IoT

MDPI

Review

A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques

Damien Warren Fernando [1,*], Nikos Komninos [1] and Thomas Chen [2]

[1] Department of Computer Science, City, University of London, London EC1V 0HB, UK; Nikos.Komninos.1@city.ac.uk

[2] Department of Electrical and Electronic Engineering, City, University of London, London EC1V 0HB, UK; Tom.Chen.1@city.ac.uk

* Correspondence: Damien.Fernando.1@city.ac.uk

Received: 29 September 2020; Accepted: 11 December 2020; Published: 15 December 2020

Abstract: This survey investigates the contributions of research into the detection of ransomware malware using machine learning and deep learning algorithms. The main motivations for this study are the destructive nature of ransomware, the difficulty of reversing a ransomware infection, and how important it is to detect it before infecting a system. Machine learning is coming to the forefront of combatting ransomware, so we attempted to identify weaknesses in machine learning approaches and how they can be strengthened. The threat posed by ransomware is exceptionally high, with new variants and families continually being found on the internet and dark web. Recovering from ransomware infections is difficult, given the nature of the encryption schemes used by them. The increase in the use of artificial intelligence also coincides with this boom in ransomware. The exploration into machine learning and deep learning approaches when it comes to detecting ransomware poses high interest because machine learning and deep learning can detect zero-day threats. These techniques can generate predictive models that can learn the behaviour of ransomware and use this knowledge to detect variants and families which have not yet been seen. In this survey, we review prominent research studies which all showcase a machine learning or deep learning approach when detecting ransomware malware. These studies were chosen based on the number of citations they had by other research. We carried out experiments to investigate how the discussed research studies are impacted by malware evolution. We also explored the new directions of ransomware and how we expect it to evolve in the coming years, such as expansion into IoT (Internet of Things), with IoT being integrated more into infrastructures and into homes.

Keywords: ransomware; crypto; locker; detection; learning algorithms; internet

1. Introduction

Ransomware is a malware type that is designed to prevent or reduce access a user has to their device, operating system, or files. Ransomware is typically found in the forms of locker ransomware and crypto-ransomware. Locker ransomware displays a lock screen that prevents the victim from accessing their computers, often pretending to be law enforcement demanding monetary payment in return for access to the computer. Crypto-ransomware encrypts key files on a user's system, using complex encryption schemes and demand fees, usually in the form of cryptocurrency to decrypt the victim's files. In its history, ransomware has become more prominent, advanced, and destructive. The rise of ransomware is attributed to many different factors since it first appeared in 1989. The emergence of ransomware as a service has also increased the availability of ransomware

to potential criminals who are less technically gifted. CryptoLocker, CryptoWall, and Locky offer this type of service with the variant CryptoWall, generating more than 320 million dollars in revenue during its lifespan [1].

Reports from early 2017 indicate total damages and profits from ransomware reaching the 1 billion dollar mark [1]. With malware often evolving and new versions of malware families behaving differently to their predecessors, traditional detection approaches will find it more difficult to detect them. Baig et al. outline methods which malware creators use to evade traditional static detection methods [2].

When compared with static code analysis techniques, machine learning techniques have proven more effective, as demonstrated by Reference [3]. Signature-based mechanisms are very easily deceived, especially when dealing with new variants of malware. According to research in Reference [3], out of the 3000 analysed exploit kits, which is how most ransomware infections are carried out (e.g., around 62% of infections of Angler exploit kit to deliver ransomware), only 6% of their signatures were found in VirusTotal. Rieck et al. propose a framework for malware analysis using machine learning approaches [4]. This approach plots malware behaviour in vector space, where the similarity of malware behaviour can be judged; this approach uses clustering, and different classification approaches, where novel classes of malware can be identified.

Machine learning has been effective in detecting malware in Windows OS systems but also in Android systems, as shown in Reference [5]. Further machine learning research into malware detection as an alternative to the use of signatures is presented in Reference [6,7], showing the effectiveness of using machine learning-based detection over signature-based approaches. The decision to evaluate machine learning and deep learning approaches as opposed to other non-machine learning-based approaches was taken because of their adaptability and strong ability to detect unseen samples of ransomware malware.

Reviewing non-machine learning approaches from Reference [8,9] was considered, but their reliance on very specific user-defined patterns and assumptions of encrypted documents made it possible to become redundant very quickly. Non-learning approaches tend to lack the ability to adapt or be retrained to a new concept quickly. These approaches would take significantly more time to recalibrate. We have an interest in the possible wide-scale integration of these solutions in IoT (Internet of Things) to prevent the infection of IoT devices.

1.1. Contribution

In this paper, we do the following:

- Review of Research: Review available machine learning and deep learning approaches to detecting ransomware; this is done in Sections 3 and 4. We assess each research paper on their algorithmic approach, feature engineering process, results, and experiments. We also evaluate the weaknesses of each approach and how improvements can be made in the future.
- Evaluate Research: We evaluate each study's strengths, weaknesses, and how they can be improved; this is included in Sections 3–5. The individual algorithms and papers reviewed are listed and broken down in Table 1. Summaries of the research papers can be found in Tables 2 and 3. Table 2 presents a summary of each paper, and Table 3 shows the statistical achievements of each research study.
- Longevity Evaluation Experiments: We evaluate the longevity of these approaches by running our experiments on current generation and older generation ransomware. The results of our independent experiments are shown in Section 7. Section 7 focuses on testing for the existence of concept drift in ransomware over four years. Our experiments introduce concept drift to the approaches we reviewed and observe their accuracy under concept drift.

Table 1. Study and algorithm index.

Study	Algorithm	Section
EldeRan [10]: 2016	Logistic Regression	Section 3.2
RansomWall [11]: 2018	Gradient Tree Boosting	Section 3.3
RansHunt [12]: 2017	SVM	Section 3.4
Behavioural-Based [13]: 2018	J48 Decision Trees	Section 3.5
Support Vector Machines [14]: 2018	Support Vector Machines	Section 3.6
SDN [15]: 2018	Random Forests	Section 3.7
NetConverse [16]: 2018	J48 Decision Tree	Section 3.8
Bayesian networks [17]: 2019	Bayesian network	Section 3.9
Analysis Framework [18]: 2018	Random Forest	Section 3.10
Feature Selection-Based Detection [19]: 2018	J48 Decision Tree	Section 3.11
Machine Learning-Based File Entropy Analysis [20]: 2019	Entropy Analysis	Section 3.12
Digital DNA Sequencing [21]: 2020	Random Forests	Section 3.13
Resilient Machine Learning [22]: 2019	Adversarial Learning	Section 3.14
API Sequence-Based Detection [23]: 2019	CF-NCF-based Machine Learning	Section 3.15
Two-stage Detection [24]: 2020	Markov Chains	Section 3.16
Multi-Tier Streaming [25]: 2020	Hybrid Learner	Section 3.17
Deep Learning [26]: 2016	Deep Neural Network	Section 4.1
Long Short Term Memory (LSTM) [27]: 2017	LSTM Neural Network	Section 4.2
Shallow and Deep networks [28]: 2017	ANNs	Section 4.3

1.2. Paper Organisation

The remainder of this study is organised as follows: Section 2 details the stages of a ransomware infection, describing how ransomware works in addition to the essential components used to detect them. Section 3 describes a variety of machines learning approaches used to detect ransomware with a summary of these research studies shown in Table 2. Section 4 details deep-learning approaches for the detection of ransomware. Section 5 discusses the overall lessons learned during the review of the research papers and their suitability for application in IoT. Section 6 presents the new directions and the evolution of ransomware. Section 7 details our experiments and analysis of ransomware detection using machine learning algorithms. Section 8 concludes what has been achieved in this study.

2. Preliminaries

2.1. Processes and Tools of Ransomware Detection

2.1.1. Cuckoo Sandbox

Cuckoo is a sandbox environment that enables the analysis of a malware or normal executable files. Analysis information is presented in a high-level way that anyone can understand [29]. API(Application Programme Interface) call analysis will isolate exactly how a malware will interact with an OS and what functions within an OS it draws on. Besides the standard Cuckoo configuration, versions that contain more anti "anti-sandbox" features are available, such as Cuckoo-modified. The basic setup of Cuckoo sandbox is displayed in Figure 1. The host uses a virtual network that is isolated, often through a host-only interface, which will allow the Cuckoo host to execute malware samples safely without giving access to the host itself. Figure 2 demonstrates the systems involved in a Cuckoo sandbox analysis task.

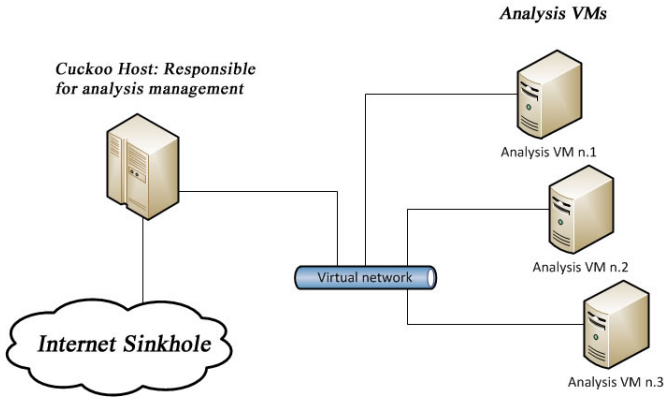

Figure 1. Cuckoo Sandbox Architecture.

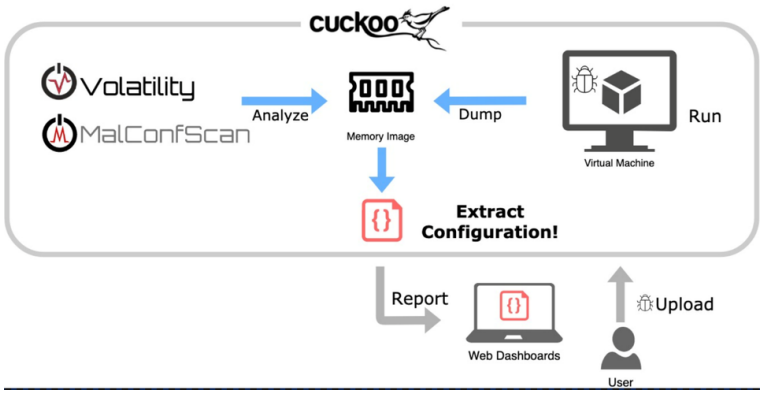

Figure 2. Cuckoo execution.

2.1.2. Machine Learning Platforms

WEKA (Waikato Environment for Knowledge Analysis): A free collection of machine learning algorithms for data mining tasks [30]. WEKA provides three main features, an interactive user interface that can apply various algorithms to datasets [31]. The system can then show the results through various visual methods and display the accuracy of the results through a confusion matrix. For the development of new algorithms, WEKA also provides a framework which provides a simple programming interface for secondary development on the WEKA system [31].

Azure Machine Learning: Azure Machine learning is Microsoft's machine learning studio based on cloud technology. While Azure is not used by any of the research papers reviewed, it is considered an alternative to WEKA. Azure is a framework that does not require the user to code at all. Processing data, training a model and then classification using the model is similar to WEKA, but the classifiers and default settings the algorithms use will differ. Azure also supports open-source Python packages, such as Scikit-learn, Tensorflow, PyTorch, and MXNet.

Scikit-Learn: Scikit-Learn is an open-source machine learning library for the Python Language. The Scikit-Learn library allows the user to implement various learning algorithms onto their data. Scikit-Learn is a library that does require the user to code. Sci-kit learn, while being more difficult to use is deployed by large companies, like J.P.Morgan, Spotify, and Inria [32]. Its commercial use shows the power to be deployed on a large scale.

2.1.3. Ransomware Detection Process For Learning Algorithms

Detection of ransomware, using either machine learning or deep learning follows a very specific pattern, as depicted in Figure 3. There must be a selection of features, be it through custom feature

selection methods or predetermined algorithms; once this is complete, and the optimal feature set is found, the data, organised by these features, will be fed into the deep learning or machine learning algorithm of choice. The algorithm will be trained and then tested. Learning algorithms allow the computer to learn by itself. Learning algorithms operate in different ways, the main ways being supervised and unsupervised. Supervised learning algorithms will require a training set, in the case of ransomware data samples of both benign and ransomware, so the algorithm can learn to identify patterns that distinguish the two from each other. Unsupervised learning methods are fed datasets that are not labelled and will attempt to find patterns which can build models, to distinguish the data types.

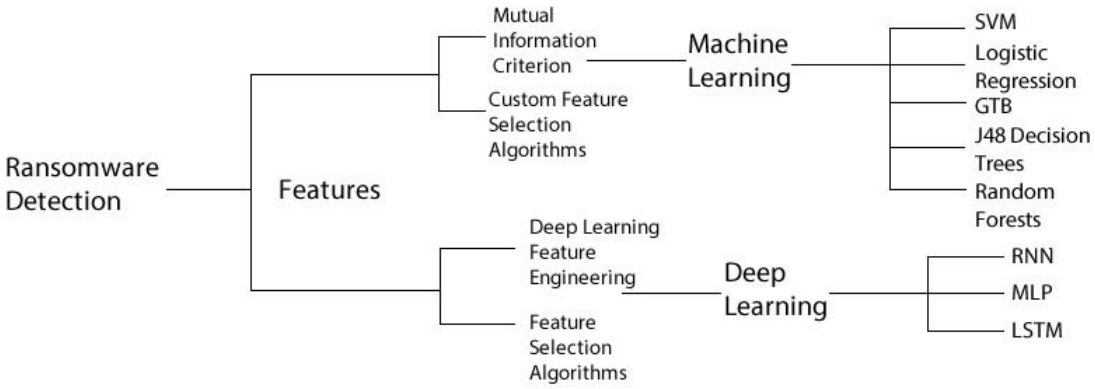

Figure 3. Ransomware detection taxonomy.

2.2. Ransomware Infection Vectors

The structure of a ransomware attack follows the methodology presented in Figure 4. Various infection vectors typically carry out ransomware infections. Firstly, the most prominent vector being malicious emails, the payload is delivered as an email attachment from emails sent through spam using botnets and other compromised hosts [33]. Exploit kits are another prominent method of infection. Exploit kits are software packages which scan a system for vulnerabilities with the intent to infect it with malicious software [34]. Another prominent method of infection is drive-by downloads, in which victims are lured to malicious websites that execute malicious code [3].

Installation occurs after the payload has been dropped into the system. One prominent method of installation is the download dropper [35]. This approach uses an initial file which involves using a small piece of code to evade detection and reach out to the command and control (C&C) centre. Ransomware authors will attempt to split execution into different scripts and processes to avoid AV (Anti-Virus) signature-based detection [35]. When an organisation is targeted in an attack, ransomware will spread through the network, determining file share locations and infecting them to maximise disruption and increase the possible ransom. The executables will not run until multiple machines have been infected.

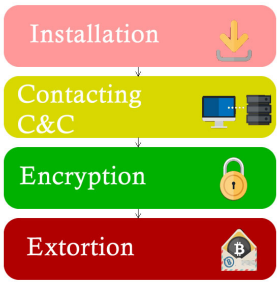

Figure 4. Ransomware methodology.

2.3. Command and Control (C&C)

Once ransomware is installed in a system, it will reach out to a C&C centre looking for instructions [35]. C & C centres will respond with a varying number of requests which will give the ransomware instructions on how to proceed with the execution. Some variants of ransomware will report back significant amounts of system information, which can give attackers an idea of what type of system they have attacked and if it is worth going beyond just a ransomware attack. The ransomware will reach out to the C&C centre for the encryption keys after installation to ensure the keys are kept secret [36]. It is almost impossible to decrypt files without the decryption keys [36]. Command and control channels differ from ransomware family to family; some will use normal HTTP (Hypertext Transfer Protocol); some use complex Tor-based services to connect [35].

2.4. Encryption and Extortion

Modern ransomware will use asymmetric encryption, so the ransomware will come with an RSA(Rivest–Shamir–Adleman)public key, used by the ransomware to establish a secure channel to its command and control server [37]. Public key encryption will mean the plaintext messaged between the server and the client (infected system) will be encrypted in a way third parties will find it very difficult to decrypt. The key factor in this process is that the public key can only decrypt messages that were encrypted by the corresponding private key. This private key is held on the server, which only the attackers have access to, therefore making it impossible for the victim to retrieve. Different variants of ransomware will encrypt files in different ways. Some will use symmetric encryption methods, and some will use asymmetric encryption methods. Symmetric methods will generate a symmetric key locally and encrypt files using this key, the major advantage of which is the lack of performance overhead, reducing the chances of being detected. Asymmetric keys will use a public key which can encrypt, but the decryption process requires the corresponding private key, which is only stored on the C&C server.

2.5. Ransomware Detection Challenges

Several challenges make ransomware detection difficult. Firstly, the idea of using heuristic-based approaches seem highly risky because of the speed at which ransomware evolves. Non-machine learning approaches do not seem appropriate with a malware strain which displays the capacity to evolve and change beyond recognition of even advanced machine learning-based solutions. Ransomware infection must be detected early because, once files are encrypted, it will be almost impossible to decrypt them without paying for a decryption key. Besides paying, victims would rely on the fact that developers of the ransomware will have made an error or decide to make the keys available. Because ransomware is propagated through various methods, it means early detection will have to take into consideration the different propagation methods that ransomware is likely to use. A larger challenge of ransomware evolution is the noticeable concept drift in ransomware, which will be elaborated on in this paper. With a noticeable concept drift detected between ransomware from 2 to 3 years ago and modern ransomware, the creation of models that do not have to be retrained constantly will prove to be a challenge.

The prediction of Ransomware evolution will also be a challenge which will prove critical when detecting ransomware in the future. The incorporation of AI(Artificial Intelligence)into ransomware attacks [9] provides the greatest challenge of all. With attackers using similar AI-based techniques to that of those employed by AI-based defence systems, the configuration of defensive measures will have to take into account adaptable AI-based attack vectors.

2.6. Ransomware in IoT

Ransomware in IoT is not widespread as yet; however, its effectiveness in causing disruption makes IoT networks a logical target. In situations where IoT is integrated into critical systems,

ransomware can be used to disable these systems and force payment. An example of ransomware in IoT is shown in Reference [38], where study exploits a vulnerability in a smart bulb, allowing an attacker to use a flatbed scanner as an entry-point into the organisation's network. The flatbed scanner used in the smart bulb was sensitive, and, using this vulnerability, the attackers can control the smart bulb remotely. The smart bulb acts as a channel into the organisation for the attacker to gain access to the organisational network. In turn, this can be used to load malware into the network.

Another example of IoT vulnerability is Frantic Locker, which first emerged in 2016. This ransomware was first seen infecting Android lock screens in 2015 and later modified to infect smart TVs. Frantic Locker will enter a smart TV and render it unusable, disabling the factory reset. The malware will stay dormant in the system for 30 min. Once the 30 min period is over, the malware will attempt to gain admin privileges to get around any sandbox features. If this is not achieved, the malware will then seize the screen. As smart TVs are constantly connected to the internet, Frantic Locker will then reach a C&C centre. This ransomware does not just hold the TV at ransom but gathers information on the user, like location and personal information stored on the device.

3. Machine Learning Detection Studies

3.1. Study Summary Table

Table 2 summarises the models we reviewed.

Table 2. Strengths and weaknesses of current studies.

Research Study	Summary	Ransomware Used	Features Used	Dataset
EldeRan [10]: 2016	• Algorithm: Regularised Logistic regression. • Feature Selection: MIC. • Able to detect ransomware in early stages of execution.	Critoni, Cryptolocker, CryptoWall, Kollah, Kovter, Locker, MARSNU, PGPCoder, Reventon, TeslaCrypt, Trojan-Ransom.	• Static features & Dynamic features	• 582 Ransomware • 942 Benign
RansomWall [11]: 2018	• Algorithm: Gradient Tree Boosting. • Feature Selection: N/A. • Multi layered approach. • Minimal system overhead. • Utilisation of backup ensures no loss of data.	CryptoWall, TeslaCrypt, Cerber, CTB-Locker, Jigsaw, TorrentLocker, Locky, CryptoLocker, CryptoDefense, Hidden Tear, CryptoFortress, CrypVault.	• Static features & Dynamic features • Honey Pot	• 574 Ransomware • 442 Benign
RansHunt [12]: 2017	• Algorithm: Support Vector Machines. • Feature Selection: MIC. • Static and dynamic analysis. • Anticipates behavioural patterns of next generation ransomware. • Kernel trick can specialise Kernel for ransomware. • Regularisation compensates for over-fitting.	GPCoder, Winlock, Reveton, DirtyCrypt, CryptoLocker, CryptoWall, CryptoWallv3, Critoni, TeslaCrypt, CryptoWallv4, Locky, CBT Locker, TorrentLocker, Cerber3, Samas, CryptXXX.	• Static features & Dynamic features • Honey Pot	• 360 Ransomware • 532 Malware • 460 Benign
Deep Learning [26]: 2016	• Algorithm: Deep Neural Network. • Feature Selection: N/A. • Network approach allows detection before encryption begins. • Can detect ransomware sequence of behaviour within network. • Identifies types of traffic to look for.	CryptoWall, TeslaCrypt, CryptXXX, Locky, CrypMIC,Cerber.	• Network features	• 155 Ransomware • Unknown Benign

Table 2. *Cont.*

Research Study	Summary	Ransomware Used	Features Used	Dataset
Long Short Term Memory (LSTM) [27]: 2017	• Algorithm: LSTM. • Feature Selection: N/A. • Compensates for ransomware that delays execution. • Feature engineering time is low.	N/A.	• API features	• 157 Ransomware • Unknown Benign
Behavioural-Based [13]: 2018	• Algorithm: J48 Decision Trees. • Feature Selection: N/A. • Allows the authors to identify ransomware that display polymorphic behaviour.	Cerber, Cryptowall, Crysis, Jaff, Jigsaw, Locky, Petya, Sage, Torrent Locker, Wannacry.	• Dynamic behavioural features	• 150 Ransomware • Unknown Benign
Support Vector Machines [14]: 2018	• Algorithm: Support Vector Machines. • Feature Selection: N/A. • Kernel trick to be used to tune the SVM. • Attempts to use API patterns to detect zero-day threats.	WannaCry, Cerber, Petya, CryptoLocker.	• API features	• 276 Ransomware • 312 Benign
SDN [15]: 2018	• Algorithm: Random Forests. • Feature Selection: Manual observation. • Pinpoints suspicious network traffic. • No need to look into packets. • Random forests can be effective in small datasets.	Cerber.	• Net-flow features	• Unknown number of Cerber Ransomware samples • 100MB of malign traffic flows • 100MB of benign traffic flows
NetConverse [16]: 2018	• Algorithm: J48 Decision Tree. • Feature Selection: TShark extractor. • Uses network conversation-based network features. • Designed to detect ransomware which evades ML techniques.	Cerber, CryptoWall, CryptoLocker, CTB Locker, Locky, PadCrypt, PayCrypt, TeslaCrypt, Torrentlocker.	• Network features extracted from TShark	• 210 Ransomware samples • 264 Benign samples

Table 2. *Cont.*

Research Study	Summary	Ransomware Used	Features Used	Dataset
Shallow and Deep networks [28]: 2017	• Algorithm: ANN trained with back-propagation and non-linear activation function. • Feature Selection: N/A. • Tests explicitly on zero-day threats from outside the time-window of the training set. • Extraordinary detection rate at 100% for zero-day threats.	Cerber, Cryptolocker, CryptoWall, Maktub, Sage, Torrentlocker.	• API features	• 755 ransomware samples • 219 benign samples
Bayesian networks [17]: 2019	• Algorithm: Bayesian Network. • Feature Selection: N/A. • Use of Packet and Traffic Analysis.	Locky.	• Traffic Flows	• Locky Ransomware network traffic • Unknown benign services traffic
Analysis Framework [18]: 2016	• Algorithm: Random Forests (used in experiments). • Feature Selection: Cosine similarity. • Framework design which can use multiple learning algorithms. • Flexible, different models can make up for the weaknesses of others. • A variety of ransomware families are used for training.	Locky, Teslacrypt, FileLocker, FileCryptor, Troldesh, CryptoWall, TorrentLocker, CryptoLocker, ZeroLocker, CryptoTorLocker, CTBLocker, Xorist, WannaCrypt.	• Assembly Code • API calls	• 178 Ransomware samples • 178 Benign samples
Feature Selection-Based Detection [19]: 2018	• Algorithm: J48 Decision Tree. • Feature Selection: Manual analysis. • Biflow packet approach. • Network features extracted from PCAP files. • Uses Cerber and Locky Ransomware.	Locky, Cerber.	• Network features	• Unknown ransomware count • Unknown benign count

Table 2. *Cont.*

Research Study	Summary	Ransomware Used	Features Used	Dataset
Machine Learning-Based File Entropy Analysis [20]: 2019	• Algorithm: Entropy. • Protects file backup systems from being compromised and overwritten by encrypted versions of the backup data. • Uses Entropy to determine whether files being backed up are encrypted. If a file is identified as encrypted by ransomware, it will not be backed up. • Detection rate of 100%.	PC BYBORG, Reveton, CryZip, May Archieve, FAVEAC, FastBsod, CyrptoLocker, GPCoder, Simple-Locker, TeslaCrypt, CryptorBit, KeRangerm CryptoWall.	• Entropy-based features	• 600 Encrypted file backups • 600 Clean file backups
Digital DNA Sequencing [21]: 2020	• Algorithm: Random Forests. • Feature Selection: MOGWO & BCS. • Complex synthetic DNA-based feature set which converts each sample to a strand of synthetic DNA following scientific constraints. • This approach looks at early detection. • This approach uses confidence values and calculates uncertain predictions, which can be used for concept drift detection in future.	Critoni, Cryptolocker, CryptoWall, Kollah, Kovter, Locker, MARSNU, PGPCoder, Reventon, TeslaCrypt, Trojan-Ransom.	• API calls • System operations • File strings	• 548 Ransomware files • 942 Benign files
Resilient Machine Learning [22]: 2019	• Algorithm: GAN. • Feature Selection: N/A. • Uses adversarial learning to explore the concept of adversarial learning being used to trick machine learning detection systems.	N/A.	• I/O Features	• 4670 Ransomware files • 1616 Benign files

Table 2. *Cont.*

Research Study	Summary	Ransomware Used	Features Used	Dataset
API Sequence-Based Detection [23]: 2019	• CF-NCF-based Machine Learning. • Feature Selection: Intel Pin tool. • Distinguishes ransomware from other malware. • Ensures preventive and reactive measures taken for ransomware will be different from other malware. • Promising results with high detection rates.	N/A.	• API Sequence	• 1000 Ransomware files • 300 Malware files • 300 Benign files
Two-stage Detection [24]: 2020	• Algorithm: Decision Tree & Markov Chains. • Feature Selection: N/A. • Uses a Markov chain and decision tree to make decisions. The decision tree compensates for the short comings of the Markov chain. • Uses static and dynamic features. • Markov chain operates by observing API call sequences.	N/A.	• API calls	• 2507 Ransomware files • 3886 Benign files
Multi-Tier Streaming Analytics Model [25]: 2020	• Algorithm: Naive Bayes & Decision Tree. • Feature Selection: N/A. • Comprehensive solution which uses a hybrid of Naive Bayes and Decision Trees. • 30 day realistic testing environment.	AiDS, GpCode, Archiveus, WinLock, Reveton, CryptoLocker, CryptoWall, RaaS, Cerber, Locky, Crysis, WannaCry, Sopra, Zeus.	• Dynamic behavioural calls • File entropy • Static features	• 35,000 Ransomware files. • 500 Malware files. • 500 Benign files.

3.2. EldeRan

The EldeRan system [10] is based on the observation that ransomware performs certain actions that are unique or significantly different from those performed by benign software [10]. The EldeRan system monitors a sandboxed environment (Cuckoo Sandbox) and extracts features in the following classes: Windows API calls, registry key operations, file system operations, directory operations, the set of operations done per file extension, dropped files, and the strings of the executable. Besides the strings, the features are gathered and analysed dynamically. Once the collection of the features is complete, this is fed into a feature selection algorithm to extract the most relevant features. Once the final set of features is extracted, the data is put through the Regularised Logistic Regression classifier, which will return either "Ransomware" or "Goodware" [10]. The system is trained offline but is run online, and new samples are classified at run time; this can be done on user PCs.

3.2.1. Feature Mapping

The feature selection component of this system uses the mutual information criterion, which allows the most discriminating features to be obtained [10]. The features used are binary; therefore, it is either the presence or absence of a feature that is the value. The mutual information criterion gives the user the ability to quantify the amount of discrimination each feature adds to the classifier. The mutual information criterion will give the system a measure of how dependant or independent features are to whether a file is either ransomware or benign. This feature set is reduced from an initial feature list of 30,967. According to the mutual information criterion, the most significant features in the final 400 features were related to registry key operations, with 48.25% of the features in the final 100 being registry key operations. The next most relevant category is the API stats features, which make up 24% of the final features. The remaining 24% of features amount to less than 10% individually. The additional features consist of directories traversed, files opened, deleted, and modified amongst other directories, and file related activity, which is not specified in the research paper.

3.2.2. Regularised Logistic Regression

The features are fed to the regularised logistic regression classifier to classify the executables as either benign or malicious. Logistic regression is known to be effective in classifying when there are multiple variables to be considered; however, because the classifier uses 400 features, the model would be very vulnerable to over-fitting issues. Over-fitting was alleviated by using a regularisation function which attaches a cost penalty function to each feature which will prevent over-fitting. The justification of the use of regularised logistic regression is that logistic regression is easier to train and add new samples to as opposed to a method, like SVM. Methods, like Naïve Bayes, will assume independence between features, but the assumption made when attempting to detect ransomware is that there is a strong dependence between the features. Because of the high volume of features in the dataset, the algorithm chosen for classification was the Logistic Regression algorithm. The method aims to model the log-posterior probability of the different classes given the data via linear functions depending on the features. Then, the posterior probability of a sample being classified as ransomware $(R = 1)$ given its feature vector x can be written as in Equation (1).

$$Pr(R = 1|x, w, b) = sgm(w^T x + b). \tag{1}$$

In the formula presented in Equation (1) w represents the vector of weights, b being the biased term, and the sigmoid function $sgm(t)$ is given in Equation (2):

$$sgm(t) = \frac{1}{1 + exp(-t)}. \tag{2}$$

The main issue with Logistic regression is it being prone to over-fitting when using the maximum likelihood of the posterior. Regularisation is introduced by adding a penalty term to the cost function;

this combats over-fitting. The cost function for the regularised regression, C' becomes Equation (3): In the formula presented in Equation (1), w represents the vector of weights, b being the biased term, and the sigmoid function $sgm(t)$ is given in Equation (2):

$$C'(w,b) = C(w,b) + \frac{\lambda}{2}\sum_{i=1}^{D} w_i^2. \tag{3}$$

The regularisation parameter is λ; this combination of a regularisation parameter which acts as a penalty function and the mutual information criterion counteracts the effects of over-fitting Logistic Regression.

3.2.3. Experiments

The Regularised Logistic Regression classifier is tested against implementations of Linear SVM and Naïve Bayes classifiers. In all cases, the Mutual Information Criterion is used to find the most relevant features before applying the classifiers to the data. The experiments also used data from the VirusTotal AV detection engines. The data from VirusTotal contains data from multiple AV engines; this was aggregated and used in a voting system. This system worked around the rule that, if the majority of the AV engines indicated that the sample was malware, then it would be decided that VirusTotal classified the sample as ransomware. For experimental purposes, the top 5 vendors with the highest accuracy rates were also used as a comparison benchmark. If an AV vendor did not provide a label, the sample is discarded from the results and not taken into account when calculating false positives; this gives the AV vendors an advantage. The experiments consistently used an 80% and 20% split for training and test data, with over 100 different combinations of this 80% and 20% split. The dataset contains 942 benign applications and 582 ransomware samples. The detection rate achieved by the EldeRan system is at 96.34%, in comparison to 92.19% for SVM and Naïve Bayes achieving 94.53% accuracy rate. The EldeRan system also achieved the lowest false positive rate, at 1.16%. In comparison to the top 5 AV vendors, EldeRan was only second to AV vendor, which had a detection rate of 96.89% and a false positive rate of 0.66%. It must be noted that the AV vendors all achieved better false positives than the EldeRan system and the other machine learning algorithms. The final phase of the experiments is on the new unknown ransomware samples. When experiments were carried out of the 11 families included in the dataset, one would be left out to test the system's ability to detect completely unknown ransomware samples which the algorithm had never been trained on. The overall detection rate goes down to 93.3% in this phase of experiments, in which the system only attempts to classify unknown ransomware samples.

3.2.4. Discussion

This approach comes with many positives while having some limitations. Firstly, the system achieves a very high detection rate (96.34%) on ransomware families it is trained on along with a 93.3% detection rate on unknown families which are zero-day threats. With its effective use of static and dynamic features, the system can achieve detection rates which are more than competitive with current Anti-Virus systems. With the inclusion of static and dynamic features, the system can identify ransomware infections in its earliest stages. Thanks to the use of logistic regression, the relationship the features have that determines whether a file is ransomware or benign is understandable. The system does have its limitations, firstly the lack of any network features in this model. The Cuckoo Sandbox provides means to extract network features from executed files; it is a big miss to leave them out of the feature set. The main problem with this approach as acknowledged by the authors is that this method struggles to detect ransomware that lies dormant for an extended period or requires user input to activate the executable because of the reliance on sandbox techniques and lack of scripting to simulate user input. Finally, regularised logistic regression is not strong with non-linear decision boundaries, meaning that the model may find it hard to find complex relationships between features. If improving

the system, certain features have been identified. Firstly, the reduction of feature analysis time and the improvement of detection techniques, e.g., using known patterns of system calls used to encrypt files.

In terms of suitability for use in IoT, this study uses a very resource-heavy feature set, which would need to be cut down to be suitable for use on more lightweight IoT devices. The use of the Cuckoo Sandbox would also have to change completely. Cuckoo is unable to simulate IoT devices. Another issue with this approach is that the features used are behavioural and static. This research method would have to be heavily modified to be ported onto the IoT domain and away from PCs.

3.3. RansomWall

RansomWall [11] is a layered system which is built to detect ransomware infections in real-time. The layers are organised in order of execution, and this system is designed to be used for Microsoft Windows. The behavioural analysis data for the files are all extracted using Cuckoo Sandbox, with the static data coming from IDA (Interactive Disassembler). The system is set up in 5 layers, the first being the static analysis layer, which analyses the executable in a static context, i.e., strings. The second layer is a trap layer which used honey files and directories. These files and directories are placed so ransomware will attack them before other user files, analysis of ransomware show that a large proportion of them use a depth-first search approach when looking for files to encrypt [11]. The third layer is the dynamic analysis engine which provides behavioural data for the executable. The final two layers are the backup layer and the machine learning layers which handle backing up of files if a ransomware infection is detected. The machine learning layer classifies samples based on the model, which is trained offline.

3.3.1. Feature Mapping

The machine learning layer comprises of logistic regression, support vector machines, ANNs(Artificial Neural Networks), random forests, and gradient tree boosting. The machine learning layer is based on "Sequential Supervised Learning with Moving Average Sliding Window" [11]. The output is either benign or ransomware; hence, this is why classifiers are used. Training is done offline, the execution of the system occurs in real-time in which the static, dynamic, and trap layers send data to a feature-collector, which converts data into the feature set. If a process is tagged as suspicious, the feature values of the data are sent to the machine learning layer, which will then process using the selection of algorithms to determine whether the sample is benign or ransomware. The exact algorithm for deciding on what features to use out of features provided by the three layers are not specified.

3.3.2. Gradient Tree Boosting

Gradient Tree Boosting works on a gradient descent type system in which it builds models progressively. Firstly, a simple model will be built, after which the error residual will be calculated for the model, and then a new model will be built to attempt to correct the errors from the first model. The algorithm will continually rebuild the model to reduce the error in the previous model until the model prediction is at an acceptable level. The gradient descent function will attempt to reduce the gradient to close the gap between real values and predicted values. The collection of weak learners is represented in Equation (4).

$$F(x, \beta, \alpha) = \sum_{i=1}^{n} \beta_i h(z, \alpha_i). \tag{4}$$

3.3.3. Experiments

Data is taken at 1-second intervals. Data from three time intervals will be taken and averaged, to avoid glitches. The model is trained on 11 out of 12 ransomware families selected, and 221 out of 442 benign files. The trained model is tested against the remaining one ransomware family and the

remaining 221 benign files; this is because of the belief that most ransomware attacks are zero-day attacks [11]. The results for this system are extremely promising with the gradient tree boosting method yielded a 98.52% detection rate, with a false positive rate of 0.0056%. The false-negative rate is because of two samples terminating early during execution making file system activity limited; therefore, they are not identified as ransomware [11].

3.3.4. Discussion

RansomWall is a comprehensive approach with many upsides but also has its limitations. In terms of strengths, RansomWall being multi-layered is its greatest strength. It uses static, dynamic, and honey pot layers to detect ransomware makes it a very unique and secure approach. The use of these layers to feed into a machine learning engine gives the system a very strong appeal. In addition to detection, the system does give a protection layer in the form of its backup layer which backs up files on detection of a ransomware infection. The system causes minimal system overhead and uses a strong array of ML algorithms to detect ransomware. Its detection rate using GTB (Gradient Tree Boosting) is exemplary, at 98.25%. In terms of limitations, the RansomWall system uses a dataset of 574 ransomware samples and 442 benign files. Because of how varied benign files can be, it raises the question as to whether the system has enough training on normal behaviour. Despite having a comprehensive multi-layered approach, the system does not use any network behaviour, which usually provides one of the earliest indicators of ransomware infections. Finally, the use of GTB can sometimes become convoluted because of it being prone to over-fitting; therefore, shrinkage and tree depth have to be carefully monitored. The expansion of RansomWall to function on large scale networks is the next step for the system, according to the authors. However, the most beneficial addition to this system in the future would be the monitoring of a set of network features, along with the dynamic and static features.

In terms of suitability for use in IoT, this system suffers from the same issue that the EldeRan system did with it being too focused on behavioural and static features with no inclusion of network-based features. The whole system would have to be calibrated to use the network protocols IoT devices use, not to mention the system itself uses Cuckoo sandbox behavioural features, which are not suitable for IoT devices. The backup layer of this system, however, might be useful to retrieve user data that might be compromised after a ransomware infection. The backup layer functionality can be used for an IoT device or network of IoT devices but would need modifying. The backup layer in RansomWall will start backing up data once it detects a ransomware infection. In terms of IoT devices, if they are trained to detect a ransomware's network behaviour, a backup layer, like the one in RansomWall, can be used to back up data and cut off infected IoT devices from a network that are compromised by a ransomware attack. Often, a ransomware attack will attempt to spread through the network it has been released into. In an IoT network, IoT ransomware will attempt to spread through the network to inflict maximum damage.

3.4. RansHunt

RansHunt [12] is a framework which is designed to identify key features which define a ransomware infection and then use these features to detect ransomware using support vector machines. This system uses both dynamic and static features extracted from 21 different ransomware families. This dataset is completed by an additional 1283 samples of benign executables and other malware. The system is compared to Decision Trees and Naïve Bayesian methods.

3.4.1. Feature Mapping

Feature selection aims to find the features which will allow the system to distinguish ransomware from benign files. The mutual information criterion is used to identify the best features for the SVM. Mutual information criteria allow the user to quantify how much discrimination each feature adds to the classifier [10]. It takes X and Y as two discrete random variables with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$ [39]. The mutual information

$MI(X, Y)$ is the relative entropy between the joint distribution and the product of the marginal distributions [39].

3.4.2. Support Vector Machines

SVM consists of a hyperplane dividing n-dimensional space, which represents data divided into two classes, in this case, either ransomware or benign files. The hyperplane is designed to maximise the distance between two separate classes with the maximal margin being defined as the largest distance between the examples of the two classes computed from the distance between the closest instances of both classes [39]. The hyperplane is represented by a vector w and a scalar m in a way that the inner products of w with vectors $\varphi(Xi)$ from the two classes are divided by an interval between -1 and 1, subject to b [39].

$$(w \cdot \varphi(X_i)) - b \geqslant +1. \tag{5}$$

For every X_i that belongs to the first class and for every X_i that belongs to the second class:

$$(w \cdot \varphi(X_i)) - b \leqslant +1. \tag{6}$$

SVM hyperplanes will have a large margin, which is supposed to reduce generalisation error and over-fitting. SVM uses hinge loss and can use different kernels. Using different kernels is applicable with datasets where the data is not linearly separable.

3.4.3. Experiments

The experiments were carried out in a dynamic and static context, static analysis was carried out on IDAPro and IDA2SQL. Dynamic analysis was carried out on Cuckoo Sandbox. The dataset contains 360 ransomware samples, 532 types of malware, and 460 benign files. Selecting features for the static analysis initially started with 64,984 features but was reduced to 100, using Mutual Information Gain. Mutual information gain measures the information you gain on one variable by learning the value of another variable. The dynamic features are taken from Cuckoo sandbox, which totalled 67 and focuses on Registry Key operations, API function features, and file operation features. These features are combined with the static features obtained and put into a hybrid dataset. The static and dynamic datasets are kept in their initial states so that tests can also be run on them. The RansHunt system is trained on 90% of the dataset using 10-fold cross-validation. The RansHunt system uses SVM with the normalised polynomial kernel and then compared to the results obtained by Naïve Bayes and Decision Trees implemented in WEKA [12]. The SVM algorithm is compared to the most prominent malware analysis platforms available in the form of VirusTotal and Malwr. The RansHunt system achieved a 93.5% accuracy rate with the static dataset and a 96.1% accuracy rate with the dynamic dataset, with a 97.1% accuracy rate with the hybrid dataset. RansHunt outperforms Decision Trees (95.6%) and Naïve Bayes (96%), along with the VirusTotal Engine (95%) and the Malwr Database (93%). RansHunt also achieves the lowest error rate, at 2.1%. RansHunt is also compared to 3 mainstream Anti-virus programs and is only outperformed by NG-AV; this is most likely because of the inclusion of R&D, along with fine-tuned signatures [12]. It must be noted that, for unknown threats, RansHunt would be more effective because it does not rely on signatures or a virus database. The inclusion of other malware, including worms, is to detect a ransomware variant named "Ransomworm", which is one of the predicted attack patterns in the coming year [12].

3.4.4. Discussion

RansHunt is a strong approach and shows very promising results; however, this is not to say that it does not come with limitations. In terms of strengths, RansHunt boasts a 97.1% detection rate with a 2.1% error rate. The training methods used for RansHunt use the concept of being future proof

with the model being trained on worms and Trojans. These behavioural patterns are taught to the model to anticipate next-generation ransomware, "Ransomworm" [12]. This new trend in ransomware behaviour is expected to be a ransomware/worm hybrid and expected to come into prominence within the next two years. In terms of ML methods, the system uses a highly robust SVM system which by default is tuned to avoid over-fitting issues posed by other models. SVM's kernel can be fine-tuned to specific problems, meaning the system can be tuned and updated to be purpose-built for ransomware. The feature set uses a hybrid of static and dynamic features, identified by the Mutual Information Criterion. In terms of weaknesses, SVM tends to underperform in comparison to deep learning approaches despite being strong in terms of speed and memory efficiency. Unmodified SVM also does not give probabilistic confidence of the values calculated, meaning the model is less understandable. When it comes to the system, the accuracy levels of detection may be deceptive because this system is not tested on zero-day threats. The dataset is not split in a way in which one family would be excluded from training and left specifically to testing. This approach, while using a hybrid approach of static and dynamic features, decides not to include any network features which could have been pivotal for early detection. The true effectiveness of this system needs to be tested on zero-day threats, while a 97.1% detection rate is very high; this statistic may not reflect real-world performance. It would be useful to test how useful the future-proofing on the system is, in terms of the ransomware & worm hybrid. As this system has a dataset of hybrid features, static and dynamic; it would make sense to include network features in the dataset also to add a layer of security to the system.

This research is interesting because it is trained on worms and trojans. The use of worms is an attempt to anticipate the next generation of ransomware, which the author predicts will behave like a worm propagating the spread of ransomware through networks. Despite this, the model created in this study would not be suitable for IoT because it uses data from the Cuckoo sandbox, and the concept of trying to simulate future ransomware which will use worms to spread through a network is an interesting proposition. With the massive expansion of IoT, ransomware designed to spread through large networks will likely use worms. The behaviour simulated for these worms carrying ransomware may need to be modified to suit IoT networks and devices; however, the concept is something to be built on for ransomware in IoT.

3.5. Behavioural-Based

Behavioural-Based Classification [13] takes into account that the use of polymorphic and metamorphic ransomware is starting to increase. This approach uses machine learning models to identify modified versions of ransomware based on their behaviour. The study uses 150 samples of ransomware from 10 different ransomware families. This research utilised some of the newest ransomware samples available at the time to get an idea of how machine learning algorithms work when trying to classify evolving ransomware. This method uses an iterative approach to identify optimum behavioural attributes which achieve the best classification accuracy. The behavioural data is taken from Cuckoo sandbox, which produces behavioural logs for executables. Unlike many other studies, this research does not attempt to classify ransomware apart from benign files. The goal is to classify ransomware into their respective families with a dataset comprising of only ransomware samples. The experiments carried out are all done in WEKA, a platform which allows the user to utilise various machine learning algorithms on a dataset.

3.5.1. Feature Mapping

The initial set of behavioural attributes is selected by taking behavioural attributes which appear in 95% of behavioural reports. The next step of the procedure was to add and remove attributes that increased the classification accuracy of the J48 algorithm. An iterative approach is used to select the attributes that give the maximum accuracy. The most important features of the feature set are located on the top levels of the tree [13]. An iterative approach utilises this method to root out irrelevant features until the attributes that appear on the top of the trees in iterations make up the bottom of the

trees, as well. The final number of features used in the dataset is 12. The 12 behavioural attributes used are not specified. All the behavioural attributes are common in all the samples; they vary in type, either nominal, binary, or numeric. This approach takes inspiration from Reference [39], which uses extensive testing on a wide variety of datasets when performing attribute selection.

3.5.2. J48 Decision Tree

As the name suggests, the data structure takes on the structure of a tree. The training dataset is used to construct a tree, which is used for making predictions on the test data. The aim is to achieve the most accurate results with the least number of decisions made. The J48 decision tree can be used to solve classification and regression problems. This method of classification relies on the concept of Entropy and Information Gain. Entropy refers to the uncertainty of the data. For example, the entropy of a coin toss is indefinite as there is no way of predicting the outcome, however, if the coin were a two-headed coin the entropy would then be zero as the outcome can be predicted with 100% certainty. The main algorithm used in decision trees is the ID3 (Iterative Dichotomister 3). The ID3 algorithm aims to start from the root and partition the data into a homogenous dataset. We want the attribute that would result in the highest information gain (return the most homogenous branches). The decision tree approach is popular because it can handle large datasets very well, deals with noise, and operates in a White Box, meaning that we can observe how exactly the outcome is obtained and what decisions lead to the said outcome. It is a popular method of tackling medical diagnosis, spam filtering, and security screening. The calculation of entropy involves splitting the dataset and calculating the entropy of each branch and then calculating the information gain of the split. Information gain is the differences in the initial entropy and the proportional sum of entropies of the branches. Attributes with the highest gain value are selected as the decision node. If a branch has an entropy of 0, it becomes the leaf node. Other branches will still require further splitting. This process runs recursively until further splitting is impossible.

3.5.3. Experiments

The J48 decision tree algorithm is compared with the K-Nearest neighbour algorithm along with the Naïve Bayes. N-Fold cross-validation is used to test the models in the training phase [13]. The n parameter is set to $n = 10$, which is the default in WEKA, which is the platform for the experiments. Overall results achieved are relatively weak with the J48 decision tree algorithm achieving the highest accuracy of a 78% detection rate, Naïve Bayes at 61%, and K Nearest Neighbour at 77.33%. This level of inaccuracy may be attributed to the use of the newer variants of polymorphic ransomware with samples of Cerber having a detection rate as low as 50% [13]. Cerber is suspected to be designed to avoid machine-learning detection techniques, and these results support this. There is a significant drop off when it comes to correctly classifying newer variants of ransomware, with Locky, Petya, Jaff, WannaCry, Sage, and Cerber variants all having classification rates well below the 80% mark. The behaviour of these variants suggests high polymorphism, with new variants not being classified correctly.

3.5.4. Discussion

This research takes an interesting approach in terms of not detecting ransomware and differentiating them from benign files but to classify which family of ransomware each sample belongs to. This approach has strengths, along with some weaknesses. The main strength of this model is the results give us a good idea of which ransomware families display polymorphic behaviour. We can get an idea of which variants of ransomware might be using machine-learning evasion techniques. The strongest machine learning algorithm used in this study, the J48 decision tree implicitly performs feature selection, thus eliminating the need for a separate feature reduction system. The preparation of data used in decision trees is fairly short, making it easier for users to feed data into this algorithm. In terms of weaknesses, the lack of significant modification or tweaking to the algorithms used means the results are naturally limited, whereas, with modification, the classification of the ransomware

families may be more accurate. The use of decision trees in this process can be quite complex because of how complex and diverse ransomware can be. Expectations in decision trees play a big part in the classification process, while expectations are realistic, the classifications are strong; however, if expectations are irrational, this can lead to errors. Decision trees can follow a natural course for events but cannot always plan for every contingency. To improve this work, the J48 algorithm could be tuned for the data presented to it. The classification of modern ransomware variants needs to be significantly improved. A method which takes into account the high polymorphism in the newer samples will greatly aid in improving the correct classification rate.

This study has similar issues to those that a lot of the prior studies do with an emphasis on behavioural data and no network data used. This study did not work on detecting ransomware but identifying ransomware families. One aspect of this study that could be very useful though is that this kind of research can distinguish different types of ransomware in the IoT domain, as well. There is a high variety of devices in the IoT domain, and this could be an important aspect of detection as different ransomware can be developed for an infinite amount of different devices. Identifying types of ransomware from its network behaviour and knowing what devices it will attempt to target will be crucial in knowing how to defend against it and stop the spread by deactivating devices that will be targeted.

3.6. SVM

Detecting ransomware using SVM is the second in this survey that utilises Support Vector Machines and the API calls used by ransomware. The idea behind this model is to train an SVM to learn the API calls ransomware makes to detect unseen ransomware (zero-day threats). The model uses a vector representation of the API calls, in which the number of API calls is counted. API calls are considered by looking into the execution logs of the samples [14]. A standardised vector representation is designed to accommodate the diversity of the programs used. The experiments carried out in this research were done in Cuckoo Sandbox.

3.6.1. Feature Mapping

The API calls used by the ransomware samples is used as the feature set for the data. However, it is not specified what exact features are used. The main concept is that the API logs of the malicious programs need to be quantified for the SVM to use them as features [14]. Because of the number of API calls differ from program to program, a standardisation of the vector representation is used. This standardisation method is described in detail in Reference [14]. Using the logs and quantifying them into vectors, the SVM can learn the sequence of API calls ransomware uses during execution.

3.6.2. Experiments

The experiments are conducted in Cuckoo Sandbox, which produces all behavioural logs. The dataset contains 276 ransomware files and 312 benign executables, various files taken from trusted sources and vendors. Cuckoo Sandbox environment deployed virtual machines with ten main folders and 1000 subfolders. Once a behavioural log is produced, it is converted into a vector and fed into the support vector machine. Once the SVM processes the features, it will then decide if the file is either ransomware of benign. The true positives and true negatives decide the validity of results. The results are compared to the approach used in Reference [4]. The SVM model used in this approach achieves a 97.48% detection rate, which is superior to the approach used by Rieck et al. that produces an accuracy of 94.18%. When using the standardised vectors designed to compensate for software diversity, the detection accuracy of the SVM-based approach decreases to 93.52%; this is because 588 samples of software are used; therefore, the dataset is not diverse enough.

3.6.3. Discussion

This approach is the second SVM-based approach reviewed in this study and has strengths and weaknesses, much like the RansHunt system, which also used SVM. Firstly, the system achieves a very strong detection rate at 97.48% with non-standardised vectors and a detection rate of 93.52% with standardised features. The use of API calls allows the system to analyse ransomware dynamically, which means the system has the potential to be used live. The use of SVM allows for the use of the Kernel trick, allowing for fine-tuning of the kernel to purpose fit the problem, in this case, ransomware. In terms of weaknesses, this model suffers from an obvious lack of diversity in training, demonstrated by its standardised vectors reducing detection accuracy. The use of 276 samples of ransomware may be acceptable, but the system only uses 312 benign samples, which is limiting. The diversity in benign software is massive, and using only 312 samples limits the model's ability to identify normal behaviour compared to ransomware behaviour. Cuckoo Sandbox is used to carry out these experiments, so network or static features could be used. In terms of future work, this research can be improved by diversifying the dataset, to make better use of their standardisation vectors. The addition of network and static features would be greatly beneficial because the trends of malware suggest that ransomware is becoming more polymorphic.

In terms of IoT, this approach is somewhat limited and does not have much that can be carried into IoT. The use of Cuckoo behavioural data and lack of network features makes this system hard to adapt for use in IoT.

3.7. SDN(Software-Defined Network)

With regard to machine learning-based detection of ransomware using SDN, Reference [15] takes the network monitoring route, using a specific type of hardware, programmable forward engines (PFEs). PFEs allow the collection of per-packet network data at high rates [15]. This hardware is used to monitor the traffic flow between the infected PC and the C&C centre, which gives ransomware execution commands. High-level network flow features are extracted from the traffic and are used for classification. This approach uses random forests which fingerprints malicious traffic. This method utilises network flows to show that a flow-based fingerprinting method is feasible and accurate enough to detect ransomware before the beginning of the encryption phase. This approach avoids the previously used method of analysing the payloads in networks, like the methodology used in Reference [15].

3.7.1. Feature Mapping

The features in the model are extracted based on the observation of the victim system's communication with a C&C. Firstly, seeing as communications between the ransomware and its C&C centre go through proxies, which will cause a higher latency [15], the measurement of packet intervals will represent this latency. The volume of incoming traffic is expected to be higher than the volume of outgoing traffic [15]. The difference in traffic is because increased inbound traffic represents the initial infection, encryption key retrieval, and the payment method notifications. The burst lengths of traffic flow are also recorded, which can help to distinguish between a clean and a malicious download. The final features consist of inflow and outflow length with inter-arrival time metrics. This feature set was decided based on its accuracy compared to other combinations of feature sets.

3.7.2. Random Forests

A random forest is a machine learning algorithm which is based on the decision trees described in the J48 decision trees. Random forests require almost no data preparation but yield strong results. Random forests are a collection of decision trees, which is why it is referred to as a "random forest", with multiple trees being built on two-thirds of the training data, in which data is chosen at random. Multiple predictor variables are randomly selected; the best split on these selected variables is used

to split the node. By using the rest of the data, the misclassification rate is calculated. The total error rate is calculated as the overall error rate. This model tunes the random forest using three parameters: the number of trees in the forest, the depth of each tree, and the number of features used in the trees. The number of trees used was 40, with a depth of 15, and a feature set to the square root of all features in the list. This combination was chosen to minimise computational overhead and learning time. Random forests use a tree-based approach with the utilisation of the Gini index to split the nodes. The Gini Index is shown in Equation (7).

$$Gini(n) = \sum_{y}^{|Y|} p_y(1 - p_y). \tag{7}$$

The experiments were conducted on 265 unidirectional unique ransomware network traffic flows. The normal baseline consists of 100 MB of non-malicious traffic, which includes traffic flows from web browsing, data downloading and file streaming. Features were decided upon using the analysis on both malign and benign traffic flows. The data is split 70% training and 30% testing. Ten-fold cross-validation is used to ensure the splitting is unbiased. The use of 28 features yields a detection rate of 89%, and the eight feature model yields an accuracy of 87%. The set of 8 features is used because it is less computationally heavy. Further experiments were carried out on only the Cerber ransomware family using the eight most prominent network features of Cerber, yielding an AUC(Area Under the Curve) of 0.987 [15].

3.7.3. Discussion

This approach to ransomware detection is unique because it takes only network behaviour, without the need to look into the payload, so it can detect ransomware. The main strength of the approach is the ability to pinpoint network behaviour on a high level that can help detect ransomware. The model is tuned so that it can detect while minimising computational overhead and maintaining a reasonable detection rate. Random forests can be useful in a dataset which is not too large, much like this one. In terms of weaknesses, the main weak point of this approach is the relatively low detection rate at 87%. This low detection rate is interesting because decision tree approaches tend to perform well when it comes to behavioural analysis. A low detection rate may suggest that network behaviour on its own may not be sufficient when it comes to detecting ransomware. Random forests tend to struggle when it comes to highly diverse behaviour, i.e., next-generation ransomware, which may break the normal trends the model knows already. This model is not tested on zero-day threats, which is a big miss seeing as its potential to detect the unknown is somewhat untested.

In terms of IoT, this study is the most relevant as it uses network traffic flow between infected devices and C&C servers so it can detect ransomware attacks. Considering an IoT will contain many devices connected to a network, the monitoring of traffic flow can stop ransomware before it even reaches other devices. The traffic flows are independent of the operating system or format of a device as it will sit within the network and monitor the outgoing traffic of each device.

3.8. NetConverse

NetConverse [16] is an analysis of machine learning algorithms on a dataset of Windows ransomware network traffic. The research takes into account the development of variants of ransomware which are now being engineered to evade machine learning detection. The dataset comprises of conversation-based network traffic. This approach acknowledges dynamic analysis techniques have limitations, and new ransomware variants can be redesigned in an attempt to decrease the rate of detection by machine learning algorithms.

3.8.1. Feature Mapping

Feature selection is done using TShark, which outputs statistical and calculated data along with static feature extraction. TShark is an extension of the network analyser, Wireshark [16]. Each network PCAP(Packet Capture)file is merged into a dataset, based on the features extracted from within the PCAP file. The features consist of the protocol used, the origin of and destination address to packets, and duration of the connections. This research takes a different approach to most of the other studies that were reviewed in this survey. They rely on a ready-made program, TShark, to do their feature extraction as opposed to using a feature selection algorithm.

3.8.2. Experiments

The data taken from the TShark is run through multiple machine learning algorithms: Bayesian networks, MLP(Multilayer Perceptron), J48, KNN(K-Nearest Neighbour), Random Forests and LMT(Logistic Model Tree). The highest accuracy achieved was by the J48 algorithm, which achieved a 97.1% accuracy rate. All experiments were carried out in WEKA with a 10-fold cross-validation approach using all ten extracted features. The J48 algorithm achieves the highest accuracy rate with a very low false-positive rate. All of the data used in the experiments are extracted from virtual machines run in VMWare workstation with all of the classifiers not being tuned. The dataset comprised of 264 benign files and 210 examples of ransomware files.

3.8.3. Discussion

This study achieves very high detection rates but comes with its own set of limitations. While achieving a 97.1% detection rate is impressive, the dataset used contains only 210 ransomware samples and 264 benign files. The extremely limited training the models receive on benign behaviour will likely lead to the model becoming confused when it comes to being deployed into the real world unless expansion and tuning are carried out. The lack of tuning also reinforces the statement of the authors that this research acts as a baseline which other researchers can build on because the algorithms have received no tuning to allow them to be purpose-built for ransomware. While these weaknesses are prominent, this work has the potential to be built upon because of the high detection rates achieved.

In terms of IoT, this is another study which can be implemented in IoT because it works exclusively with network-based features. This approach calculating ten optimal features bases on a statistical approach is also positive as this reduces complexity and serves as a lightweight solution using significantly fewer features than most other studies that have been reviewed. The software used would have to be different, of course, because TShark cannot run to monitor IoT a wide range of devices; a method of feature extraction would have to be developed that could be implemented in an IoT environment.

3.9. Bayesian Networks

This multi-classifier detection system [17] is a network-based detection system based on the Locky ransomware. This approach is implemented by using close observations of malware behaviour on the network to develop a set of observable features which can identify and differentiate between network traffic generated by Locky ransomware from normal network traffic. The choice of isolating Locky was made because of its prominence as a crypto-ransomware. The multi-classifier works in multiple layers with one classifier monitoring packets and one classifier monitoring network flow.

3.9.1. Feature Mapping

The features used in this system were not built or decided on by using an algorithm but by using observations on the packets and the network flow. The first distinguishable feature identified is the IP(Internet Protocol)-wise reset connections ratio [17]. The authors noticed that the in 15-min time frame the malicious traffic would repeat the same set of IP addresses with a high reset connection ratio

in nearly every time frame. This feature does not have significant longevity because future variants of Locky could easily terminate their connections early using TCP(Transmission Control Protocol) and FIN(Finished Flag) without RST(Reset Flag) to make their traffic seem more benign. The second distinguishable feature is the increase in the number of HTTP-POST within the traffic stream because of Locky. It is observed that most Locky variants use the HTTP-POST without specifying the User-Agent with no such instances being found in normal traffic. The third distinguishable feature is the large volume of DNS(Domain Name System) name errors in malicious traffic. This feature is identified as a behavioural feature because Locky is based on the DGA algorithm, which is designed to generate a lot of pseudo-random domain names. The underlying technology behind the malware would have to change for this feature to be classed as non-behavioural. The next feature found was the DNS labels used by the malware. It was noted by the authors that Locky only used DNS names with two labels, whereas benign traffic used multiple labels for DNS names. The final feature used was the presence of NBNS (NetBIOS Name Service) packets in 6.05% of malicious traffic.

3.9.2. Bayesian Networks

A Bayesian network (BN) is a relationships network that uses statistical methods to represent probability relationships between different nodes. It is a compact representation of the joint probability distribution for reasoning under uncertainty. The mathematical notation for a Bayesian network is shown in Equation (8), as displayed in Reference [40]. Bayesian networks tend to be computationally expensive because they rely on many data samples to train a network effectively. The computational complexity of a Bayesian network should be considered when using it. In addition to the complexity, the Bayesian network will not be able to model cyclic relationships, and, if three variables correlate to each other, a Bayesian network will not be able to model this. Bayesian networks will enforce a cause and effect relationship on variables; therefore, when modelling using a Bayesian network, the variables must have a cause-effect relationship, or this will be enforced when it does not exist.

$$p(h|e) = \frac{p(h|e) \cdot (h)}{p(e)}. \tag{8}$$

Here, $p(h)$ is the prior probability of the hypothesis, $h : p(e)$ is the prior probability of evidence, e; $p(h|e)$ is the probability of h given e; $p(e|h)$ is the probability of e given h [40].

3.9.3. Experiments

The experimental setup used in this system consists of 5 PCs, the first PC, PC1 being the victim machine where the ransomware is infected. The second PC, PC2, hosts two virtual machines, VPC1 and VPC2, which represent three clean machines on the network and will show how the ransomware attempts to spread through the network. PC3, the third PC, will use Wireshark to capture traffic for analysis. PC3 will run Ubuntu, and its NIC(Network Interface Controller) is set to promiscuous mode to prevent infection. The extracted data is fed into the packet and flow-based classifiers which will then make a decision based on the choices of the two classifiers; these classifiers are built using random forest, LibSVM, Bayesian networks, and random trees. The technical aspects of how these classifiers contribute are not specified. In terms of the packet-based classifier, the random trees prove most effective, with an accuracy of 98.72%. The flow-based classifier displays the most success with Bayesian networks, with an accuracy of 99.83%. On average, the flow-based classifiers were more accurate than the packet-based classifiers.

3.9.4. Discussion

This approach comes with several positives, while also having drawbacks; the system does achieve a high accuracy rate at 98.72% for packet-based detection and 99.83% for flow-based detection for Locky ransomware. The novelty of this approach is that it uses two vectors to detect the ransomware,

a packet-based detection algorithm and a network flow-based detection algorithm. The multi-layered approach allows not one but two angles to monitor for ransomware activity.

The main issue with this research is that it only uses one family of ransomware, Locky to perform all tests and training. The research was published in 2019, so it would be very limited when it came to detecting any other type of ransomware because of how many ransomware variants exist. Using Locky alone seems redundant, which brings us to the second largest issue with the system. The features were all collected based on the observations of the authors and not based on any mathematical feature engineering/extraction approach, which would have chosen from many extracted network features. Using only observed features could bypass valuable information, a feature extraction method would have obtained. For the system to move forward, it would need to be exposed to more ransomware families because of the vast number of them in the modern environment. It is redundant to train a detection algorithm to only detect one family out of dozens of existing ransomware families. Using a mathematical feature extraction method as opposed to manual observations would enhance the system.

This approach used Bayesian networks as the detection algorithm. The Bayesian algorithm has drawbacks when being used for a live system like this. A Bayesian network is computationally expensive, and with a live system like this which will be sitting on the network, detection will need to be fast, especially if acting as the first line of defence. A Bayesian network will be unable to capture cyclic relationships. Cyclic relationships limit the complexity of the network behaviour it can watch out for. Overall, for a live detection system, Bayesian networks may not be the optimal choice.

The approach used in this study could be used for IoT ransomware. The setup can be replicated in an IoT environment with a machine set up to monitor ransomware behaviour using a network monitoring tool, like Wireshark. The exact setup cannot be replicated because of the nature of virtual machines, but the concept is a viable method of using machine learning methods to extract network stream data and train models. Much like the other network-based studies, it is transferable; however, there is not a concept alone which makes it stand out from the other studies, like the SDN concept.

3.10. Analysis Framework

For the Analysis Framework [18], the authors have developed a framework which performs multi-level analysis on a sample. The multi-level analysis involves the analysis of binaries, assembly code, and function calls, such as API calls. The framework uses a combination of these features to make an informed decision on ransomware detection with the use of different machine learning classifiers. They have achieved strong results, with detection rates over 90% for almost all of the machine learning algorithms used.

3.10.1. Feature Mapping

The feature set consisted of reverse-engineered binary features which are constructed from assembly code and DLL(Dynamic Link Library)features extracted from the executables. The feature extractor would extract the DLL and assembly features from the executable. Cosine similarity is used to measure the similarity of two samples based on their DLL and assembly features. Cosine similarity will give a good idea of which features give the most discrimination between benign and ransomware, as shown in Equation (9).

$$\cos \varphi = \frac{P \cdot Q}{|P||Q|}. \tag{9}$$

3.10.2. AdaBoost

The most successful approach in this research is AdaBoost, an iterative ensemble learner. The AdaBoost learning method will learn from mistakes of previous classifiers, considered weak and will turn these classifiers into stronger ones. In a random forest, boosting would be applicable for

classifiers which have a non-satisfactory accuracy. All the trees would learn from each other's incorrect classifications to build an overall stronger classifier.

$$Gini(n) = \sum_{y}^{|Y|} p_y(1 - p_y). \tag{10}$$

3.10.3. Experiments

The experiments consisted of a dataset, containing 302 malware samples [18]. The experiment aimed to determine whether the classifier could tell ransomware apart from other types of malware. The total number of ransomware samples used was 178 from this dataset with an uneven split from various families, most of which consisted of Locky and Teslacrypt. There was also an additional 178 benign samples chosen specifically to act similarly to ransomware to harden the training of the classifiers further. There are three sets of experiments carried out, one with assembly code features, one with DLL features, and the last with both sets of features combined. The third round of experiments is the most successful, with the accuracy of 97.95% with Random Forests; similarly strong results are achieved with J48 and Random Forests which use AdaBoost. In terms of accuracy, when identifying individual families (Locky and TeslaCrypt), the most accurate algorithm is the Random Forest with AdaBoost.

3.10.4. Discussion

This research has positives and negatives to it. The use of a variety of learning algorithms shows the coverage of options when it comes to detection solutions. Because the model is created as a framework, it increases in flexibility in terms of the algorithms that can be used with it. The first issue with this study is that they use very few ransomware samples to train their classifier. While using a variety of different families, they make the mistake of heavily weighting their samples towards just Locky and TeslaCrypt and much less for the rest of the ransomware families they use. The features used are restricted to assembly and DLL features; these features could be further expanded to provide a complete picture of ransomware and better means of detecting them.

In terms of applicability in IoT, this kind of framework could be useful if the feature set could be converted to be applicable in IoT devices. In its current state, the framework would not port directly to IoT devices. In addition, the framework would have to be more calibrated to work in a network scenario as opposed to relying heavily on static and DLL features. The use of an ensemble may be too complex in lightweight devices.

3.11. Feature Selection-Based Detection

Regarding feature-selection-based ransomware detection with machine learning of data analysis, Reference [19] focuses on the network aspect of ransomware detection, which bases its methods on big data. This approach uses a BiFlow concept to replace packets because of size reduction in data, with data size being reduced by a ratio of 1000:1 when using BiFlows over packet data. This system uses six selection algorithms for classification purposes and an additional decision tree model to enhance the performance of the intrusion detection system.

3.11.1. Feature Mapping

The features used are 36 network features identified by analysis of PCAP files, which builds the basis for the research as this is where all of the data comes from. The algorithms were matched to the features based on six characteristics which were gain ratio, information gain, correlation ranking, OneR feature, ReliefF ranking, and symmetrical [19]. The six feature selection algorithms were used together, and their scores normalised to give a combined score for each feature.

3.11.2. Experiments

The dataset contains a combination of Cerber, Locky, and benign software. The experiments carried out are not extensive but use different amounts of attributes, depths, and number of leaves. The most successful configuration of this was using 25 of the 36 attributes, with 19 leaves and tree size of 31, achieving a precision rate of 90.62%. There is a variation of around 2% for precision between the different configurations used for the decision trees, which varied in leaves and tree depth.

3.11.3. Discussion

This study is interesting in because their focus is on the feature selection process and alteration of network data. They transform packet data to BiFlow format to reduce data size. The main issue with this research is simply the lack of information provided about their data and their approach. The reasoning behind only selecting Locky and Cerber is not provided, and the dataset is not described at all. There is also a lack of justification for their selection of features. The use of network features alone might not be a complete solution as there are behavioural and static components to ransomware detection, which could be considered.

J48 decision tree implicitly performs feature selection; this makes using a feature selection algorithm on top of it unnecessary. The omission of a feature selection algorithm in this research paper could be attributed to this; however, this is not specifically mentioned. The use of decision trees in this process can be quite complex because of how complex and diverse ransomware can be; this study uses Cerber and Locky, which means the trees will be biased towards detection of these two families. The diversity of ransomware families has not been captured, meaning the random forests will have trees which cannot classify the vast amount of unseen ransomware. Expectations in decision trees play a big part in the classification process, while expectations are realistic, the classifications are strong; however, if expectations are irrational, this can lead to errors. There is no mechanism to compensate for unreliable predictions decision trees can be prone to, especially if there is a shifting concept. Decision trees can be adapted to shifting concepts through local replacement of nodes; however, this is not addressed in this research.

In terms of the use of this research in IoT, it could be viable because it focuses on network features. While it could be appropriate to include behavioural features in IoT device detection of ransomware, these features would have to be heavily modified if coming from a regular PC detection system. Network features, such as those in this research, can be easily taken and adapted to IoT. The use of BiFlows significantly reduces the size of data, which shows that the approach is capable of being lightweight.

3.12. Machine Learning-Based File Entropy Analysis

In Machine Learning-Based File Entropy Analysis for ransomware Detection in Backup Systems, Reference [20] focuses on ransomware infections in cloud backups. This research recognises that cloud services cannot recognise that the files it is backing up are possibly encrypted by ransomware. The researchers of this paper propose using entropy analysis of infected files to distinguish between infected files and normal files. The main motivator behind this is to protect back up systems and back up files if a ransomware infection has occurred as the file backups will be crucial to system restoration. Encrypted and infected files must be kept away from back up systems. It is difficult to point to negatives of this research, besides that they need to make sure that they test and train this system on a very large range of ransomware which ensures they can keep track of the possible effects different types of ransomware can have on files.

3.12.1. Feature Mapping

This research does not use traditional features, like the other studies we reviewed. Since it is not a traditional detection system which monitors the endpoints of a network, it does not necessarily need

to have static, network or behavioural features. Instead of these traditional features, the system uses a measure of entropy which measures the entropy between the files incoming from and endpoint, and the corresponding backup on the backup server. Through the entropy measure, it is possible to detect whether the incoming file is infected by ransomware. If system resources permit, machine learning models will be added to this detection system, and the synchronised files in the backup server will be used as a test set to detect ransomware infections [20].

3.12.2. Entropy

Entropy is a measure of randomness or uncertainty in the outcome of an event. Entropy can be derived in many different ways. The formula for measuring entropy is shown below in Equation (12).

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i. \tag{11}$$

The backup system will store the data for the user, and when detecting ransomware, the entropy reference value of each user's backup data is measured. The detection module detects infected files by comparing the measured entropy of the synchronised files with the reference value of the file format received from the backup system [20]. It is also proposed to store the entropy reference value as the average of a collection of files. It should be noted that the entropy characteristics for each user can differ, so this should be taken into account when calculating these figures. The backup system learns and measures the entropy of the files stored in the backup system for each user, thereby deriving an optimal reference value specific to the user files, which leads to more accurate detection of the files infected by ransomware [20]. The data used for the machine learning models in the proposed system consists of file format, the entropy measured by the most common value estimate, the entropy measured by the collision test estimate, the entropy measured by the Markov test estimate, the entropy measured by the compression test estimate, and whether or not the file is infected by ransomware [20].

3.12.3. Experiments

The results obtained in these experiments appear very impressive with the majority of the machine learning models obtaining test set detection rates of 100%. The dataset is comprised of 1200 files in total, with 100 examples of each file format included, additionally with 100 encrypted examples of each file format. The results are promising as the dataset is varied and not small, and it is comprised of over 1000 files.

3.12.4. Discussion

This research provides promising results, with a near 100% detection rate of infected files on all the detection algorithms they have used. This aspect of detecting ransomware has not been covered well, and this solution does protect backup systems from taking on encrypted or infected files because backups are the last line of defence against ransomware. Backups must be protected from infection and being overwritten with encrypted files.

This research is different from the other research studies that have been reviewed. The main focus of detection is on the file entropy and how the underlying machine learning algorithm interprets the differences in entropy to detect files which have been encrypted by ransomware. The main weakness of this study is that there is no consideration of how ransomware encryption may be altered to avoid a system like this. The entropy calculations between benign and infected files will rely on specific differences and patterns which define how an infected and encrypted file is identified. It remains to be seen whether the system would be resistant to attempts to mask encrypted files as normal files.

In terms of future work, it would be advisable to ensure the system is trained on the most up to date ransomware in combination with older samples. The system needs to be ready to deal with any eventuality when it comes to the way the ransomware may encrypt and alter files. While this system

shows promising results, all it would take is for a strain of ransomware to alter the way it encrypts files, in a method which the system is unaware of, to trick the system into accepting the encrypted files as backups.

In terms of IoT implementation, this approach would be suitable for IoT devices and networks which have data stored, which is of value. Data backups in IoT can depend heavily on the type of system is involved, so this implementation may not be needed in lightweight IoT networks. However, in larger-scale networks which will store mass amounts of data about millions of users, and will have subsequent backups of all of their data, it will be applicable.

3.13. Digital DNA Sequencing

The digital DNA sequencing engine for ransomware detection using machine learning is a research approach which uses a DNA sequencing approach to ransomware detection. The ransomware sample data is converted to resemble a DNA sequence to detect ransomware before the initial infection stage. This approach is because current ransomware detection studies rely on the analysis of ransomware network or behavioural patterns, which imply the attack is already taking place during detection. Code obfuscation makes it difficult to detect based on code. It is known that only portions of code are revealed during certain parts of executions [21].

3.13.1. Feature Mapping

This system uses MOGWO (Multi-Objective Grey Wolf Optimiser), and BCS (Binary Cuckoo Search) used to select the relevant features for the framework. MOGWO builds on GWO (Grey Wolf Optimiser), which is an algorithm inspired by grey wolves. GWO searches for the optimal method for hunting prey. MOGWO divides the objective space into a grid which contains all possible solutions. The archive decides if a solution should be added or not depending on whether the solution dominates any solutions already in the grid. BCS builds on CS (Cuckoo Search) algorithm, which is adopted from the reproduction strategy of cuckoo birds. Along with these two methods, the accuracy determines which function is maximised and which features are chosen.

3.13.2. Digital DNA Sequencing

The novelty of this study is the use of digital DNA sequencing, a synthetic DNA representation of a digital artefact rather than biological DNA. A digital artefact is represented by only the characters A, G, C, T, where adenine (A), guanine (G), cytosine (C), and thymine (T) molecules, commonly referred to as "bases" [21]. An example of this process of converting a regular dataset to a DNA sequence is: a sample with the binary features '0 1 0 0 0 1 0 1 0 1 0' would be converted to 'CGAACGCGCG' which is a digital DNA sequence containing features from the "bases". A DNA sequence should follow constraints which take into account H-Measure, continuity, melting temperature, and GC Content. DNAct-Ran uses three constraints for the DNA sequences it produces. The *TmConstraint* as represented in Equation (13), the *GCConstraint* as shown in Equation (14) and the $AT_GCRadioConstraing$, as shown in Equation (15).

$$Tm = 4\,^{\circ}\mathrm{C}(G + C) + 2\,^{\circ}\mathrm{C}(A + T), \tag{12}$$

$$GCC = \frac{G + C}{A + T + G + C} \times 100\%, \tag{13}$$

$$(A + T)/(G + C). \tag{14}$$

Using the constraints defined by the DNAct-Ran system, a regular dataset is converted to a synthetic DNA dataset, and, from there, machine learning algorithms are trained on the synthetic dataset to detect ransomware.

3.13.3. Experiments

The detection algorithm uses an active learning approach which is broken down into four strategies:

- Query strategies based on uncertainties, where instances with the lowest prediction confidence are queried.
- Query strategies based on disagreement which queries the instances on which the hypothesis space has the most disagreement degree on their predictions.
- Minimise the expected variances and error by labelling the instances on the pool of unlabelled instances.
- Exploiting the structure information among the instances.

The initial classifier uses a linear regression model which will be used with the active learning principles to train and predict the classification of the data. Once the DNA sequence of the ransomware is learned, the individual families are analysed by machine learning algorithms. The dataset used for this study is the dataset from the EldeRan system, which contains 1524 samples, split into 582 ransomware and 942 benign software samples. The testing was carried out on 150 ransomware and 150 benign samples. Overall the active learning solution achieved an overall accuracy of 87.91%, which is promising because of the novelty of the approach used. However, there are detection studies which achieve much higher early detection results with much less complex systems.

3.13.4. Discussion

In terms of positives, this study uses a highly sophisticated DNA-based detection system which focuses on early detection. The detection rates are on the lower end of the scale with an accuracy of 87.9%, which is lower than the majority of the studies that have been reviewed. This research was published in the year 2020; however, it uses a dataset with software from the years 2013–2015. The age of the data would be considered unsuitable, especially because, through our experiments on the same dataset, we saw significant concept drift in ransomware since 2015. The representation of the data as DNA is a very novel angle on this topic; however, it is unknown how this approach will react when faced with significant concept drift. It is an unknown as to how the changes represented by concept drift will translate to a system which used a "DNA" approach. In terms of future work, the system would need to be trained on more up to date ransomware, and the system itself would need to present a higher detection rate. The use of the linear regression algorithm could be changed to something more suitable for ransomware data. Taken into consideration, this is the only system we reviewed that uses linear regression; it may be wise to alter the data to try the machine learning algorithms others have used.

The most successful algorithm used by the DNA-based system was Random Forests compared to Naive Bayes and the Decision Stump. The random forest will be able to capture complex patterns displayed by a variety of ransomware samples in the EldeRan dataset, regardless of how old they are. The main issue faced by this DNA-based approach is the complexity of it while using Random Forests with a synthetic DNA dataset. DNA is highly diverse and complex; the additional complexity of representing ransomware as DNA coupled with a random forest can represent a high level of complexity. It is uncertain whether the complex and specific patterns generated by the synthetic DNA dataset may cause over-fitting in decision trees. However, because there will be multiple trees in a Random Forest, each tree will have captured different patterns and characteristics in the synthetic DNA which define ransomware, which makes the trees more robust and stable when dealing with unseen ransomware as concept drift will not affect every tree at once.

In terms of usage in IoT, this approach may be too resource-heavy to port to lightweight IoT devices. The synthetic DNA and usage of multiple layers of learning algorithms may indeed be too heavy. For larger-scale IoT networks, it is suitability may be higher; however, its low detection rates in comparison to its high complexity makes it doubtful as to whether it can be used for critical systems.

3.14. Resilient Machine Learning

This research paper evaluates whether machine learning approaches on their own are resilient enough. The authors of this paper test the resiliency and trustworthiness of machine learning algorithms. They use a generative adversarial network (GAN) to carry out these tests. The GAN generates dynamic features which would reduce the efficacy of black-box ransomware classifiers [22]. The main objective of this is to prove that machine learning classifiers need to be further reinforced and be resilient to what the GAN samples represent. The quality of each GAN samples is examined to measure their statistical similarity with real ransomware. The sample space the GAN samples lie in are also investigated; we get an insight into why these samples cause machine learning models to degrade.

3.14.1. Feature Mapping

The execution log of each sample contains a timestamp, event name, targetted file, and file entropy. The four file actions are "create", "delete", "rename", and "change". The entropy level of the file is combined with the event of a file change [22]. Each execution log is a sequence of events, and the maximum length of a sample is capped at 3000 events. Samples with a sequence of events less than 3000 are padded with zeros, so all samples are the same length.

3.14.2. GAN

Adversarial Machine Learning is a method of fooling machine learning models by giving the model an input which is specifically designed to deceive the classifier. The reason behind this is that machine learning models are trained to work on and classify very specific problems. Often, they do not take into consideration that when these models are applied to the real world, the data they deal with may be different or in the case of malware, evolve. Once these classifiers are identified, adversaries will attempt to use data, or malware which does not follow the statistical assumptions made by the model.

A GAN (Generative Adversarial Network)is a connected neural network architecture for both discriminator and generator [22]. The generator produces samples from the generated distribution P_G, which is supposed to be as close to the real distribution P_R. The discriminator will classify the samples generated by the generator and decides whether the sample is from P_G or P_R. The discriminator's aim to find the real from the fake and the generator's aim is to fool the discriminator. By the end of the training, the generator is supposed to maximise fooling the discriminator. These generated samples are put through a quality assessment phase, which is a sample-based adversarial quality metric. Each sample will be quality tested in order to be sure it is fit for use in testing. This process is repeated until a defined number of samples which pass the quality assessment are generated.

3.14.3. Experiments

There are seven models trained on the initial dataset. The models used are Text-CNN (Convolutional Neural Network), XGB (XGBoost), LDA (Linear Discriminant Analysis), Random Forest, Naive Bayes, SVM-linear, and SVM-radial. The ransomware samples are from late 2017 to 2018. The training set contains 1292 benign samples and 3736 ransomware samples. The test set contains 324 benign samples and 934 malicious samples. The training to test ratio is 80% to 20%. On the test set, the detection rates were promising with Text-CNN achieving an accuracy of 98.9%, with a true positive rate of 99.98%. The highest performing algorithms were then tested on the adversarial samples which were produced by the GAN; the results showed significant degrading with Text-CNN detecting none of the adversarial ransomware samples, XGB detection rate falls to 12.73%, and random forest falls to 36.35%, whereas SVM radial detects 100% of the adversarial samples. The experiments have proved the lack of resiliency in machine learning models and how the models can be reinforced with the samples generated by the GANs.

3.14.4. Discussion

This study takes steps to address what may be a growing issue in the future which is adversarial machine learning. The methods used to prove the weaknesses in machine learning have effectively proved an interesting point. The experiments carried out did focus on the ransomware, with the datasets being heavily skewed towards ransomware with relatively little benign software. In a real-life scenario, the ratio of ransomware to benign files a system would encounter would be the opposite. It would be wise to include more diverse benign files in training. The samples used are varied and will give the model a good spread of ransomware families, which display diverse and complex behavioural patterns. The features are somewhat limited as they do not take any network or static features into account but do use I/O features. The reliance on file entropy and file interaction would suggest needing the ransomware to begin execution before detection. The most important theoretical issue with this study is the need for the generation of potential samples. When it comes to ransomware, there is no guarantee that the changes or evolution of the ransomware can be predicted. Realistically, the GAN has to be heavily tuned and specialised to operate within strict parameters that would ensure it behaved like real ransomware while displaying some behaviour which would make it appear benign to a classifier. The next issue with this approach would be the fact that, when altering the statistics of each sample, the actual behaviour it is representing has to be taken into account. Regardless of the potential issues, the overall concept of the work is promising and raises valid issues.

In terms of IoT, there is no reason why the technique could not be applied when it comes to defending against potential IoT ransomware. The features would need to be adapted, but the concept of adversarial machine learning can be used to exploit IoT detection mechanisms, so there would be no reason a technique like this would not be applicable.

3.15. API Sequence-Based Detection

This study takes an approach that a lot of the other reviewed studies have not considered, which is differentiating ransomware from other malware, not just benign software. This research uses API execution sequences which are converted to n-gram sequences. These n-gram sequences are used to identify ransomware and differentiate them from other malware and benign software. Each element of the input can be represented as "1" if an n-gram appears in the n-gram sequence or as "0" if the n-gram is not present in the sequence [23].

3.15.1. Feature Mapping

The features used in this research are Windows API calls. The sequences of API calls of each executable are gathered and then converted to n-grams. The n-gram sequences are used to differentiate between malware, ransomware and benign software. Extraction of the Windows API calls is done by the Intel Pin tool. CF-NCF (Class Frequency Non-Class Frequency) values are calculated for each sample, and these values act as the weights of each element.

3.15.2. CF-NCF (Class Frequency-Non-Class Frequency)

CF-NCF acts as an indicator for classification models, based on the Term-Frequency-Inverse Document Frequency. This technique is used to emphasise the features of each class, therefore giving further ability to differentiate between malware, ransomware and benign files. CF-NCF calculates weights on an element in a class, to provide a higher accuracy on classification experiments [23]. The calculation of this equation can be seen in Equations (16)–(18).

$$CF(s, C) = f(s, c), \tag{15}$$

$$NCF(s, N) = \log(\frac{1}{0.001 + f(s, N)}), \tag{16}$$

$$CF - NCF = CFxNCF. \quad (17)$$

In these equations, s is an n-gram, and $f(s, C)$ is the number of times that the n-gram s occurs in the n-gram sequence C of a particular class. N is the n-gram sequences of the other classes, and 0.001 serves to prevent the denominator from becoming zero [23].

3.15.3. Experiments

The experiments used six machine learning algorithms which included Random Forests, Logistic Regression, Naive Bayes, Stochastic Gradient Descent, K-Nearest Neighbours, and SVM. The dataset was comprised of 1000 ransomware samples, 900 other malware files and 300 benign files. Only 2064 of the 2200 samples were used due to improper execution of some of these files. The API invocations were limited to 50,000 to limit the size of the samples and regulate the number of n-grams produced. In regard to n-grams, the best results were obtained when the n-value was set to 4, and it was Random Forests which displayed the best rate of accuracy, with 97.29%. When detecting ransomware against other malware, it was Random Forests which had the highest accuracy rate of 96.61%. The algorithms were from the Scikit learn libraries.

3.15.4. Discussion

This study focuses on the need to distinguish ransomware from other types of malware. Detection ideally should stop a ransomware attack before infection or encryption, but, in the event of some encryption, the type of reactive measures that need to be taken out if ransomware is attacking your machine will be different than other types of malware. The detection of a ransomware attack on one machine can stop it spreading through a network, which is a key aspect of detection. The permanent nature of ransomware attacks would make a system like this useful. The results obtained are promising; however, this study lacks any future-proofing and acknowledgement of evolving ransomware. The dataset used in this research is skewed heavily towards ransomware and malware, with only 300 benign files. In reality, the model is far more likely to encounter benign files than malware. The lack of variety in the benign samples used means the model could display high false positives when coming across unseen types of benign files. It is important to include a high number of benign files, so the model captures at least some of the diversity that exists in benign files.

In terms of applicability in IoT, the main issue with this study would be the use of Windows API calls. The feature-set would have to be modified to use network activity sequences to pick up ransomware infections on IoT devices. If the features were changed, the use of certain sequences of behaviour would work very well if used in IoT.

3.16. Two Stage Ransomware Detection

This research uses a two-staged approach to detecting ransomware [24]. The authors acknowledge the increasing diversity in ransomware and the difficulty in detecting ransomware. This research uses a Markov model in combination with a Random Forest to detect ransomware. The Markov model focuses on the API call sequences of the ransomware, and the Random Forest is used to model the remaining data to reduce the false positive and false negative rates.

3.16.1. Feature Mapping

This study uses Windows API calls; it is not limited to a specific set or number of API calls. The researchers acknowledge that there are varying numbers of API call sequences for each executable; however, these are not limited or normalised. The additional features extracted from the analysis tasks are registry operations, file system operations, strings, file extensions, directory operations, and dropped file extensions.

3.16.2. Markov Chains

A Markov chain is a state machine system in which events transition from state to state. The Markov chain operates on the assumption that regardless of what the present state is, the future states are all fixed, and no possibilities outside these states can occur. The probability of future events depends on the state in the previous event. In this case, the state space is 303 of the most common API calls used as decided by the authors. The Markov chain is built to predict a classification, using the built probability model which uses the sequence of API calls.

3.16.3. Experiments

The dataset used in the experiments contained 2507 ransomware sample and 3886 benign samples. The Markov chains are used to decide on a classification based on the API call sequences of a sample and the Random Forests to classify based on the remaining features. The Markov chains provide a low rate of false negatives, and the Random Forests provide a low rate of false positives. The Markov chain will classify in the first stage, and the Random Forests will classify all the samples the Markov chain classifies as benign to provide a second layer of security. During the experiments, the threshold value of the Random Forest was altered to extract the best results. With a threshold of 0.2, the highest achieved is 97.28%.

3.16.4. Discussion

This study takes a double-layered approach to detect ransomware. The Markov chains combined with a Random Forest has shown to have promising results although it is not specified on which samples the Random Forest is trained on. The Markov chain may be limited when it comes to ransomware which will change over time as its future states are set, and there are limited futures available for each state to go to next. The reliance on certain API calls may be a risk considering any drastic changes in the way the ransomware uses API or changes in the sequence of API calls will compromise the Markov chain's ability to predict what will happen next in the sequence. In terms of the features and dataset used, this research is comprehensive and covers static and dynamic features. The only lacking component is a set of network features. The dataset uses a larger amount of benign software than ransomware, ensuring the models will be trained to identify diverse benign files from the many families of ransomware that exists.

In terms of IoT, the use of the Markov chains could be used to identify sequences of behaviour within an IoT network which may be unique to ransomware. The Random Forest component may not be applicable because of the nature of the features used in this component of the detection system. The applicability of the system in IoT networks would depend on where the system is deployed and the likelihood of an attack on said system. The sequence of attack would be different if the attack was on an IoT device and act more like Locker ransomware.

3.17. Multi-Tier Streaming Analytics Model

In this paper, the authors propose a hybrid machine learner model, which is a multi-tiered streaming analytics model [25]. The model classifies each sample into 1 of 14 ransomware families by using 24 different static and dynamic features. The model classifies ransomware versions into their ancestral families numerically. For ransomware which descends from multiple families, the model fuses the multi-descendant families statistically [25]. This model attempts to categorise 0-day models into their respective ancestral family or identify if it has multiple ancestors. This research uses the most extensive dataset of ransomware reviewed in this paper, using a dataset of 35,000 ransomware samples, 500 malware samples, and 500 benign files.

3.17.1. Feature Mapping

The feature set used in this research includes 14 dynamic features and ten static features. The dynamic features work through the accessing queries of files and directories, read/write/delete operations, edit the system's digital certification, and modify system files' headers, as well as the entropies of buffering data. The static features involve file content, and file paths, as well as spoofing the links to particular directories [25]. Each sample will have a classification of "01" to "14", so the family it belongs to can be identified. The label "00" is assigned to benign software and the label "99" to other malware types. Naive Bayes is impractical with large feature sets and heterogeneous trait values. However, it spends a short computation time in learning the training vectors of traits and predicting their actual classes by using Bayes' probabilistic theorem with the assumption that all the examined traits are independent of each other [25]. Thus, NB (Naive Bayes)is applied by HML(Hybrid Machine Learner) to trace the predictive classes of all overlooked traits in the indecipherable nodes of DT(Decision Trees) that optimises the adaptive classification [25].

3.17.2. HML Learner

The proposed learner, called HML, is a hybridised version of Naive Bayes and Decision Trees. The main benefit of combining these two classifiers is so they can compensate for each other's weaknesses. Decision trees are known to classify quickly and efficiently on large training sets; however, they are not as effective against previously unseen threats with irrelevant traits [25]. HML trains the fetching batch of trait vectors through cutting the decision edges off the Decision Tree with Naive Bayes pruning margins in an iterative splitting of the training trait vectors into sub-training vectors [25]. The training feature matrix $(T = T_1, ..., T_K)$ such that $(T_i = T_{ij_{i\in K, j\in|T_i|}})$ with the predictive labels $(P_c lass = C_1, C_2 : C_1 = 1$ and $C_2 = -1)$. Each feature vector is represented as $(T_i = C_m, T_{i,j_{i\in|T_K|m\in|C_M|}})$. The prior class probability $P(C_m)$ is computed in Equation (19). This predicts how often a class occurs over (T) in relation to its trait vector (T_i). The conditional probability is shown in Equation (20). Equation (20) represents the relevance between the predicted class (C_m) and its corresponding trait $(T_{i,j})$ as indicated by $P(T_{i,j}|C_m)$.

$$P(T_i|C_m) = P(C_m) \prod_{e=1->p} (T_{i,j}|C_m), \tag{18}$$

$$C_m = C_i -> P_{ms}(T_i, C_m). \tag{19}$$

3.17.3. Experiments

The system initially trains itself by building the classifier and the relationships between attributes and their relation to specific families of ransomware. The system will also identify samples which descend from multiple families and what relationships constitute this. The experiments are carried out in three phases; phase one compares HML's accuracy and detection rates to other popular machine learning algorithms. The second phase tests HML against signature-based ransomware detection solutions. The third phase of experiments tests HML against machine learning-based solutions like EldeRan and RANDS(). HML consistently outperformed each of the comparative machine learning algorithms and also outperformed the machine learning solutions it was compared to which were RANDS and EldeRan. The accuracy of the HML system was tested in a realistic environment over 30 days and maintained an accuracy rate of above 97.4% and a mistake rate between 2.2 and 2.6%.

3.17.4. Discussion

This research goes far to ensure the system produced is resilient to varied ransomware. The system uses a hybridised version of Decision Trees and Naive Bayes to compensate for each algorithms shortcomings. The extent of testing carried out in this research is impressive, with the system tested out in a realistic scenario for 30 days to monitor accuracy and mistake rates over 30 days. The other

unique aspect of this study is the system's ability to identify which ransomware the samples are related to. Being able to identify the ancestry of the sample will make zero-day samples easier to identify. Unless a sample is truly unique and a new family which does not derive from any previous ransomware families, it is highly likely to be detected by the HML system. A problem may arise if the zero-day samples it encounters are not descendants of any previous ransomware family. This system also attempts to classify as fast as possible, and its results are impressive with HML classifying faster than every system it is compared to. The time it takes to classify is crucial as ransomware can do damage if not detected quickly. In terms of negatives, this research study has few, besides the questions over whether the system will be as effective on strains which have no known ancestors. The decision to include 35,000 ransomware samples is a step towards the system being trained comprehensively; however, the decision to only include 500 benign software is hard to justify. In a realistic setting, the system would come across far more benign software than ransomware. Therefore it would be logical to train using a large number of diverse benign software, as well as a large number of ransomware samples. There could be more focus on training the system with benign software which behaves similarly to ransomware.

In terms of IoT integration, this system would need to be shifted to work with only network features. This system could be considered computationally expensive to work with lightweight IoT devices due to using two machine learning algorithms working as one. The system also relies on the combination of many types of features, and the effectiveness with a reduced feature set would be unknown. In the event, this system was deployed on IoT the ancestral aspect of the system would be limited for a considerable amount of time due to the scarcity of IoT ransomware to descend from. A system like this would have to have reduced complexity and would need more ransomware to be released, which targets IoT devices specifically in order to be effective in IoT.

4. Deep Learning Detection Studies

4.1. Deep Neural Networks

In this study, the authors use a network-based approach and use a combination of network monitoring with a deep neural network to detect ransomware. The common approach to detection today used by AV is matching binary patterns and monitoring API calls [26]. These are recognised as signature-based approaches and would not be effective in detecting novel types of malware. If ransomware changes their behaviour in terms of API calls or begins to change its binaries, then they would no longer be seen by the AV until a signature update is released [26]. The usage of API calls and binary patterns relies on detection after execution starts and cannot detect immediately before infection starts [26].

Using the analysis of network behaviour in ransomware, a deep neural network is constructed, which is trained on critical payloads selected from packets which were extracted from real network traffic [26]. This method is designed to detect the infection as soon as it starts and is designed to be implemented on SDN switch, giving it potential to be easily integrated into real network architectures. In terms of the network, ransomware can enter a system in many ways: malicious emails, drive-by downloads, and compromised websites, to name a few. Through the analysis of ransomware network behaviour, it is evident, upon infection, ransomware will request a DNS query to a DNS server for the C&C information for a configuration file. The ransomware will then contact the C&C servers [26], which will give the ransomware further instructions on how to behave. Through this analysis, it is decided that the DNS query and HTTP requests are what is most important for the analysis of ransomware network traffic. Domains for C&C servers can vary; HTTP requests become very important [26].

4.1.1. Feature Mapping

Deep learning models use a method of feature engineering which utilises carefully crafted feature detectors [41]. The neural network is capable of starting with raw data and developing features as it goes along. If one is using deep learning, the use of separate feature engineering is considered obsolete [41].

4.1.2. DNP

Deep Neural networks were chosen for the task of classification because of their ability to build a better feature set than a shallow model. The dropout method was introduced to reduce over-fitting, dropout will randomly ignore neurons during the training process, ignoring their contribution on the forward pass with weight updates being introduced during the backward pass. Dropout is represented in Equation (21).

$$O_i^h = f(y_i^h) = f(\sum_{l<h} \sum_j w_{ij}^{hl} b_j^l O_j^l). \tag{20}$$

where Oh, i is the output of unit i in layer h, y, i is the output vector of the ith layer, w are the weights, and f serves as the activation function. blj is the Bernoulli random vector, where $P(blj = 1) = pl, j$, the dropped neuron, will depend on the Bernoulli random vector.

The use of root mean square propagation (RMSprop) is used to adjust the learning rate, so the step size stays on the same scale as the gradient where λ is the forgotten coefficient, $Q(w)^2$ is the gradient at w. The use of the root square propagation is similar to the use of weight decay; it keeps the moving average of the gradient; this is presented in Equations (22) and (23).

$$V(w,t) = \gamma V(w,t-1) + (1-\lambda) \nabla Q(w)^2, \tag{21}$$

$$w^+ = w = -\frac{n}{\sqrt{(w,t)}} \nabla Q(w)^2. \tag{22}$$

4.1.3. Experiments

The dataset constructed comprises of 23 different types of ransomware, the most prominent being CryptoWall, TeslaCrypt, CryptXXX, Locky, CrypMIC, and Cerber [26]. The majority of ransomware traffic was obtained on traffic analysis websites, along with traffic obtained from user's sandbox environment. The normal traffic was captured by capturing network data of users carrying out normal tasks. Initial extraction yielded over 600,000 malware packets; however, under the assumption that the essential packets were DNS and HTTP requests, this is reduced to 0.3% of the initial number down to 2631 packets. The training set used 80 samples of ransomware found between February 2015 and May 2016, whereas the test set used 77 samples of ransomware which were found post-June 2016. All inputs are normalised to make inputs set to equal vector; this is achieved by using Maximum Transmission Unit. The noise which is caused by samples differing in length is resolved with Central-Slide shifting all inputs to the middle and padding them with zeros. The strongest results obtained from the deep learning neural network was the accuracy of 93.92%, with a false positive rate of 0.12%.

4.1.4. Discussion

The deep learning approach is becoming more and more popular for malware detection, and research into it is also increasing for it in ransomware. This approach has a lot of positive aspects to it while also having some limitations which need to be monitored. Firstly, the deep learning approach used for this model uses ReLU(Rectified Linear Units), which increases the speed of the training rate, slow training rates being one of the main hindrances of deep learning models. The use of network features allows detection before encryption can begin or even earlier in the process, which gives this model an edge over those that do not have any network monitoring capabilities. The approach taken by

the authors also isolates types of network communications to look for as opposed to analysing massive amounts of packets. The usage of deep learning also removes the need for the author to engineer their features. While this model does have a lot of positive aspects to it, it does have limitations. Firstly, over 70 thousand inputs are making the feature engineering process undertaken by the algorithm longer; this can easily be streamlined instead of feeding raw sandbox data into the neural network. The amount of ransomware samples used is very low in comparison to other studies. Deep learning requires large amounts of data. The relatively small amount of data used in this dataset may not be enough to create a strong model. The future of this system is to move execution from a static state to a dynamic execution on an SDN switch which is set in a real network environment [26]. The systems use of network features alone could be further enhanced by the addition of behavioural and static data to take advantage of the power of Deep Neural Networks.

In terms of IoT, this research can easily be ported over as it used network features, in this case, HTTP requests which are network protocols. Through experimentation in IoT networks, it is feasible to identify which protocols are most prominent when it comes to ransomware attacks. The number of times certain protocols appear or the pattern and order in which protocols appear can be key in detecting ransomware attacks or ransomware being spread through the network. Like previous studies, we looked at the patterns in network traffic that are useable in IoT ransomware detection. The research in this study would have to adapt because of the use of Sandboxes using VMs and not a simulation of an IoT network; however, the concept is a viable one.

4.2. Long Short Term Memory (LSTM)

This deep learning LSTM designed for ransomware detection uses the analysis of an API call sequence to create a metric to identify the behaviour of a process [27]. Having analysed multiple ransomware families, the fact that they share common behaviours is identified. Ransomware nearly always makes a short term connection to a command and control centre, delete shadow volume copies of files, and cause a large system overhead by conducting a massive amount of file operations. This deep learning approach creates a model which can identify these properties and classifies a sample as either ransomware or benign [27]. The data for analysis is obtained in Cuckoo Sandbox Modified, a modified version of the cuckoo sandbox which is designed to circumnavigate anti-sandbox measures.

4.2.1. Feature Mapping

The exact features used in this model are not specified, from the specification on columns used in the dataset, being 73,989 would suggest that there are this amount of characteristics which contribute to the final feature set used. Deep learning models use a method of feature engineering which utilises carefully crafted feature detectors [41]. While using feature detectors, the neural network is capable of starting with raw data and developing features as it goes along. Despite this, the feeding of 73,989 columns of raw data may be excessive and produce a massive overhead when the LSTM is engineering features from a massive amount of raw data. The research does not specify the exact amount of features and their type, which is a shortcoming.

4.2.2. LSTMs

LSTM neural networks are a form of Recurrent Neural Network (RNN) which is capable of learning long term dependencies [27]. They utilise a memory unit called a cell to retain information. The LSTM has four neural network layers interacting with each other deciding whether to add or remove information to a cell state. The issue with a regular RNN is that it does not look back far in terms of time steps. LSTMS will include functions which hold on to certain predictions from multiple time steps ago for them to continually contribute to predictions the neural network makes until those predictions are deemed to be forgotten. In this sense, an LSTM works much like the human brain in the way it learns through experience.

Initially, an LSTM will determine how much information from the previous layer a cell must retain. Information retention is decided by a sigmoid layer which acts as a forgetting mechanism. The sigmoid activation function ∅ takes x_t and h_{t-1} and returns a value between 0 and 1 as output, which is multiplied by the value of c_{t-1}. An output of 1 for the sigmoid will completely retain the previous value, and a value 0 will remove it [27].

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f). \tag{23}$$

The next step is updating the memory cell, the sigmoid layer which decides on which values to update, and the tanh layer, a hyperbolic tangent function which returns values between 1 and -1. The tanh layer creates c_t, which is the vector of new candidate values.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \tag{24}$$

$$C_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{25}$$

The old state c_{t-1} is multiplied with f_t, which forgets the values we decided in the previous step. The results is then added to $i_t \cdot c_t$. The resulting candidate value will then be stored in the cell. This stage gathers new information in the cell.

The output h_t is a filtered version of the values in the cell state. The sigmoid layer will decide which part of the cell is going to be output. The value of the cell is passed to the tanh function, the output will be multiplied with the output of the sigmoid layer, and the output is then passed to the next layer in the network [27].

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \tag{26}$$

$$h_t = o_t \cdot \tanh(C_t). \tag{27}$$

4.2.3. Experiments

The dataset contains 157 ransomware executables and benign files which were obtained from online repositories and default programs obtained from a fresh Windows install. When executed, in Cuckoo modified, 239 different API calls were identified in different frequency and order. From the 239 API calls, 38 of them are common in all executed samples; these are labelled between 1 and 39. The difference in API sequence length from executable to executable the API sequence is converted into an integer and appended with 0s to match the longest sequence length. Each sequence is labelled and converted to an integer the dataset is split 80/20 with the 20% being for testing purposes. The first column of the input is the label (benign or ransomware), with the remaining 73,989 columns being the input sequence [27]. The LSTM model chooses the most accurate model during the training phase to be put forward for testing. The training layer is built on three layers with 64 nodes. The highest accuracy training model is used to achieve a test accuracy of 96.67%. The sigmoid function and Adam optimiser are applied to this network [36].

4.2.4. Discussion

The use of LSTM deep learning networks to detect ransomware is a particularly novel approach because of how powerful LSTM is. This model is a strong one yielding very good results; however, it does have limitations. The detection rate for a start is very high, at 96.67%; however, it must be noted it is not specified whether these results are on zero-day ransomware samples. The authors used sequences of API calls to train the model to detect ransomware. The sequences can make it possible for the system to detect ransomware early in its execution based on API calls. This system uses an extended timeout to compensate for ransomware which delays execution. The feature engineering system used

by deep learning models greatly reduces the time taken by creators of the model to reduce features and carry out manual feature engineering. While being a strong model, it does have its limitations. Firstly, the dataset uses common API between all samples. The system relies on ransomware following particular patterns of API usage which could be rendered obsolete in future especially if polymorphic ransomware arises which can alter its behaviour dynamically. Finally, this model makes the mistake of having a very limited dataset, using only 157 ransomware samples and benign files from a fresh Windows installation. Basic Windows programs cannot produce the diverse behaviour required to build a strong deep learning model. In the future, this system should be trained on more ransomware samples and a more diverse set of benign programs. The benign programs only being taken from a fresh Windows installation limits the dataset and gives a false representation of how high the detection rate is. Little to no programs in a fresh Windows install will behave in any way close to how a ransomware executable will behave. The use of only 38 types of API calls could be expanded, or a new metric could be proposed to identify ransomware because there is no solid rule that ransomware will always rely on the same set of API calls to execute their payloads.

In terms of IoT usage, these concepts would struggle because of the focus on API calls used by the ransomware samples in Windows systems. The behavioural characteristics of ransomware in IoT would differ significantly with the feature-set having to be completely redesigned to fit IoT devices. The main issue is that the system's approach in its current form is flawed in that it only looks at 38 common API calls which may limit its ability to deal with new emerging ransomware threats. New samples only have to use a few different API calls to that of the defined 38 API calls to cause the classifier difficulties. In addition, relying on behavioural characteristics alone can cause issues as this implies the system waits until infection to act. If it monitors the network traffic or flows of traffic, it is possible to be more preventive than reactive.

4.3. Shallow and Deep Networks

This study uses deep and shallow networks to detect and classify ransomware. The models classify ransomware and identify which family of ransomware they belong to. All experiments are run up to 500 epochs with a learning rate in the range [0.01–0.5] [28]. The model claims 100% accuracy when classifying ransomware from benign and a 98% accuracy rate when classifying ransomware into their respective families. The dataset is comprised of API calls from Cuckoo Sandbox.

4.3.1. Feature Mapping

The exact selection of the features is not specified. The alternative approaches used, such as SVM, would require some feature selection process; however, this is not specified either. The features used considered around 131 API calls, and the frequency of them from the Cuckoo Sandbox logs are selected and considered features [28]. The feature set does not appear to use any feature reduction methods to reduce the feature count.

4.3.2. ANN

Artificial Neural Networks represent a directed graph in which a set of artificial neuron generally called units in a mathematical model connected by edges [28]. The most common variants of ANNs are RNNs (Recurrent Neural Networks) and FFNs (Feed-Forward Networks). The algorithm used for this model, Multi-Layer Perceptron (MLP) is a subset of FFN. This network will contain three or more layers [28]. The minimal three layers will contain an input layer, at least one hidden layer and an output layer [28]. The number of hidden layers will depend on the complexity of the data. All these units will form an acyclic graph with signals moving forward from layer to layer.

4.3.3. Experiments

The models used are trained using backpropagation with non-linear activation function on TensorFlow [42]. The gradient descent calculations are accelerated by running TensorFlow in a single

NVidia GK110BGL Tesla k40 [43]; this allows increased training speed [44]. The dataset contains seven different ransomware families comprising of Cerber, CryptoLocker, CryptoWall, Maktub, Sage, and TorrentLocker. The 131 API calls used are extracted from the Cuckoo Sandbox logs. The training data will be ransomware samples found from January 2015 to March 2016. Testing data uses samples from April 2016 to May 2016. A training rate of 0.1 is used with three network topologies: MLP 1 layer with 1000 units, MLP 2 layer with 1000 and 500 units, and MLP 3 layer with 1000, 500, and 250 units. All MLP network topologies have used ReLU activation functions which increases training rate. Dropout is used to prevent overfitting [43]; using this approach randomly removes units during training in the forward pass. The MLP with three hidden layers performs the best in binary and multi-class classification. The MLP has a 100% accuracy rate when classifying files as either benign or ransomware. It has a 98% accuracy rate when classifying ransomware into their respective families.

4.3.4. Discussion

This model boasts very good results and has many strengths, while also having some weaknesses. The main strength of this approach is that the MLP model with three hidden layers manages to achieve a 100% detection rate on the test data when it came to classifying between ransomware and benign files. The model allows the classification of files into their respective ransomware families. However, it is not mentioned specifically how many ransomware samples or benign files are used, making it difficult to judge how strong the model is. Multi-layer perceptrons (MLP) as an approach is strong when it comes to classifying problems with a large number of parameters, such as this one. In terms of weaknesses, the simplicity of MLP in comparison with other more complex deep learning approaches may undo it when it comes to dealing with next-generation ransomware. The reliance on just API calls and no kind or network or static features could be an area in which this approach misses out on.

This approach has similar issues as the other deep learning approaches that use Windows API calls as the main feature set. In terms of samples, the issue with deep learning approaches is that there is a significant lack of IoT ransomware which can be used to train a strong model. The scarcity of IoT ransomware means the researchers would have to either spend a long time waiting for malware creators to create more samples of IoT ransomware which will attack different types of IoT devices or researchers will have to create these samples themselves. Because of the lack of IoT ransomware available, deep learning and neural network approaches may not be the ideal approach for now.

5. Experimental Observations & Open Issues

The main lessons learned from the reviewed literature are:

- Effective ML and DL: Machine learning models can be effectively trained to detect ransomware, albeit with some issues which can be addressed.
- Effective Feature Types: The research reviewed points towards the idea that the use of behavioural, network, and static features can all prove effective, despite none of the research papers combining the use of all three types of features.
- Evolution: There is a lack of emphasis on the evolution of ransomware and how the models created would become obsolete over time.

Overall, the lessons learned are that machine learning algorithms can be trained to detect ransomware without a huge amount of tuning; however, to achieve longevity and adaptability to new threats, it would require research and modification of how the algorithms work. Table 3 shows the detection statistics for all of the studies we reviewed. Table 4 shows how the key statistics are composed.

Table 3. Current ransomware detection studies with learning algorithms detection rates.

Research Studies	Detection Rate	Recall	FPR	FNR	Precision	F1 Score
ElderRan [10]: 2016	96.34%	96.33%	0.16 %	3.66%	0.9983	0.9805
RansomWall [11]: 2018	98.25%	97.28%	0.056%	2.75%	0.9994	0.9884
RansHunt [12]: 2017	97.1%	97.04%	2.1%	2.9%	0.9788	0.9749
Deep-Learning [26]: 2016	93.92%	88.76%	38%	7.08%	0.7119	0.8099
Long Short Term Memory (LSTM) [27]: 2017	96.67%	N/A	N/A	3.33%	N/A	N/A
Behavioural-Based [13]: 2018	78% Ransomware family classification rate.	N/A	N/A	N/A	N/A	N/A
Support-Vector Machines [14]: 2018	97.18%	97.13%	1.64%	2.82%	0.9834	0.9772
SDN [15]: 2018	87%	85.14%	12.5%	2.9%	0.8744	0.872
NetConverse [16]: 2018	97.1%	97.05%	1.6%	2.9%	0.9838	0.9774
Shallow and Deep Networks [28]: 2017	100% 98% Ransomware family classification rate.	98.01%	1%	2%	0.99	0.9950
Bayesian Networks [17]: 2019	99.83%	97.1 %	2.09%	0.17%	0.979	0.971
Analysis Framework [18]: 2018	N/A	N/A	N/A	N/A	0.9062	N/A
Feature Selection-Based Detection [19]: 2018	97.95%	N/A	N/A	2.05%	N/A	N/A
Machine Learning-Based File Entropy Analysis [20]: 2019	100%	N/A	N/A	0%	N/A	N/A
Digital DNA-Sequencing [21]: 2020	87.9%	87.9%	10%	12.1%	0.897	0.888
Resilient ML [22]: 2019	98.90%	99.89%	3%	1.1%	0.995	0.979
API-Sequence-Based Detection [23]: 2019	99.53%	99.35%	N/A	0.47%	0.994	0.997
Two-Stage Detection [24]: 2020	98.8%	96.65%	6.93%	1.2%	N/A	0.974
Multi-Tier Streaming [25]: 2020	N/A	N/A	N/A	N/A	N/A	N/A

Table 4. Formula composition.

Metric	Calculation	Value
Detection Rate	TP/(TP+FN)	Correct classification of Ransomware.
False Positive Rate (FPR)	FP/(FP+TN)	Benign software classed as Ransomware.
False Negative Rate (FNR)	100-Detection Rate	Ransomware classed as benign.

Open Issues

After reviewing prominent papers which use machine learning and deep learning approaches for ransomware detection, there are open issues which will need to be addressed in future research. The biggest open issue is lack of acknowledgement of concept drift that has occurred in ransomware over the last 4 to 5 years. The lack of concept drift incorporation in testing does create a significant gap in the domain because of the zero-day testing carried out by some of the papers. They do not explicitly take into account the testing of ransomware from a future period, only testing on zero-day threats from a similar period. Testing on zero-day threats does not appear a priority, as the majority of the studies we reviewed do not explicitly simulate zero-day performance. The most consistent issues are the limitations in the datasets use and the fact they do not take into account the concept shift in ransomware across the years. The ransomware samples used could be further diversified because of the vast amount of ransomware which exists on the internet. We also found that there is a considerable amount of research put into the mathematical theory behind the machine learning algorithms that are used, which we view as appropriate and necessary. However, there is a lack of research into the execution of malware and how it interacts with a system. We find background research on ransomware in all of the studies we review, but not all of them have had an in-depth look into the different aspects of ransomware for which machine learning can be used to detect. The feature-sets used in future studies need to have as much work put into them as the mathematical theory behind the detection algorithms.

6. New Directions/Ransomware Evolution

In terms of the directions ransomware detection is heading, there is compelling evidence that machine learning and deep learning will play a pivotal role in the future of ransomware detection, based on the research we reviewed. Judging from the research studies surveyed, the accuracy and detection rates provided by these approaches are very difficult to dispute. How machine learning approaches are used to detect ransomware will need to adapt as ransomware evolves. Machine learning approaches will have to be trained to learn and anticipate new ransomware trends which will appear in the future. The evolution of ransomware may attempt to avoid or trick machine learning techniques with adversarial learning, like in [22]. Ransomware which will display polymorphic behaviour and hybrid ransomware. It is essential that machine learning and deep learning models be fine-tuned and modified to take into account polymorphic and hybrid ransomware strains. Deep learning may become more prevalent for next-generation ransomware which can be polymorphic or hybrid strains; this is because of how efficient they are with determining relationships that exist among features. As ransomware evolves patterns of behaviour will diversify, and deep learning may be required to spot these highly complex behavioural patterns. Hasan et al. carry out the exercise of predicting the future of ransomware, the RansHunt system [12] is trained on ransomware and worm data to anticipate "ransom worm", which is a ransomware and worm hybrid, which the authors predict to be a future strain of ransomware. Researchers being able to test systems on new threats can be very helpful when hardening detection systems. The research provided in Reference [16] also acknowledges the use of techniques to avoid ML-detection and proposes approaches and a feature set which attempts to remedy this. One of the most significant limitations of the current research is probably the coverage of the datasets used. Currently, the data used is limited in terms of the depth of the number of ransomware and benign samples used. Therefore, it is hard to predict how viable current solutions are. Only two of the studies reviewed tests explicitly against zero-day attacks, which is a huge oversight.

Testing on known samples within a very limited dataset can very easily lead to over-fitting and, therefore, deceptive results. None of the approaches reviewed takes into consideration the use of hardware data to detect ransomware with machine learning. Ransomware will affect on the computing device, not only in terms of behavioural data but in terms of hardware and network behaviour.

It would be useful to use network features, static and behavioural data to have a comprehensive method of detection rather than rely on just static and behavioural or just network features alone. It would be advisable to have systems with multiple Machine Learning or Deep learning components, each handling one aspect of the detection process. For example, one algorithm would handle behavioural data, another would handle hardware, and a third algorithm would handle the network data. The use of multiple algorithms is present in [24,25] where the systems use a Markov Chain and Decision tree, and a Naïve Bayes and Decision Tree hybrid, respectively. The most suitable machine learning or deep learning approaches would have to be selected and aggregated to take decisions from all three components and then decided whether a sample is a ransomware or not. The RansomWall system [11] takes an approach which attempts to be comprehensive as it uses multiple layers; however, the machine learning component aggregates all data as opposed to being assigned to individual aspects of the system.

A gap in the majority of these research techniques is because detection seems to depend on the execution of the ransomware, with the features taken from the sandboxes all during full execution. Unless the samples are executed on sandboxes before installation or execution, these methods would still lead to the infliction of some damage to the system when relying on behavioural patterns, such as read/write frequencies and file access patterns. The execution within sandboxes could lead to performance overhead and increases the time required to execute the files. The majority of the studies we reviewed do not address early detection directly, and the studies which do not provide why their chosen time-period of detection is optimal for ransomware. It is useful for systems to detect ransomware before they encrypt files; therefore, researchers need to state how long their test and training samples took on average, to start encrypting files. The time-sensitive aspect for ransomware needs to be looked at in-depth to know how soon a sample needs to be stopped to preserve files. The EldeRan system [10], for example, addresses early detection and states it monitors the first 30 s of execution; however, this time frame needs to be justified. Early ransomware detection is relative; early detection can be defined as identifying an infected machine and preventing the infection from spreading across the network or can be defined as identifying a ransomware executable in the earliest stages of its execution before encryption. Early Detection solutions, like Reference [10,16,21,39], are critical in ransomware detection and will be critical in ransomware detection going forward. The early detection approach is important because of the damage ransomware can do to a system and businesses. For early detection systems, various approaches, like identifying behavioural traits the malware exhibits during initial execution before encryption or static analysis, can be applied. Studies we reviewed do take into account binary features, network features, and behavioural features. A combination of these three features can be used to identify traits of ransomware before encryption of files. Static features alone will not be enough for early detection due to the increase of file-less attacks; however, ransomware execution methodology allows early detection as it carries out tasks before encryption, such as calling out to C&C servers and receiving encryption keys. The steps before infection will require specific network behaviour patterns and the invocation of specific API calls, which makes early detection a strong possibility, given a classifier knows to identify this behaviour. It must be acknowledged that intelligent ransomware strains will be more difficult to detect as they will aim to obfuscate samples and mask malicious behaviour; however, despite this, ransomware core behaviour must be retained; otherwise, it cannot fulfil its purpose. Ransomware will still attempt to call to a C&C server for encryption keys and will require iteration of folders and files to encrypt files. Theoretically, an AI-based ransomware strain will attempt to fulfil the core ransomware functionality in the most covert and disguised way possible. AI-based malware cannot change completely beyond the point that its core behaviour is irrelevant to its goals; therefore, machine learning-based detection systems need

to be able to see through obfuscation and disguises generated by AI-based malware. The possibility of AI-based malware does not detract from the importance of machine learning detection systems, as they are equipped to identify the signs of disguised malicious behaviour and obfuscated samples. Moreover, there is a high possibility that ransomware could begin to use adversarial machine learning or deep learning techniques. With the advancements and integration of AI, machine learning, and deep learning into mainstream technology and life itself, it is only a matter of time until malware developers begin to create more and more intelligent strains. A solution to adversarial ransomware is addressed in Reference [22], in which a GAN is used to generate potential samples to trick a ransomware classifier. Using trained samples generated by a GAN, the system in Reference [22] proves effective, but this is not to say this area of research cannot be expanded further. With the increased use of AI-based techniques in the detection of malware and ransomware, in particular, the inevitable step of malware creators integrating machine learning techniques into ransomware draws closer; this stems from the need for malware to become more intelligent to keep pace with the intelligence of detection systems ever increasing. The need to research and analyse the behaviour of next-generation ransomware will increase. It will be necessary to train models to take into account adversarial learning and intelligent AI-based malware, especially training models, to spot the subtle differences between malign software and ransomware data which is modified to look benign. The creation of intelligent machine learning or deep learning-based ransomware is still theoretical.

In addition to intelligent ransomware strains, the introduction of ransomware in IoT is almost a certainty. The expansion of IoT into critical infrastructure will pose the biggest vulnerability with attackers targetting critical systems, like transport, airports, smart grids, and entire smart cities, in which attacking IoT devices could cripple economies. Mass IoT integration means a high reward for attackers and more incentive for victims to pay the ransom. Locker type ransomware could be used to maximum effect when targetting critical infrastructure. This steps away from the more popular crypto-ransomware; locker ransomware will prevent access until a ransom is paid as opposed to encryption of files. In the event of infection of critical infrastructures, such as power plants, airports, hospitals, or entire smart grids, victims may be inclined to pay. IoT is becoming more integrated into the automotive industry with predictions of up to 14 million semi- or fully autonomous vehicles on the roads in the U.S. by the year 2025 [45]. This IoT integration is not only limited to personal transport, as the New York Metropolitan Transportation Authority is planning full integration of IoT into public transport through built-in WiFi, security cameras, and charging. IoT connectivity will allow manufacturers to release software updates, with companies, like BMW, leading the way in this technological revolution in the automotive industry. Alphabet's Waymo and Apple are also investing heavily in the development of autonomous vehicles, and Waymo is expected to be worth $105 billion and hold an 18% stake in the AV market by 2030 [45]. This IoT connectivity in vehicles will create vulnerabilities and opportunities for attackers to exploit these vulnerabilities. If a car relies on remote updates, there is no reason an attacker cannot use this to upload ransomware onto a car's computer system. If the ECU (Electronic Control Unit) is encrypted or locked, the car's drivetrain will fail to function and render the vehicle useless. This approach is not very different from traditional Petya ransomware corrupting a Windows system MBR (Master Boot Record) to prevent it booting normally. In a scenario where an autonomous or IoT integrated car's ECU is disabled by ransomware, the car will be disabled until the ECU is allowed to function as normal. If an attack scenario involving an autonomous car happens in a deserted location, where the victim will struggle to get assistance, the victim will be inclined to pay immediately [46]. In the event a victim can get help, they will still be left with a car which has an encrypted or locked ECU. An encrypted ECU will require a replacement, and, unless the manufacturer can decrypt the ECU data or replace under warranty, the victim will be inclined to pay, as long as the cost of the ransom is less than that of the ECU replacement. If this type of attack was carried out on public transport on a train, an attacker could cause severe disruption on train lines by encrypting or locking a train's control unit. The scenarios described in [46] are possible

as systems become more integrated with IoT, with the public and personal transport scenarios being on the milder end of the spectrum.

7. Our Experiments and Results

In the interest of research, we ran our experiments on a modified version of the dataset used by the EldeRan system [10]. The dataset used was available in a raw format, with the inclusion of all the behavioural data from Cuckoo sandbox executions of ransomware samples found from the years 2013 to 2015. This dataset alone was used without the use of any other datasets from papers reviewed in this survey because none of the authors of the other papers has made their datasets available to the public. The main motivation for carrying out these experiments is to demonstrate how concept drift can degrade a machine learning model. By studying concept-drift, we will be able to understand how far back we need to go and use data for training learning algorithms. The models used for these experiments were trained on the EldeRan dataset, which includes ransomware from 2013 to 2015. We then test these models on ransomware from 2016 and 2017 to observe how severely the models degrade. All experiments were run on WEKA, using the most possible approaches from the studies we reviewed. All of the behavioural data was taken from Cuckoo sandbox. The original dataset consisted of 581 ransomware samples and 942 benign samples. The benign samples included variants of AxCrypt, Bitlocker, 7zip, and VerCrypt. We included genuine encryption software so the classifiers could be trained to distinguish between genuine encryption software and ransomware.

WEKA follows a five-step process when analysing data.

- Data Entry: WEKA can take dataset inputs in the form of WEKA's Attribute-Relation File Format(ARFF) format or Comma-Separated-Value (CSV) files.
- Data Mining: The data mining phase includes pre-processing, classification, clustering, association rules, and regression [31].
- Data Evaluation: Data Evaluation assesses the models of the results of the algorithm used.
- Visualisation: WEKA is capable of providing visual representations of the data and the results obtained by the algorithm.
- Storage: Once results are obtained, the data can be stored to be viewed.

In our initial experiments, we used the whole dataset and split or cross-validate it, depending on the method of detection used. The parameters used for our models are provided in Table 5. In the second round of experiments, the whole EldeRan dataset is used as a training set with the test set being a dataset of 100 modern ransomware samples and 300 benign files. The samples in the test set are from the families WannaCry, Cerber, and Locky. The Locky and Cerber variants are from 2016, and the WannaCry variant is from 2017. Out of the experiments we carried out, the best detection rates are achieved by regularised logistic regression and a multi-layer perceptron with 20 hidden layers. Overall, the results obtained through WEKA with minimal modification to the algorithms themselves are promising, albeit with slightly high false-positive results. Table 6 illustrates how the detection, false negative, and false positive rates differ between ransomware samples up to a year apart. The feature set we use for these experiments is a collection of 318 API calls used by Windows executables. These 318 API calls were selected as they are the API calls hooked by Cuckoo Sandbox.

As it can be seen in Table 6 and Figure 5, the detection rate of all the algorithms used in the experiments falls when faced with ransomware variants from a different era to the ones it has been trained on. In this instance, the test ransomware was mainly from 2016 with the WannaCry Variant from 2017. With the detection rate falling, false negatives sharply increase, and the increase in false negatives is uniform and across all algorithms. This conclusion leads us to believe that the rapid evolution of ransomware means that models can be out of date within the space of a year. How ransomware evolves can also be rapid; therefore, new unseen ransomware may emerge just after training a model. The algorithms used may lack the longevity required to detect on a long term basis. Of course, this is without modification, with modification and "future-proofing" machine learning and deep learning

should indeed be adapted into a long term solution. As demonstrated in Table 6, the multi-layer perceptron, which achieved an initial detection rate of 96.4% using ten hidden layers, appears to degrade the least when exposed to ransomware from a different era. In Table 6, it can be seen that the results display a high level of false-positive. The false positives are because the dataset created contains a small number of benign samples which display similar behaviour to ransomware; however, it can be seen in Figure 6 that the level of false-positives increase during concept drift.

Table 5. Independent experiment parameters.

Method	**Parameters**
GTB	• ZMax Value: 3.0 • Random Tree Classifier • Likelihood Threshold: -1.7976931 • Weight Threshold: 100
Random Forest	• Bag Size Percent: 100 • Max Tree Depth: 0 • Iterations: 100
SVM	• SVM Type: C-SVC • Eps: 0.001 • Gamma: 0 • Kernel Type: Radial Basis Function: $Exp(-gamma \cdot \lvert u - v \rvert^2)$ • Loss: 0.1 • Nu:0.1
Logistic Regression	• Max iterations: -1 • Ridge: 1.08×10^{-8}
J48 Decision Tree	• Confidence Factor: 0.25 • MinNumObj: 2 • Folds: 3 • Use Sub Tree Raising
Deep Neural Network (10 fold Cross Validation)	• Instance Iterator: Default • Neural Network configuration - Leaky RELU alpha: 0.01 • Optimisation: Stochastic Gradient Descent • Updater: ADAM • Beta1MeanDecay: 0.9 • Epsilon: 1.08×10^{-8} • Learning Rate: 0.1 • Weight Initialisation Vector: XAVIER • Gradient Normalisation threshold: 1.0 • Intermediate evaluation: 5
MLP (10 Hidden Layers)	• Hidden Layers: 10 • Learning Rate: 0.3 • Momentum: 0.2 • Training Time: 500 • Validation Threshold: 20
MLP (20 Hidden Layers)	• Hidden Layers: 20 • Learning Rate: 0.3 • Momentum: 0.2 • Training Time: 500 • Validation Threshold: 20
Bayesian networks	• Estimator: BMAEstimator, alpha : 0.5 • Search Algorithm: K2

Table 6. Independent experiments.

Method	Detection Rate on Samples	Detection on 16/17 Samples	13–15 False Positive	16/17 False Positive	13–15 False Negative	16/17 False Negative	13–15 Precision	16/17 Precision
Regularised Logistic Regression	88.6%	62.3%	13.6%	32.1%	11.4%	37.3%	86.4%	92.8%
Gradient Tree Boosting	94.6%	77.4%	5.5%	19.2%	5.4%	22.6%	95.5%	96.6%
Random Forests	94.0%	79.2%	5.6%	17.6%	6.0%	20.8%	94.5%	96.9%
SVM	33.1%	17.0%	36.1%	70.4%	76.9%	83.0%	74.9%	84.0%
J48 Decision Tree	89.8%	79.2%	8.0%	17.9%	10.2%	20.8%	92.2%	94.6%
Deep Neural Network Using 10 (Fold Cross Validations)	95.1%	84.9%	9.6%	13.6%	4.9%	15.1%	90.6%	93.4%
Multi-Layer Perceptron (10 Hidden Layers)	96.4%	85.0%	8.0%	14.0%	3.6%	15.0%	92.0%	92.2%
Multi-Layer Perceptron (20 Hidden Layers)	95.8%	58.5%	9.0%	35.5%	4.2%	41.5%	91.2%	90.7%
Bayesian Networks	93.4%	83.0%	13.3%	17.4%	6.4%	17.0%	87.1%	88.1%

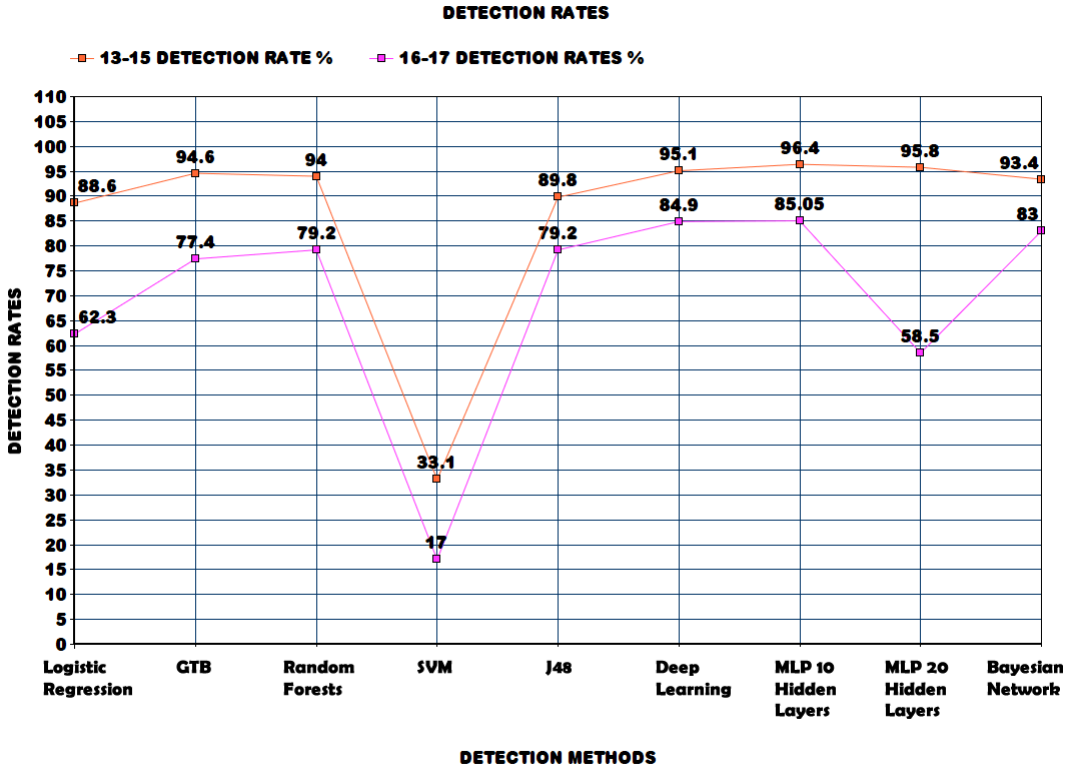

Figure 5. Detection rates.

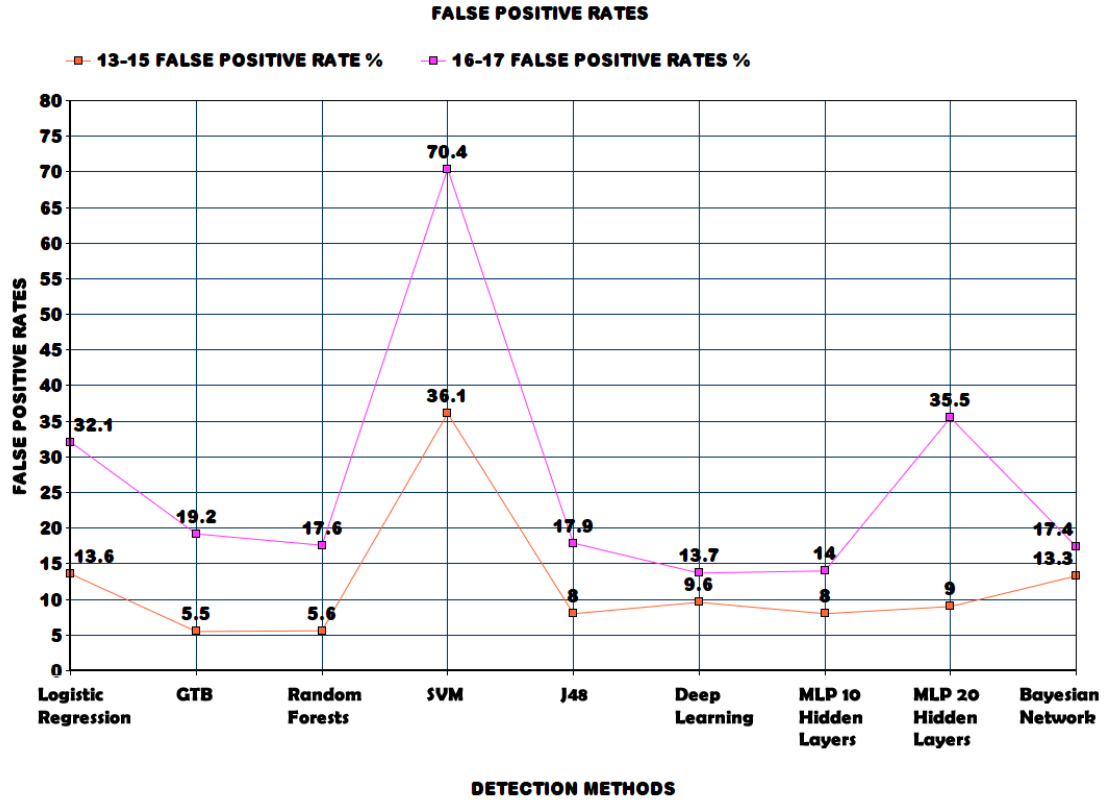

Figure 6. False positive rates.

Our results provide an insight into what gaps exist in machine learning-based ransomware detection. We observe that detection studies that focus on zero-day detection have strong results; however, we observe the sudden evolution of ransomware may prove to challenge ransomware detection systems. Concept-drift has always existed in systems in which there are machine learning systems; therefore, evolving malware behavioural patterns were always likely to exist in ransomware. Our experiments are designed to highlight this, and we suggest that machine learning approaches detecting ransomware take concept drift into account explicitly in order to adapt to sudden concept drift. Our experiments demonstrate that machine learning approaches need to take sudden evolution into account, not that machine learning is obsolete when detecting unseen strains of ransomware. We believe that machine learning still has excellent potential when detecting unseen variants of ransomware but will struggle to adapt to concept drift without mechanisms in place to counter concept drift.

These experimental results show the speed at which ransomware has evolved over the last decade or so. The complexity of ransomware on Windows OS alone has increased rapidly. The possibility of ransomware attacks in IoT must be acknowledged. With IoT networks being on a massive scale, the risk/reward pay-off is high with larger targets like smart cities and smart airports being critical systems that cannot afford to go offline. Attackers are likely to take advantage of the connectivity of millions of devices to use advanced propagation techniques to spread through vast networks, causing massive amounts of damage. There is a need for solutions tailored to networks which will hold thousands of different types of devices which not only operate on an endpoint level but sit within the network monitoring the flow and behaviour of traffic to identify potential threats.

8. Concluding Remarks & Future Work

We reviewed research which uses machine learning and deep learning for the detection of ransomware. In general, the approaches reviewed boast high detection rates in the mid to high 90s. These models are all trained on a mix of network, behavioural, or static features. While most are conceptual systems, like RansomWall and the EldeRan system, some have been tested via deployment. The results achieved give us confidence that machine and deep learning models can be deployed to detect ransomware. However, their ability to stand the test of time and evolve with the rapid evolution of ransomware is debatable. We also seen the introduction of GAN and adversarial machine learning being used to deceive ransomware detection systems, which is an area which will need to be addressed as urgently as concept drift. We explore the concept of AI being integrated into ransomware to create intelligent strains and how this will change the detection space. To test the longevity of the models, we carried out new experiments using the ransomware and benign samples from the EldeRan dataset and data we gathered ourselves. We simulated the effects of concept drift and observed the behaviour of machine learning and deep learning classifiers under concept drift. We acknowledge the rapid evolution of ransomware and introduce this evolution to trained classifiers by testing models on ransomware from a few months to a year ahead of their time. Our experiments demonstrate possible flaws in the approaches taken in all the proposed methods. Our experiments show that longevity should be taken into account when learning algorithms are trained, and that machine learning and deep learning algorithms will need to take concept drift in ransomware into account, to be considered a long-term solution. Moreover, we looked into the suitability of these detection techniques when applied to the IoT sphere. We believe the integration of detection systems in IoT is crucial because of how expansive IoT is becoming; it is being implemented in critical infrastructure and people's day to day lives. The threat of ransomware is becoming very real and more dangerous with billions of dollars at stake along with the risk of human lives being lost during services and system becoming non-operational during ransomware attacks. We evaluated scenarios in which IoT can be compromised by ransomware and how these scenarios can become widespread. Overall, we believe that ransomware will continue to evolve and become more challenging for intelligent detection systems. The evolution of ransomware and the advancement of detection evasion mechanisms follow the general pattern of

malware evolution. It is important to acknowledge that signature-based detection was once the most commonly used method of detection and how it became obsolete over time. The creators of intelligent detection systems which use machine-learning and deep-learning systems will have to create ways in which these systems can stand the test of time and will not be left behind, like heuristic approaches; studies we evaluated in this paper have touched on adapting to change, but we believe it is an aspect of detection which is not researched enough. To conclude, this paper has thoroughly summarised the ransomware detection research space and conducted experiments to propose open issues which must be addressed in the future.

Future Work

For our future work, we would like to compile IoT ransomware to test the applicability of the detection methods we explored in this paper. We acknowledge the scarcity of ransomware in IoT currently; therefore, it is impossible to evaluate the current ransomware detection approaches in an IoT context. The viewpoint that ransomware will not target IoT devices and instead attack their back end servers could well come to fruition, so we must take into account aspects of IoT back end architecture might be vulnerable to ransomware. If the IoT ransomware threat does not emerge immediately, we do aim to create or simulate IoT ransomware and their operation to analyse and prepare for what we believe is an eventuality. We aim to analyse the viability of ransomware targetting IoT devices and the data centres and servers which keep them running, as well as using this analysis to determine which of the two is the more likely target.

Author Contributions: The contributions of the three authors are as follows. D.W.F. contributed 50% with the research and experiments with 40% contribution from N.K. with contributions to the paper's structure, research, design, experiments, and supervision. T.C. contributed 10% with supervision and related studies. D.W.F.; data curation, D.W.F.; investigation, D.W.F.; formal analysis, D.W.F.; experimental analysis, D.W.F.; original draft preparation, D.W.F.; Final write-up, N.K.; writing review and editing, N.K.; experimental analysis, N.K.; supervision, N.K.; investigation, T.C.; supervision. All authors have read and agreed to the published version of the manuscript.

Funding: Funding by the School of Mathematics, Computer Science and Engineering at City, University of London.

Acknowledgments: This work is supported by the School of Mathematics, Computer Science and Engineering (SMCSE) at City, University of London.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programme Interface
AUC	Area Under the Curve
AV	Anti Virus
BCS	Binary Cuckoo Search
BN	Bayesian Network
C&C	Command and Control
CF-NCF	Class Frequency-Non-Class Frequency
CNN	Convolutional Neural Network
CS	Cuckoo Search
DLL	Dynamic Link Library
DNA	Deoxyribonucleic Acid
DNS	Domain Name System
DT	Decision Trees

ECU	Electronic Control Unit
FFN	Feed Forward Network
FIN	Finished Flag
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GTB	Gradient Tree Boosting
GWO	Grey Wolf Optimiser
HML	Hybrid Machine Learner
HTTP	Hypertext Transfer Protocol
IDA	Interactive Disassembler
IoT	Internet of Things
IP	Internet Protocol
KNN	K-Nearest Neighbour
LDA	Linear Discriminant Analysis
LMT	Logistic Model Tree
LSTM	Long Short Term Memory
MBR	Master Boot Record
MLP	Multilayer Perceptron
MOGWO	Multi-Objective Grey Wolf Optimiser
NB	Naive Bayes
NBNS	NetBIOS Name Service
NIC	Network Interface Controller
OS	Operating System
PC	Personal Computer
PCAP	Packet Capture
PFE	Programmable Forward Engines
ReLU	Rectified Linear Units
RSA	Rivest–Shamir–Adleman
RST	Reset Flag
SDN	Software Defined Network
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
WEKA	Waikato Environment for Knowledge Analysis
XGB	XGBoost

References

1. De Groot, J. A History of Ransomware Attack: The Biggest and Worst Ransomware Attack of All Time. 2017. Available online:Https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time (accessed on 22 November 2018).
2. Baig, M.; Zavarsky, P.; Ruhl, R.; Lindskog, D. The study of evasion of packed PE from static detection. In Proceedings of the World Congress on Internet Security (WorldCIS), Guelph, Ontario, 10–12 June 2012; pp. 99–104.
3. Zakaria, W.Z.A.; Mohd, M.F.A.O.; Ariffin, A.F.M. The Rise of Ransomware. In Proceedings of the 2017 International Conference on Software and e-Business, ICSEB 2017, Hong Kong, 28–30 December 2017; pp. 66–70.
4. Rieck, K.; Trinius, P.; Willems, C.; Holz, T. Automatic Analysis of Malware Behavior Using Machine Learning. *J. Comput. Secur.* **2011**, *4*, 639–668. [CrossRef]
5. Milosevic, N.; Dehghantanha, A.; Choo, K.K.R. Machine learning aided Android malware classification. *Comput. Electr. Eng.* **2016**, *61*, 266–274. [CrossRef]
6. Anderson, B.; Quist, D.; Neil, J.; Storlie, C.; Lane, T. Graph-based malware detection using dynamic analysis. *J. Comput. Virol.* **2011**, *7*, 247–258. [CrossRef]

7. Kolter, J.Z.; Maloof, M.A. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.* **2006**, *7*, 2721–2744.
8. Honda, T.; Mukaiyama, K.; Shirai, T.; Ohki, T.; Nishigaki, M. Ransomware Detection Considering User's Document Editing. In Proceedings of the 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), Krakow, Poland, 16–18 May 2018; pp. 907–914.
9. Olenick, D. AI Use in Ransomware Attacks and Sextortion Schemes Top Malwarebytes 2018 Report, Malwarebytes. Available online: Https://www.scmagazine.com/home/security-news/malware/ai-use-in-ransomware-attacks-and-sextortion-schemes-top-malwarebytes-2018-report/ (accessed on 1 October 2019).
10. Sgandurra, D.; Munoz-Gonzalez, L.; Mohsen, R.; Lupu, E. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and Use for Detection. Available online: https://arxiv.org/abs/1609.03020 (accessed on 14 December 2020).
11. Shaukat, S.; Ribeiro, V. RansomWall: A Layered Defence System against Cryptographic Ransomware Attacks using Machine Learning. In Proceedings of the 10th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, 3–7 January 2018; pp. 356–363.
12. Hasan, M.; Rahman, M. RansHunt: A Support Vector Machines Based Ransomware Analysis Framework with Integrated Feature Set. In Proceedings of the 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 22–24 December 2017; pp. 1–7.
13. Daku, H.; Zavarsky, P.; Malik, Y. Behavioural-Based Classification and Identification of Ransomware Variants Using Machine Learning. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security and Privacy, New York, NY, USA, 1–3 August 2018; pp. 1560–1564.
14. Takeuchi, Y.; Sakai, K.; Fukumoto, S. Detecting Ransomware using Support Vector Machines. In Proceedings of the 47th International Conference on Parallel Processing Companion, ICPP '18 Comp, Eugene, OR, USA, 13–16 August 2018; ACM: New York, NY, USA, 2018; 6p.
15. Cusack, G.; Michel, O.; Keller, E. Machine Learning-Based Detection of Ransomware Using SDN. In Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks &Network Function Virtualization, SDN-NFV Sec'18, Tempe, AZ, USA, 19–21 March; pp. 1–6.
16. Alhawi, O.M.K.; Baldwin, J.; Dehghantanha, A. Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection. In *Cyber Threat Intelligence. Advances in Information Security*; Dehghantanha, A., Conti, M., Dargahi, T., Eds.; Springer, Cham, Germany, 2018; Volume 70.
17. Almashhadani, A.O.; Kaiiali, M.; Sezer, S.; O'Kane, P. A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware. *IEEE Access* **2019**, *7*, 47053–47067. [CrossRef]
18. Poudel, S.; Subedi, P.; Dasgupta, D. A Framework for Analyzing Ransomware using Machine Learning. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bengaluru, India, 18–21 November 2018.
19. Chang, J.-C.; Wan, Y.-L.; Chen, R.-J. Feature-Selection-Based Ransomware Detection with Machine Learning of Data Analysis. In Proceedings of the 2018 3rd International Conference on Computer and Communication Systems (ICCCS), Nagoya, Japan, 27–30 April 2018.
20. Lee, K.; Lee, S.-Y.; Yim, K. Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems. *IEEE Access* **2019**, *7*, 110205–110215. [CrossRef]
21. Khan, F.; McNube, C.; Lakshmana, R.; Kadry, S.; Nam, Y. A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning. *IEEE Access* **2020**, *8*, 119710–119719. [CrossRef]
22. Chen, L.; Yang, C.-Y.; Paul, A.; Sahita, R. Towards resilient machine learning for ransomware detection. In Proceedings of the KDD 2019, Anchorage, AK, USA, 4–8 August 2019.
23. Bae, S.I.; Lee, G.B.; Im, E.G. Ransomware detection using machine learning algorithms. In *Concurrency and Computation: Practice and Experience*; Wiley: Hoboken, NJ, USA, 2019; Volume 32.
24. Hwang, J.; Kim, J.; Lee, S. Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques. *Wirel. Pers. Commun.* **2020**, *112*, 2597–2609. [CrossRef]
25. Zuhair, H.; Selamat, A.; Krejcar, O. A Multi-Tier Streaming Analytics Model of 0-Day Ransomware Detection Using Machine Learning. *Appl. Sci.* **2020**, *10*, 3210 [CrossRef]
26. Tseng, A.; Chen, Y.; Kao, Y.; Lin, T. Deep Learning for Ransomware Detection. Available online: https://www.semanticscholar.org/paper/Deep-Learning-for-Ransomware-Detection-Aragorn-Yun-chun/cc3a41b37230861cfe429632744e0d1db19256b7 (accessed on 14 December 2020).

27. Maniath, S.; Ahok, A.; Poornach, R.P.; Sujadev, V.G.; Sankar, P.; Jan, S. Deep Learning LSTM based Ransomware Detection. In Proceedings of the Recent Developments in Control, Automation & Power Engineering (RDCAPE), Noida, India, 26–27 October 2017; pp. 442–446.
28. VinayKumar, R.; Soman, K.P.; Senthil Velan, K.K.; Ganorkan, S. Evaluating Shallow and Deep Networks for Ransomware Detection and Classification. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017.
29. Guarnieri, C.; Tanasi, A.; Bremer, J.; Schloesser, M. *Cuckoo Sandbox Book*; Cuckoo Foundation: Berlin, Germany, 2018.
30. Weka 3: Machine Learning Software in Java. 2018. Available online: https://www.cs.waikato.ac.nz/ml/weka/index.html (accessed on 21 November 2018).
31. Yanguan, S.; Liu, J.; Shen, J. The Further Development of Weka Base on Positive and Negative Association Rules. In Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation, Changsha, China, 11–12 May 2010; pp. 811–814.
32. Alernative.me. Who Uses Scikit-Learn, Scikit-Learn. Available online: https://scikit-learn.org/stable/testimonials/testimonials.html (accessed on 11 October 2019).
33. Zimba, A. Malware-Free Intrusion: A Novel Approach to Ransomware Infection Vectors. *Int. J. Comput. Sci. Inform. Secur.* **2017**, *15*, 317–325.
34. CyberPedia. What Is an Exploit Kit. 2018. Available online: Https://www.paloaltonetworks.com/cyberpedia/what-is-an-exploit-kit (accessed on 21 November 2018).
35. Liska, A.; Gallo, T. *Ransomware: Defending Against Digital Extortion*, 1st ed.; O'Reilly M., Ed.; O'Reilly Media: Sebastopol, CA, USA, 2017.
36. Sophos Knowledge Base: Ransomware: How an Attack Works. 2016. Available online: Https://community.sophos.com/kb/en-us/124699 (accessed on 21 November 2018).
37. Taile, J.P.; Patel, A.D. A Comprehensive Survey: Ransomware Attacks Prevention, Monitoring and Damage Control. *Int. J. Res. Sci. Innov.* **2017**, *4*, 2321–2705.
38. Nassi, N.; Shamir, A.; Elovici, Y. emphOops!...I Think I Scanned a Malware. *arXiv*, **2017**, arXiv:1703.07751.
39. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2006.
40. Wen, L.I.; Lingdi, P.; Wu, C.; Ming, J. Distributed Bayesian Network Trust Model in Virtual Network. In Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, Wuhan, China, 24–25 April 2010; pp. 71–74.
41. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
42. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. Tensorflow: A system for large-scale machine learning in OSDI. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
43. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
44. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
45. Meola, A. How 5G & IoT Technologies Are Driving the Connected Smart Vehicle Industry. 2020. Available online: Https://www.businessinsider.com/iot-connected-smart-cars?r=US&IR=T (accessed on 1 November 2020).
46. Dickson, B. The IoT Ransomware Threat Is More Serious Than you Think. 2019. Available online: Https://www.iotsecurityfoundation.org/the-iot-ransomware-threat-is-more-serious-than-you-think/ (accessed on 1 November 2020)