IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 September 2022, accepted 27 October 2022, date of publication 1 November 2022, date of current version 11 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3218779

RESEARCH ARTICLE

Machine Learning Algorithms and Frameworks in Ransomware Detection

DARYLE SMITH, SAJAD KHORSANDROO, AND KAUSHIK ROY
Department of Computer Science, North Carolina A&T State University, Greensboro, NC 27411, USA
Corresponding author: Daryle Smith (dsmith18@aggies.ncat.edu)

ABSTRACT Ransomware has been one of the biggest cyber threats against consumers in recent years. It can leverage various attack vectors while it also evolves in terms of finding more innovative ways to invade different cyber security systems. There have been many efforts to detect ransomware within the workforce and academia leveraging machine learning algorithms, which has shown promising results. Accordingly, there is a considerably large body of literature addressing various solutions on how ransomware threats can be detected and mitigated. Such large and rapidly growing scientific and technical materials start to make it difficult in knowing the actual ML algorithm(s) being used. Hence, the aim of this paper is to give insight about ransomware detection frameworks and those ML algorithms which are typically being used to extract ever-evolving characteristics of ransomware. In addition, this study will provide the cyber security community with a detailed analysis of those frameworks. This will be augmented with information such as datasets being used along with the challenges that each framework may be faced with in detecting a wide variety of ransomware accurately. To summarize, this paper delivers a comparative study which can be used by peers as a reference for future work in ransomware detection.

INDEX TERMS Artificial Neural Network (ANN), cyber security, deep convolutional neural network (DCNN), deep neural network (DNN), Hardware Performance Counter (HPC), Long Short Term Memory (LSTM), machine learning (ML), ransomware, Recurrent Neural Network (RNN), Sum of Product (SOP), Support Vector Machine (SVM), Term Frequency and Inverse Document Frequency (TF-IDF), The Onion Routing (TOR).

I. INTRODUCTION

Ransomware has been a threat against typical end users, business units, and the government in recent years. For example, it has targeted medical centers, schools [1], universities [2], and police departments [3], to name a few. It was even predicted that ransomware would account for around $20 billion in loss alone towards organizations in 2021 [4]. Ransomware is a form of malware designed to control access to data or a system until a requested ransom amount from the attacker is satisfied [5]. Detection of ransomware is tricky and a resource hungry task because it is hidden within the application layer payload. Mitigation can also be difficult because of the use of encryption against the application. Though more studies and evaluations have been involved in other areas of malware, ransomware specifically has not been the focal point, and the push to improve security measures and discovery have been stagnant [5]. There are two types of ransomware: first, locker-ransomware, which is designed to lock the victims' computer, to prevent them from using it; Second, and most common nowadays, is crypto-ransomware, which encrypts personal files to make them inaccessible to its victims [55]. Frameworks applying static and dynamic analysis, as well as, ML algorithms, have been aiding with ransomware detection, and due to the nature of executing the ransomware, a high accuracy rate would be expected. However, analysis takes a relatively long time, leaving gaps where the malicious payload can intrude the sandbox system without detection. This entire process alone is overly complicated. In this paper, we focus on those ML algorithms that are mostly used in ransomware detection. We also provide a brief review of common ransomware frameworks using such algorithms, along with their results.

When it comes to ransomware attacks, cybercriminals have perfected these techniques over the years. However, both academia and industry have been trying to address these threats and protect victims by learning from past experiences and utilizing technological advancements over time [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wu.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

Nonetheless, these attacks are growing daily. The underlying reason is that combating ransomware is challenging [7]. Ransomware typically relies on strong encryption that is easy to accommodate due to the vast amounts of open-source implementations. It also makes use of most infection techniques that are employed by modern malware families. Ransomware benefits from the common elusive methods utilized by modern malware and it frequently uses application programming interfaces (APIs) to carry out malicious actions that make it difficult to differentiate from benign applications. Furthermore, it uses TOR networks (The Onion Routing networks) to keep its communication anonymous, and unregulated payment techniques like cryptocurrencies to get paid without easily disclosing the identity of the attackers [8].

The remaining structure of this paper is organized as follows and investigated perspectives can be found in Figure 1: Section 2 reviews required research background and provides a comprehensive review of different ML algorithms used in ransomware detection. Section 3 will provide an analysis of ransomware frameworks that use ML algorithms, their challenges, evaluations, and results. This section also expands on the importance of this research, consolidating all the mentioned frameworks in a table, providing a description of each framework's name, the algorithm(s) of choice, the year it was created, overall results, and challenges. Section 4 will speak about some future concerns and defense topics of ransomware, ending with concluding the paper in section 5.

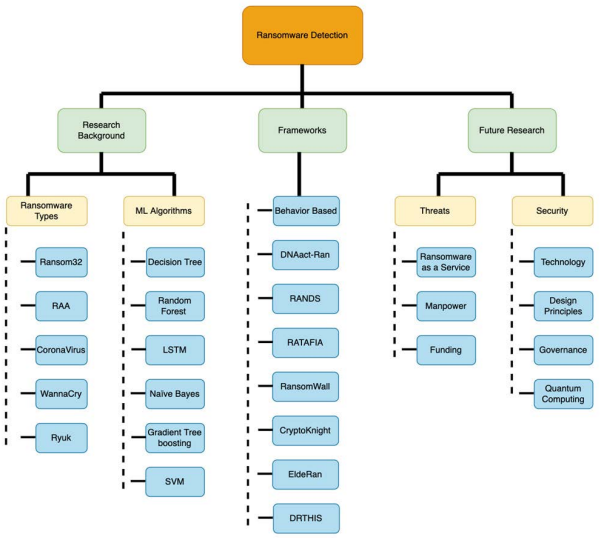

FIGURE 1. An overview of discussed ransomware.

II. RESEARCH BACKGROUND

This section will briefly cover different types of ransomware that are common across the cyber security community. It will also cover typical machine learning algorithms used in ransomware detection.

A. Ransom32

With the emergence of social media and its popularity in the younger generations, new ransomware families are being created by cybercriminals to target web pages with the use of JavaScript. One such example is Ransom32, which first appeared in late 2015. Until its discovery, no other ransomware attack was used with that programming language [9]. This type of ransomware-as-a-service is unique because, being written in JavaScript, it uses a web browser to initiate its attack. The impact of this threat is far superior in nature because it can be used theoretically on any operating system where a web browser exists. This grants Ransom32 so-called ''write-once-infect-all'' capabilities [9]. Nonetheless, Ransom32 has only been detected on a Windows-based platform thus far. It can be found on most underground TOR sites and can be downloaded by the affiliated user. To download the executable, one must have a bitcoin address, as this is the way payments of ransom are made.

The developers of the Ransom32 software take a 25% cut of any ransom made, and the rest goes to the user of the affiliated program [9]. When the Ransom32 executable runs, it extracts several files. During this process, a shortcut is created in the start menu, and the ransomware will start at login which guarantees the malware will be executed every time the system is started. The shortcut points to a chrome.exe executable file that is typically an NW.js package. This package contains JavaScript code used for encryption using AES and extracts to folders such as %AppData% and %Temp%. Furthermore, this is the piece that contributes to performing the harmful events towards the compromised system [10]. With NW.js being a legitimate framework and application, antivirus coverage in this area is still very weak in nature. Any black hat or white hat developer can use this executable to create and distribute native apps that run just like normal executables [11]. Furthermore, when looking closer into Ransom32, it runs under the context of the user without having any administrative rights or permissions. Figure 2 gives a general idea of how a member can join the affiliate network, then ultimately be granted access to download the malicious code for use. The member would also be able to see statistics related to the software such as the number of payments that

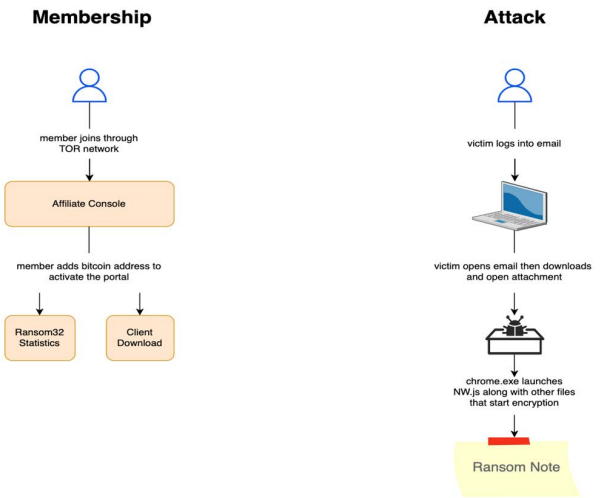

FIGURE 2. Ransom32 membership and attack process flows.

have been made and the number of installations that have been completed. The figure also shows the process of how a Ransom32 attack can occur.

```
function (NOWuidbsIBSbdbiubix() {
    var d_pn = "dosbiuiyvuvWIYVUWGTVICydbsbobsiodbwyev";
    var flo = new ActiveXObject (ADODB.Stream");
    var runer = WScript.CreateObject("WScript.Shell");
    var wher = runer.SpecialFolders("MyDocuments");

    wher = wher + "\\" + "st.exe";
    flo.CharSet = "437";
    flo.Open();
    var pony = d_pn.replace(/NMSIOP/g, "A");
    var pony_ar = CryptoJS.enc.Base64.parse(pony);
    var pony_dec = pony_ar.toString(CryptoJS.enc.Utf8);
    flo.Position = 0;
    flo.SetEOS;
    flo.WriteText(pony_dec);
    flo.SaveToFile(wher,2);
    flo.Close;
    wher = "\"" + wher + "\"";
    runer.Run(wher);
}
```

FIGURE 3. A deobfuscated function installing Pony.

B. RAA

The second example of JavaScript ransomware is RAA, which spreads via email attachments pretending to be legitimate document files. These files typically have a valid file format with a JavaScript extension, making the victim believe its authenticity. Once the file is opened, it works just as any other ransomware attack. The victim host's files will be encrypted, and a ransom will be demanded. RAA also infects the victim's computer by installing Pony, a well-known password-stealing malware embedded in a JavaScript file. A sample code of how this happens can be found in Figure 3. This malware can collect browser passwords and other critical information on infected systems. Two security researchers initially discovered RAA and according to them, it encrypts files using code from an open-source library called CryptoJS [12]. This code handles cipher algorithms such as AES, DES, to name a few [12]. RAA targets images, Ms-Word, Ms-Excel, Photoshop, .zip, .rar, sparing program files, Windows files, AppData, and Microsoft files by appending a ''.locked'' to the end of the filenames [12]. Upon further analysis, Trend Micro™ discovered that the RAA ransomware is written in Jscript and not JavaScript [13]. Jscript is designed for Windows® systems and executed by the Windows Scripting Host Engine through Microsoft Internet Explorer (aka IE), but not via the Microsoft Edge browser. Jscript carries some semblances with JavaScript because they are both derived from ECMAScript.[1] Jscript is the implementation of ECMAScript while JavaScript is the Mozilla implementation of ECMAScript [53]. Jscript can access objects exposed by IE and some systems objects such as the Wscript[2] [54]. It is believed that the attackers behind the RAA ransomware are using the Jscript scripting language to make detection more difficult and to make complications easier. Most malware attacks are written in compiled programming languages with ransomware often disguised as executables. Nonetheless, using languages which are not typically used to deliver malware, such as scripting languages, could be less prone to detection [13].

C. CoronaVirus

Ransomware affiliates switched to COVID-19-themed social engineering tactics during the 2020 pandemic to carry out threats [14]. Mobile applications that looked legitimate would download various forms of ransomware using spam attachments that claim to provide health and safety information about COVID-19 [15]. As the global pandemic increased the need for health centers, the exposure to cyber-attacks also boosted. This situation increased the number of ransomware attacks within the health sectors, and Corona ransomware was born [16]. This was a new strain that focused specifically on hospitals and the encryption of patients' health records. After the host became infected, it displayed a COVID-19-themed ransom message and demanded payment in Bitcoin [14].

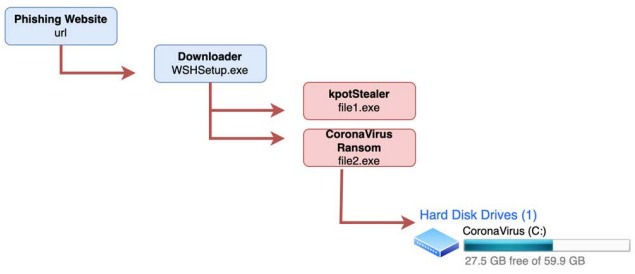

FIGURE 4. CoronaVirus delivery flow.

The Covid-19 pandemic also opened the doors for many ransomware attacks against employees. Due to the threat of catching the virus, many companies began to offer employees the opportunity to work remotely [43]. This increases exposure to cyber-risks because individuals connect through less reliable and unsecured Internet connections. Employees that accessed corporate networks using personal devices provided a way to get into the hands of unauthorized individuals through unsanctioned channels [43]. Attackers also focused heavily on sophisticated phishing techniques. According to [44], an APWG report showed 267,372 phishing campaigns were reported in the first quarter of 2020, increasing (19.06%) over 2019 during the same period. In Figure 4 below, the CoronaVirus delivery flow begins with a phishing website, locking the file system, then fully compromising the hard drive until the ransom has been paid.

[1]ECMAScript is a Javascript standard that helps ensure the interoperability of web pages across different browsers.

[2]Wscript are generic scripts specifically executed in Windows based platforms.

D. WannCry

WannaCry, introduced itself and targeted computers running the Windows operating system [45]. It encrypts a victim's data using Microsoft's flawed protocol EternalBlue, then demands payment in Bitcoin once the infection has taken place [46]. This vulnerability allows the adversaries to execute a remote code on the infected machines by sending specially crafted messages to an SMB v1 server, connecting to TCP ports of unpatched Windows systems [47]. WannaCry also works as a network worm because it includes a transport mechanism to automatically spread itself, Figure 5 shows this visual. This feature makes the attacks more effective and requires defense mechanisms that can react quickly and in real time. Furthermore, WannaCry has an encryption component that is based on public-key cryptography. The virus impacted more than 200,000 computers in over 150 countries [48]; Ukraine, India, Russia, and Taiwan were the four most affected countries [49]. Vladamir Putin, president of Russia, blamed the United States for the attack due to their involvement with developing EternalBlue, but it was later determined a group of North Korean hackers were responsible [50]. Microsoft began providing patches for older system versions on the day of the outbreak, but the count of attacked systems continued to rise as new versions and variants of the ransomware were constantly released [51]. The spread of the virus was slowed by the work of Marcus Hutchins, who discovered a ''kill switch'' inside the virus [52], [60].

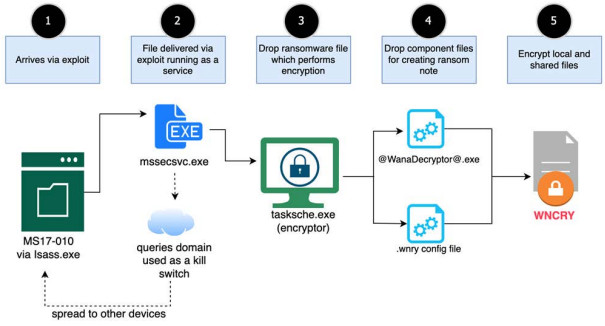

FIGURE 5. WannaCry process flow as a network worm.

E. RYUK

In mid to late 2018, a new type of ransomware started targeting specific victims and carried out its attacks against enterprises [56]. Ryuk mostly infected its targets via other malware [57] and attacks would disable the Windows system restore option, making it impossible to restore encrypted files without a backup [58]. During infection, Ryuk first shuts down 180 services and 40 processes [59]. These services and processes could prevent Ryuk from doing its own work and are needed to facilitate the attack. At that moment, the encryption logic can begin. Ryuk encrypts files such as photos, videos, databases, and documents – all the data you care about – using AES-256 encryption. The symmetric encryption keys are then encrypted using asymmetric RSA-4096. Ryuk can encrypt remotely and perform Wake-On-Lan, waking computers for encryption [59]. These abilities contribute to the effectiveness and reach of its encryption and the damage it can cause. Ryuk accounts for three of the top 10 largest ransom demands of the year in the CrowdStrike 2020 Global Threat Report, with amounts of USD $5.3 million, $9.9 million, and $12.5 million [59]. The Russian hacker group, WIZARD SPIDER, is said to be the creator of Ryuk, and in 2020 during the coronavirus pandemic an attack was targeted against Universal Health Services [59]. The fortune 500 company has health care facilities in both the US and UK, with exposure stemming from a phishing email [61]. Figure 6 demonstrates how Ryuk can attack an Active Directory that has been misconfigured.

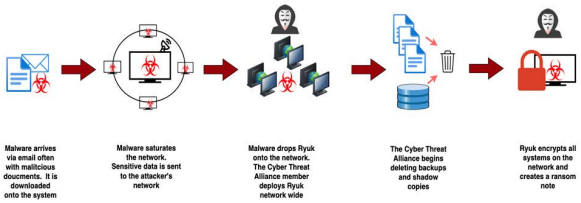

FIGURE 6. Common Ryuk attack with regards to a misconfigured active directory.

F. MACHINE LEARNING WITH RANSOMWARE

Machine Learning (ML) has become a mature technology that is being applied to a wide range of business problems such as web search, online advertising, product recommendations, object recognition, and so on. As a result, it has become imperative for researchers and practitioners to have a fundamental understanding of ML concepts and practical knowledge of end-to-end modeling [17]. Machine Learning contains the use of statistical methods for the detection of patterns within data and those patterns are constructed against mathematical models [17]. These models are then used to make predictions against future data. Machine Learning is being used extensively by companies across a broad spectrum of applications and there are many other areas such as game playing, unmanned cars, and automated question answering where ML is poised to drastically change the way technology affects our lives.

There have been multiple attempts to detect ransomware, and a plethora of researchers have tested against several frameworks. Various surveys that have condensed ransomware characteristics and attacks to provide a full spectrum of what ransomware really is, how it works, and how to limit its threat. However, a study against Machine Learning algorithms specifically used to detect ransomware and the classification of those frameworks have not been addressed directly. Due to the advancement of social media and enriched websites involving user input and interaction, it is important to understand what algorithms are providing the best results in detecting ransomware so that the research community can improve the areas that are not working. Furthermore, as the threat of ransomware continues to grow, having a direct go-to-guide of algorithms that have proven to be effective will

help the research community spend less time analyzing irrelevant information. To fill this research gap, a broad study of ransomware detection frameworks and tools that utilize ML algorithms are reviewed. This research shows how those frameworks are used in relation to ransomware detection, the ML algorithm of choice, and the results of accuracy to predict and classify ransomware.

The following sections will discuss the typical algorithms that are used in detecting ransomware. These algorithms are performed under multiple trials of data sets and are often combined with other algorithms using cross validation analysis.

1) DECISION TREE

Decision Tree algorithm belongs to the family of supervised learning algorithms where learning and prediction steps are performed. The model in the learning step is developed based on given training data, and the model in the prediction step model is used to guess the response for given data [18]. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving both regression and classification problems. When using a Decision Tree, the goal is to build a training model that can predict the class or value of the target variable by learning simple decision rules inferred from prior data. For predicting a class label for a record, one would start from the root of the tree. The values are then compared with the root attribute along with the record's attribute. The branch which links to that value is followed, based on the comparison, and jumps to the next node [18]. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves. There are two types of Decision Trees based on the type of target variable being used: Categorical Variable Decision Tree and Continuous Variable Decision Tree. A categorical variable decision tree is illustrated by Figure 7.

Decision Trees follow the Sum of Product (SOP) representation. Every branch from the root of the tree to a leaf node having the same class is the conjunction (product) of values, and different branches ending in that class form a disjunction (sum). The main purpose of a Decision Tree is to detect which attributes are needed to consider as the root node. The tree's accuracy is dependent upon its decisions on how it splits its nodes, and they use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes, meaning the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in the most homogeneous sub-nodes [18]. One problem that must be accounted for is overfitting, which happens when a tree is overly complex and does not generalize against the trained data. To correct this problem, a data compression technique called pruning is performed to reduce the size of the tree by removing sections that provide limited value [62].

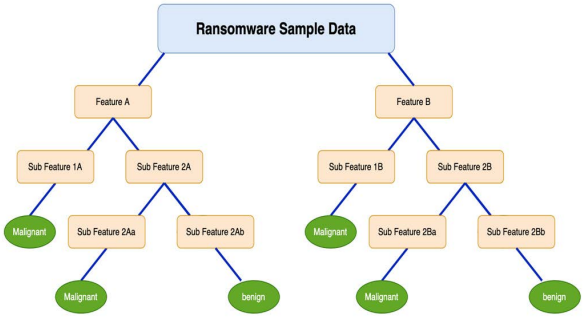

FIGURE 7. A breakdown of decision tree using ransomware sample data.

2) RANDOM FOREST

Random Forest is a supervised learning algorithm that builds a forest of decision trees, usually trained with the bagging method. This method gives awareness that a mixture of learning models increases the overall result. Random Forest can be used for both classification and regression problems and typically has the same hyperparameters of a Decision Tree [19]. While growing trees, this algorithm adds additional randomness to the model in hopes of producing an even better model. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features [63]. Trees can become more random by using random thresholds for each feature rather than searching for the best possible thresholds. Random forest is a great algorithm to train early in the model development process, to see how it performs. Its ease makes building a good random forest quite simple. The algorithm is also a great choice for developing quick models and showing good metrics of the importance it assigns to features. The performance of Random Forest is quite consistent and is difficult for other algorithms to achieve [19]. Random Forest algorithms are not ideal in the extrapolation of data, nor does it produce satisfactory results with sparse data. They typically will spend more time when compared to a decision tree and require more resources for computation [20]. Figure 8 provides an example of how ransomware data may be used with the random forest tree.

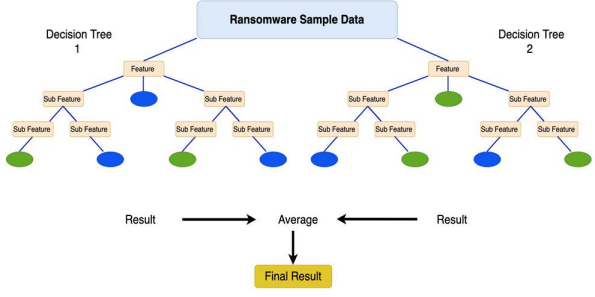

FIGURE 8. Random Forest tree operations.

3) LONG SHORT TERM MEMORY

Long Short Term Memory is a type of recurrent neural network (RNN) capable of learning order dependence in

sequence prediction problems. They use special units and standard units, which include a memory cell that can maintain information in memory for long periods of time. The core concept of an LSTM is the cell state and three gate phases [21]. The cell state acts as a gateway that transmits relative information all the way down the sequence chain. It can be thought of as the memory of the network. The cell state carries information throughout the sequence processing, allowing data from an earlier time step to be present later. This process helps reduce the effects of short-term memory. As the cell state goes on its journey, information is either added or removed to the cell state via gates. These gates are different neural networks that decide which information is allowed on the cell state [21].

The first gate, the forget gate, decides whether the data should be kept from the previous timestamp or forgotten. Information from the previous hidden state and information from the current input is passed through a sigmoid function where values come out between 0 and 1. The closer to 0 means to forget the data, whereas the closer to 1 means to keep it [22]. The second part is called the input gate, and it is used to quantify the importance of the new information carried by the input. It passes the previous hidden state and current input into a sigmoid function that decides which values will be updated. It also transfers the hidden state and current input into the tanh function which helps regulate the network. The sigmoid output will decide which information is important to keep from the tanh output and it uses the same 0 and 1 approach as the forget gate. The output gate passes the updated information from the current timestamp to the next timestamp, deciding what the next hidden state should be. Because LSTM can give more accurate predictions from recent information, it solves the problem of long-term dependencies by trying to predict words in long term memory. LSTM can maintain information for a long period of time and is used for processing, predicting, and classifying time-series data [21], [22], [64].

4) NAÏVE BAYES

Naïve Bayes is a classification algorithm based on the Bayes Theorem for calculating probabilities and conditional probabilities [23]. It is not a single algorithm but a family of algorithms that share a common principle. This algorithm is extremely fast and is mainly used with large datasets. It assumes that the occurrence of a particular feature does not affect the other and is known to outperform some of the better classification methods [24]. A Naive Bayes model consists of a large block that includes an input field name, an input field value, and a target field value. The model is used to record how often a target field value appears together with a value of an input field. The value of the probability-threshold parameter is used if one of these fields of the block is empty, which occurs when a training-data record with the combination of an input field value and target value does not exist. The NB algorithm using ransomware data is shown in Figure 9.

```
1. Load ransomware data
        def sampleset(Rdata)
        Input:      Ransomware Sample Data

        Output:     Farray  //features
                    Larray  //labels | classes

2. Split data into training and test sets
        def tts(data, Larray, size, state)
        Input:      size = percentage of test data used
                    state = randomness value
                    data, Labels_Array

        Output:     A_train
                    A_test
                    B_train
                    B_test

3. Create the NB classifier
        nb = nbC()

4. Train the model
        nb.fit(A_train, B_train)

5. Predict the response for test dataset
        B_pred = predict(A_test)
```

FIGURE 9. Code snippet of a NB algorithm using ransomware data.

5) GRADIENT TREE BOOSTING

Gradient Tree boosting is a machine learning algorithm used for building predictive models regarding its prediction speed and accuracy, especially with large and complex data. It works with both classification and regression problems that utilize weaker learners to generate a more accurate predictor. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. A gradient-boosted trees model is made in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of a random differentiable loss function [25]. It is composed of three elements: a loss function to be optimized, a weak learner to make predictions, and an additive model to add weak learners to minimize the loss function.

Overfitting is a problem in Fitting the training set too closely can lead to degradation of the model's generalization ability. Some regularization techniques reduce this overfitting effect by constraining the fitting procedure. One way to achieve this goal is by using the number of gradient boosting iterations for its regularization parameter [25]. Increasing this reduces the error within the training set. However, it must not be set too high. Monitoring the prediction error on a separate validation data set can also aid in selecting the correct number of iterations. Several other regularization techniques can be used such as the depth of trees. A higher value in this regularization parameter typically shows that the model will overfit the training data.

6) SUPPORT VECTOR MACHINE

Support Vector Machine is an algorithm used for both regression and classification tasks but is primarily used in classification objectives. It does not require high computation power but still produces significant accuracy. The support vector machine algorithm finds a hyperplane in an N-dimensional space that classifies the data points. The classification is

performed by finding the hyper-plane that differentiates the two classes very well [26]. They are effective in high dimensional spaces and can still be useful in cases where the number of dimensions is greater than the number of samples. SVMs use support vectors, a subset of training points in the decision function, providing memory efficiency. The versatility of the algorithm is also a key point as it can use different kernel functions against the decision function or even use custom kernels. However, overfitting will occur if the number of features is much greater than the number of samples. SVMs do not provide probability metrics and must use five-fold cross validation. The SVM algorithm has been applied in biological science for use in the categorization of protein; it has also been widely used with the classification of images, producing higher accuracy results than traditional query refinement schemes [26]. In Figure 10, pseudo code of the SVM algorithm using ransomware sample data is shown.

```
1. Load ransomware data
        def dataset(ransomwareData)
        Input:    Ransomware Sample Data
        Output:   Features_array, Labels_Array

2. Split data into training and test sets
        def train_test_split(data, Labels_Array, size, state)
        Input:    size = percentage of test data used, state = randomness value
                  data, Labels_Array
        Output:   X_train, X_test, y_train, y_test

3. Create the SVM classifier
        svm = svmC()

4. Train the model
        svm.fit(X_train, y_train)

5. Predict the response for test dataset
        y_pred = predict(X_test)

6. Evaluate accuracy using actual and predicted values
        accuracy_score(y_test, y_pred))
```

FIGURE 10. Code snippet of a SVM algorithm using ransomware data.

III. RANSOMWARE DETECTION FRAMEWORKS

This section investigates several ML-based frameworks which are widely used in detecting ransomware. Some of the reviewed frameworks utilize one ML algorithm while the others might use multiple. These frameworks yield promising results in detection of different types of ransomware and have potential to be used in future research works by the cyber security community. In what follows, eight state-of-the-art frameworks including Behavior Based [27], DNAact-Ran [28], RANDS [30], RATAFIA [32], RansomWall [33], CryptoKnight [34], EldeRan [35], and DRTHIS [40] will be studied and compared.

A. BEHAVIOR BASED

A proposed behavior-based framework was built for defining dynamically monitored valuable features of high survivable ransomware (HSR) [27]. The analysis of HSR was conducted within an isolated sandbox environment, through the Term Frequency-Inverse document frequency (TF-IDF). By doing so, the most relevant features that provided optimal performance in detecting new ransomware were extracted. Detection models were also developed for HSR, and they utilized supervised machine learning algorithms on many prominent features. It was proven that this framework's detection method achieved high accuracy and less false positive rate for detecting HSR in the early phases of ransomware. These methods have also been validated with an extensive experimental evaluation to show their effectiveness. Lastly, the capabilities of the proposed method were compared to the results of previous work, other classifiers, and VirusTotal. The framework itself is broken into 3 phases. The first phase includes gathering ransomware and benign data from a variety of sources. Once gathered, the data is checked and labeled under a particular malware family using VirusTotal software. The second phase analyzes the samples using a Cuckoo sandbox and generates a report in JSON format of its findings. Within the sandbox, log files are submitted through pre-processing tasks, and when finished, the relevant features are extracted to get valuable feature sets. Those features are applied against the term frequency and inverse document frequency (TF-IDF) algorithm for feature selection. The last phase uses supervised machine learning algorithms Support vector machine (SVM), and Artificial Neural Network (ANN) to derive statistics of the data.

Three different experimental evaluations were conducted to measure the performance of the framework. The first evaluation used the train-test splitting method which divides the whole data set into training and testing data. The dataset was split randomly with a uniform distribution of 80:20 ratio as training and testing, respectively. The experimental results of ANN showed an accuracy of 0.958 with 0.101 false positive rates, while SVM presented a higher false positive of 0.109 compared to ANN and an accuracy of 0.932. In the train-test splitting method, the data can become obscure and irrelevant, which is why the second experimental evaluation is used. The 10-Fold cross-validation technique can prevent the overfitting problem and estimate the effectiveness of the detection models. The best accuracy obtained by SVM is by presenting 0.982 of area under the curve (AUC) with less than 0.035 of false positive rate [27]. It is important to examine the ability of the classifiers to distinguish the ransomware from benign samples. Therefore, precision and recall are applied to both datasets and presents 0.945 and 0.942 respectively. SVM also showed better accuracy of 0.952 when compared to MLP that showed 0.945 of detection rate and 0.036 of false positive rates.

The last evaluation used selected subset features which eliminate the redundant and irrelevant features and reduces the dimensionality of the dataset. The features were divided into seven subset features (top20, top30, top40, top50, top60, top70, top80) by considering their importance and ranking based on phase 2 processing. The results of the experiment demonstrated that ANN showed the highest accuracy of 98.79% when the top30 of the feature set was used as training and testing [27]. However, this classification

accuracy had dramatically decreased to 95.63% when the top20 of the feature set was used. The best model of SVM presented an accuracy of 97.6% when top40 of the feature was applied while training the model [27]. ANN and SVM both had low classification accuracy when the top 80 of the feature set was used to train and test, which indicates that more features do not improve the performance of the classifiers.

B. DNAACT-RAN

DNAact-Ran is a Machine Learning-based digital DNA sequencing engine used in classifying and detecting ransomware. It uses an active ML approach for sequencing its digital DNA and detects ransomware in three key process steps: Feature Selection, DNA Sequence Generation and Ransomware Detection. Feature selection removes irrelevant features and reduces storage and computational cost. It is considered the most important process of machine learning. Multi-Objective Grey Wolf Optimization (MOGWO) and Binary Cuckoo Search (BCS) algorithms are used to select the relevant features from the collected dataset. MOGWO uses a grid and archive approach with selecting the most dominant features, while BCS uses a heuristic search approach to determine its features [28]. Figure 11 gives the complete architecture of DNAact-Ran.

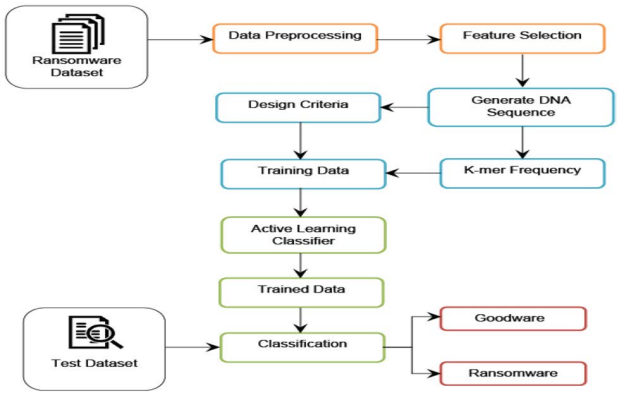

FIGURE 11. DNAact-Ran architecture.

In the digital DNA sequence generation step, a new dataset is used to generate the digital DNA sequence after the feature selection process is completed. The design constraint of digital DNA is then computed, and the k-mer frequency vector is generated for the DNA sequence. A new dataset is then generated for the ransomware detection training phase based on those calculations. A synthetic DNA representation of a digital artifact is used because it does not represent the content of biological DNA. DNA is represented computationally by character strings containing only the characters A, G, C and T, therefore, Pedersen et al. [29] created a reversible translation of the byte sequence of a digital artifact which mapped binary pairs into those string characters [28]. The DNA sequence design is used as an approach of control and DNAact-Ran uses 3 constraints (Tm, GC Content, and AT_GC Ratio) to avoid inadequate data.

The last step used by DNAact-Ran is the actual ransomware detection step. The dataset is trained using an active learning classifier. Once this process is done, digital DNA sequences are randomly generated from the test data where it is classified as good-ware or ransomware. Finally, the ransomware family is detected using the traditional ML classification algorithms. Machine Learning applications struggle with the amount of time and effort required to interpret large amounts of data sets that are required for supervised learning in the process of training a high-accuracy classifier. To solve this issue, active learning has been proposed and designed to decrease the cost by finding data points to be used by the learning algorithm. The active learning algorithm uses three parameters for determining accuracy: Smoothing Parameter (SP), Regularization Parameter (RP), and Learning Rate (LR). The data was tested against traditional ML algorithms and the experiment showed a 78.5% detection accuracy for Naïve Bayes, 75.8% for Decision Stump, 83.2% for AdaBoost, and 87.9% for the proposed solution [28]. This experiment partially proves that active learning classifiers are better at detecting ransomware more efficiently.

C. RANDS

RANDS is a windows-based anti-ransomware tool that implements a multi-tier framework with ransomware traits archive and machine learning algorithms. The architecture of RANDs is classified in three tiers: Analysis, Learning, and Detection. The first tier checks the traits of different ransomware families in a recursive test routine in a virtual test environment. The virtual environment is utilized to avoid the severe damage and malfunctions of ransomware on the platform system. The second tier studies the combined traits from the archive using a hybrid machine learning algorithm to generate the classification model. The generated classification models will be used to detect any suspicious activity against the actions or traits in the last tier too. The last tier applies the classification model to detect any unknown ransomware variant via a computer scan [30] which alerts the system's user that a ransomware is going to possibly infect the system. RANDS machine learning algorithm uses a hybrid approach. It uses both Decision Tree and Naïve Bayes decision functions due to their pruning margins for more accurate categorization. The Decision Tree algorithm generates its predictions of the traits within a tree structure of nodes, leaves, and branches throughout the pruning and tree building process [31]. The Naïve Bayes algorithm is used for predicting the actual category of the overlooked traits in the vague nodes of Decision Tree. The Bayes' probabilistic theorem is used when a trait that goes unclassified cannot be classified.

To test and demonstrate whether RANDS could adapt the zero-day ransomware variants and their corresponding families, performance metrics including Detection Accuracy Rate, Mistake Rate, Miss Rate, and Elapsed Time along

with plots of ROC curve have been utilized through experiments [31]. The ten-fold cross evaluation routine inferred that RANDS could manifest its adaptive and effective classification against zero-day ransomware. That was accredited by the hybrid machine learning approach that RANDS utilized. RANDS implemented ransomware traits to distinguish different ransomware families and identify their related variants. However, the performance trend line reported at certain days showed erratic behavior caused by the qualities of ransomware families and their corresponding variants that might be varied. Results showed a 96.27% average accuracy rate and 1.32% average mistake rate throughout the real-time assessment [31].

D. RATAFIA

An unsupervised detection framework RATAFIA uses a DNN architecture and Fast Fourier Transformation to develop a highly accurate, fast, and reliable solution to ransomware detection using minimal trace-points. The advantage of using an unsupervised technique is that the learning process does not require a labeled dataset, which is often difficult to obtain considering the occurrences of several newer unknown varieties of ransomware. RATAFIA specifically was created to learn the behavior of a system under observation with the statistics obtained from a cluster of Hardware Performance Counter (HPC) events [32]. The first phase of RATAFIA tests its robustness and uses an analysis in the presence of expensive SPEC benchmarks. It is observed that the execution behaviors of HPC events are significantly different from normal observations, and the sequences of time-series data in which RATAFIA processes are treated as being malicious due to reaching computational thresholds. However, these are simply the SPEC programs creating false positive errors. The second phase uses FFT to try to eliminate the false positive by changing the HPC values from time domain to frequency domain. This is done to understand the repetitive pattern within the values because ransomware executable runs encryption repeatedly on multiple files. The entire detection procedure does not need any template of the malicious process from beforehand. Instead, it thrives on an anomaly detection procedure to detect infectious ransomware in as less as 5 seconds with almost zero false positives, using frequency analysis [32]. The proposed detection method works on any platform having HPCs. However, the tunable hyper-parameters will be different for different systems. The determination of values for these parameters is a one-time process, which will be accomplished during the training of autoencoders. RATAFIA uses a template of the normal system behavior in terms of HPC values to train the autoencoders. The advantage of using HPCs is that they are difficult to tamper with. While one may increase some HPC values by a program, it is difficult to reduce the HPC values without explicitly targeting the HPC registers.

E. RANSOMWALL

RansomWall protects against cryptographic ransomware using a layered defense system. The features that are generated during the sample's execution aid in organizing the layers in a computational order. It is implemented solely for Windows operating system. RansomWall's architecture models that of a hybrid approach utilizing a joint static and dynamic analysis to compute values of the selected feature set [33]. The Machine Learning Engine is used to develop a generalized model which is effective against zero-day ransomware attacks. It takes feature values collected by static, dynamic and trap layers as input and classifies the executable as ransomware or benign. The engine is trained offline using supervised algorithms and the training data consists of feature values with ransomware and benign labels. The Trained Machine Learning Engine then uses the learned model to classify executables in real-time based on input feature values. The following supervised machine learning algorithms are evaluated based on performance: Logistic Regression, SVM, ANN, Random Forests, and Gradient Tree Boosting.

The ransomware sample set has a 12-Fold Cross Validation performed on it. In each test run, Machine Learning Layer is trained on all samples from 11 out of 12 Cryptographic ransomware families and 221 out of 442 samples from benign software [33]. The learned model is tested against remaining benign samples on the evaluation setup and all samples from the last ransomware family. Since most of the successful ransomware attacks are zero-day intrusions, this process of evaluation is selected, with samples from an entirely new ransomware family or its upgraded variant. During the evaluation, the functionality of the File Backup Layer is verified to check if the files are correctly backed up for suspicious processes after receiving classification output from the Machine Learning Layer.

The metrics show the best results with Gradient Tree Boosting Algorithm. RansomWall attains a detection rate of 98.25% with near-zero false positives with this algorithm. The Gradient Tree Boosting algorithm provides effective handling of heterogeneous data, high predictive power and robustness to outliers resulting in high performance [GG]. Analysis of false negatives show that two ransomware samples abruptly terminated after encrypting only a few files. As the resulting file system activity is reduced, samples do not get detected. Limited file system activity is leading to false negatives due to the low number of such files on the user's system. The rest of the false negatives come from decision boundary errors.

F. CRYPTOKNIGHT

CryptoKnight was built to classify cryptographic primitives in compiled binary executables using the Dynamic Convolutional Neural Network (DCNN) algorithm. It introduces a learning system that can easily integrate new samples through the scalable synthesis of customizable cryptographic algorithms. CryptoKnight's architecture is intended to limit human interaction, allowing the structure of an effective model at run-time [34]. The entire system is comprised of three stages:

1. Procedural generation guides the synthesis of unique cryptographic binaries with variable obfuscation and alternate compilation.
2. Assumptions of cryptographic code aid the discrimination of diagnostics from the dynamic analysis of synthetic or reference binaries to build an 'image' of execution.
3. A DCNN fits variable-length matrices for ease of training and the immediate classification of new samples.

CryptoKnight was tested on many applications using non-library linked functionality and analysis showed that it is a viable solution that can quickly learn from new cryptographic execution patterns to classify unknown software [34]. CryptoKnight also demonstrated that it could classify results faster compared to that of previous methodologies and is considerably re-usable. At a 96 % accuracy rate, CryptoKnight confirmed that cryptographic primitive classification in compiled binary executables could be successfully achieved using a DCNN algorithm.

G. ELDERAN

In 2016, EldeRan was developed to identify the most significant ransomware features and use them to detect ransomware [35]. The framework is based on the observation of actions or events that typically occur within ransomware and goodware samples in its early stages. Ransomware and goodware sample datasets are dynamically analyzed in a sandbox environment first. From the two datasets, EldeRan retrieves and analyses the following classes of features: *Windows API calls*, *Registry Key Operations*, *File System Operations*, *the set of file operations performed per File Extension*, *Directory Operations*, *Dropped Files*, and *Strings*. Other than *Strings,* the rest of the features are collected while dynamically analyzing the ransomware. Once the monitoring phase has completed, the Mutual Information criterion [37], a feature selection algorithm is used to choose the ones that are most relevant. The feature selection process is not always used in machine learning algorithms, but for EldeRan, it helps with performance and provides more competence in the algorithm [38]. Finally, the matrices containing these features are used in a *Regularized Logistic Regression* classifier. This classifier will return ransomware or goodware once detected and is also run online on a PC to classify new samples, which can come from infected websites or multiple infected vectors. The training set is analyzed offline and completed within minutes in the sandbox environment while new applications are classified at run-time through an online classifier, which is also fast [36].

EldeRan was conducted in three different experiments. The first experiment tested how performant EldeRan was compared to two other classifiers, SVM (Support Vector Machine) and NB (Naive Bayes) [36]. It was determined that both SVM and EldeRan outperformed NB, and EldeRan slightly edged SVM. These metrics were evaluated against the AUC (Area Under the Curve) using between 50 and 1500 features, all supported by MI criterion. A structure of 100 random splits was introduced for each explored value, where 80% of the samples were used for training and 20% as test samples. It is determined that all three classifiers show maximum performance at 400 features, and the accuracy showed no improvement beyond that number [36]. The second experiment observes the performance of the previous classifiers along with VirusTotal. The top 400 features were used for the original three classifiers, and the test covered all the methods averaging the results over 100 independent train/test splits with 80% of samples for training. It is determined VirusTotal shows better performance when compared to the other algorithms, although EldeRan is just slightly behind. The last experiment tests how effective EldeRan can detect new families of ransomware. For new families of ransomware, it is common for them to share the same characteristics and goals of previous classes [37]. Datasets were clustered into 11 classes using their known family name, since the naming conventions of the antivirus (AV) vendors are not always consistent or compatible amongst them. Two cases were considered by selecting the top 100 and 400 features according to the MI criterion. For eight of the ransomware families the detection rate is above 90% and for ten families the detection rate is above 80%, both occurring when using 100 features. When using 400 features, the detection rates become worse with only five families achieving 90% and eight families achieving 80%. The average detection rate is higher (93.3%) when using only the top 100 features than in the case of using 400 features (87.1%) [36]. Figure 12 below shows an average ROC of the test samples.

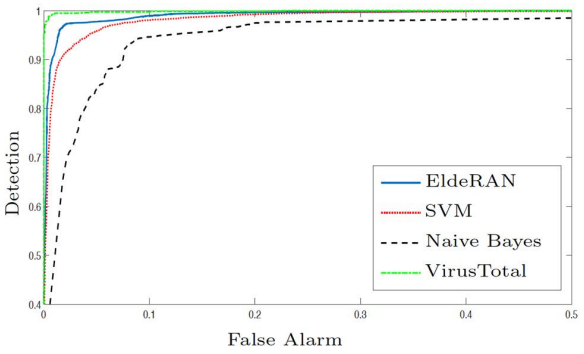

FIGURE 12. Average ROC for the test samples over 100 random splits for EldeRan, the SVM, NB, and VirusTotal [36].

Some limitations of EldeRan existed. If the ransomware remained silent or waited for user interaction within the sandbox environment, EldeRan does not properly extract the ransomware, therefore, goes undetected. Secondly, no other applications were running within the sandbox environment, which was purposely done to eliminate the ransomware checks to evade detection. Lastly, the original ransomware and goodware data samples were limited because EldeRan could not process empty API calls efficiently during the dynamic analysis phase. This reduced the dataset by half the samples. Ultimately, EldeRan can only detect ransomware once the infection occurs [39].

H. DRTHIS

DRTHIS was developed to determine ransomware from goodware and to identify their families. It uses LSTM and CCN deep learning techniques for classification. Based on the application sequence of activities, a binary classifier, a Deep Feature Extractor (DFE), and a One Class Classifier (OCCs) are all used for hunting ransomware samples and identifying their families. The system records executed events when a user launches an application, and within the first 10 seconds of application execution, the captured sequence is transformed to detect if a given sample is considered ransomware [40]. Ransomware samples that have been identified are sent to the system to categorize its family. During the threat intelligence phase, DFE is used to extract a vector to feed the OCC, and it contains the pre-trained model LSTM or CNN. This is the step that produces the family. DRTHIS does a data transformation task to transform textual sequences of events into a numerical form. Then, the combining and labeling component combines input datasets into one integrated dataset suitable for our deep learning tasks. It is notable that combining, and labeling creates two separate datasets with the same samples but different class labels [40].

DRTHIS uses both ransomware samples from new families and unforeseen benign applications. The created OCC generated 24% wrong prediction when it classified 16 out of 66 Locky samples as goodware. DRTHIS also wrongly classified one Cerber sample and two TeslaCrypt samples as a new family of ransomware [40]. DRTHIS takes advantage of One Class Classifier to determine if a sample belongs to a known family of ransomware or whether it belongs to a new family. Samples from CryptoWall, TorrentLocker and Sage families are used for evaluating the performance of the system against samples from unforeseen families. DRTHIS correctly recognized these three families as a new family without any conflict with the trained families. 99% of CryptoWall, 75% of TorrentLocker and 92% of Sage samples are correctly detected as samples from a new family of ransomware. DRTHIS wrongly classifies 1 CryptoWall sample (1%), 4 TorrentLocker samples (14.2%), and 1 Sage sample (1.2%) as goodware. DRTHIS identifies 2 TorrentLocker samples (7%) and 1 Sage sample (1.2%) as Cerber samples. Three samples (3.8%) of Sage and 1 sample (3.5%) of TorrentLocker are also detected as TeslaCrypt [40]. Due to the fast classification of new instances, DRTHIS can be considered as a basic method in the cyber security industry for implementing new threat hunting and intelligence tools.

I. A COMPARATIVE ANALYSIS OF FRAMEWORKS

Section 3 has provided details of several different frameworks. The below table captures those frameworks and consolidates data points and metrics for quick references. The table is composed of the author's reference of work, the name of the framework, the dataset quantity, the number of features (if applicable), the type(s) of machine learning algorithms used, the year it was created, the results, and the challenges each framework faced. The first important factor about these frameworks is the datasets. Most have been manipulated based on feature selection to improve the accuracy of each of their respective models. However, the framework RATAFIA is very different from the others. For example, it does not follow the traditional procedures of using ML algorithms because it strictly factors in performance based on unsupervised learning.

Another observation between these frameworks in Table 1 below is the ML algorithms used. Each framework in this research was specifically chosen to bring about nuances in the related field, therefore, the approach of each framework is different and all use different algorithms. As this paper is collectively bringing about research geared towards ransomware detection, providing a variety of different frameworks that are not producing the same results gives better insight as to what is happening in this area of research study. Lastly, this paper gives awareness as to how ransomware detection was conducted years ago versus how it has been advanced and improved in recent years.

IV. FUTURE OF RANSOMWARE

Trends show that ransomware attacks will continue to surge in 2022 and will continue to be a top threat for all sectors [41]. Ransomware-as-a-Service, which is a powerful asset, will also boost malicious attacks on end users, as it does not require anyone to have any technical knowledge. An increase in the usage of Initial Access Brokers (IAB) is also projected to peak, as they gain access to a victim's network and then sell it to open ransomware markets for profit. The Covid 19 pandemic will continue contributing to more ransomware related activities as it has caused a drop in employment. Many companies currently do not have the workforce to increase their cyber security measures or provide awareness, and mitigation tactics are lacking. Financial impacts have reduced funds for companies to invest in such software and are steering smaller companies in the direction of bypassing security measures altogether. It is also noted that ransomware operators are likely to intensify the ways that they pressure victims into paying ransoms by contacting customers of interest, engaging in media sources and journalists, or simply calling victims directly [41].

Defenders of ransomware will have to stay ahead of the advancement of ransomware schemes. For instance, IoT devices and 5G networks create many loopholes for ransomware intrusions. Integrating security technology, secure design principles, and governance at each phase of an organization's IoT and 5G landscapes will also need to be set forth in each sublayer. Quantum computing is also being used in the detection of ransomware and is projected to reach maturity within the next few years [42]. Detection algorithms could be enriched with quantum technology to expedite the identification and decryption of encrypted malware. Defenders would also be able to use quantum computers to decrypt malice communications using proactive monitoring. Leveraging quantum could also disrupt the flow of ransomware in its attack sequences once it has encrypted its targeted file.

TABLE 1. ConsolidateD view of frameworks.

Author	Framework	Dataset	Features	ML Algorithm	Year	Results	Challenges
[35]	EldeRan	582 malicious 942 benign	50 - 1500	Regularized Logistic Regression	2016	96.3% accuracy	detects after the system is infected unable to detect silent ransomware
[40]	DRTHIS	879 samples (219 benign)	Multiple, specifically for hunting	CNN LSTM	2018	97.2% accuracy	considered as a mediocre system for executing new threat hunting techniques
[34]	CryptoKnight	750 original samples 150 trained samples	N/A	DCNN	2018	96% accuracy	can only classify known samples
[33]	RansomWall	1,016 samples (442 benign)	Multiple across 5 different stages	Gradient Tree Boosting	2018	98.25% accuracy	Windows OS only
[32]	RATAFIA	4 ransomware programs	N/A	DNN FFT	2019	ransomware detection within 5 seconds	unable to pinpoint HPC events to one individual process being executed determining disk encryption processes from malicious ones
[30]	RANDS	10,000 samples (500 benign)	N/A	Naïve Bays Decision Tree	2019	96.27% average accuracy	Windows-based platform only
[28]	DNAact-Ran	300 out of 1,524 used (150 malicious) (150 benign)	16,383 out of 30,970 used	MOGWO BCS active learning	2020	87.9% accuracy	Partially proves active ML algorithms detect ransomware better must improve performance by constantly adjusting the active learning algorithm parameters
[27]	behavior-based	1,254 malicious samples 1,308 benign samples	3,930 that was relevant	SVM ANN TF-IDF	2020	98.7% accuracy	More added features decrease performance

In [65], the authors present a new type of framework called Detection Avoidance Mitigation (DAM). It can handle classification, detection, and mitigation all in one go. Its architecture consists of typical detection techniques using static and dynamic analysis, avoidance techniques such as system updates and patches, and mitigation techniques such as reverse engineering. DAM evaluated different combat strategies for preventing ransomware attacks and widespread financial losses, proving that avoidance techniques are the most desirable in protecting users and organizations from ransomware.

Lastly, the first blockchain-based ransomware schemes were introduced in [66]. The authors focused on smart contracts to contribute to the paying of single files or refunding the ransom payment back to the victim if the decryption keys were not sent within a reasonable time. The results of this research showed no practical countermeasures when using public blockchains, therefore, more research interests in the area is needed as this shows a concern.

V. CONCLUSION

In recent years, ransomware has continuously been a top topic in cybersecurity and attacks are now taking place not only on individuals but organizations as well. Ransomware has evolved from elementary scareware and locker related user interfaces, to cryptographic and fileless ransomware. In this paper, we provide a comprehensive survey of ransomware types, common frameworks that are used to detect ransomware, and the ML algorithms in which they use. A detailed list of all pertinent information is gathered and arranged in a table. Though other research papers have provided reviews with similar concepts, these surveys have not captured the explicit details in one place as this research.

By collecting such material and providing a comparative study, this paper provides a means for others to foresee an area of interest and investigate parts where improvements can be made due to poor results or limitations. Ultimately, this paper can provide direction to those who are looking to utilize one of the mentioned frameworks for advancement in future work.

REFERENCES

[1] L. Abrams. (2020). *SunCrypt Ransomware Shuts Down North Carolina School District*. Accessed: Jan. 11, 2021. [Online]. Available: https://www.bleepingcomputer.com/news/security/suncrypt-ransomware-shuts-down-north-carolina-schooldistrict/

[2] BBC News. (2020). *Northumbria University Hit by Cyber Attack*. Accessed: Jan. 11, 2021. [Online]. Available: https://www.bbc.com/news/uk-england-tyne-53989404

[3] B. Fraga. (2013). *Swansea Police Pay $750 'Ransom' After Computer Virus Strikes*. Accessed: Jan. 11, 2021. [Online]. Available: https://www.heraldnews.com/x2132756948/Swansea-police-pay-750-ransom-after-computer virus-strikes

[4] L. Freedman. (2020). *Ransomware Attacks Predicted to Occur Every 11 Seconds in 202 With a Cost of $20 Billion*. Accessed: Jan. 11, 2021. [Online]. Available: https://www.dataprivacyandsecurityinsider.com/2020/02/ransomwareattacks-predicted-to-occur-every-11-seconds-in-2021-with-a-cost-of-20-billion/

[5] K. Savage, P. Coogan, and H. Lau. (2015). *The Evolution of Ransomware*. [Online]. Available: https://its.fsu.edu/sites/g/files/imported/storage/images/information-security-and-privacy-office/the-evolution-of-ransomware

[6] I. Segun, B. I. Ujioghosa, S. O. Ojewande, F. O. Sweetwilliams, S. N. John, and A. A. Atayero, "Ransomware: Current trend, challenges, and research directions," in *Proc. World Congr. Eng. Comput. Sci.*, 2017, pp. 169–174.

[7] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "UNVEIL: A large-scale, automated approach to detecting ransomware," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 757–772.

[8] D. Y. Huang, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A. C. Snoeren, and D. McCoy, "Tracking ransomware end-to-end," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 618–631.

[9] L. Abrams. (Jan. 4, 2016). *Ransom32 is the First Ransomware Written in Javascript*. BleepingComputer. Accessed: Jan. 12, 2021. [Online]. Available: https://www.bleepingcomputer.com/news/security/ransom32-is-the-first-ransomware-written-in-javascript/

[10] KnowBe4. (2021). *Ransom32 Ransomware-as-a-Service*. Accessed: Jan. 12, 2021. [Online]. Available: https://www.knowbe4.com/ransom-32-ransomware-as-a-service

[11] S. Sjouwerman. (Feb. 5, 2019). *First Javascript-Only Ransomware as a Service Poses New Threat*. TechBeacon. Accessed: Jan. 12, 2021. [Online]. Available: https://techbeacon.com/security/first-javascript-only-ransomware-service-poses-new-threat

[12] M. J. Schwartz and R. Ross. (Jun. 20, 2016). *Latest Ransomware Relies on JavaScript*. Bank Information Security. Accessed: Dec. 2, 2021. [Online]. Available: https://www.bankinfosecurity.com/latest-ransomware-relies-on-javascript-a-9212

[13] (Jun. 16, 2016). *New RAA Ransomware Uses Only JavaScript to Infect Computers*. Accessed: Jan. 12, 2021. [Online]. Available: https://www.trendmicro.com/vinfo/mx/security/news/cybercrime-and-digital-threats/new-raa-ransomware-uses-only-javascript-to-infect-computers

[14] J. Tolbert. (2020). *Malicious Actors Exploiting Coronavirus Fears*. Accessed: Jan. 12, 2021. [Online]. Available: https://www.kuppingercole.com/blog/tolbert/maliciousactors-exploiting-coronavirus-fears

[15] Brooke Crothers. (2020). *Apps Designed to Track COVID-19 Might be Full of Ransomware, Report Says*. [Online]. Available: https://www.foxnews.com/tech/apps-track-covid-19-full-ransomware

[16] Acronis. (2020). *Digital CoronaVirus: Yet Another Ransomware Combined With Infostealer*. Accessed: Jan. 12, 2021. [Online]. Available: https://www.cbronline.com/news/tesla-cyber-attack

[17] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). New York, NY, USA: Springer-Verlag, 2006.

[18] N. Chauhan. (Jan. 2020). *Decision Tree Algorithm, Explained*. KDnuggets. Accessed: Jan. 22, 2021. [Online]. Available: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[19] N. Donges. (Jun. 22, 2021). *A Complete Guide to the Random Forest Algorithm*. Accessed: Jan. 22, 2021. [Online]. Available: https://builtin.com/data-science/random-forest-algorithm

[20] O. Mbaabu. (Dec. 11, 2020). *Introduction to Random Forest in Machine Learning*. Accessed: Jan. 22, 2021. [Online]. Available: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

[21] J. Brownlee. (Jul. 7, 2021). *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. Machine Learning Mastery. Accessed: Jan. 24, 2021. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[22] S. Saxena. (Mar. 16, 2021). *Introduction to Long Short Term Memory (LSTM)*. Analytics Vidhya. Accessed: Jan. 24, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[23] S. Ray. (Sep. 11, 2017). *6 Easy Steps to Learn Naive Bayes Algorithm With Codes in Python and R*. Analytics Vidhya. Accessed: Jan. 24, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[24] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Mach. Learn.*, vol. 29, pp. 103–130, Nov. 1997.

[25] C. Li. (2016). *A Gentle Introduction to Gradient Boosting*. Accessed: Jan. 26, 2021. [Online]. Available: https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

[26] R. Gandhi. (Jul. 7, 2018). *Support Vector Machine—Introduction to Machine Learning Algorithms*. Accessed: Jan. 26, 2021. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learningalgorithms-934a444fca47

[27] Y. A. Ahmed, B. Kocer, and B. A. S. Al-rimy, "Automated analysis approach for the detection of high survivable ransomware," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 5, pp. 2236–2257, 2020, doi: 10.3837/TIIS.2020.05.021.

[28] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry, and Y. Nam, "A digital DNA sequencing engine for ransomware detection using machine learning," *IEEE Access*, vol 8, pp. 119710–119719, 2020.

[29] J. Pedersen, D. Bastola, K. Dick, R. Gandhi, and W. Mahoney, "Blast your way through malware analysis assisted by bioinformatics tools," in *Proc. Int. Conf. Secur. Manage.*, 2012, p. 1.

[30] H. Zuhair and A. Selamat, "RANDS: A machine learning-based anti-ransomware tool for Windows platforms," in *Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques*, vol. 318, 2019.

[31] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on Windows platforms," *J. Inf. Secur. Appl.*, vol. 40, pp. 44–51, Jun. 2018.

[32] M. Alam, S. Bhattacharya, S. Dutta, S. Sinha, D. Mukhopadhyay, and A. Chattopadhyay, "RATAFIA: Ransomware analysis using time and frequency informed autoencoders," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2019, pp. 218–227, doi: 10.1109/HST.2019.8740837.

[33] S. K. Shaukat and V. J. Ribeiro, "RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning," in *Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2018, pp. 356–363.

[34] G. Hill and X. Bellekens, "CryptoKnight: Generating and modelling compiled cryptographic primitives," *Information*, vol. 9, no. 9, p. 231, Sep. 2018.

[35] Z.-G. Chen, H.-S. Kang, S.-N. Yin, and S.-R. Kim, "Automatic ransomware detection and analysis based on dynamic API calls flow graph," in *Proc. Int. Conf. Res. Adapt. Convergent Syst.*, Sep. 2017, pp. 196–201, doi: 10.1145/3129676.3129704.

[36] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," 2016, *arXiv:1609.03020*.

[37] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Cham, Switzerland: Springer, 2015, pp. 3–24, doi: 10.1007/978-3-319-20550-2_1.
[38] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *J. Mach. Learn. Res.*, vol. 7, pp. 2721–2744, Dec. 2006.
[39] G. Cusack, O. Michel, and E. Keller, "Machine learning-based detection of ransomware using SDN," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2018, pp. 1–6, doi: 10.1145/3180465.3180467.
[40] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K.-K. R. Choo, and D. E. Newton, "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Gener. Comput. Syst.*, vol. 90, pp. 94–104, Jan. 2019.
[41] QuoIntelligence. (Jan. 18, 2022). *Ransomware is Here to Stay and Other Cybersecurity Predictions for 2022*. Accessed: Jan. 31, 2021. [Online]. Available: https://quointelligence.eu/2022/01/ransomware-and-other-cybersecurity-predictions-for-2022/
[42] D. Golden and K. Norton. (2021). *Defending Against Ransomware in an Age of Emerging Technology*. Deloitte. Accessed: Jan. 31, 2021. [Online]. Available: https://www2.deloitte.com/us/en/pages/risk/articles/defending-against-ransomware.html
[43] L. Simonovich. (Jan. 15, 2020). *Are Utilities Doing Enough to Protect Themselves From Cyberattack?*. World Economic Forum. Accessed: Apr. 4, 2021. [Online]. Available: https://www.weforum.org/agenda/2020/01/are-utilities-doing-enough-to-protect-themselves-from-cyberattack/
[44] APWG. (May 11, 2020). *Phishing Activity Trends Report in Q1 of 2020*. Accessed: Apr. 4, 2021. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2020.pdf
[45] Q. Chen and R. A. Bridges, "Automated behavioral analysis of malware: A case study of WannaCry ransomware," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 454–460, doi: 10.1109/ICMLA.2017.0-119.
[46] (May 22, 2017). *WannaCry Ransomware Campaign Exploiting SMB Vulnerability*. Accessed: Apr. 4, 2021. [Online]. Available: https://cert.europa.eu/static/SecurityAdvisories/2017/CERT-EU-SA2017-012.pdf
[47] M. Akbanov, V. G. Vassilakis, and M. D. Logothetis, "WannaCry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms," *J. Telecommun. Inf. Technol.*, vol. 1, no. 2019, pp. 113–124, Apr. 2019.
[48] L. J. Trautman and P. Ormerod, "Wannacry, ransomware, and the emerging threat to corporations," *SSRN Electron. J.*, vol. 86, p. 503, Jan. 2018, doi: 10.2139/ssrn.3238293.
[49] S. Jones and T. Bradshaw. (May 14, 2017). *Global Alert to Prepare for Fresh Cyber Attacks*. Accessed: Apr. 4, 2021. [Online]. Available: https://www.ft.com/content/bb4dda38-389f-11e7-821a-6027b8a20f23
[50] M. V. Liy. (May 15, 2017). *Putin Culpa a Los Servicios Secretos de EE UU Por el Virus 'WannaCry' Que Desencaden? el Ciberataque Mundial*. Accessed: Apr. 4, 2021. [Online]. Available: https://elpais.com/internacional/2017/05/15/actualidad/1494855826_022843.html
[51] S. K. Sahi, "A study of wannacry ransomware attack," *Int. J. Eng. Res. Comput. Sci. Eng.*, vol. 4, no. 9, pp. 5–7, 2017.
[52] R. Collier, "NHS ransomware attack spreads worldwide," *Can. Med. Assoc. J.*, vol. 189, no. 22, pp. E786–E787, 2017.
[53] JavaScript|MDN. (Feb. 18, 2022). *JavaScript Language Resources—JavaScript: MDN*. Accessed: Apr. 4, 2021. [Online]. Available: https://developer.mozilla.org/enUS/docs/Web/JavaScript/Language_Resources
[54] J. Gerend. (Mar. 3, 2021). *Wscript*. Microsoft Docs. Accessed: Apr. 4, 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/wscript
[55] T. McIntosh, A. S. M. Kayes, Y.-P.-P. Chen, A. Ng, and P. Watters, "Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–36, Dec. 2022, doi: 10.1145/3479393.
[56] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A survey on ransomware: Evolution, taxonomy, and defense solutions," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 1–37, Jan. 2022, doi: 10.1145/3514229.
[57] CIS Security. (2019). *Fall 2019 Threat of the Quarter: Ryuk Ransomware*. Accessed: Apr. 5, 2021. [Online]. Available: https://www.cisecurity.org/white-papers/fall-2019-threat-ofthe-quarter-ryuk-ransomware/
[58] H. Ke, H. Wu, and D. Yang, "Towards evolving security requirements of industrial internet: A layered security architecture solution based on data transfer techniques," in *Proc. Int. Conf. Cyberspace Innov. Adv. Technol.*, New York, NY, USA, Dec. 2020, pp. 504–511, doi: 10.1145/3444370.3444620.
[59] Trend Micro. *What is Ryuk Ransomware*. Accessed: Apr. 5, 2021. [Online]. Available: https://www.trendmicro.com/en_us/what-is/ransomware/ryuk-ransomware.html
[60] WannaCry Ransomware. (May 15, 2017). *WannaCry Ransomware—LogRhythm*. Accessed: Apr. 14, 2021. [Online]. Available: https://logrhythm.com/blog/wannacry-ransomware/
[61] A. Kujawa. (Jan. 8, 2019). *Ryuk Ransomware Attacks Businesses Over the Holidays*. Malwarebytes Labs. Accessed: Apr. 14, 2021. [Online]. Available: https://blog.malwarebytes.com/cybercrime/malware/2019/01/ryuk-ransomware-attacks-businesses-over-the-holidays/
[62] R. Nimbalkar. (Jul. 13, 2021). *Decision Tree Algorithms-Machine Learning*. Accessed: Apr. 14, 2021. [Online]. Available: https://medium.com/appengine-ai/decision-tree-algorithms-machine-learning-9e2e8cadfcae
[63] S. India. (Jul. 4, 2020). *Hands-on Training With Machine Learning Algorithms: Decision Tree and Random Forest*. Springboard Blog. Accessed: Apr. 14, 2021. [Online]. Available: https://in.springboard.com/blog/machine-learning-algorithms-decision-tree-random-forest/
[64] G. Van Houdt, C. Mosquera, and G. Npoles, "A review on the long short-term memory model," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, 2020, doi: 10.1007/s10462-020-09838-1.
[65] A. Kapoor, A. Gupta, R. Gupta, S. Tanwar, G. Sharma, and I. E. Davidson, "Ransomware detection, avoidance, and mitigation scheme: A review and future directions," *Sustainability*, vol. 14, no. 1, p. 8, Dec. 2021.
[66] O. Delgado-Mohatar, J. M. Sierra-Cámara, and E. Anguiano, "Blockchain-based semi-autonomous ransomware," *Future Gener. Comput. Syst.*, vol. 112, pp. 589–603, Nov. 2020.

DARYLE SMITH was born in Lenoir, NC, USA, in 1985. He received the B.S. and M.S. degrees in computer science from Winston-Salem State University. He is currently pursuing the Ph.D. degree in computer science with North Carolina A&T State University. Since 2010, he has been starting his IT career as a Software Engineer and has been involved in every aspect of e-commerce since. He is currently an E-Commerce Architect with the Peapod Digital Laboratories, Salisbury, NC headquarters.

SAJAD KHORSANDROO received the Ph.D. degree in computer science from The University of Texas at San Antonio, in 2019. Currently, he is an Assistant Professor of computer science at North Carolina A&T State University, where he is also an Associate Director of the Cyber Defense and AI Laboratory. He has already secured $1.7M in funds from NSF, DoD, Palo Alto Networks, and Carolina Cyber Center. His current research interests include the application of AI/ML in cyber security, next-generation network infrastructures, cloud computing, and secure cyber physical systems.

KAUSHIK ROY is currently a Professor and the Interim Chair of the Department of Computer Science, North Carolina A&T State University (NCAT). He has over 140 publications, including 35 journal articles and a book. His current research interests include cybersecurity, cyber identity, biometrics, machine learning (deep learning), data science, the IoT, cyber-physical systems, and big data analytics. His research is funded by the National Science Foundation (NSF), Department of Defense (DoD), National Security Agency (NSA), and Department of Energy (DoE). He is the Director of the Center for Cyber Defense (CCD). He also directs the Cyber Defense and AI Laboratory.