi-class classifiers, i.e., selection between ransomware and benign. Consider mis-classification in more detail. Here, incorrectly classifying ransomware as benign is much more problematic than mis-classifying it as another family (as it may bypass network or host defenses). To quantify this, a *ransomware detection rate* (RDR) is defined:

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

where T_{rs} is the total number of ransomware testing samples across all families that are classified as (any family of) ransomware, and F_{rs} is the total number of

ransomware testing samples across all families that are mis-classified as benign, i.e., total number of ransomware testing samples is $(T_{rs} + F_{rs})$. This metric is similar to recall and treats all ransomware families as a single malicious class, i.e., tracks false negatives. Similarly, a *benign detection rate* (BDR) is also defined:

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

where T_{bn} is the total number of benign testing samples classified as benign, and F_{bn} is the total number of benign testing samples mis-classified as ransomware, i.e., total number of benign testing samples is $(T_{bn} + F_{bn})$. Note that the false negative classification of benign applications is generally less of a security threat than false negative classification of ransomware programs.

Table 3. Results for FL with varying client sites, K (FedAvg)

2 Client Sites	Accuracy	Precision	Recall	F1 Score
Global Avg. Model	91.87%	86.01%	86.67%	91.82%
Client site 1	86.52%	78.57%	79.37%	86.30%
Client site 2	88.34%	78.59%	79.22%	87.34%
3 Client Sites	Accuracy	Precision	Recall	F1 Score
Global Avg. Model	92.11%	87.40%	86.57%	91.90%
Client site 1	86.02%	77.86%	75.64%	85.33%
Client site 2	82.90%	73.89%	74.56%	82.13%
Client site 3	81.47%	73.55%	75.95%	81.49%
4 Client Sites	Accuracy	Precision	Recall	F1 Score
Global Avg. Model	92.46%	87.69%	86.40%	92.43%
Client site 1	79.93%	72.22%	72.61%	79.53%
Client site 2	83.70%	73.29%	74.41%	83.48%
Client site 3	85.48%	71.91%	71.73%	83.76%
Client site 4	85.49%	72.23%	71.41%	80.20%

Detailed results for the FL scheme are first presented in Table 3 for varying numbers of client sites (K values). In each case, the table lists the accuracy, precision, recall, and F1 scores for the global averaged FNN model (in bold) followed by the individual local client site models. These findings show vastly improved model generalization with FL, with the global models exceeding the individual client models by sizeable margins for all metrics. For example, average accuracy is 3-12% higher, precision is 7-15% higher, recall is 7-15% higher, and F1 scores are 4-13% higher. These are very impressive results and indicate that organizations can greatly improve their ransomware defenses by participating in FL-based schemes, i.e., while retaining data privacy and maintaining smaller datasets. Furthermore, FL accuracy is also very high. For example, the accuracy (and F1 score) with 4 client sites is close to 92.5%, i.e., correct attribution of over 18 out of 20 samples. These results also closely match some centralized ransomware classification schemes (Section 2.1) many of which implement heavier

feature extraction and ML computation algorithms, e.g., image and entropy-based features, deep NN designs, etc [14],[15]. By contrast, the FL setup herein uses very small feature vectors (15 parameters) and basic FNN models.

Table 4. Results for ML models (averaged over 50 independent runs)

Distd. FL	Accuracy	Precision	Recall	F1 Score	Binary	
					RDR	BDR
2 clients (FNN)	91.87%	86.01%	86.67%	91.82%	94.92%	94.99%
3 clients (FNN)	92.11%	87.40%	86.57%	91.90%	93.04%	95.86%
4 clients (FNN)	92.46%	87.69%	86.40%	92.43%	93.27%	96.43%
Centralized	Accuracy	Precision	Recall	F1 Score	Binary	
					RDR	BDR
FNN	91.48%	86.84%	84.68%	91.27%	92.06%	96.69%
SVM	90.44%	91.16%	75.60%	89.86%	80.39%	97.85%
RF	96.02%	94.41%	92.07%	95.98%	95.72%	99.05%

Next, the performance of all ML classifiers is presented in Table 4. Here the respective FL percentages are the same as the global model averages from Table 3. First consider multi-class attribution, as measured by the accuracy, precision, recall, and F1 scores. Foremost, the FL approach outperforms its centralized FNN counterpart for varying numbers of client sites. In particular with $K=4$ client sites the accuracy and F1 scores are 1% higher. These gains come despite using much smaller training datasets at the client sites, e.g., only 270 ransomware samples for $K=4$, Table 2. Again, this is another key result as it demonstrates the ability of decentralized FL setups (with smaller client sites) to achieve similar or better ransomware attribution compared to less practical centralized setups (requiring much more training data). Furthermore, the FL approach also outperforms the SVM algorithm by over 2% in terms of accuracy. However, the centralized RF scheme (trained with global data) gives the best results, with accuracy and F1 scores averaging about 3.5% higher than FL.

Also, Table 4 presents *binary* detection results, as measured by the RDR and BDR metrics (Eqs. 1, 2). The former is deemed more important as it captures the mis-classification rate of ransomware. Overall, FL gives very good RDR values, up to 94.92%, and within 1% of the RF scheme which has the best binary results. Namely, this scheme yields a false negative rate of about 1 in 20 malicious samples. These results are impressive and match those from earlier studies on binary detection of older ransomware threats, Section 2. Also, the BDR values are higher than the RDR values since training datasets have a larger proportion of benign data as per real-world settings (Table 2). Note that the FNN-based schemes (including distributed FL) give slightly lower BDR values than the other ML algorithms, i.e., by about 1-2.5%. Nevertheless, the related BDR percentages are still close to 97%, i.e., 1 error in about 33 benign samples.

To investigate detailed per-class behaviors, Fig. 6 shows a sample averaged confusion matrix for FL with 4 client sites (rows 0-9 represent ransomware fami-

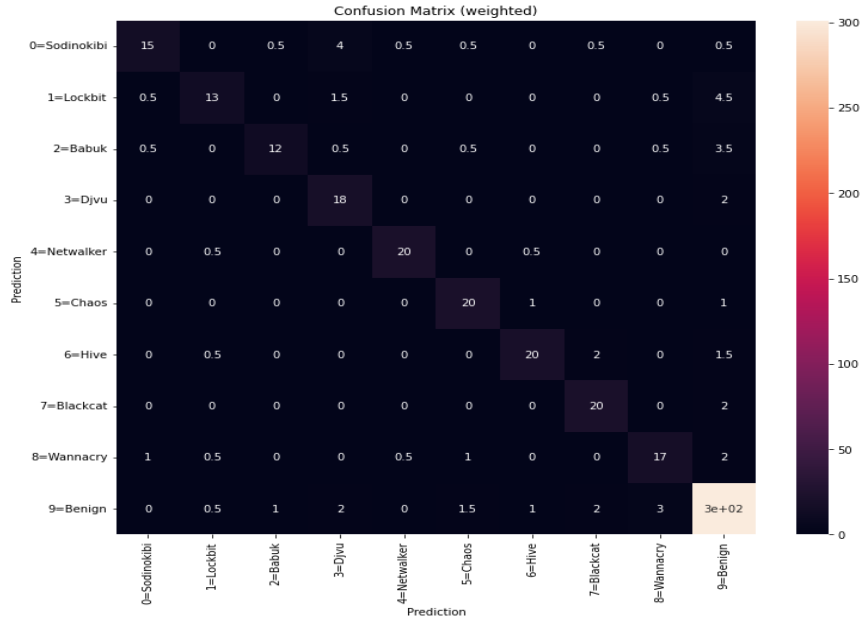


Fig. 6. Sample multi-class confusion matrix for FL ($K=4$ client sites, $E=1$ epoch)

lies and row 10 represents benign applications). Note that the numbers in row 10 are larger as there are more benign testing samples, Table 2. Here, the majority of ransomware and benign application samples are correctly classified, i.e., diagonal entries dominate. Furthermore, most mis-attributed ransomware samples are still classified as some form of ransomware, although LockBit and Babuk show higher averages, i.e., 4.5 (22.5%) and 3.5 (17.5%) out of 20, respectively. Hence the potential damage from false negatives is relatively small for most ransomware cases. Meanwhile, Fig. 7 plots the average accuracy and F1 scores for the global FL model with $K=4$ client sites and varying epoch counts (E). These results shed light on the local FL training process and indicate better performance with $E=4$ epochs (also observed for several other K values).

Finally, Fig. 8 plots the accuracy, precision, recall, and F1 scores at the end of each communication round for the global FL model with $K=4$ client sites (and $E=4$ epochs). For reference sake, corresponding accuracy values for the centralized FNN classifier are also plotted. The objective here is to observe the FL training process over multiple rounds. These plots show very rapid progression, with performance improving rapidly within the first 10 rounds and stabilizing by about 15 rounds. By contrast, the centralized FNN takes more rounds to improve, with accuracy picking up after 9 rounds. Overall, these results demonstrate improved responsiveness and learning with the distributed FL approach, further complementing its inherent privacy and scalability benefits.

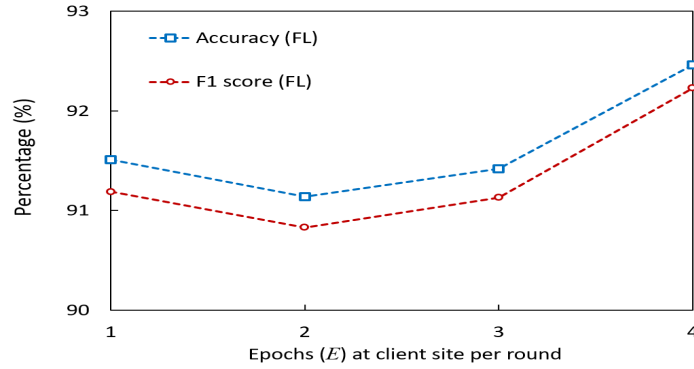


Fig. 7. Accuracy and F1 score results for $K=4$ client sites (averaged over 50 trials)

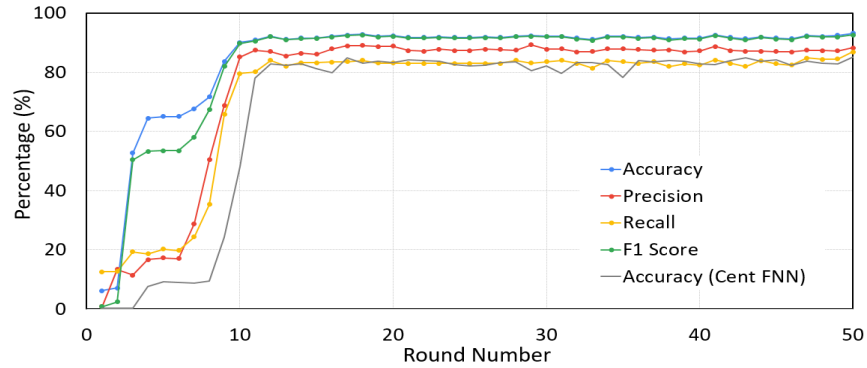


Fig. 8. Sample progression of FL training (FedAvg, $K=4$ client sites, $E=4$ epochs)

6 Conclusions & Future Work

Ransomware presents a persistent threat to users and organizations. Researchers have developed various solutions to detect and classify this malware using *machine learning* (ML) schemes to analyze binary files and system/network activities. However, growing scalability and privacy concerns make it difficult to collect and analyze massive amounts of data at a centralized site. Hence this study presents a novel *distributed ransomware analysis* (DRA) framework for detection and attribution using *federated learning* (FL). This architecture embodies a generic approach which can implement both static and dynamic analysis schemes. A realistic dataset repository comprising of some of the latest ransomware threats is used to conduct a detailed performance evaluation study for the case of static analysis with rapid/lightweight feature extraction from Windows *portable executable* (PE) format files. Overall findings confirm superior performance for the FL-based approach, with global ML model performance notably exceeding locally trained models. The FL approach also closely matches

or outperforms some centralized ML algorithms in terms of attribution accuracy and binary detection.

Overall, this effort presents one of the first studies on ransomware detection and attribution using the decentralized FL framework and provides a strong basis for further study. Foremost, a broader set of static and dynamic features can be used to improve local model training (leveraging existing work on ransomware detection). Further efforts can also address FL bias and variability concerns.

7 Acknowledgements

This work has been supported in part by Cyber Florida. The authors are very grateful for this support.

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. Yayimli, “Ransomware detection and classification strategies,” in *IEEE Black Sea Communications Conference 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. H. McMahan, “Communication-efficient learning of deep learning from decentralized data,” in *AISTATS 2017*, Ft. Lauderdale, FL, April 2017.
5. Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, and B. He, “A survey on federated learning systems: Vision, hype and reality for data privacy and protection,” in *arXiv 1907.09693*, December 2021.
6. N. Hudson, M. Hossain, M. Hosseinzadeh, H. Khamfroush, M. Naeini, and N. Ghani, “A framework for edge intelligent smart distribution grids via federated learning,” in *IEEE ICCCN 2021*, July 2021.
7. Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, “Multitask network anomaly detection using federated learning,” in *10th Intl. Symposium on Information and Communication Tech. (SoICT 2019)*, Ha Long Bay, Vietnam, December 2019.
8. B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, “Deepfed: Federated deep learning for intrusion detection in industrial cyberphysical systems,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, August 2021.
9. T. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. Sadeghi, “Diot: A federated selflearning anomaly detection system for iot,” in *IEEE Intl. Conf. on Distributed Computing Systems (ICDCS) 2019*, Dallas, TX, July 2019.
10. V. Rey, P. Sanchez, A. Celdran, G. Bovet, and M. Jaggi, “Federated learning for malware detection in iot devices,” in *arXiv 2104.09994*, November 2021.
11. S. Poudyal, K. P. Subedi, and D. Dasgupta, “A framework for analyzing ransomware using machine learning,” *IEEE 2018 SSCI*, Nov. 2018.
12. 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.

13. D. Mulders, "Network based ransomware detection on the samba protocol," *MS Thesis, Department of Mathematics, TU Eindhoven*, 2017.
14. 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.
15. 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.
16. D. Kim, S. Woo, D. Lee, and T. Chung, "Static detection of malware and benign executable using machine learning," in *8th International Conference on Evolving Internet*, Barcelona, Spain, November 2016.
17. Y. Liao, "Pe-header-based malware study and detection," in *Semantic Scholar*, 2021.
18. 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.
19. A. Vehabovic, H. Zandizzari, F. Shaikh, N. Ghani, M. S. Pour, E. Bou-Harb, and J. Crichigno, "Data-centric machine learning approach for early ransomware detection and attribution," in *8th IEEE/IFIP NOMS Workshop on Analytics for Network and Service Management (AnNet 2023)*, Miami, Florida, May 2023.
20. A. Almashhadani, 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.
21. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, and S. Hashemi, "Drthis: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, pp. 94–104, Jan. 2019.
22. O. Alhawi, J. Baldwin, and A. Dehghantanha, "Leveraging machine learning techniques for windows ransomware network traffic detection," *Advances in Information Security*, p. 93–106, July 2018.
23. K. C. Roy and Q. Chen, "Deepran: Attention-based bilstm and crf for ransomware early detection and classification. information systems frontiers," *Information Systems Frontiers*, vol. 0, pp. 1–17, 2021.
24. 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.
25. E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "Paybreak: Defense against cryptographic ransomware," in *Asia CCS 2017*, Abu Dhabi, UAE, April 2017.
26. R. M. A. Molina, S. Torabi, K. Srieddine, E. Bou-Harb, N. Bouguila, and C. Assi, "On ransomware family attribution using pre-attack paranoia activities," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 19–36, 2022.
27. R. M. A. Molina, "Rpm: Ransomware prevention and mitigation using operating systems sensing tactics," *submitted*, 2022.
28. P. Kairouz, "Advances and open problems in federated learning," in *arXiv 1912.04977*, March 2021.
29. A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Second Edition*. O’Reilly Media, 2019.
30. A. Ng, "Ai minimalist," *IEEE Spectrum*, vol. 59, no. 4, pp. 23–25, April 2022.
31. Q. Li, *Definitive Guide to Windows PE large systems security technology*. Machinery Industry Press, 2000.