Computer Science Review 47 (2023) 100529

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

A comprehensive survey on deep learning based malware detection techniques

Gopinath M., Sibi Chakkaravarthy Sethuraman (Ph.D.)[*]

Center of Excellence, Artificial Intelligence and Robotics (AIR) & Center of Excellence, Cyber Security and School of Computer Science and Engineering VIT-AP University, Andhra Pradesh, India

A R T I C L E I N F O

Article history:
Received 18 September 2021
Received in revised form 7 December 2022
Accepted 8 December 2022
Available online 21 December 2022

Keywords:
Malware detection
Deep learning
Mobile malware
IoT malware
Windows malware
APTs
Ransomware

A B S T R A C T

Recent theoretical and practical studies have revealed that malware is one of the most harmful threats to the digital world. Malware mitigation techniques have evolved over the years to ensure security. Earlier, several classical methods were used for detecting malware embedded with various features like the signature, heuristic, and others. Traditional malware detection techniques were unable to defeat new generations of malware and their sophisticated obfuscation tactics. Deep Learning is increasingly used in malware detection as DL-based systems outperform conventional malware detection approaches at finding new malware variants. Furthermore, DL-based techniques provide rapid malware prediction with excellent detection rates and analysis of different malware types. Investigating recently proposed Deep Learning-based malware detection systems and their evolution is hence of interest to this work. It offers a thorough analysis of the recently developed DL-based malware detection techniques. Furthermore, current trending malwares are studied and detection techniques of Mobile malware (both Android and iOS), Windows malware, IoT malware, Advanced Persistent Threats (APTs), and Ransomware are precisely reviewed.

Contents

1. Introduction 2
2. Malware detection and techniques 3
   2.1. Malware obfuscation methods 3
   2.2. Sandboxing techniques 3
   2.3. Data mining models and datasets 3
   2.4. Currently trending prominent malware types 5
3. Malware detection and classification approaches 5
   3.1. Static analysis 5
   3.2. Dynamic analysis 6
   3.3. Image processing 6
4. Deep Learning models 6
   4.1. Deep neural network 6
   4.2. Convolutional neural network 7
   4.3. Recurrent neural network 8
5. Machine learning based malware detection techniques 8
   5.1. Sandboxing based malware detection 8
      5.1.1. Static analysis based malware detection techniques 8
      5.1.2. Dynamic and hybrid analysis based malware detection techniques 9
      5.1.3. Image based malware detection techniques 10
   5.2. Mobile malware detection 11
      5.2.1. Android malware detection techniques 12
      5.2.2. iOS malware detection techniques 14
   5.3. Windows malware detection techniques 15

[*] Corresponding author.
E-mail address: sb.sibi@gmail.com (S.C. Sethuraman).

https://doi.org/10.1016/j.cosrev.2022.100529
1574-0137/© 2022 Elsevier Inc. All rights reserved.

5.4. IoT malware detection techniques ... 16
5.5. APT detection techniques ... 18
5.6. Ransomware detection techniques ... 19
6. Deep learning based malware detection techniques ... 21
6.1. Sandboxing based malware detection ... 21
6.1.1. Static analysis based malware detection techniques ... 21
6.1.2. Dynamic and hybrid analysis based malware detection techniques ... 22
6.1.3. Image based malware detection techniques ... 23
6.2. Mobile malware detection ... 24
6.2.1. Android malware detection techniques ... 24
6.3. Windows malware detection techniques ... 25
6.4. IoT malware detection techniques ... 26
6.5. APT detection techniques ... 28
6.6. Ransomware detection techniques ... 29
7. Inference ... 29
8. Conclusion ... 30
Declaration of competing interest ... 30
Data availability ... 30
Acknowledgments ... 30
References ... 30

1. Introduction

Malware or malicious software is intentionally developed to cause harm to digital devices. It is referred to by a variety of different names, such as Trojan horse, worm, virus, bots and botnets, ransomware, adware, and spyware. Attackers target individual computers and networks which eventually leads to an increase in security loopholes. Privacy and personal information leakage are among the main problems faced by legitimate users. The damage incurred by cyber frauds has doubled as compared to traditional frauds [1]. There has been a significant increase in financial losses due to various attacks such as phishing, pharming, and misuse of payment cards. Financial damage done by malware all over the world is increasing proportionally. On a daily average, 10 lakh malwares are generated as per scientific and business reports, and cybercrime cost is estimated to be 6 trillion dollars in the year 2021 [2]. According to the McAfee Covid-19 threat report, in the past 4 quarters malware attacks have increased by 1902% and 375 new cyber-crimes per minute have been recorded worldwide [3]. During the period 2019 to 2024, the market for malware analysis is expected to increase from 3 billion USD to 11.7 billion USD at a 31% CAGR [4]. AV-Test Institute's recent estimate reports that more than 3.6 lakhs of new malwares are produced daily, approximately 4.2 malwares each second [5]. Every year, the growth of malware surges by 100 million based on the report of the past 5 years. To provide a comprehensive response to cyber threats, which is challenging and dangerous, several security firms are implementing various tools. Researchers use Machine Learning and Deep Learning approaches to create an effective model to resolve these issues by involving detailed studies [6]. Malware mitigation techniques are also becoming more sophisticated with the increase in malware diversity.

Earlier, malwares were written using simple codes and were detected easily. Whereas nowadays, malware creators increase the complexity of code as a result even advanced techniques fail to detect them. It is difficult to identify the next-generation malwares compared to traditional malwares which is designed to execute on the kernel. These types of malwares easily escape from security systems like firewalls and antivirus. They persist in a system or network permanently, spread through multiple extensions, and attack the targeted user. There are various kinds of malware detection approaches based on miscellaneous features like the signature, heuristic, behaviour, model checking, cloud, mobile devices, Internet of Things (IoT), Machine Learning, and Deep Learning [7]. Initially, methods for detecting malware that uses signatures are frequently employed. Despite being quick, it cannot identify sophisticated malware [8]. Signature-based traditional methods like pattern matching failed to meet the requirements of malware detection [9]. Consequently, it is a nightmare for both end users and security providers to implement advanced security approaches [10]. Further new techniques are proposed based on various features like heuristics and model checking. Advanced data mining and Machine Learning algorithms are incorporated within those techniques to detect malware.

The recent technologies deliver higher efficiency than traditional technologies. Still, it is difficult to conclude malware detection based on any one of these approaches, as they are not fully effective in combating new-generation malware. By automatically extracting and statically analysing the characteristics of Application Programming Interface (API) function calls where various dangerous features of API calls are employed, malwares are discovered. [11]. To analyse harmful behaviour, this method entails four steps; unpacking malware, retrieving an assembly program, extracting API calls, and mapping API call with Microsoft Developer Network (MSDN) library to evaluate dangerous behaviour. Additionally, an advanced new method is implemented with five steps which makes use of Machine Learning algorithms like Support Vector Machine (SVM) with n-gram features and 10-fold cross-validations [12]. For the detection and classification of zero-day malware, a framework is implemented with the help of various traditional Machine Learning classifiers [13]. To understand the behaviour of malware, both static and dynamic analysis approaches are utilized to extract features such as Windows API calls, from the malware. For profiling and classifying the behaviours of malware, resemblance-oriented mining and Machine Learning algorithms are used [14]. It is dependent on the sequence of API calls and their respective frequency. True and false-positive rates are achieved as 0.94 and 0.051 respectively from the result of detecting malware.

To achieve high accuracy in malware detection, a hybrid wrapper filter model is proposed [15]. The highest significance and lowest redundancy are considered in it with the help of the SVM wrapper. It is a fully automated framework and employs a signature-free method to detect malware that evades identification by methods relying on signatures. Alazab et al. describe many types of obfuscated malware and their development in depth, as well as the significance of anomaly detection [16]. Neural networks are incorporated to detect malware where domain knowledge is taken from Portable Executables (PE) header and networks learn entirely from raw bytes [17]. The challenges and many issues with the dynamic analysis approach are covered extensively in [18]. A reinforcement learning agent is also employed to operate on PE files [19]. Knowledge extraction of raw data can be either structured or unstructured where data science approaches are utilized [20]. Deep Learning comprising multiple processing layers and deep convolutional networks provides possibilities to process different forms of files like text, image, video, audio, and PE [21].

This paper presents a detailed survey of recent Deep Learning-based malware detection techniques. The progression of innovations from the beginning to the present is demonstrated. To improve the study, the benefits and shortcomings of each technique are also explored. The major contributions of this paper are as follows

- It provides an advanced and detailed review of the evolution of Deep Learning-based malware detection techniques from the traditional levels. It also elaborates the discussion based on sandboxing approaches, Deep Learning models, and current trending malwares such as Ransomware, APT and traditional malwares like IoT, Windows, Android, and iOS.
- There have been several reviews carried out on malware detection techniques in recent days [7,22,23]. In these works, there are a few limitations on focuses like file-less malware and APTs including those which were not covered even in a single deep learning technique [23]. Review works based on Machine Learning and Deep Learning are carried further but those are not mapped with trending malwares like Ransomware, APTs, etc. whereas this survey correlates them [24–26].
- Research gaps in the recent mitigation techniques are also discussed which will be helpful for the enhancement of future reviews. Taxonomy is proposed in Fig. 1 derived from recent research works and the relationship between these works is highlighted.
- Finally, the claim for contribution is extended as this survey will guide researchers in the right direction for building mitigation techniques for traditional and advanced malwares. Also, it will be helpful for those who are working in Deep Learning-based malware detection and prevention.

The rest of the paper is categorized into the following sections: Section 2 provides brief information about malware obfuscation methods, datasets, sandboxing techniques and currently trending prominent malware types, Section 3 details malware classification approaches, Section 4 describes various Deep Learning models for malware detection, Section 5 explores Machine Learning-based malware detection techniques on the basis of Sandboxing techniques, malware detection techniques of Mobile malware including Android and Mac OS, Windows malware, IoT malware, APT, and Ransomware, In a similar vein, Section 6 examines Deep Learning-based malware detection techniques based on Sandboxing techniques, malware detection techniques for Mobile malware, including Android and Mac OS variants, Windows malware, IoT malware, APT, and Ransomware, Section 7 details the inferences observed from this survey and finally the conclusion of this review work is presented in Section 8.

2. Malware detection and techniques

2.1. Malware obfuscation methods

A virus was the first type of malware to be discovered around the end of the 1980s, and studies show that finding viruses is a difficult task which is also non-Polynomial complete [17,27–32]. Malware writers are utilizing Antivirus evasion (AV) techniques to avoid antivirus software-related applications. Likewise, security researchers and penetration testers also use such techniques for security implementation. Attackers leverage static as well as dynamic AV techniques to exploit the target machines by executing generated payloads. Static analysis bypasses antivirus signature scanning algorithms used for malware detection. Dynamic execution avoids behaviour detection during sample execution. Malware samples are executed within a sandbox or AV emulator. The advanced next-generation malware adopts obfuscation methods like encryption, oligomorphic, metamorphic, stealth, and packaging. These obfuscation methods help malwares to evade malware detection techniques. Using encryption techniques, malware can be concealed somewhere inside the entire program [18]. Encryption is also used for packaging. It allows malwares to remain hidden from detection techniques by creating four diverse types of packers like compressors, crypters, protectors, and bundlers [20,21].

In the oligomorphic method, two different keys are used with a payload of malware whereas separate keys are used for performing encryption and decryption [33]. This makes malware detection harder. The polymorphic method is similar to oligomorphic method; the dichotomy between them is taking more copies of the encrypted payload. Thus, it makes detecting malware even more complex [34,35]. While incorporating the metamorphic method, the dynamicity of malicious code is enabled for each iteration of the malicious process [19]. The stealth method hides malware from a secure system by adopting several counter methods like code modification and data encryption [33].

2.2. Sandboxing techniques

Malware detection is a decision-making process. At the end of this process, the malicious program is identified. Malware researchers utilize a sandbox environment to execute malicious code obtained from unknown attachments or suspicious URLs for observing the behaviours of malware code. As a sandbox is an emulated environment, it is easy to observe the suspicious code without acquiring access to data, network and other applications. It acts as quarantine over unknown emails and attachments and plays a key role in isolating suspicious executable programs. Analysing the malware, extracting the features and classification are the three basic steps of the detection process. While analysing malwares, it is important to study their behaviour. The working of malwares and the damage caused by them are studied extensively in [36,37]. Static and dynamic analyses are the two momentous techniques for analysing malwares [37]. Static analysis examines the executable file without running the program to analyse behaviours. The dynamic analysis evaluates behaviours at the time of execution of the program and it is performed in a Virtual sandboxed environment. The malware analysis cycle starts with simple static analysis and ends with an efficient dynamic analysis.

2.3. Data mining models and datasets

Data mining methods are utilized for extracting the features of malware. It creates more semantic information from large-scale datasets. Recently developed and significant data mining models for generating datasets and features include the N-gram model and the graph model [38].

n-gram model: Features are generated based on characteristics of static and dynamic analysis. APIs and system calls are combined to provide features.

graph model: Graph G(V,E) is generated from the system calls in which V denotes nodes that specify system calls and E

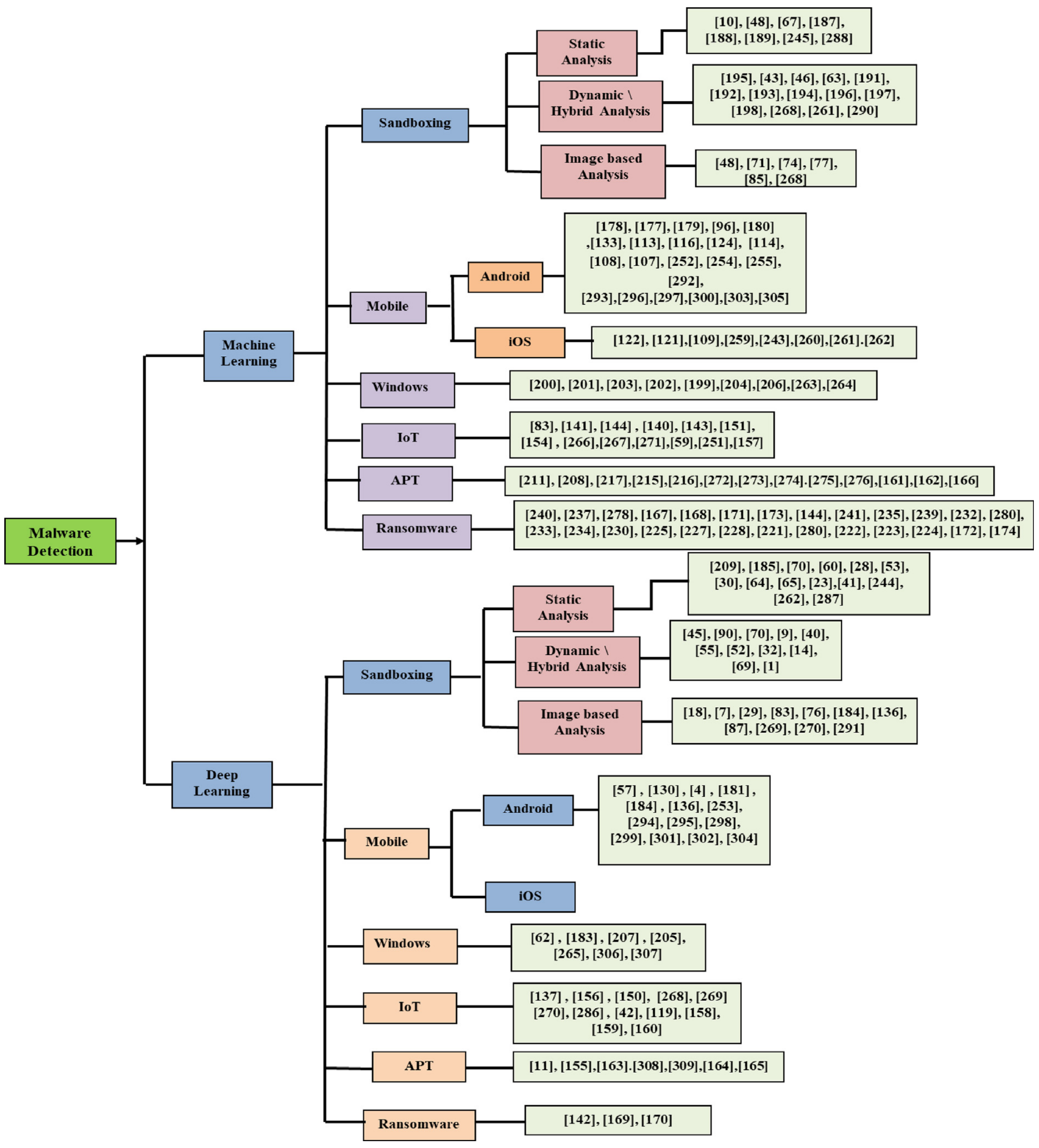

Fig. 1. Taxonomy of research.

is denoted as the edges helpful for relating these system calls. Sub-diagram is needed when the size is increased and it is Non-deterministic Polynomial (NP) complete.

Malware dataset is a key factor for malware detection. The formats of existing datasets can occasionally make it challenging to use them for mining. NSL KDD, Drebin, MS malware classification challenge, ClaMP, AAGM, and EMBER are some examples of datasets used so far. In the classification stage, Machine Learning algorithms are used. Machine Learning is applied for carrying out operations like classification, regression, and clustering over data. Bayesian Network (BN) [39], Naive Bayes (NB) [40], C4.5 Decision Tree variant (J48) [41], Logistic Model Trees (LMT), Random Forest Tree (RF) [40,42,43], K-Nearest Neighbour (KNN) [44,45], Multi Layer Perceptron (MLP) [46,47], Simple Logistic Regression (SLR), Support Vector Machine (SVM) [48,49], and Sequential Minimal Optimization (SMO) [50] are examples of Machine Learning classifiers that utilize behaviour to make predictions.

The NSL KDD-2009 dataset is used for IDS purposes and has approximately 1, 25,000 records and 41 features [28]. The Drebin-2014 dataset has 5560 malwares over 20 families and it is used for smartphone purposes [51]. The MS Malware Classification Challenge-2015 dataset contains disassembly code and byte code for 20,000 different types of malware [52]. The ClaMP-2016 dataset comprises 5184 records and 55 properties [53]. The AAGM-2017 dataset is associated with Android malware which has 400 malwares from 12 families [54]. The EMBER-2018 dataset is created with more than 1 million records covering the features of benign and malware for detecting Windows malware files [55]. There are several datasets created by the Canadian Institute of Cyber security (CIC) [56]. CIC-MalMem-2022, an obfuscated malware dataset, was developed by implementing memory-based detection methods. It contains 29,298 benign and 29,298 malware samples. Android malware dataset CCCS-CIC-AndMal-2020 has a total of 400k apps in which 200k samples are benign and 200k samples are malware.

2.4. Currently trending prominent malware types

In the year 2019, 4,368,921,256 records were assessed by security experts [5]. 10 lakh spam messages and 5 lakh URLs are evaluated. Around 3 million files were tested every day with a size of 2500 TB. In earlier 2020, the malware growth reached over 43 million which implies the most dangerous scenario in 2020 with a development of 4.2 samples per second and most attacks are targeted at the Windows operating system [5]. Global malware strikes reached 2.8 billion in the first half of 2022, up 11% year to date from 2021, according to threat experts at SonicWall Capture Labs [57]. Malware exploitation is concentrated in two different areas by the attackers. Initially, by creating automated bulk malware, widespread web attacks are targeted. Further, sophisticated malware is created to target specialized attacks, which use a specially created attack tool based on previously determined victim infrastructure. Microsoft systems are the most targeted for launching attacks by cybercriminals, according to statistics [5].

Trojans

Trojans are a type of malicious software that typically deceives people into thinking they are not at all harmful. Trojans usually spread through spam emails and downloadable files by hiding themselves. Trojans are usually spread by visiting contaminated websites, opening bulky spam mail, downloading illegitimate apps and software, and running suspicious files of movies and songs. Also, there are possibilities from QR coders, storage devices like USB drives, and external hard disks. Once an adequate count of infected systems is identified, attackers will proceed with special codes. According to ThreatFabric, more than 3 lakh customers of the Google Play store contracted Mobile banking Trojans in 2021.

Ransomware

Ransomware malware denies access to the device and private files and demands a ransom fee which should be paid to gain access back. Several types of ransomware are available like Android ransomware, Mac ransomware, Scareware, and so on. They aim to deny users or organization access to files on their computers. Ransomware became a profitable source of business for cybercriminals in 2019, tripled in 2020 and reached a peak with 900,000 malware samples [58]. In the first half of 2022, 23% drop in ransomware attacks globally [57].

Advanced Persistent Threats (APT)

APTs utilize unbroken, stealthy, and complicated approaches to hacking to get permission into the system and stay within the extended period in the company of high-level vulnerabilities [5]. The enormous count of APTs is a challenging task to tackle in the security system. APTs are planned attacks and are purposely targeted towards multiple firms and institutions for obtaining valuable information and also, including target areas of public, financial, and research.

Potentially Unwanted Applications (PUA)

Potentially Unwanted Applications (PUA) is another kind of malware attack in the digital world [5]. It is spyware that is intentionally installed in devices along with the package of software while the apps are downloaded. Advertisement industries make use of PUAs for the detection and investigation of individual behaviour for obtaining information. PUAs target users with personal advertisements by injecting unnecessary secret questions.

3. Malware detection and classification approaches

Today malware is coded to present features from multiple families at the moment of discovery, making malware classification extremely challenging. As advanced malwares are executed in the kernel it is hard to discover them when compared with classical malwares. These types of malwares easily escape from security systems like firewalls and antivirus. In this section, the advantages and drawbacks of classification techniques for detecting malware based on static and dynamic analysis are discussed. Further for malware detection, image processing-based techniques are also utilized with big data to enhance the visualization of data and efficient decision-making.

3.1. Static analysis

Earlier malware classification approaches used various features like n-gram which includes multi-byte identifiers and strings [53]. Sequence CNN is applied for classification where features are extracted from the bytes of binaries [35]. Ember large-scale dataset is used for training ML models and detection of PE files [54]. It has 1.1 million binary files in which 900k are training samples and 200k are testing samples. Other than a benchmark dataset, it is useful for end-to-end deep learning. A byte/entropy 2histogram is computed for malware detection using deep neural networks [58].

Han et al. implemented a program profiling model MalInsight for detecting malware. Profiling is performed by considering the structure, low-level, and high-level behavioural features of malwares [42]. Structural information is obtained from basic structural profiling. Operations between programs and OS lead to low-level profiling and the operations on files, registry, and network lead to high-level profiling. Machine Learning-based classifiers like KNN, DT, RF, and Extreme Gradient Boosting are trained by the resultant feature set to detect or classify malwares. Evaluation is carried over the dataset which includes 4250 samples from operating systems like VirusShare and Windows 7 Pro. It detects new malware types with 97.21% accuracy as per the results.

Kim et al. implemented a multimodal deep learning scheme to detect Android mobile malwares [59]. It used many features like string, method opcode, method API (Application Programming Interface), function opcode of the shared library, permission, component, and environment. Various features are extracted, and a feature vector was generated which was used to provide training to the initial networks. The final network was trained by using the results of the initial networks. An evaluation was carried out with 41,260 samples.

Fang et al. built Deep Q learning-based Evading Anti-malware engines (DQEAF) framework with the help of reinforcement learning for exposing and demonstrating the weakness of supervised learning-based malware detection models [60]. DQEAF trained an Artificial Intelligence (AI) agent, the default primary component for interacting with malwares through neural networks, which plays a vital role in choosing the optimal sequence of safety actions by incorporating reinforcement learning. Results of DQEAF show a 75% success rate which is high in Portable Executable (PE) samples. Later, they introduced Deep Q learning-based Feature Selection Architecture (DQFSA), a further architecture for feature selection using reinforcement learning [40]. An AI agent interacts with the samples' feature space to obtain the optimized reasonable features continuously where human intervention is avoided. Classifiers like KNN, DT, RF, NB, and SVM are utilized, and better accuracy is obtained with them than with existing architectures.

3.2. Dynamic analysis

RNN is trained for extracting process behaviour features after which CNN is trained for categorizing images [28]. For the projection stage, Echo State Network (ESN) and RNN are used where features are extracted by training 150k samples. For detecting malware and benign, a shallow multi-task deep learning architecture is proposed [38] based on natural language modelling in which the language of malware is trained to extract features. To get optimal malware classification results convolutional and neural networks' recurrent layers are combined which improves modelling and classification of system call sequence [52] and outperforms Hidden Markov Models and SVM.

An RNN-based malware prediction model is proposed with two various datasets and malicious files are predicted before the execution of the payload to prevent attacks [34]. The performance of the classical Machine Learning classifiers is also evaluated. Results have shown 94% accuracy with an execution time of 5 s. Behavioural data collected through dynamic analysis of all the models are examined and Hidden Markov Model is trained by employing both static and dynamic analysis. Comparison of features is performed to detect In order to identify malware families, comparisons of attributes are made, and the results show that the dynamic approach performs better [61].

A Multi-Level Deep Learning System (MLDLS) was devised by Zhong et al. to improve the efficiency of Deep Learning-based Malware Detection Systems (MDSs) [62] that utilized a tree structure to coordinate multiple deep learning models. Instead, each deep learning model focuses on learning a particular data distribution for a particular class of malware, and all deep learning models in the tree collaborate to reach a decision. Although the network behaviour is impacted, Shibahara et al. developed a method that improves the effectiveness of dynamic analysis [51]. This approach is focused on identifying the two unique aspects of malware communication, which are variations in the communication mechanism and latent function. The same data structures in malware communication and natural language are monitored and evaluated with 29,562 samples of malware.

Vinaya Kumar et al. built a malware detection framework that consists of two stages [63]. In order to create a highly scalable malware detection framework, classical Machine Learning methods as well as deep learning frameworks based on static, dynamic, and image processing are explored. Initially, malware classification is performed by implementing static and dynamic analyses. Using the image processing method, malwares are grouped into their respective families in the second stage. The resilience of the suggested framework in the work [64] must be taken into account since it is intended for an efficient zero-day malware detection mechanism.

3.3. Image processing

Today detecting malicious files by examining large-sized data has become a difficult challenge for security providers. For analysing vast amounts of malwares, a malware classification model is proposed by utilizing deep learning which is based on images [65]. Orthogonal methods for malware detection are based on signal and image processing. Azmoodeh and Choo presented a scheme to detect malwares for the Internet of Battlefield Things (IoBT) devices by implementing deep learning through the sequence of the device's opcode.

Li et al. designed a Machine Learning framework for detecting malwares based on Domain Generation Algorithm (DGA) [8]. Many DGA domains are classified using the deep learning method by collecting one year's real-time traffic data. This framework is made with a two-level model and a prediction model. DGA domains are segregated from normal domains in the two-level model where the clustering method is used for identifying algorithms to generate DGA domains. Prior to designing a time sequence model oriented on a Hidden Markov Model (HMM) to anticipate incoming domain features, a Deep Neural Network (DNN) system is adopted in the prediction model. Classification by this machine learning framework's result is shown to have an accuracy of 95.89%.

Similarities in the structure of malwares are utilized in the works [66,67]. Signal and image processing methods are implemented for detecting and classifying malware where malwares are treated like images or signals. This method, which is based on meta-learning, is expanded for forensics and the classification of data types. Search and RetrieVAl of Malware (SARVAM) is presented to perform searching and retrieving online malwares [68]. Executable binary files can be uploaded by any person and can be searched from the database of 70 lakh samples of malware with the help of Image Similarity metrics.

An automated malware classification method proposed in the work [69] uses image texture analysis. The outcomes demonstrate more effectiveness than dynamic analysis using a sizable dataset with over 685k malwares. For quick recognition of packed executables, core binary data is evaluated and SVM is used for both training and testing purposes [70]. By expressing binary bytes as audio signals, an approach for detecting malware is proposed [71]. Music Information Retrieval (MIR) techniques are then used to identify musical patterns. Further, ML classifiers are used. A slight change in the original code is made as malicious by malware authors. These minor changes are captured as images and similar images are formed as a family. Features are extracted by classification or clustering techniques.

Image-based malware analysis is fast and provides a lot of information regarding the structure of malware. SigMal [72] is a fast signal processing technique applicable for both packed and unpacked samples which is a resemblance-based framework for malware detection. Signal processing-based features are improved by the heuristics of PE structure information. Malware detection techniques use a variety of image-based malware datasets, including Malimg, VGG16, and ImageNet [37]. The accuracy of VGG16 with SVM is higher than VGG16 with scratch [36]. Later, theYongImage method [73] is implemented for detecting malwares.

4. Deep Learning models

Machine Learning, which is employed in conjunction with artificial intelligence technology and relies on samples for learning, gives way to Deep Learning. The process of extracting features from an input involves establishing numerous layers in a hierarchy. There are many applications of Deep Learning like automated cars, image processing, and language processing. Deep Learning models are built based on either Supervised or Unsupervised Learning [8,13,27,35,36,52,53,74–79]. They differ based on the training methodology. Supervised model training is carried through the examples of specific datasets whereas only input data is given for unsupervised models. Here, deep neural networks, convolutional neural networks, and recurrent neural networks are also discussed which are useful for real-time environments. Fig. 2 shows the basic Deep Learning architecture.

4.1. Deep neural network

A deep neural network is an artificial neural network that is incorporated with many layers amid input layers and output layers of the network. Sample DNN with n hidden layers (h_1 to h_n) is shown in Fig. 3. By using random projections, the input size

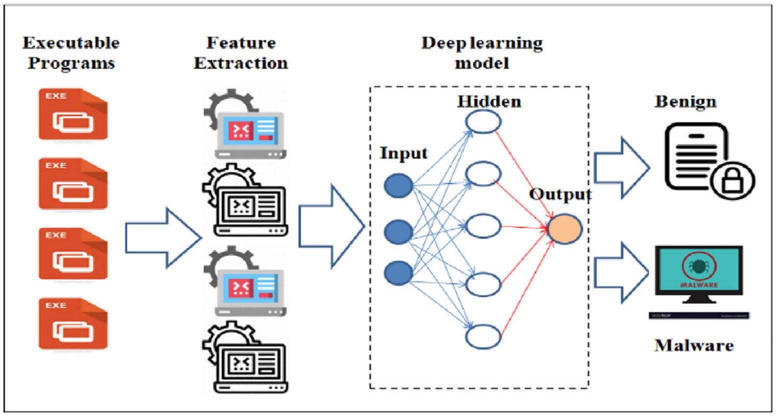

Fig. 2. Deep Learning architecture.

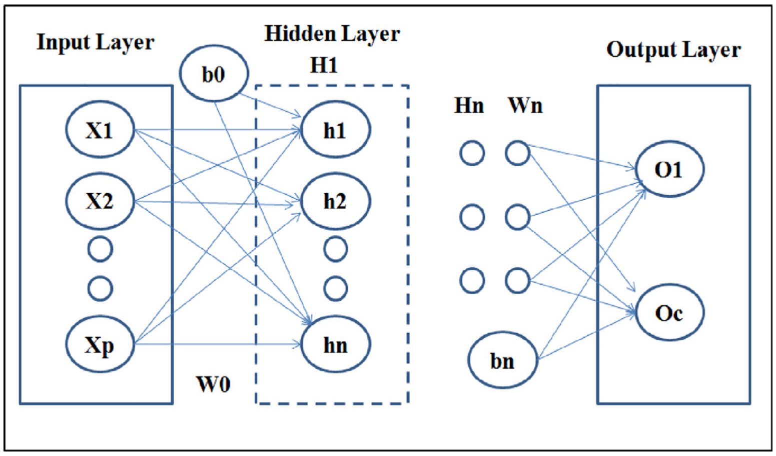

Fig. 3. Deep Neural Network (DNN).

of the large-scale network is reduced with a factor of 45 over 2.6 million samples. It shows the result in terms of two-class error rate as 0.49% and 0.42% for solo neural networks and ensemble of neural networks respectively [80]. The accuracy of detection does not depend on incrementing the hidden layers. Two and more layered neural networks result in lesser performance than single-layered neural networks.

Droid-Sec makes use of deep learning in which results are compared with earlier 5 machine learning models like SVM, C4.5, Naïve Bayes, LR, and MLP [81] and is designed for Android malwares. Implementing a Deep Belief Network (DBN) allows for the creation of the deep learning model [82]. In addition, Restricted Boltzmann Machines (RBM), components, and algorithms are taken into account. Comparing the results to those of more established techniques like SVM, C4.5, and Naive Bayes, the results obtained are more promising.

The features of 2D binary programs are used for detecting malwares that use a deep neural network [58]. The whole framework is separated into 3 stages. In the first stage, four various features are extracted from benign and malicious binaries. The deep neural network is built in the second stage with an input layer, an output layer, and 2 hidden layers. The third one is a score calibrator in which the probability of malware files and is measured by translating the outputs of the neural network. Over 4 million software binary files, 95% DR (Detection Rate), and 0.1% FPR (False Positive Rate) were attained in this. Cross-validation-based methods show high accuracy compared to split validation methods.that implies multiple benign samples need to be analysed to produce improved accuracy. A deep learning-based model called Multi-Task Neural Network (MtNet) has been proposed to classify malware [38] where benign and malicious are classified through dynamic analysis. 4.5 million files are trained, and 2 million files are tested. MtNet attained a 0.358% binary classification error rate and a 2.94% malware family classification error rate based on the results.

4.2. Convolutional neural network

A convolutional neural network is derived from DNN which is used for enhancing visualization. Fig. 4 depicts the sample CNN. CNN was applied for image recognition that employs a unique feature extractor and image size normalization [83]. CNN has trained over 1.2 million HD images for classification [84]. For faster training, non-saturated neurons, and highly configured Graphics Processing Units (GPUs) are used. Davis et al. applied CNN for malware detection and features were extracted by disassembling byte codes [27]. Similarly, more researchers applied CNN to the byte code of malware to represent 2-Dimensional Grayscale images [29,30,66,68–70]. Each sample is normalized with a size of $32 * 32$ pixels by implementing down sampling.

Expert domain knowledge is not required for CNN-BiLSTM (Bi Long Short-Term Memory) model which depends on data for feature extraction [85]. Similarly, Kolosnjaji et al. proposed a model by merging CNN and LSTM where API call sequences are observed by CNN and the relationships are carried by LSTM [52]. Outputs from the convolutional layers are connected to forward and backward layers of LSTM in the CNN-BiLSTM model [85]. Outputs of those LSTM layers are combined to provide an output.

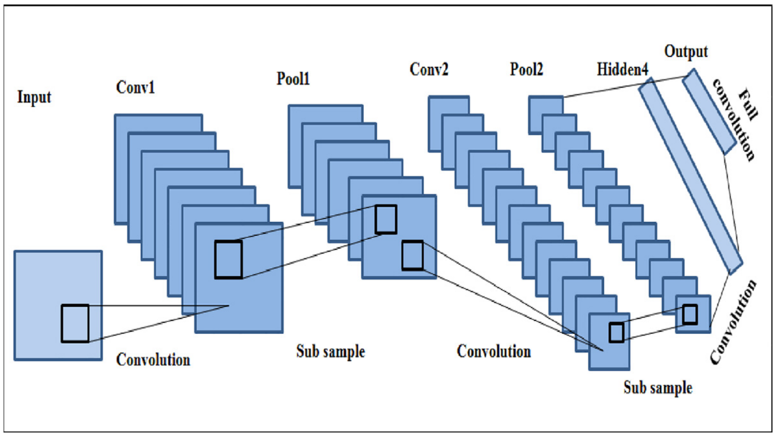

Fig. 4. Convolutional Neural Network (CNN).

CNN-BiLSTM provides the best performance in accuracy when compared with CNN. The embedding layer is incorporated with the neural network whereas max pooling is applied for the convolutional layer. The Convolutional layer learns with n-gram signatures to detect opcodes of malwares [74]. Hierarchical Convolutional Neural Network (HCNN) [13] is proposed for treating the hierarchical configuration of Program Executables that uses CNN mnemonics to record features at the functional level

4.3. Recurrent neural network

RNN is a form of Artificial Neural Network (ANN) which is applied in recognition of speeches along with natural language processing. Sequential events are observed to predict the next action.RNN is used to carry all information from the beginning to the end of time which is referred to as the Long–Short Term Memory Module (LSTM) and is utilized to avoid the gradient problem [86]. Sample RNN is depicted in Fig. 5. For time-based sequential prediction, RNN is employed where features are taken from previous inputs [27]. It has more application areas like speech recognition and synthesis, robot control, machine translation, music composition, etc. CNN is initially employed for chunk analysis.

In API call tracing for detecting malware, RNN with integrated LSTM and GRU, performed better (Gated Recurrent Unit) [87]. It has 2 stages, LSTM and GRU pick up features in the initial stage, and features are categorized in the subsequent stage. Sequential dependencies are modelled in API calls where previous inputs provide output [28]. Gradient problem is avoided by combining 4 layers of LSTM with RNN. Before training CNN, RNN is applied to build a behavioural model with data. And then the output of RNN is transformed into feature images for CNN. RNN is utilized for detecting unknown malware/threats in the IoT environment [88], where opcodes are used for the classification of malwares. Further SVM is employed to assess opcodes and discover the feature set of malware. The use of the n-gram method enables it to achieve accuracy at the level of 96%. True +ve values and false +ve values of malware samples are computed by applying deep learning methods. It gives 98% of malware detection accuracy.

Neural networks are utilized to classify malware into their predefined families. Convolutional and recurrent layers are analysed to model call sequences. The convolutional layer's output is connected to the recurrent layer [52]. On top of CNN, RNN is applied in LSTM in which a feature vector is obtained by considering the whole file contents [85]. Moreover, input to the RNN is classified into 9 classes of malware before feature vector creation by RNN.

5. Machine learning based malware detection techniques

5.1. Sandboxing based malware detection

5.1.1. Static analysis based malware detection techniques

Graph similarities are computed by utilizing an altered histogram to analyse and classify malware [89]. Experimental results show that the accuracy of detection is high. Utilizing control flow graphs, quick malware detection is put into place to find malware and classify it according to its family (CFGs) [90]. Every family's signature is created using the Bloom filter, which also helps to reduce feature complexity and hence the cost of computation. Features are summarized with the help of CFGs to define the similarities. Identical block numbers, edge directions, and sequences of instructions are also considered. For similar and different families, the ratio of similarity is obtained as 0.82315 and 0.01970 respectively. The malware classification method is proposed with the extraction of malware features where every library of malware is counted [91]. Massive amounts of malwares are obtained from McAfee, Kaspersky, Norton, and F-Secure for authenticity purposes. Similarities among malware families are analysed using API calls per DLL (Dynamic Link Library) and library functions are imported for the run time environment. Euclidean distance and multi-layer perceptrons are utilized to classify the different malwares.

Every value printed in the destination registers of runtime traces is utilized for malware classification [92]. Complete profiling is examined to recognize the functions of every program. The similarity of traces is computed by implementing a two-step procedure depending on the values of the sequence. Clustering is applied to group the samples; also, the reuse of code is ensured. Various malware families are differentiated by implementing a method that converts malware programs into a visualized image [93]. Furthermore, visualized images are transferred to entropy graphs for finding the similarities between malware families and classification.

To detect malware and benign, feature extraction is implemented in android applications. Explainable machine learning is utilized to develop a lightweight android malware detection system named PAIRED [94]. Recursive feature elimination (RFE) is utilized for maintaining the minimum possible critical features. For training purposes, 35 static features of applications are extracted. Further making use of Shapley Additive Explanation (SHAP) values, the presented classifier model is explained with the selected features. Based on experiment results more than 98% accuracy is attained while a small footprint is maintained over the device and False Negative rate is obtained as

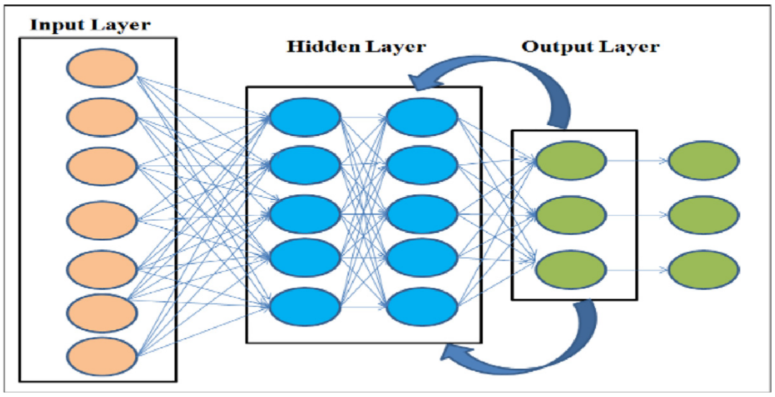

Fig. 5. Recurrent Neural Network (RNN).

Table 1
Comparison of static analysis techniques using ML.

Research paper	Features	Type of analysis	Algorithms of classification
Alani et al. (2022) [94]	RFE, SHAP, Drebin-215 dataset	Static	PAIRED
Fang et al. (2019) [40]	PE files, RL	Static	DT, RF, SVM, DQFSA
Raff et al. (2018a) [53]	Byte sequence	Static	CNN
Davis et al. (2017) [27]	Byte's sequence	Static	CNN with RNN
Ahmadi et al. (2016) [30]	Bytes 1-g, metadata features, entropy statistics, Haralick and Local Binary Pattern 0. features, ASCII strings	Static	Ensemble of Gradient Boosting Trees
Han et al. (2015) [93]	Visualized image, entropy graphs	Static	Clustering
Annachhatre et al. (2015) [92]	Profiling functions, sequence	Static	Hidden Markov models
Gonzalez et al. (2013) [91]	API calls per DLL, library feature	Static	Euclidean distance and artificial neural networks

0090. The Drebin-215 dataset is utilized for experimental purposes. Table 1 compares static analysis-based malware detection techniques using ML.

5.1.2. Dynamic and hybrid analysis based malware detection techniques

Dynamic analysis involves executing malwares in a real-time environment or sandbox-like virtual environment for extracting features. Earlier most of the traditional techniques extract API calls from executing malware processes and used them as input. The ML-based malware detection method is proposed by utilizing Anubis which was earlier called TT analyse for extracting report files of XML [41]. The dataset consists of 220 malicious samples and 250 legitimate samples. Based on the reports, term frequencies are generated which are used as a feature set. Utilization of the J48 model provides a good result with a True +ve rate of 96% and a False +ve rate of 2.4%.

In the work [95], CW Sandbox is utilized for extracting system calls of the dataset which is sized with 3133 samples of malware. From the sequence of system calls, an N-gram feature set is developed. Also, a clustering algorithm is utilized to map malwares with similarities. 33,698 anonymous samples are used to verify this framework, and 99.7% of the F -score is attained. The Android malware classification model [96] is proposed in which API functions of android apps are classified and sources and sinks of android apps are differentiated. It also involves a flow-tracking mechanism. Malware behaviour is discovered by the sources and sinks. A similar rate of precision and recall is attained with 92.3% at the end of the analysis. TrumanBox is utilized [97] to extract the features of the network like request count, data size, and requests received per day for detecting malicious activities of Hypertext Transfer Protocol (HTTP) request communication. Irregularity behaviours of the network like tunnelling and backdoors are identified by using the dataset of 695 samples of malware.

Memory dumps are retrieved from the virtual environment during the execution of malicious and legitimate samples [98]. WinDbg is utilized for redesigning the system calls by tracing them. The Random Forest model is used for testing. RF tests 10 samples which include ransomware and Remote Access Trojan (RATs), and 6 legitimate samples. The experimental result shows a true +ve rate of 98% and a false +ve rate of 0%. Research work is carried out on the application of Machine Learning in Side-Channel Analysis (SCA) [99]. Cryptographic keys of cryptographic devices are extracted by applying SCA. Consumption of power is considered a feature set. The Least Squares Support Vector Machine (LS-SVM) classifier is utilized for malicious classification that obtains a success rate of 75%.

The online malware detection method is implemented over android and Linux OS [100]. 210 legitimate software and 503 malicious samples with 2 rootkits are involved in an investigation with this model. With the help of different algorithms such as decision tree, the rate of detection is attained as 83% and a False +ve rate of 10%. EM-Based Detection of Deviations in Program Execution (EDDIE) is proposed where the emission of electromagnetic IoT devices is monitored and the statistical irregularities are defined with the EM emissions, which are based on the injection of malicious code in the app's runtime environment [101]. Intentions for malevolent conduct will be defeated in this way. In the work [80] neural networks are trained by using the feature of API calls in the hidden layers. Null terminated strings are also extracted for feature extraction from the memory of processes. The projection technique is applied for minimizing the dimension of features and those are managed by the neural network.

A Hybrid IoT botnet detection method is proposed for the Industrial IoT environment [102]. Printable string information (PSI) rooted sub-graph features are generated during static analysis and they are enhanced in dynamic analysis. PSI is utilized for

Table 2
Comparison of recent dynamic and hybrid malware detection techniques using ML.

Research paper	Features	Type of analysis	Algorithms of classification
Nguyen et al. (2022) [102]	PSI rooted sub graph features, Industrial IoT	Both static and dynamic	Hybrid botnet detection
Husainiamer et al. (2020) [103]	Mobile behaviour and vulnerability exploitation	Both static and dynamic	POC
Jeon et al. (2020) [104]	Behaviours associated with memory, process, and system call	Dynamic	DAIMD, CNN
Han et al. (2019a) [42]	Static and Dynamic API sequences	Both static and dynamic	K-NN, RF DT, Extreme Gradient Boosting
Han et al. (2019b) [43]	Sections size of PE, DLL information, APIsequences, IP, DNS, port and domain Request operations	Both static and dynamic	K-NN, RF, DT, Extreme Gradient Boosting
Pekta, s and Acarman (2017) [105]	File system, API call N-grams, network and registry features	Dynamic	PA-I, PA-II, CW, AROW, NHERD
Nissim et al. (2018) [98]	Memory dumps, WinDbg, RATs	Dynamic	Random Forest model
Nazari et al. (2017) [101]	EM emissions,	Dynamic	EDDIE
Rasthofer et al. (2014) [96]	API functions, android	Dynamic	Flow tracking mechanism
Demme et al. (2013) [100]	Decision tree, android	Dynamic	Online malware detection
Hospodar et al. (2011) [99]	SCA, Cryptographic keys, power consumption	Dynamic	LS-SVM
Schwenk et al. (2011) [97]	HTTP request communication, network behaviours	Dynamic	TrumanBox
Rieck et al. (2011) [95]	CW Sandbox, N-gram feature set	Dynamic	Clustering algorithm
Firdausi et al. (2010) [41]	Anubis, XML reports	Dynamic	J48 model

traversing the static analysis-based graph and graph-based features are useful in classifying the samples of benign and malware. A dataset with the size of 83 430 executable samples is used for experimentation. Among the samples 5531 are IoT botnet samples and 2799 are IoT benign samples. Results of the hybrid method show that 98.1% detection accuracy and 91.99% classification accuracy are attained for IoT botnets. Methods of dynamic and hybrid analysis are compared in Table 2.

5.1.3. *Image based malware detection techniques*

Classification of malwares becomes more crucial in these years. Malware classification is always linked to the process of spotting newly emerging malware. Fig. 6 portrays various malware detection methods utilizing image processing. According to a study by Microsoft, security service providers face a challenging problem when managing vast amounts of data for malware detection [106]. Classical techniques failed to fulfil the demands and requirements while analysing malwares, which is based on signature and behaviour, hence effective methods are necessitated [107]. The core bytecodes of the program are converted into grayscale images where pixel representation is applied for every byte code. Then the sequence of bytes is wrapped artificially to form a two-dimensional array. Classification and clustering methods are used for extracting features and machine learning techniques are applied to them [108].

By utilizing visualization techniques, the classification model for malwares is simplified by adapting image processing techniques [69,108]. Grayscale images are obtained as a result of conversion from malware binaries [108]. GIST descriptor is employed to calculate texture features by decaying images. The K nearest Neighbours algorithm with Euclidean distance is used for classification purposes. To evaluate the performance bigram distributions are computed from core data. The accuracy of malware classification is attained as 0.98 with time consumption of 56 s while considering feature vectors of bigram distributions.

NatarajImage is the first implemented malware embedding method which utilized binary files [108]. The malware binary of a Portable Executable file, which is 8 bits in size, yields a grey picture vector [30]. It is found that the texture pattern of various families of malwares becomes the same in the NatarajImage. Despite NatarajImage's flaws in packing and obfuscation methods, it inspired additional research [30,109–111]. By using a black-and-white image vector obtained from hexadecimal-sized instruction in the disassembly file, a further embedding method is proposed namely AndrewImage for malware detection [27]. Comparing these mentioned embedding methods; AndrewImage provides better performance in terms of robustness and interpretability than NatarajImage in which instruction-level information is embedded. Kaggle dataset is used for malware identification in the work [30]. Features are extracted from raw binary files as well as disassembled files and represented as images. XGBoost classifier is used to extract the features. The accuracy of performance is 95.5% according to the 5-fold validation result.

An automotive framework for finding anonymous vulnerable activities is achieved using methods of neural networks like computer vision and classification of images [76]. Malware binaries are transferred into 8-bit vectors for feature extraction purposes and grayscale images are prepared as input to new training sets of an ML algorithm. For both the training and testing phases of classification, the Malimg dataset is used. The results have an accuracy of 0.0856, an average training time of 0.218 s, and a 0.0211 s testing time while making use of the nearest centroid. The highest result of accuracy is obtained while using random forest i.e., 0.0916 with 1.72 s average training time and 0.0063 s testing time. However, when using different algorithms for classification like perceptron, stochastic gradient, decision tree, and multilayer perceptron, accuracy is obtained as 0.0905, 0.087, 0.088, and 0.087 respectively. Furthermore, classification performance is lacking for stochastic gradient and perceptron.

Time consumption and manpower over feature extraction become a major concern apart from accuracy [30,110]. It causes the training and detection phases to be less effective.

Impressed by NatarajImage, to classify the different malwares, a random forest method is introduced [112]. Lempel–Ziv Jaccard

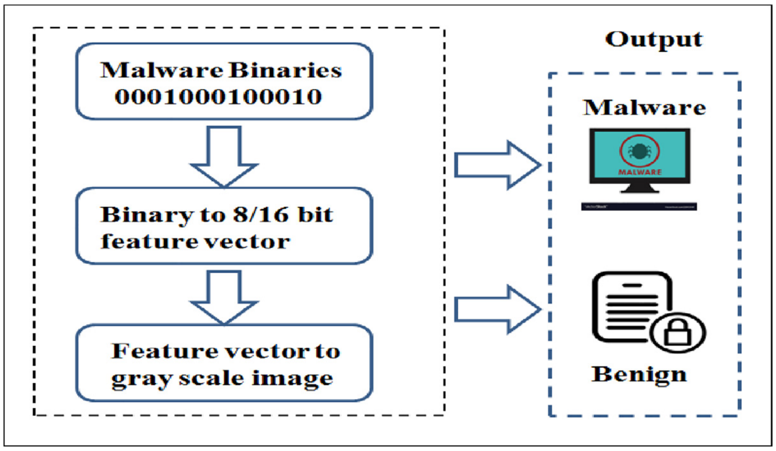
Output
Malware Binaries 0001000100010
Binary to 8/16 bit feature vector
Feature vector to gray scale image
Malware
MALWARE
Benign

Fig. 6. Malware detection process based on image processing.

Table 3
Comparison of recent image-based malware detection techniques using ML.

Research paper	Features	Type of analysis	Algorithms of classification
Jeon et al. (2020) [104]	Behaviour images	Image-based	DAIMD, CNN
Raff et al. (2017) [113]	NCD	Image-based	LZJD
Drew et al. (2017) [114]	Time consumption	Image-based	Strand gen sequence method
Kosmidis et al. (2017) [76]	Grayscale images Malimg dataset	Image-based	Stochastic gradient, DT, and multilayer perceptron
Ahmadi et al. (2016) [30]	Grayscale images Kaggle dataset	Image-based	XGBoo st
Garcia et al. (2016) [112]	Grayscale images	Image-based	RFmethod

Distance (LZJD) is proposed for malware classification from core data, which provides an improvised performance than Normalized Compression Distance (NCD) [113]. The strand gene sequence method is introduced regarding malware classification with an accuracy level of 98.89% and requires minimum time for training [114]. However, when dealing with large-sized malwares, the time consumption for detection is substantially longer. Table 3 contrasts recent image-based malware detection techniques using Machine Learning.

5.2. Mobile malware detection

Usage of mobile devices has increased in everyone's daily activities. Bank transactions, ticket booking, social network follow-ups, online learning, online gaming, and so many other activities are involved with mobile devices like smartphones, tablets, and laptops. According to the statistics, the economy of smartphones is higher than PCs [115]. Recent approaches implemented for defending malware fail to provide effective security by using signature-based techniques. Based on the recent survey, the false-positive ratio is higher for the malware detection models proposed while considering obfuscation techniques. Attackers eagerly target mobile devices compared to PCs because mobile devices contain with the more sensitive and confidential information of users [9,116–118]. Besides, smartphones are associated with SIM cards which are responsible for transactions based on one-time passwords also known as OTP [119].

Based on the projection for the six months of apps downloaded by users worldwide, 25% of apps are used at most only once. This environment of rising users of mobile devices provides an attraction for malware coders. Malware writers implement their code to take personal and banking information. However, hundreds of apps have been removed by Google's play store and Apple's app store considering security concerns. As per the McAfee report, in 2006 the largest threat was identified regarding malware infection over iOS apps and android apps.

Elimination of malicious apps is carried out by app stores of Google and Apple but a few infected apps escape during the screening process. The signature-based technique is utilized to protect from malware attacks currently. The malicious behaviour of every malware sample is verified with the repository of anti-malware where behaviour is obtained in the form of string. The signature-based technique has two major drawbacks. The first one is that it is difficult and a time-consuming method. The second one is that it is not suitable for obfuscation techniques applied by the attackers [120]. With these incorporated drawbacks malware attackers generate various variants to showcase similar behaviour by employing obfuscation techniques in an automated manner. Thus, it becomes a herculean task for anti-malware providers to concentrate on those weaknesses, while researchers focus on finding a way to develop defence mechanisms against mobile malware attacks.

In common by applying repackaging techniques malicious activities are carried out in genuine applications. Nearly 86% of Android malware is acquired by using this technique [121,122]. Malware coder changes the app downloaded from famous app sales platforms like Google's play store [123]. Attacker applies reverse engineering techniques to the disassemblers which are cost-free like ApkTool [124]. The malicious payloads are then integrated into the app's original code [125,126], and the attacker uploads the maliciously coded apps to the marketplace. This prototype can fool legitimate users who are unaware of the discrimination between the legitimate app and the maliciously coded app. Malicious codes are added during the installation of malicious apps and the intended attacks of various types are implemented. Tools utilized for performing reverse engineering on android and apple apps are verified by creating a model and the process of creating repacked malware is automated [127]. Malicious payloads are injected by utilizing GUI interfaces by the proposed frameworks like DroidJack and AhMyth [128–130]. After the installation of the repackaged app, malicious attackers will get an alert regarding the maliciously affected device and will extract information from that device by recording the microphone, taking images, sending messages, and interpreting the contacts with the use of the GUI framework.

Providers of mobile OS seek solutions to this unconditional flood of malware by monitoring the malware-affected users. Google, which has been permissioned to perform this role, is required to offer deep scanning for each new app that is made available in the Play Store in order to find nefarious actions. Further, newly developed apps should be surrendered to Bouncer [131]. In the year 2012 automated scanning system was implemented with typical features as follows

- Static analysis while searching for identified vulnerabilities
- Behaviour identification by executing and analysing the app with a virtual emulator (QEMU)
- 5 min of tracking the behaviour of the app
- Investigation over each button of the app

Static analysis is performed by a bouncer by utilizing anti-malware providers like VirusTotal which evaluates with 60 various anti-malwares. By using signature-based techniques, a malicious sample is discovered only if the signature is available in the repository. Thus, it is effective in combating against zero-day threats [132,133]. While performing dynamic analysis, the application is executed for a time limit of 5 min. There is a possibility of the app not disclosing its malicious behaviour in this period. Also, malware can be aware when it is run virtually, in which malicious activity can be carried out to boycott the sandbox's detection [134]. Android and iOS malware detection and mitigation techniques are further explained in detail as follows.

5.2.1. Android malware detection techniques

From 2019 onwards, Android has been the most dangerous OS [5,135,136]. Malware for Android continues to evolve at a rapid rate, continuing a four-year trend. The first highest development of android malware was recorded in the mid of 2016. Because of its security breaches, it remains in first place of the infected OS with the highest vulnerabilities. As per the CVE (Common Vulnerabilities and Exposures) database, over 417 security leak scenarios were recorded in 2019.

Trojans occupy a major part in the distribution of android malwares in 2019, precisely 93.93%. An android Trojan named "Hiddad" is one of the top 10 android malwares of the year 2019 which is defined as an advertising specialist, hiked its position from 8 to 2 from the previous year with 18.7% infection. It confuses users and disables security software while being concealed in other Google Play Store apps. The second-ranked Android ransomware in the top 10 list of android malwares was created in 2019 with the intention of extorting legitimate users who earn more money and consequently fall victim to the attackers. Due to the extensive use of mobile devices rather than PCs and notebooks, economic damage also increases. Most ransomwares involve blocking devices by considering the features of the camera and backup. More than 78k new ransomware deployment of Android was found in 2019.

Online banking and purchasing environments place Android Password Trojans in the third place. Login credentials are skimmed, and multiple user accounts are targeted by cyber attackers because of the lack of security features associated with these apps. Though there are updates and upgrades in the versions of android, it is still a well-suited environment for illegal cyber businesses. A prototype of the application and its disassembled code are analysed for feature extraction in static analysis. The behaviour of the application is investigated at run time by applying the dynamic analysis technique. Hybrid analysis techniques analyse the behaviour of any application before the execution and after the execution. Individual flaws of static and dynamic analysis are resolved by combining these two.

AndroDialysis is a classification model based on the intents of android [39]. It concentrates on the runtime binding messaging system which insists on the processes of application mined from the malicious code and K-fold cross-validation is implemented by Bayesian Network for performance evaluation. While utilizing permissions, it attains an improved detection ratio [137]. Static analysis methods do not intend to infect a device and have no standard proposal for gathering information while executing in a restricted environment. However, code obfuscation techniques cannot be avoided by static methods [130,133,138–140]. Epic build pattern for each source and destination of ICC is included with numerous factors like location of Inter Component Communication (ICC), source and destination, operations of ICC, data type, and other useful information for analysis [141]. FlowDroid [142] also executes ICC analysis over the code and configuration files of the testing application. It decreases false positives. By collecting features from both static and dynamic analysis new methods for classifying malwares are introduced, which evaluate the malicious behaviour of android applications [143].

Machine Learning methods are applied for classification purposes in which a feature set is obtained by employing both static and dynamic analysis for familiar malicious and non-malicious applications [144]. BRIDEMAID framework makes use of dynamic analysis for detecting malicious behaviour at the time of execution, where multiple levels of monitoring are carried over devices, applications, and behaviours of users [145]. Exclusively dynamic features are only considered for classification by the Andromaly framework [146]. It is a host-oriented system for detecting malwares. Various features are monitored like usage of CPU (Central Processor Unit), packet count sent via the network, count of live processes, and the level of the battery. Besides, events acquired from mobile devices at the time of execution are also monitored. Sandbox is a system where malicious apps are analysed by combining static and dynamic methods [147]. Application is executed in a sandbox environment and hazardous events like file open processes and remote server connections are investigated. The denial of Service process is implemented by evaluating the method in a sample application for commencing the exterior binary program and creating sub-processes in an indefinite loop.

Power consumption is considered a classifying feature by multiple techniques proposed amid malicious and legitimate mobile Apps. It is assumed that there is a high association between the power consumption prototype of mobile devices and locality [148]. 20 smart phones' power consumption data are analysed over the course of three months. Nokia 5500 mobile is utilized for testing purposes to assess the mobile malware in a real-time environment. Jackdaw tool is implemented by correlating control and data-flow information obtained from binary files, with the employability of both static and dynamic analysis [149]. This hybrid analysis helps to discover API call groups that are associated with high-level actions. Evaluation is carried out with 2136 different binary samples of both legitimate and malicious binaries.

TaintDroid is proposed, which is the expansion of Android OS [150]. It is possible to trace the stream of private sensible information with the help of third-party apps that are downloadable malware. Besides, the real-time application environment is monitored with application accessibility and users' data manipulation. By utilizing predefined patterns, research work is carried out on the detection of suspicious temporal patterns for detecting intrusion [151]. Features like memory on demand, CPU usage, and keyboard utilization are also considered. The data set is derived from developing and accessing five infected android apps. DroidRanger [152] is a behaviour and heuristic-based approach proposed for android malware detection. Experimentation is done over 204, 040 apps of 5 distinct markets of android from May 2011 to June 2011. Among those apps, 211 malicious apps are effectively detected and 2 complicated zero-day malware over 40 samples are discovered where 11 apps are from a legitimate market.

SmartDroid is implemented to detect android malwares where behaviours of automotive Android app user interface interactions are prioritized, and problems with them are attempted to be fixed [153]. The hybrid analysis is applied to expose the UI-related trigger conditions of android apps. Activities and graphs of function calls are analysed statically to extract feature-like paths of activity. The dynamic analysis helps in tracing the elements of UI. Andrubis [154] is a free and complete automotive system for analysing android related applications. It makes use of hybrid analysis techniques over Dalvik VM and the system level. The dataset is obtained by examining more than 10 lakh android applications where 40% of which are malicious. Recent malware behaviours are defined by this dataset.

Mobile-Sandbox is an automatic analysis system implemented for android applications [155] using static and dynamic analysis techniques. The Manifest of the application is parsed and decompiled in static analysis. Every completed action is recorded from API calls by applying dynamic analysis. Machine Learning techniques are employed in the results of both analyses. Evaluation is done in more than 69 000 apps. Supervised and unsupervised learning classification approaches are applied for detection when malicious behaviour is revealed by applications [156]. It is different from previous works as it employs a visualization component. Also, various methods are proposed using static analysis to detect the interaction pattern between the modules of the application and the lack of privacy.

An image-based method for detecting android malwares is developed from executable binaries, which classifies malicious apps from legitimate apps [157]. The textual image classification method is utilized to extract the micro-level patterns of images. Retrieval techniques utilized for music information are applied to executable files to distinguish between malicious and legitimate apps [158]. The feature set is the results obtained from audio, which is created from executable files. Machine Learning algorithms are applied regarding the examination of performances attained by considering feature vectors collected from audio signals.

A reinforcement learning-based android malware detection model is proposed for combating adversarial attacks [159]. Single-policy and multi-policy are suggested for the scenarios with complete knowledge and limited knowledge respectively, on the basis of evasion attacks. Four conventional classifiers are utilized in this which include traditional ML, bagging, boosting and deep neural network. Based on the experimental results, single-policy utilization attained a 90% fooling rate with 10% modification. On other hand for multi-policy, 95% fooling rate is achieved with 10% modification.

Finding the best features that distinguish malware in a distinctive way is tough using machine learning techniques and to overcome the difficulties, an Android malware detection method is proposed by extracting malicious system call codes from typical system call sequences built by applications [160], depending on the occurrence of malicious system call codes. System call sequences are represented as first-order ergodic Markov chains. Reliable accuracy is obtained at about 0.95 for the data sets of balanced as well as unbalanced. Considering precision, it is obtained above 0.90 for balanced and slightly unbalanced datasets but 0.72 precision is attained for highly unbalanced data sets which is somewhat lower.

HAWK, a malware variant detection framework is proposed to overcome the malicious issues related to the current android applications [161]. A heterogeneous Information Network is modelled by combining android entities and relationships among behaviours. To handle the dynamicity of applications, an incremental learning model is built without modernizing the complete HIN and following the embedding model. Proximity amid new and existing applications is identified quickly and numerical embeddings of different semantics are cumulated. Above 80,860 malicious samples and 1,00,375 benign application samples are investigated during the experimentation. HAWK achieved the highest detection accuracy by taking 3.5 ms only to detect application samples which implies that its augmented training time is 50 times faster compared to the existing methods.

SEDMDroid is developed which is an enhanced stacking ensemble android malware detection framework [47]. Random feature subspace and bootstrapping samples techniques are adopted for generating subsets and Principal Component Analysis (PCA) is executed over every subset. The accuracy of SEDMDroid is explored by observing the entire principal components and utilizing the dataset for every MLP base learner during training. Further SVM is used as the fusion classifier for learning and result prediction. On the basis of experimental results, SEDMDroid accomplished 89.07% accuracy for the dataset of multi-level static features like permission, sensitive API and monitoring system event and then 94.92% accuracy for large-sized public datasets.

Fast Android Malware Detector (FAMD) is proposed with the aim of fast malware detection by using a combination of various features [162]. In the first step, the original feature set is created by extracting the permissions and Dalvik opcode sequences. The Dalvik opcodes are then subjected to pre-processing using the N-Gram approach. Further feature dimensionality is reduced by employing a symmetrical uncertainty-based FCBF (Fast Correlation-Based Filter) algorithm. At last, features with reduced dimensions are forwarded as input for the CatBoost classifier to detect and classify malware. The Drebin dataset is used for experiments, and improved results are obtained in terms of detection accuracy and classification accuracy, which are 97.40% and 97.38%, respectively.

GDroid is proposed on the basis of a Graph Convolutional Network (GCN) in order to detect and classify the families of android malware [163]. Apps and android APIs are mapped as a big heterogeneous graph in which the original problem is converted as a node classification task. On the basis of invocation relationships and API usage patterns, edges are built as "App-API" and "API-API". Following that, a heterogeneous graph is forwarded to the GCN model where nodes are generated by incorporating topological structure and node features. According to the experimental results, the GDroid system achieved a detection accuracy of 98.99% and a false +ve rate of less than 1%. In addition, for malware family classification average accuracy is attained as 97%.

Profile-hidden Markov model based android malware detection model ProDroid [164] is a behavioural method proposed for malware detection and classification. An encoded list is created by decompiling the android malware dataset for discovering suspicious API classes or methods. Encoded patterns are used for creating several sequence alignments for various malware families and they are applied for generating profile-hidden Markov models. According to the log-likelihood score, unknown benign and malicious applications are classified by the model. The accuracy of ProDroid is attained as 94.5% which is higher when compared with existing frameworks.

A solution to detect android malwares is proposed making use of Semantic Distance based API Clustering (SDAC) in the work [165]. The value of new APIs to malware detection is assessed on the basis of current API-spanning API clusters. When assigning APIs to vectors on sequences, a single neural network is used, and discrepancies across API vectors are viewed as semantic distances. API clusters and classification models are updated on the basis of true labelled training data and pseudo-labelled test data. Performance evaluation is implemented over 70k android app samples in a 5 years period time. Results depict that SDAC attained accuracy with a 97.49% F-score and slow ageing speed with a 0.11% reduction in F-score.

A Contrastive Learning-based strategy is suggested to identify and categorize android malware to support model pre-training without using labels and reduces the effects of prior knowledge [166]. According to the results, more than 96% malware detection accuracy is attained from the public dataset and more than 98% accuracy is achieved from malware class and family. To detect hazardous fake android anti malwares, HamDroid [167] is employed on the basis of static permissions and applications making use of MLP (Multi-Layer Perceptron) neural networks. A new dataset is created with the collection of harmful and benign android anti-malware applications by using Virus Total. Further, it is applied to train and evaluate MLP neural networks. As permissions are extracted from manifest files before the installation of android anti-malware, HamDroid seems feasible. Results of 98.62% accuracy, 95.56% precision, and 97.73% recall, and 96.63% F1-score were obtained by experimentation.

MSerNetDroid [168], an Android malware detection framework, used a multi-head squeeze-and-excitation residual network to extract features from the manifest file permissions, API calls and hardware features for app classification. Multi-Head Squeeze-and-Excitation Residual block (MSer) architecture is designed for learning essential correlation amid features. MSerNetDroid framework is evaluated by analysing 2126 malware apps and 1061 benign apps which are obtained through VirusShare and Google Play Store. The experimental outcomes reveal that the MSerNetDroid framework achieves 96.48% accuracy. Recently implemented android malware detection techniques making use of ML are depicted in Table 4.

5.2.2. iOS malware detection techniques

Mac OS environment is more secure compared to the other environments like windows and android [5]. 107 malicious samples are programmed for breaking Mac OS in 2019. As there are no malware defence systems, it is crucial to concentrate on Mac OS. Confidential information theft is performed by more distinct malwares in iOS devices. Based on the report of Shannon, watering hole spyware is reported in 2019 which is an iOS zero-day malware. It has the ability to grasp private information like iMessage photos as well as GPS locations [169]. In the same year researchers also found OceanLotus, a new iOS malware that uses cutting-edge tactics to evade malware analysis.

Similarly, exodus malware obtains information about phone contacts, audio recordings, photos, videos, GPS location and devices [170]. It spreads via phishing websites which impersonate the mobile carriers of Italy and Turkmenistan. FinSpy is a similar kind and is capable of stealing confidential information such as SMS or MMS messages, phone call records, emails, phone contacts, files, videos and GPS locations [171]. As per Kaspersky's report, Myanmar users are targeted for spying by using the latest versions of FinSpy variants of iOS and Android devices.

In 2019, Trojans hold the top spot with 59.87% of the market share, while 34.66% of backdoors in second place pose serious security risks [5]. It is known from the list that trojan and backdoor play specific roles because of the high-level explosion. Trojan namely Flashback with 38.36% has different variants that are involved in the infection of Mac OS over the last 10 years. It uses java for breaking and once java is enabled in browsers it starts its activation and infection by reading passkeys through keyloggers. More research is needed to understand Mac OS malwares because it has been there for ten years and is still active.

Considering the Apple environment, the ISAM model [172] for detecting malwares performs wireless infection and self-propagation among iPhone devices. Six malicious payloads of integrated malware were connected to the bot master server for updating the logic of programming [173]. iOS malware features are explored and after classification, 36 families of iOS malwares are identified between 2009 and 2015. Besides, these results are disseminated over legitimate markets and used for reducing the infections of iOS devices. By incorporating machine learning techniques and an opcode-oriented feature set, malware detection methods detect attacks from the malicious samples of iOS [174]. The histogram is created as the result of analysing these applications. The dimension of every histogram denotes how many times the opcode is displayed in the corresponding code. The OneR algorithm is utilized with 0.971 precision and recall as 1 for classifying the samples as malicious and benign.

Research works are concentrated on issues and open problems like stealing sensitive confidential information of the user which should be resolved [175]. In order to address this, remedial purpose dynamic analysis – which is appropriate for iOS environments – is used to propose solutions in iOS applications. Assessment processes concentrate on the need for user interfaces during dynamic analysis among iOS apps [176]. A high interaction Honeypot is implemented based on Mac OSX. More than 6000 blacklisted URLs are evaluated for predicting malware spread in Mac OS X.

Kernel-based SVM is applied for detecting the Mac OS malware using library calls [177]. A dataset of 152 malicious and 450 legitimate files is used for both training and testing. General supervised learning is used over the dataset. Detection accuracy at the rate of 91% and FPR of 3.9% is attained based on the performance of the system. Further, the Synthetic Minority Over-sampling Technique (SMOTE) is also used. While employing SMOTE, the accuracy rate increased to 96%, and FPR was obtained at less than 4%.

Android and iOS-based Mobile Banking Applications (MBA) of various banks are completely analysed by applying hybrid analysis and a threat model namely Vulnerability Assessment and Penetration Testing of Android and iOS Mobile Banking Apps (VAPTAi) is proposed [178]. Based on the systematic investigation of MBAs it is found that MBAs can be affected by Man-in-the-Middle Attacks (MitM). It is observed that few MBAs used simple HTTP protocol for transferring data without any security features. Forged or self-signed certificates are received by MBAs in many cases which will lead to SSL or TLS MitM attacks.

ChanDet, a channel model is introduced on the basis of five conditions while analysing the complete operation method and comparing the correspondences and modifications of several applications [179]. It is found that iMessage has a lack of confidentiality which implies it is not a potential channel. It provides a solution for testing the application as a potential channel. With the inspiration of the phylogenetic concept, the iOS malware classification method is implemented on the basis of mobile behaviour and vulnerability exploitation [103]. A Hybrid analysis is applied for experimental purposes. The significance of the proposed iOS malware classification towards malware detection is proved by conducting Proof of concept (POC).

iOS Chameleon apps are systematically studied and the Chameleon-Hunter detection tool is utilized for discovering the PHI-UIs and semantic feature set created with the help of nontrivial techniques [180]. Chameleon-Hunter is a static analysis-based method which identifies the maliciousness of PHI-UI. It was discovered that among 28 000 iOS apps on the app store, 142 Chameleon apps were involved in malicious activities. After disclosing issues with Apple, most of the iOS Chameleon apps are removed or suspicious UIs are disabled based on the investigation. Chameleon-Hunter is very active and achieved precision and recall at 92.6% and 94.7% respectively.

Also, the unpredictability of OS X malware is depicted by those research works. Complicated approaches are used by few

Table 4
Comparison of recent Android mobile malware detection techniques using ML.

Research paper	Features	Type of malware	Algorithms of classification
Zhu et al. (2023) [168]	Multi-head squeeze-and-excitation residual network, API calls and hardware features	Android	MSerNetDroid
Seraj et al. (2022) [167]	MLP, fake android anti malwares, static permissions and applications	Android	HamDroid
Yang et al. (2022) [166]	Past knowledge, model pre-training	Android	Contrastive Learning
Xu et al. (2022) [165]	Semantic distance-based API clustering, API vectors	Android	SDAC
Kumar Sasidharan et al. (2021) [164]	Profile hidden Markov model, encoded patterns, log likelihood score	Android	ProDroid
Bai et al. (2020) [162]	Permissions and Dalvik opcode sequences, N-Gram technique, FCBF, CatBoost classifier	Android	FAMD
Zhu et al. (2020) [47]	Random feature subspace and bootstrapping samples techniques, PCA, MLP, SVM	Android	SEDMDroid
Hei et al. (2021) [161]	Behaviours	Android	HAWK
Surendran et al. (2021) [160]	System call codes, system call sequence	Android	Ergodic Markov chain, system call pattern analysis
Rathore et al. (2021) [159]	Single policy and multi policy	Android	Traditional ML, bagging, boosting and deep neural network
Hashemi et al. (2019) [157]	Micropatterns	Android	Textural image classification
Farrokhmanesh et al. (2019) [158]	Audio files	Android	Music Information Retrieval
Feizollah et al. (2017) [39]	Messages, Bayesian Network	Android	AndroDialysis
Faiella et al. (2017) [145]	Behaviours of users	Android	BRIDEMAID
Ferrante et al. (2016) [156]	Change in environment	Android	Supervised and unsupervised
Lindorfer et al. (2015) [143]	Behaviour	Android	ML methods
Polino et al. (2015) [149]	API calls groups	Android	Jackdaw tool
Lindorfer et al. (2014) [154]	Malware behaviours	Android	Andrubis
Enck et al. (2014) [150]	Private sensible information	Android	TaintDroid
Spreitzenbarth et al. (2014) [155]	Manifest of app	Android	Mobile-Sandbox
Zhou et al. (2012) [152]	Behaviour and heuristic	Android	DroidRanger
Zheng et al. (2012) [153]	Behaviours of automotive User Interface	Android	SmartDroid

Table 5
Comparison of recent iOS mobile malware detection techniques using ML.

Research paper	Features	Type of malware	Algorithms of classification
Lee et al. (2021) [180]	iOS Chameleon apps, UI based illicit activity threats, suspicious PHI-UI	Mac OS	Chameleon-Hunter
Husainiamer et al. (2020) [103]	Mobile behaviour and vulnerability exploitation	Mac OS	POC
Zhou et al. (2019) [179]	iMessage	Mac OS	ChanDet
Pajouh et al. (2018) [177]	Library calls	Mac OS	Kernel-based SVM, SMOTE
Bojjagani et al. (2017) [178]	Mobile banking apps	Android and iOS	VAPTAi
Cimitile et al. (2017) [174]	Opcode Histogram	Mac OS	OneR
Garcıa et al. (2016) [173]	Malware families	Mac OS	KeyRaider analysis, IDA, Hoper
Lindorfer et al. (2013) [176]	A Mac OSX	Mac OS	iHoneypot

malware families to threaten the users of OS X and many of them are not successful; however, they will affect the tasks by rebooting. Important features of iOS mobile malware detection techniques are enumerated in Table 5.

5.3. Windows malware detection techniques

Based on the CVE database, over 660 official cases were recorded the previous year about the risks in the security of the system [5]. Windows 10 OS is targeted for vulnerable attacks with a count of 357. Also, high levels of security breaches are identified in Windows Server 2016 and Windows Server 2019. Considering the first quarter of 2020, Windows 10 is used by 51.38% around the world. Windows 7 remains with 30% utilization despite its security loopholes. The exploitation of Windows operating system vulnerabilities is the primary target for cybercriminals. Further backdoors associated with OS and firmware attached to the devices infect the entire security environment. IBM obtained 2nd place from 6th place in the first 10 manufacturing industries with security breaches in 2019 where most popular companies like Google, Oracle, Adobe, and Cisco are also on the list. For Adobe Reader, more than 342 susceptible cases were registered. Compared to the previous year in 2019, Windows trojan's percentage has increased by 35% with a total of 64.31% of malwares. Besides, ransomware, bots, crypto miners, and password Trojans are on the list of attacks.

Various detection methods are implemented for Windows malware. Anonymous malware detection in the Windows OS environment is achieved by implementing a malware detection framework using active learning [181]. Support Vector Machine classifier is utilized for classifying all anonymous malwares into legitimate and malicious. Runtime environment anonymous sample files are analysed and categorized by implementing a model that is employed both static and dynamic analysis techniques [182]. Static analysis is used to analyse binary data with the purpose of defining a feature module, and dynamic analysis is used to determine behaviours. Resultants of both analyses are used by ML classifiers to differentiate the sample as either benign or malicious.

A hybrid analysis-based malware detection model is developed for detecting botnets of 8 bits size in the work [183]. In the Windows OS environment, it aids in the detection of malware samples by utilizing both static and dynamic analysis techniques. The article [184] proposes a model for Windows malware detection model by utilizing features like strings and n-grams. ML models are utilized for malware classification in the Windows OS environment, and features are retrieved by running files in the run-time environment. A windows malware detection system is implemented by utilizing a directory of dependent files of prefetched files [185]. The minimum value of the false +ve rate is attained as 0.001. It is defined as a light-weighted malware detection method based on behaviours. For quick dynamic detection of malware, a framework namely a hybrid multi-filter is implemented to identify the behaviours of the runtime environment [186]. Various hybrid algorithms are utilized by combining the filtering methods like low redundancy and fisher relief F score. SVM wrapper is used for increasing the relevance level.

To observe and identify strange risks related to energy conservation, a power-aware malware detection framework is proposed [187]. It is framed with a combination of a power monitor and data analyser. Power samples are collected, and energy consumption history is created by the power monitor wherein the data analyser is utilized for power signature creation. By using noise-filtering and data-compression methods, power signatures are produced, reducing the difficulty of detection. HP iPAQ is utilized for experimental purposes over the Windows OS mobile environment. Malicious code analyser Cuckoo sandbox is utilized for analysing malware samples [49] in a Windows OS environment where two distinct methodologies are deployed — one for feature extraction and the other for behaviour analysis. Genetic algorithms are utilized for extracting features and for classification purposes, 3 classifiers are used — SVM, Naïve Bayes, and RF. This system attains classification accuracy with SVM at 81.3%, NB at 64.7%, and RF at 86.8%.

Recently windows malware forensic analysis methods are proposed without applying any ML techniques [188,189]. A comprehensive method is presented to renovate/retrieve vital information associated with malicious programs from different hidden locations of the windows system that expose the traces of malicious programs using available remnants of a computer [188]. Analysing procedures for various Windows 10 artefacts are offered, as techniques for looking into the digital footprints of suspect computers. Experiments are carried out for discovering the traces of deleted or existing malicious programs. Remote Access Trojan (RAT) families are studied comprehensively between 1996 and 2016 [189]. On the basis of the study RATSCOPE is proposed which is a forensic system. It is capable of recording and precisely reconstructing RAT attacks making use of the semantics of windows in a fine-grained manner. A novel Aggregated API Tree Record Graph (AATR) model is proposed for classifying PHFs of RATs by enhancing the performance and quality of intrinsic Event Tracing for Windows (ETW). Test results of RATSCOPE beat earlier forensic systems by achieving 90% TPR (True Positive Rate) over cross-family evaluation and 80% TPR over 2 years spanning temporal evaluation. Runtime overhead is gained as 3.7% which is very low. Table 6 depicts the comparison of recent Windows malware detection techniques using ML.

5.4. IoT malware detection techniques

The prevalence of malware on linked Linux devices has doubled in the IoT context. 57 million IoT malware attacks were discovered in the first half of 2022, up 77% from the previous half, according to a SonicWall analysis [57]. The growth curve of the malware development graph in the IoT environment is still rising [190]. More than 4 lakh samples were produced in 2019 which was nearly 2 lakhs in 2018. Trojans occupy the top place with 41.78% of the share in infection. Mirai malware is holding its first position in the top ten malware. It drew global attention due to the catastrophic DDoS attack on major web services like Twitter, Amazon, Dyn, Spotify, and the routers of Deutschland and Telekom. No awareness regarding this threat has been created by the producers of IoT devices. Mirai malware continued to create and spread botnets throughout 2019 [5].

Novel codes and techniques are injected to generate new samples like accessing the functions of Industrial IoT (IIoT). DDoS attacks are applied by cyber thieves to exploit the loopholes in the security system. Crypto miners also target the unsecured infrastructure of IoT. New IoT malwares are generated as a result of these and make the defence of internet-connected devices questionable. Gafgyt, a Trojan that ranks second globally with 15.04% of infections, and Tsunami and Hajime are also responsible for the destruction of IoT networks. Coinhive is a crypto miner that targets IoT apps. Code written in JavaScript is utilized for computing browser-oriented digital currency business services. With the comparison of 2020 results, sample counts have increased by 46% from 2019 [5]. Machine learning is used in malware analysis because it eliminates the need to acquire signatures for each type of malware for each detection technique, which aids in resolving some complications. The Basic IoT malware detection method with the utilization of different ML classifiers is depicted in Fig. 7.

Security-providing organizations implement two key methods for detecting malwares, which are matching the signatures and analysing based on heuristics [191]. The heuristic-based analysis approach becomes less effective as it is evaded by the size of the IoT setup and utilization of approaches. An immediate detection method is required for protecting the networks of IoT without any delay [192]. IoT environment-related threats and deployment of tools, and techniques against attackers are detailed [193,194]. CNN is utilized for detecting IoT malware by transferring binaries into grayscale images [195]. Malware detection traffic is analysed using the Bayesian model updating method and the result is compared with ML classifiers like KNN and SVM [45]. The K-Means Clustering algorithm is utilized for finding the abnormality of the power indulgence prototype to detect IoT malware [196]. Machine learning classifiers have been utilized by multiple research communities to detect IoT malwares for the past few years. Machine learning models are trained for malware prediction by concentrating categorization over existing malware samples [107] in which various methods utilized for choosing features are also detailed. Furthermore, classification algorithms like ANN, Decision Tree, and SVM are also discussed. Further, data privacy protection methods like secure offloading, access control and authentication are concentrated [197].

A cross-platform approach to analysing IoT malware where printable strings are first extracted from ELF files after IoT malware collection [198]. Printable strings are utilized as a cross-platform feature for identifying IoT malware over different platforms. ML algorithms are incorporated to verify the ability of strings in the classification of malware families of cross-platform environments. Based on the results, 99% accuracy is attained in a similar platform of training and testing with the data set which contains 120k IoT malware executable files. For different IoT platforms, 96% accuracy is attained. A unified IoT malware detection

Table 6
Comparison of recent Windows malware detection techniques using ML.

Research paper	Features	Type of malware	Algorithms of classification
Yang et al. (2020) [189]	AATR, ETW	Windows	RATSCOPE
D. S et al. (2020) [188]	Footprints	Windows	comprehensive method
Irshad et al. (2019) [49]	Cuckoo sandbox	Windows	SVM, Naïve Bayes, and RF
Huda et al. (2018) [186]	Behaviours of a runtime environment	Windows	Hybrid-multi filter-wrapper framework, SVM wrapper
Alsulami et al. (2017) [185]	Directory of dependent files	Windows	Lightweight behavioural malware detection
Mithal et al. (2016) [184]	Strings and n-grams	Windows	ML
Satrya et al. (2015) [183]	Malware behaviours	Windows	Hybrid analysis
Shijo et al. (2015) [182]	Behaviours	Windows	ML
Nissim et al. (2014) [181]	Active learning	Windows	SVM

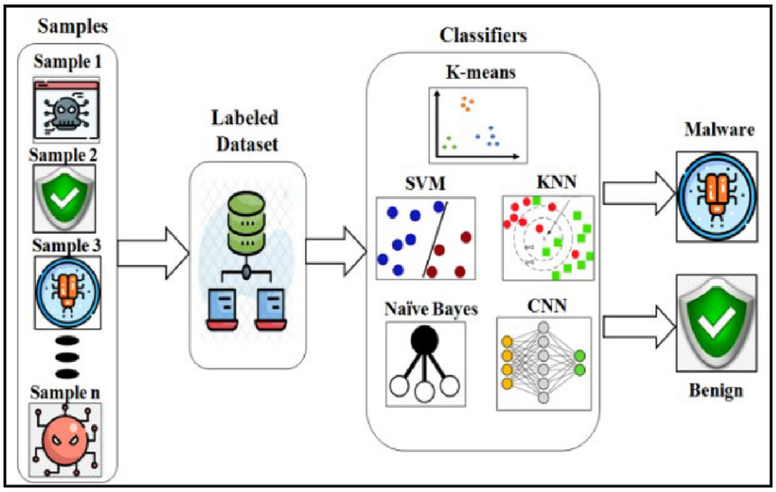

Fig. 7. IoT Malware detection using ML classifiers.

method is proposed by focusing on the detection of IoT malware at the node level as well as the network level [199]. For node-level malware detection, a Lightweight runtime malware detector is deployed which works with Hardware Performance Counter (HPC) values. To resolve the malware imprisonment problem of IoT networks, heuristic end-to-end detection and defence approaches are implemented. The scalability of network topology is maintained making use of multi-attribute graph translation.

Recently without applying ML techniques, IoT-based secure patching framework (IoT-Proctor) is proposed with various network isolation levels to moderate and control device-to-device (D2D) malware propagation [200]. Identification of compromised devices and malicious activity origin is performed using remote attestation. In addition, virtual patching is proposed through physical unclonable functions (PUFs) to avoid malware spread. The SEIR (susceptible, exposed, infected, and resistant) model is utilized for representing the levels of isolation and performance evaluation. These types of studies are done using an access control logic model, which reduces patching time and improves framework performance. Similarly another framework namely IoT-BDA is also developed without the involvement of ML techniques.

IoT Botnet Detection and Analysis (IoT-BDA) framework is an automated manner to capture, analyse, identify and report IoT botnets [201]. For providing support towards a broad range of hardware and software configurations honeypots are integrated with novel sandboxes in the framework. In addition, it involves discovering Indicators of Compromise (IoC) and Indicators of Attack (IoA) with in-depth analysis by implementing the techniques of anti-analysis, persistence and anti-forensics. The need for multi-architecture botnet analysis is ensured because 52% of the botnets are adopted with evading techniques for hiding C2 communication and 67% of botnets are targeting several CPU architectures. For experimentation 4077 distinct IoT botnet samples are involved for capturing, analysing and reporting in the time period of 7 months and better results are obtained.

To detect DDoS attacks in IoT networks, SDN-based framework is proposed by using SDNWISE customized controllers [202]. Other than IoT device vulnerabilities, malicious traffic of IoT networks is also investigated at an earlier stage by analysing packet logs employing the IP packet counter and IP payload detection approaches which are deployed over the SDNWISE controller. Massive traffic between nodes is induced by configuring part of the model's nodes in the COOJA simulator, which is used to develop software-defined IoT networks. While considering the results of the DDoS attack detection framework, 98% accuracy and 100% detection rate are attained with a low false +ve rate.

To mitigate the critical impacts of IoT botnets, ML-based IDS is developed with the aim of accurate IoT botnet malware detection [203]. Feature set minimization is focused and implemented for ML tasks by formulating 6 distinct binary and multi-class classification problems on the basis of botnet life cycle stages. For every classification problem, an optimal feature set is obtained by applying filter and wrapper methods with chosen ML methods. Based on experimentation results high detection rates are achieved in a restricted number of features and wrapper methods outperform filter methods in terms of performance. Channel-based features are preferred during the feature selection stage of post-attack detection. N-BaIoT and Med-BIoT datasets are used for the experimentation purposes. DT-based SBS (Sequential Backward Selection) offers excellent detection accuracy in a short amount of time; for the N-BaIoT dataset with 9 class classification types, this accuracy was obtained at 99.57%. Table 7 compares the recent techniques implemented for Machine Learning based IoT malware detection.

Table 7
Comparison of recent IoT malware detection techniques using ML.

Research paper	Features	Type of malware	Algorithms of classification
Kalakoti et al. (2022) [203]	Botnet life cycle stage, wrapper methods, Channel based features	IoT	ML based IDS
Bhayo et al. (2022) [202]	DDoS attack, SDNWISE customized controllers, IP packet counter and IP payload, COOJA simulator	IoT	SDN based framework
Trajanovski et al. (2021) [201]	IoC and IoA, in-depth analysis, multi architecture botnet analysis	IoT	IoT-BDA
Aman et al. (2021) [200]	Remote attestation, PUFs, SEIR model-based isolation	IoT malware	IoT-Proctor
Dinakarrao et al. (2020) [199]	HPC values	IoT malware	Multi-attribute graph translation
Lee et al. (2020) [198]	ELF files	IoT malware	ML algorithms
Wu et al. (2019) [45]	Bayesian model	IoT malware	KNN and SVM
Papafotikas et al. (2019) [196]	K-Means Clustering	IoT malware	ML
Al-Asli et al. (2019) [191]	Heuristics	IoT malware	ML
Pajouh et al. (2019) [192]	Heuristics	IoT malware	two-tier classification
Azmoodeh et al. (2018) [204]	Power consumption prototypes	IoT malware	KNN
Xiao et al. (2018) [197]	Control methods	IoT malware	ML
Su et al. (2018) [195]	Images	IoT malware	CNN

5.5. APT detection techniques

APT is a sophisticated stealthy threat actor which is a highly vulnerable kind of threat that should be proactively avoided. The most valuable information of legitimate users is targeted and creates financial issues for them. Various machine learning techniques are utilized for detecting advanced persistent threats. MLAPT is implemented by detecting and forecasting APT with 3 phases of execution [205]. The first phase involves the process of detecting threats which are developed with 8 sub-modules that are utilized for analysing the traffic in a real-time environment. Alerts are correlated in the second phase where the results of the first phase are used to discover the alert's type. In the final phase, the attack is forecasted by using ML algorithms which helps to detect the threat earlier. Based on the implementation results it attains maximum accuracy of 84.8% and a low false +ve rate is obtained. APT is a type of cyber attack that relies on targets to collect sensitive data using a bot. For detecting APT, methods that utilize signatures are not capable. APT assessment is implemented in the Hadoop platform by creating a definite search machine [206]. Information related to HTTP logs, like cookies, are accessed to gain confidential information by bot-infected systems. HTTP-based Command and control (C&C) is used for securing the affected system.

By combining the concepts of clustering and correlation to identify the system hosting the root infection, SPuNge is offered as a method for detecting APT [207]. The threat information connected to the attack's target and anticipated vulnerable behaviours helps to identify the impending attack. Attacking spot is defined by mapping the information with K means. SPuNge detects the collection of malware hyperlinks which are having similar domains. Domain graph-based APT detection scheme is proposed over an information-centric IoT environment [208], where connections of malware-infected domains and the respective address of IP are identified. To create domain graphs, mostly connected subgraphs were used, and computation and consumption were kept to a minimum. Based on the results of experiments it provides better accuracy and is applicable for the IoT environment's sensor networks.

An APT defence mechanism is proposed by analysing and utilizing the concept of dynamic deception [209]. Messaging system is implemented by applying both encryption and decryption mechanisms based on sockets for authentication of indenting legitimate users. SM4 is used for creating dynamic IP addresses wherein the Viterbi algorithm-oriented timing selection method is utilized dynamically. Issues on dynamic policy generation are resolved by applying both of them. Dynamic policy allocation is implemented by Dynamic Host Configuration Protocol Version 6 (DHCPv6). At last, strategies are generated and policies are assigned dynamically. For detecting APT-based malicious open XML documents, an intelligent framework is proposed based on flexibility and ease of configuration [210]. Reports are generated by highlighting information to analyse open XML documents without manual intervention. Configured scanner module provides the detection of APT. 9376 execution logs retrieved from 2010 APT campaign materials make up the APT-EXE dataset [211]. Statistical analysis is applied to the APT-EXE dataset. It is applicable for detecting general Windows APTs. An optimized feature set is obtained by accessing locations, file types, and operations in a loop. While testing 152 samples through campaigns in 2018 and 2019, 0.7697 sensitivity is attained for detecting run-time events. To defend against APT attacks, a game theory-based model is proposed [212] that resolves the issues related to APT attack paths. A network security defencive architecture is designed based on the analysis of the attack paths of attackers. Subsequently, using game theory as a foundation, the attack path prediction model (OAPG) is developed and an APT instance assault is employed to evaluate the model.

APT attack on the basis of time synchronization is a dangerous threat [213]. To simulate that for security research purposes, two devices are proposed which are a programmable Man-in-the-Middle (pMitM) and a programmable injector (pInj). They allow the implementation of various kinds of attacks over Precision Time Protocol (PTP) networks for security research purposes. On the basis of APT attack guidelines and emerging detection technology, an APT assault analysis framework has been proposed [214] where collected data is repeatedly matched with manually built cyber security knowledge graphs. In order to link new knowledge with past knowledge and provide real-time network security status, the framework leverages self-defined rules. This framework involves a lot of manual activities, which could cause human intervention errors that can be mitigated by implementing automation. An enhanced response system for APT is proposed [215] in which anomalous changes in the system are

Table 8
Comparison of recent APTs detection techniques using ML.

Research paper	Features	Type of malware	Algorithms of classification
Al-Saraireh et al. (2022) [218]	ANOVA, 5 stages, variance feature selection method	APT	XGB based
Halabi et al. (2021) [217]	IoV, optimal mixed strategy , DOBSS	APT	Bayesian Stackelberg Game
Yang et al. (2021) [216]	SDN, game-theoretic problem, Nash equilibrium solution	APT	DBAR
Hong et al. (2021) [215]	Changes in system	APT	AMHIDS,
Qi et al. (2020) [214]	Self-define Rules and Mapreduce	APT	Cyber security knowledge graph
Alghamdi et al. (2020) [213]	pMitM, pInj	APT	–
Su (2020) [212]	APT attack paths	APT	Game theory, OAPG
Coulter et al. (2020) [211]	APT-EXE	APT	–
Liu et al. (2019) [209]	Socket based	APT	SM4 + Viterbi algorithm
Ma et al. (2019) [208]	IP address	APT	Large-Scale Domain Graph
Sun et al. (2019) [210]	Open XML documents	APT	Framework
Ghafira et al. (2018) [205]	Traffic in a real-time environment	APT	MLAPT
Balduzzi et al. (2013) [207]	K means	APT	SPuNge

monitored by developing a real-time Anti-Malware Host Intrusion Detection system (AMHIDS). It is cost-effective in preventing APT attacks while considering security measures related to government operations.

APT defence mechanism recently developed on the basis of data backup and recovery (DBAR) techniques [216] is implemented to overcome the issues in the paradigm of software-defined networking (SDN) without employing ML techniques. It addresses the DBARS problem with the aim of identifying cost-effective DBAR technique. To describe the evolution of the predicted security status associated with the network of organizations, a dynamic model is presented. Further the problem is reduced to a differential game-theoretic problem with the implementation of the Nash equilibrium solution concept to attain cost-effective DBAR technique. The problem's optimality system is then derived using the Nash equilibrium solution notion.

To optimize the defence system of IoV networks and model the interactions amid various kinds of attackers, a repeated Bayesian Stackelberg game is developed [217]. Distributed sophisticated attacks in the Internet of Vehicles (IoV) are investigated and the severe impacts of APTs are highlighted. To predict the probability of various attack types over the IoV network, risk assessment is implemented at RSU levels. Results obtained from the risk assessment phase are integrated by the game in order to create a defence system with optimal mixed strategy. To increase the attack detection, an optimal Bayesian Stackelberg solver (DOBSS) is utilized for solving the game. It aims to offer a solution for balancing IoV's defence workload while limiting service delivery performance deterioration.

APT detection model is proposed on the basis of eXtreme Gradient Boosting (XGB) and variance feature selection method-based analysis [218]. A new dataset is created by considering the various stages of APT like normal (stage 0), reconnaissance (stage 1), initial compromise (stage 2), lateral movement (stage 3) and data exfiltration activity (stage 4). Further, strategies, procedures, techniques and Indicators of Compromise (IoC) of APTs are also considered during dataset generation. The produced dataset is then used to apply the suggested ML model. To choose the finest feature subset from the dataset, the ANOVA feature selection method is utilized. According to the obtained results, the accuracy score is attained as 99.89% by using 12 features alone among 65 features from the dataset. Table 8 depicts the summary of recent APT detection techniques on the basis of Machine Learning.

5.6. Ransomware detection techniques

Protection against ransomware is the most laborious task. Most ransomware attacks are targeted at naïve users who do not keep backup data and malicious users use these victims as a means of generating revenue [5]. Ransomware acts benign and it is difficult to detect because of its invariance in the environment [219]. Implementing zero-day attack protection techniques is significant for automatic ransomware detection.

The system of anti-ransomware typically aims to avoid damage, protect against zero-day ransomware attacks, and raise user awareness. Ransomware is speculated to use evasion mechanisms such as delayed execution, self-destruction, polymorphism, and network traffic encryption [220]. Sharing malware payloads through simple actions like clicking on a malicious link or opening a spam email attachment would result in a ransomware attack [221]. Due to the nature of ransomware, standard malware detection methods are insufficient. Hence additional skill enhancement is required.

Early indications of ransomware outbreak can be obtained with the use of CryptoDrop [219] since it keeps track of changes to user files and data in a real-time environment. Specific file systems-based attributes are recommended which are relevant to most crypto miners. Primary and secondary indicators play a key role which is identified via extensive analysis of behaviour. Change in the file system, Hash similarities, and Shannon Entropy are the primary indicators, and file destroying and tunnelling are secondary terms. CryptoDrop attained a 100% of True Positive Rate (TPR), 1% of False Positive Rate (FPR) and 1–16 ms overhead based on the results.

UNVEIL [222] creates an automatic virtual environment for imitating the environment of the user. For detecting ransomware two techniques are presented in UNVEIL which are file lockers and screen lockers. File lockers focus on the files of the victim's system and file system accesses of the system are monitored. Screen lockers concentrate on ransomware which avoids system access and it relies on the dissimilarity scores of screenshots obtained from the system's desktop analysis before, during, and after running the malware sample to find locked desktops. Based on the experimental results, UNVEIL attained a 96.3% True Positive Rate at zero False Positives.

R-Locker is based on the concept of deception where a honey file trap is implemented in a FIFO manner to block ransomware [223]. The maximum degree of dependency on the OS is considered as its feature. It does not need pre-training and aims for zero-day attacks. It is observed that based on results, R-Locker achieved 100% TPR and 0% of FPR. The method for identifying zero-day attacks is implemented by applying behaviour detection and anomaly detection [224]. Classifiers such as SVM are utilized for behaviour analysis and it generates predictions. The majority voting strategy is used for merging the outcomes of classifiers and OR logic is used for combining outcomes of anomaly-based predictions. Experiment results depict 99% TPR and 2.4% FPR.

The ransomware detection method is implemented by utilizing the characteristics of human file handling as a whitelist and it focuses on the dissimilarities between "humanness" and "ransomwareness" [225]. By prohibiting file deletion and overwriting, it avoided acquiring files encrypted by crypto-ransomware and it is implemented for protecting text files. For encrypted ransomware, a context-aware entropy analysis-based detection method is implemented [226]. It is discovered that the header fields of encrypted files of ransomware are having high entropy compared to the normal legitimate file's header and this information is utilized for abnormal behaviour detection. 100% TPR and 0% FPR are attained as results after experimentation.

RWGuard is proposed for detecting crypto-ransomware over a real-time environment by utilizing 3 techniques of monitoring namely trap monitoring, monitoring the changes of files, and monitoring the running processes [227]. To find changes in a file, the number of indicators are tracked, including entropy, patterns that are similar, and changes in the type and size of files. The threshold value is computed based on those changes. While classifying the samples into legitimate and malicious, profiling is completed by considering the encryption behaviour. RWGuard achieved 100% TPR, 0.1% FPR and 1.9 ms overhead according to the results.

ShieldFS [228], a runtime environment for a ransomware detection system, utilizes classifiers based on supervised learning making use of the input–output request packet obtained from benign and malicious systems. Two distinct models are developed for classification based on systems and processes. By involving various levels of assessment of files, every classifier is trained. ShieldFS achieves 99%–100% TPR, 0–0.2% FPR, and 0.3–3.6x overhead dependent on results. To recover from the damages of ransomware, mitigation methods that guarantee data reliability and accessibility are put forward in the work [229] that observes the IRP of every user file. A mini-filter driver effectively prevents hackers from having access to read or write files and experimental results are attained as 100% TPR and 0% FPR. RedFish is an algorithm that can identify ransomware activities and stop them from spreading to shared documents [230]. It is a network traffic monitoring-based ransomware detection algorithm used for analysing the traffic of the network. Basic activities of ransomware like modification of files by read or write operation, content removing, or overwriting are monitored. Tracing information on benign and malicious networks is utilized for training purposes. RedFish also attained the same results as previous work as 100% TPR and 0% FPR.

SDN-based detection method by analysing the HTTP request and response messages [231,232] is proposed where HTTP packets are extracted and pre-processed by the traffic analyser of HTTP which reconstructs the information of the server, size, and IP. The ransomware detection method is proposed by mixing the features of both static and dynamic analysis [233]. In static analysis, the concept of deception of the file is implemented by placing trigger programs with honey files in an artificial network. Additionally, network packets are observed and features are mined from the header of packets to create a dataset for malicious behaviour detection. NetConverse employs Machine Learning techniques by utilizing the information of network traffic [234] in which a Decision Tree classifier is used to trace the parameters of the ransomware-affected system. NetConverse attained 95% TPR and 1%–6% FPR.

Security measures are designed to guard against ransomware attacks on flash-based storage devices [235]. The type of ransomware is discovered by pattern-based detection. Ransomware activities are monitored by tracking the operations like reading and writing of files with the help of the policy of buffer management. To protect the ransomware attacks in Solid State Drive (SSDs) based on NAND flash, a security approach is implemented [236]. For determining the precise details of ransomware, features including the address of the block, its size, and the types of IO requests it makes are taken into consideration. Binary DT with ID3 is utilized for the detection of ransomware. Additionally, impacted files are recovered by constructing a Flash Translation Layer (FTL), which prevents the delay of ransomware detection but imposes an extra burden on SSD. According to experimental results, 100% TPR and 0%–5% FPR is attained.

Grand Unified Regularized Least Square (GURLS) based method is proposed for ransomware detection which utilizes supervised learning and Regularized Least Square algorithm [237]. Training is performed by considering features like API call's regularity and strings; the adequate level of detection is provided when combined with the Radial Basis Function kernel. Experimental results are provided as 87.7% TPR and 7.5% FPR. A Hybrid android ransomware detection and mitigation method is proposed [238] where Opcode's frequency is considered in static analysis whereas usage of memory, network, and CPU are considered for dynamic analysis. 2-gram opcode frequencies are taken as input for the binary classifier. TPR at 100% and FPR at 4% are attained as experimental results.

R-PackDroid is meant for avoiding ransomware attacks in android phones [239] where ML is employed with the consideration of API packs for feature extraction and Random Forest Classifier is used for classifying distinct ransomware based on the training results. R-PackDroid attained TPR and FPR of 97% and 2% respectively. For IoT environments, the ransomware detection method is implemented based on power consumption prototypes [204]. KNN provides a high level of detection accuracy with measures of time and similarities and 95.65% TPR is attained. It is suggested that ransomware can be detected by looking for anomalies in file directory activity [240]. Use of storage, use of processor and IO rates are also utilized. Testing is performed with single custom ransomware purposely developed for research.

By applying static analysis crypto-ransomware is detected from the collection of malicious and benign files [241]. The histogram density of opcodes is considered as the feature set and the SMO classifier is incorporated along with WEKA. 95% TPR and 0%–5% FPR is attained as results. UShallNotPass [50] uses the concept of encryption of random numbers used by ransomware and implements a security mechanism based on this. Most operating systems typically generate pseudo-random integers to increase randomness. 94% TPR is attained after experimentation.

Reinforcement learning is incorporated for anti-ransomware testing [242] where a simulation environment is built by including two main components. The first component of ransomware simulates an attack, while the second component of ransomware detects an attack. RL agent is introduced in order to attack optimally by obfuscating the detector and then the user's target files are encrypted. It will be helpful for identifying the loopholes of anti-ransomware defence and overcoming them before the occurrence of real-time attack.

Sequential Pattern Mining based feature identification and ML-based classification are combined together for detecting ransomware [243]. The environment is set by collecting the activity log samples of 517 locky ransomware, 535 Cerber ransomware and 572 TeslaCrypt ransomware. Maximal Frequent Patterns (MFP) are discovered using Sequential Pattern Mining among various ransomware families. J48, Random Forest, Bagging and MLP algorithms are used for classification. The accuracy of ransomware detection is achieved at 99% in this method. An adaptive pre-encryption crypto ransomware early detection model is proposed by combining the concepts of pre-encryption and crypto-ransomware population drift [244]. It provided a remedy for the shortcomings of the earlier models with the help of precise pre-encryption boundary definition. This model's adaptability feature has up-to-date information on attack behaviour, which will be useful in identifying polymorphic ransomware.

Ransomware detection framework Autonomous Backup and Recovery (AMOEBA) is proposed which is based on content [245].

It provides solutions at the device level and extra backup storage is not required. AMOEBA depends on two main components which are the hardware accelerator and fine-grained backup mechanism. For high-speed ransomware detection, content-based detection algorithms are executed with the help of hardware accelerators. The space overhead of data backup is reduced by a fine-grained backup control mechanism. In addition, it is prototyped with the OpenSSD platform.

In order to perform ransomware detection and classification, the DNAact-Ran method is proposed by using ML [246]. Digital DNA Sequences computation is established with the help of algorithms like MOGWO and BCS and also Digital DNA sequencing design constraints and k-mer frequency vector are utilized by DNAact-Ran. On the basis of the digital genome, the uniqueness of ransomware is identified and ransomware is classified into well-known families. The performance of the proposed method is evaluated with the evaluation metrics such as precision, recall, f-measure and accuracy over 582 ransomware instances and 942 goodware instances. Proposed method attained 87.9% detection accuracy which depicts better results when compared with ML algorithms like Naïve Bayes, Decision Stump and AdaBoost.

For detecting IoT ransomware attacks, the Intrusion Detection Honeypot (IDH) is developed by utilizing Social Leopard Algorithm (SoLA) [247]. IDH is comprised of Honeyfolder, Audit Watch and Complex Event Processing (CEP). The Honey folder acts as a trap and prior warning system while suspicious file activities are encountered where SoLA is utilized. In Audit Watch, the entropy of files and folders is verified. Finally, the CEP engine cumulates the data of various security systems and the behaviour and attack pattern of ransomware are confirmed. This proposed design is validated with more than 20 ransomware variants. Experimental results depict better results than existing systems in terms of precision, recall and accuracy.

To combat smart healthcare-based ransomware attacks, Blockchain-Enabled Security Framework (BSFR-SH) is proposed [248]. Security analysis is conducted in BSFR-SH and proved its security against ransomware attacks. BSFR-SH comprises 5 phases. In the first phase, healthcare data backups are generated via blockchain. Blockchain-based data collection and signature generation are performed in the second phase. In the third phase, ML is applied for ransomware detection and analysis. Blockchain-based ransomware mitigation and data recovery are implemented in the fourth and fifth phases respectively. While comparing the performance with other existing schemes, BSFR-SH provides better results in terms of accuracy and F1-score.

A multi-feature and multi-classifier network-based system (MFMCNS) is proposed to detect Ransomwares [249]. Wannacry and NotPetya Ransomwares are considered for providing complete behavioural analysis of the network traffic of Ransomwares. On the basis of session and time flow levels, two sets of 21 related features are extracted and every set is included with an independent classifier by utilizing the majority voting rule. On the basis of results, detection accuracy is obtained as 99.88% and 99.66% for the session and time-based classifiers respectively. On the basis of file sharing traffic analysis, a security tool is proposed for detecting and blocking crypto-ransomwares [250]. ML techniques are utilized for monitoring traffic and traffic patterns related to ransomware activities are identified during file reading and overwriting. Features are extracted on the basis of activities which include opening, closing and altering files. For training and testing purposes, above 70 ransomware binaries of 33 distinct strains are used. False positive rate and encrypted file information are considered as parameters for validating algorithms.

An ontology driven framework is developed for digital extortion threats (i.e Rantology) which focus on windows ransomware attacks [251]. Proposed ontology utilizes API functions and behaviours for evaluating the maliciousness of programs. It facilitates communication between cyber security experts and ransomware knowledge-based systems and also identifies sensitive areas for investigation. Evaluation is performed with several measures like clarity, consistency, modularity, richness and coverage of the inheritance. Table 9 provides a summary of recent ransomware detection techniques.

6. Deep learning based malware detection techniques

6.1. Sandboxing based malware detection

6.1.1. Static analysis based malware detection techniques

Automatic malware detection in the work [252] is accomplished using the deep learning approach on the basis of static analysis. It consists of two components: a feature extraction technique and a deep learning model. Grayscale images and OpCode 3-gram are combined to extract features of the malware. The feature set is used for training purposes by the deep learning model. A deep feedforward neural network is used for classifying benign and malware [58].

Deep Learning Framework for Intelligent Malware Detection (DL4MD) extracts API calls of malware libraries for learning [31]. API call references of size 32 bits are extracted using a database. These features are given as input for DL4MD architecture which is based on stacked autoencoders. In the work [27], Convolutional Neural Network is employed for the classification of binaries into disassembled byte codes. Further processes are applied to the raw disassembled code for feature generation. Instructions of X86 processors of different lengths are extracted to set features the disassembled code is parsed to extract features. With the features of a linear support vector machine, a deep learning model is implemented in which cross-entropy loss is reduced [253]. The use of SoftMax is replaced by L2-SVMs, which displays better results over famous datasets of deep learning like MNIST and CIFAR-10. It produces an error rate of 11.9% which is comparatively better than CNN with SoftMax. CNN-BiLSTM model [85] is entirely a data-driven method for feature extraction.

Grayscale graphics are used to represent malicious software because they can be used to detect differences since they can capture subtle changes while preserving the overall structure [77]. The MalImg dataset and the Microsoft Malware Classification Challenge dataset are used as benchmarks to assess the feasibility of the approach. Similarly, by highlighting the fact that most variations of the malware are created using standard obfuscation techniques so the compression and encryption algorithms retain some attributes inherent in the original code, a file-agnostic deep learning strategy for classifying malware is presented [78]. With the use of the data supplied by Microsoft for the BigData Innovators Gathering Anti-Malware Prediction Challenge, it enables the discovery of discriminative patterns that practically all variants in a family share. A file-independent end-to-end deep learning method is suggested for classifying malware from raw byte sequences without the use of manually created features [75]. It has two essential parts: a denoising autoencoder that discovers a hidden representation of the binary content of the malware, and a dilated residual network that serves as a classifier. The experiments perform admirably, classifying malware into families with almost 99% accuracy.

A holistic multi-dimensional deep learning framework is implemented in the work [79] to classify IoT malware and family attribution by using Deep Learning architectures. Static features of executable binaries are extracted and used in the format of strings and images. More than 70k IoT malware samples are involved in the analysis which is related to the four major malware families such as Mirai, Gafgyt, Tsunami and Dofloo. After

Table 9
Comparison of recent ransomware detection techniques using ML.

Type of method	Research paper	Features	Algorithms of classification
Behaviour-based	Keshavarzi et al. (2023) [251]	Knowledge-based systems, API functions and behaviours	Ontology driven framework
	Almashhadani et al. (2022) [249]	Wannacry and NotPetya, majority voting rule	MFMCNS
	Al-rimy et al. (2018) [224]	Majority voting strategy, OR logic	SVM
	Gomez-Hernandez et al (2018) [223]	Honey file trap, zero-day attacks	R-Locker
	Jung et al. (2018) [226]	Header fields, high entropy	Context-aware entropy analysis
	Scaife et al. (2016) [219]	Primary & secondary indicators, Hash similarities, Shannon Entropy	CryptoDrop
	Kharaz et al. (2016) [222]	File lockers, screen lockers,	UNVEIL
IRP monitoring based	Mehnaz et al. (2018) [227]	Trap monitoring, entropy, patterns, profiling	RWGuard
	Bottazzi et al. (2018) [229] Continella et al. (2016) [228]	IRP Supervised learning	Mini-filter driver ShieldFS
Network traffic monitoring based	Morato et al. (2018) [230]	Traffic monitoring, file modifications	RedFish
	Cabaj et al. (2018) [231] Alhawi et al. (2018) [234]	HTTP traffic ML, network traffic, DT	SDN based NetConverse
Memory-based	Min et al. (2021) [245]	Device level storage, hardware accelerators	AMOEBA
	Baek et al. (2018) [236]	SSDs, NAND flash, FTL	Binary DT with ID3
API calls based	Harikrishnan et al. (2018) [237]	Regularized Least Square, Radial Basis Function	GURLS
Android environment	Ferrante et al. (2018) [238]	Opcode's frequency, memory, network, and CPU	Hybrid method
	Scalas et al. (2018) [239]	ML, API packs, RF	R-PackDroid
	Azmoodeh et al. (2018) [204]	Power consumption prototypes	KNN
Others	Berrueta et al. (2022) [250]	Crypto-ransomwares, ML, file activities	File sharing traffic analysis
	Wazid et al. (2022) [248]	Smart healthcare-based, 5 phases, ML	BSFR-SH
	Sibi Chakkaravarthy et al. (2020) [247]	IDH, Honeyfolder, Audit Watch, CEP	SoLA
	Khan et al. (2020) [246]	Digital DNA Sequences, MOGWO, BCS	DNAact-Ran
	Homayoun et al. (2020) [243] Baldwin et al. (2018) [241] Gen et al. (2018) [50]	ML, MFP WEKA, histogram density Random number encryption	Sequential Pattern Mining SMO UShallNotPass

performing in-depth experiments, the accuracy of the framework is accomplished at 99.78%. To label the fresh "unknown" malware samples, this IoT-tailored method is used. Table 10 provides summary of recent static analysis based malware detection techniques using DL.

6.1.2. Dynamic and hybrid analysis based malware detection techniques

Features of API calls are utilized by Huang and Stokes [38]. By applying a sandbox to the created log files, Deep Belief Network (DBN) is employed [32]. Similarly, Recurrent Neural Network (RNN) is employed over events of API calls. It is a bidirectional RNN that observes system calls. These events are encoded as high-level events (represented as 114) from malware analysers. The relationship among events and similarities of functions of LSTM is captured by RNN [28]. The true +ve rate of this model is 82% and the false +ve rate is 0.5% as per the experimentation results.

Raw malware byte code is considered for malware detection which uses a down-sampling approach before the pre-processing stage. Down-sampling is a generic data scaling approach [33]. After that, a deep learning model can be applied. Evasion attacks make it harder to detect malwares while using deep learning. The investigation is done on the viability of adversarial crafting in favour of deep neural networks by Grosse et al. [254]. Crafted inputs avoid machine learning models, and it leads to failure of classification. While using the DREBIN dataset, the misclassification rate is raised to 80%. So adversarial crafting is a major threat that affects security concerns. The gradient-based attack is also possible in a deep network. Change in particular bytes of the malware sample leads to attack. According to Kolosnjaji et al. the probability of a malware attack is high for adversarial malware binaries [52]. For the above similar problem, a gradient-based method [255] is implemented for classifying malware samples. MalConv network is utilized for this

Table 10
Comparison of Static analysis techniques using DL.

Research paper	Features	Type of analysis	Algorithms of classification
Dib et al. (2021) [79]	Static features, strings and image format	Static	Multi-dimensional deep learning framework
Lee et al. (2021) [180]	PHI-UIs and semantic feature	Static	Chameleon-Hunter
Fang et al. (2019) [60]	PE, RL	Static	DQEAF
Gibert et al. (2019) [13]	Mnemonics sequence	Static	Hierarchical CNN
Gibert et al. (2018a) [75]	Byte sequence Denoising	Static	AE with ResNet
Gibert et al. (2018b) [78]	Structural entropy	Static	CNN
Gibert et al. (2018c) [77]	Grayscale image	Static	CNN
Li et al. (2019) [8]	HMM, domain features, two level model	Static	DGA + DNN
Křcal et al. (2018) [35]	Bytes	Static	sequence CNN
Rezende et al. (2018) [36]	Grayscale	Static	image ResNet-50
Gibert et al. (2017) [74]	Mnemonics sequence	Static	Shallow CNN
Kolosnjaji et al. (2017) [52]	PE header features, Instruction traces, imported functions and DLL files	Static	CNN with Feedforward NN (Nearest Neighbours)
Liu et al. (2017) [252]	Grayscale images	Static	Opcode 3 gram with CNN
Yin et al. (2017) [88]	Opcodes	Static	RNN + SVM

implementation with a 60% rate of evasion. This byte-oriented approach provides better security for the vulnerable environment.

Though malware detection using Deep Learning is powerful, efficient, and decreases the size of features heavily, it does not provide resistance to evasion attacks. The performance of the model does not depend on adding hidden layers. More advanced research is needed in Deep Learning-based malware detection. LSTM classifier is used by extracting HTTP traffic for malware classification in the dynamic analysis [256]. Similarly, network behaviour is analysed and a combination of Autoencoders (AE) and Nearest Neighbour (NN) classifiers is used [257]. Kumar et al. implemented a malware detection technique by applying both static and dynamic analysis where the file's metadata, network data, system calls, process, and registry features are extracted [258]. MalShare and ViruShare are utilized as classification sources in conjunction with the XG Boost classifier, RF classifier, DT, NN, and KNN classifiers. Similar to Kumar et al. Rhode et al. [259] applied both analysis methodologies to extracting features such as API calls and classifying the data from 6809 malware samples using RF, NN, and SVM classifiers.

By combining visualization and convolutional neural network techniques, a deep learning-based method for detecting Windows malware is put into practice [260]. The structure of the neural network is defined on the basis of the VGG16 network and it supports the hybrid visualization of malwares. To analyse the samples dynamically, Cuckoo Sandbox is utilized and after that results obtained from the dynamic analysis are transformed as RGB colour images on the basis of developed a new static visualization algorithm. For encoding additional information from raw files, the designed approach employs the byte frequency information of PE files and RGB colour pictures, which makes use of all three colour channels. Neural networks are trained using static and hybrid visualization images. Results depicted that accuracy is attained by the method for static visualization as 91.41% and for hybrid visualization at 94.70%. Recently proposed dynamic and hybrid malware detection techniques using DL are compared in Table 11.

6.1.3. Image based malware detection techniques

Comprehensive representations of malware process behaviours are provided in the work [28]; for each operation's execution, information such as the name, ID, name of the event, and current directory path are recorded. After that RNN is applied to build a behavioural language model where outputs are interpreted as images. The CNN-trained features of images are used to finally classify any discovered as benign or malicious. In a similar vein, 2D grayscale images are produced using CNN by Tobiyama et al. [261].

Deep learning suits well for GPU environments that would employ a model with ease of training and efficient malware detection. Thus, current research of analysing malwares converted to the field of deep learning from static and dynamic analysis. The use of the Microsoft malware classification challenge dataset (BIG2015) has been the subject of comparative research [262]. When comparing the research from 2016 to 2017, the majority of the researchers focused on deep learning. For accurate malware identification, a model with automotive feature extraction is required.

YongImage method is employed in a DL-based malware classification model [73] that embeds malware where instruction-level information is related to disassembly metadata produced from the IDA disassembler for converting image vectors. Finally, with a simplified design and a high rate of convergence, a deep neural network is formed namely malVecNet. It is motivated by better performances provided by applying image-related conventional neural networks. YongImage method transfers the tasks of malware analysis to the problems of image classification where minimum domain knowledge and simple methods for extracting features are enough. Optimization of malVecNet is ensured by utilizing the idea of sentence-level categorization of Natural Language Processing [263]. Effective training is enabled by malVecNet. Evaluation of the model is carried out by utilizing 10-fold cross-validation over the malware dataset of Microsoft. Results show an accuracy level of 99.49%. Utilizing the Malimg dataset, a malware classification deep learning model is proposed in which an SVM classifier is added for classifying malwares [37]. Convolutional Neural Networks (CNN), Multilayer Perceptron (MLP), and Gated Recurrent Unit (GRU) are some various Deep Learning models employed for performing classification. Implementation results depict that GRU has better performance than other models with an accuracy of 84.92% approximately.

A light-weighted model for the IoT environment is introduced to detect distributed denial of service (DDoS) malware [195]. A well-defined convolutional neural network is designed to classify the families of malware making use of image recognition methods where grayscale images are created from malware binaries. Resultant grayscale images are given as input for machine learning classifiers. If a malicious file is found it is sent to the remote cloud server for further classification. The size of the database of the signature matching system is very large because it includes information about every malware sample. So, it is not efficient for the IoT environment because the resources of IoT devices are fixed with certain limits. Small-sized two-layered shallow CNN is needed for building lightweight malware detection and classifying malicious behaviour with a 94.0% accuracy rate.

A malware classification model is proposed by combining the feature of CNN and SVM [48]. SVM is utilized for differentiating

Table 11
Comparison of recent Dynamic and Hybrid malware detection techniques using DL.

Research paper	Features	Type of analysis	Algorithms of classification
Huang et al. (2021) [260]	Visualization, CNN, VGG16, RGB colour images	Both static and dynamic	Deep Learning based
Vinaykumar et al. (2019) [63]	PE	Both static and dynamic	LSTM+CNN, MLA
Kumar et al. (2019) [258]	system calls, metadata of PE file, Network data, process and registry features	Both static and dynamic	RF, XGBoost, DT, NN, K-NN
Rhode et al. (2019) [259]	API calls Machine metrics	Both static and dynamic	NN, RF, SVM
AL-Hawawreh et al. (2018) [257]	Network behaviour	Dynamic	AE with NN
Kolosnjaji et al. (2018) [255]	Byte-oriented approach, gradient-based	Dynamic	MalConv
Raff et al. (2017) [33]	Raw malware byte code		Down sampling
Athiwaratkun et al. (2017) [87]	API call sequence	Dynamic	LSTM, GRU
Prasse et al. (2017) [256]	HTTP traffic	Dynamic	LSTM
Kolosnjaji et al. (2016) [52]	API call sequences	dynamic	CRNN
Huang, et al. (2016) [38]	4.5 M files training, 2 M testing	Dynamic	MtNet
Grosse et al. (2016)_ [254]	Adversarial crafting, DREBIN dataset	Dynamic	DNN

the families of malware. Malware and its family is mapped with the mathematical function $f : n \rightarrow z$ where n denotes the new arriving malware and z denotes its family. Malimg dataset with the size of 25 families and 9339 samples are utilized for implementation. 97.5% accuracy is obtained by this CNN-SVM model. Mobile apps are represented in terms of grayscale images obtained from mobile binaries [264]. Image processing-based malware detection techniques still require difficult methods for extracting features to detect malware. BIG2015 is utilized for performing malware analysis by multiple methods recently in different experimentation environments. To detect Android malware and classify the family of Android malware, VisDroid is implemented which is an image-based malware detection method [265].

An automated vision-based Android Malware Detection (AMD) model is proposed by developing 16 distinct fine-tuned deep learning-based CNN algorithms in order to attain effective classification of benign apps from malware apps with minimal computation in the stages of reverse engineering and feature extraction [266]. Grey-coloured malware images are generated to achieve accurate prediction making use of balanced or imbalanced datasets. Before the CNN classification stage, the byte-codes associated with "classes.dex" files are extracted from the malware and benign apps and transformed into grayscale and colour visual images. An imbalanced benchmark Leopard Android dataset composed of 14 733 malware app samples and 2486 benign app samples is utilized to evaluate the detection efficiency AMD model. Several experimental scenarios are implemented with the help of balanced and imbalanced android samples of the Leopard dataset. On the basis of the results, 99.40% accuracy is attained for balanced android samples and 98.05% accuracy is achieved for imbalanced android samples. Table 12 compares the recent image-based malware analysis techniques making use of DL.

6.2. Mobile malware detection

6.2.1. Android malware detection techniques

Android phones lead in mobile phone sales with 71.62% of sales and iOS of Apple takes next place with 27.73% of sales all over the world [269]. The first version of iOS was introduced in 2007, which is noted as a revolution in the mobile market because of the implementation of novel features like touch screen user interfaces and virtual type keyboard. It elevated Apple to the top of the mobile device market and forced other manufacturers to develop new products and operating systems in response to consumer expectations. After 3 years of the apple device's introduction in 2010, there were 150 apps available in the apple market and attained 1.5 million in 2015. Droid-Sec is the foremost android malware classification model implemented by applying deep learning [81]. More than 200 features are extracted by implementing static as well as dynamic analysis. Then the feature set is provided as input for DNN to classify android malwares. By experimenting with 250 malware apps and 250 legitimate apps performance is shown to reach an accuracy of 96.5%.

Binaries of Windows applications are analysed by modelling DNN [58]. The feature vector of size 1024 bit is obtained as the result of the extraction of 4 distinct sets of static features. Hidden layered DNN involves the classification of malwares by utilizing feature vectors. Android implements inter-component communication (ICC) analysis and as a result, flow graphs of data and dependency are obtained [270]. DroidDetector is a similar method developed and uses the same procedure for classifying android malwares like Droid-Sec [271]. CNN is constructed with the result of 192 extracted features obtained at the end applying static and dynamic analysis.

A malware classification method for the mobile environment is proposed on the basis of grayscale images [264]. To provide input for the classifiers, grayscale images are obtained after executing all the samples, and features are extracted. The implemented model is tested with a dataset that includes android samples of 50,000 and apple samples of 230. Among android samples 24,553 are malicious and 25,447 are genuine samples over 71 families. For apple 115 samples are distinguished from 10 families.

VisDroid [265] is an android malware detection scheme that utilizes an image-oriented analysis approach which classifies malware families. From various malware samples with a size of 4850 samples of android, 5 datasets of grayscale images are developed. Two kinds of image feature extraction are performed and trained with 6 distinct ML classifiers like RF, KNN, DT, Bagging, AdaBoost, and GBC. The development of the colour histogram is based on both local and global factors, with local features connected to robustness. To evaluate the efficiency of classifiers hybridized ensemble voting classification is implemented. Deep learning models RNN and Inception3 are utilized for the classifying of malwares with the help of datasets.

MobiTive [272] is proposed with the aim of enhancing android malware detection which influences the modified neural networks for affording real-time mobile environments. MobiTive detects mobile malwares which are either previously or dynamically installed. Though deep learning models are efficient in

Table 12
Comparison of recent Image-based malware detection techniques using DL.

Research paper	Features	Type of analysis	Algorithms of classification
Almomani et al. (2022) [266]	16 CNN algorithms, Grey coloured malware images, Leopard Android dataset	Image-based	An automated vision-based Android Malware Detection
Yuan et al. (2021) [267]	Multidimensional markov images	Image-based	LCNN
Li et al. (2021) [268]	binary, assembly and visible string	Image-based	Enhanced CNN
Gurumayum et al. (2020) [48]	Malimg dataset	Image-based	CNN+SVM
Francescoet al. (2020) [264]	Grayscale images	Image-based	CNN
Bakour et al. (2020) [265]	5 types of image datasets VisDroid	Image-based	RNN and Inception3
Yongkang et al. (2019) [73]	malVecNet	Image based	CNN
Su et al. (2018) [195]	Light-weighted model IoT environment	Image-based	CNN
Agarap et al. (2017) [37]	Malimg dataset	Image-based	CNN+MLP+GRU
Tobiyama et al. (2016) [261]	2D grayscale images	Image-based	CNN
Pascanu et al. (2015) [28]	Behavioural language model	Image-based	CNN

server-side malware detection, still there are some limitations like computation power, memory size, and energy while deploying in mobile devices directly. To overcome those shortcomings, the proposed system calculates and examines 5 important factors. The performance and accuracy of various feature types with neural networks are evaluated. Making use of multi-classification tasks, 70,130 android malwares are classified for evaluation.

A hybrid DL-enabled intelligent multi-vector malware detection method is presented [273]. It is developed by combining the salient features of CNN and Bidirectional Long Short-Term Memory (BiLSTM) for effective malware identification. Publicly available state-of-the-art datasets like Androzoo and AMD are utilized for experimentation to evaluate the detection method. For effective feature selection, 5 distinct feature selection algorithms are employed. The proposed mechanism outperforms the existing hybrid DL-enabled architectures like hybrid DNN-GRU and LSTM-GRU. 10-fold cross-validation is implemented for depicting the performance of the proposed method and the results are obtained as 99.05% accuracy, 99.39% precision and 99.41% F1-score.

To classify the android malware family, a multi-stream-based deep learning network is utilized [274]. Input data of CNN is obtained from a few files of every android malicious app in string format. A dimensional convolution filter-based network is applied over files or sections to classify the malware family. Further, gradient analysis is utilized for visualizing the main files and sections. DREBIN and AMD malware datasets are used for validating the effectiveness during experimentation. The findings show that the 1D CNN model achieves 93.2% accuracy, which is higher than the 2D CNN model's accuracy.

With the detailed analysis of the android permission system, the deep layer clustering technique is applied for identifying permission usage patterns of several groups of android apps [275]. A large size dataset is built with 16 000 apps and 118 features. Automatic identification and classification of permission usage patterns is aimed by utilizing the combination of Self Organizing Map (SOM) and K-means clustering algorithms. To validate the technique, the SVM classifier is employed with the identified patterns in terms of coherence and generalizability during malware detection. Results exhibit 93.5% accuracy for the model without potential malware and 94.1% accuracy for the model with potential malware.

One dimensional CNN-based android malware detection framework DroidMalwareDetector [276] is developed with the aim of automated feature extraction and selection, 1D data execution with CNN and permissions-based comprehensive malware analysis using intents and API calls. De-facto standard dataset with the size of 14,386 apps is utilized for training and testing purposes during experimentation. DroidMalwareDetector achieved an accuracy of 0.9, which is regarded as high, according to the results.

SOMDROID, an android malware detection framework by applying an unsupervised ML algorithm ANN is proposed in the work [277]. For model development, 5 lakhs different android apps of 30 distinct categories are collected and 1844 distinct features are extracted. Six various feature ranking methods are applied to choose significant features or feature sets. The Self-Organizing Map (SOM) algorithm is implemented over the selected features (i.e permissions, app ratings, number of downloads and API calls) or feature sets. To check the significance of selected features, test analysis is applied. According to the results, the detection rate is attained as 98.7% for the unknown malware apps in the real-time environment.

MGOPDroid [278] an android malware obfuscation variants detection system is proposed on the basis of multi-granularity opcode features [278]. The opcode feature distribution difference index that is obtained before and after the deployment of obfuscation is used with the TFIDF technique to get the opcode feature weight. On the basis of opcode encoding mapping rules, opcode features are then translated as sequences, and sequences are then turned into grayscale images to achieve feature visualization. DL model is integrated with Resnet and the global average pooling layer is developed for detecting malware variants. MGOPDroid can be used in mobile devices and ensures automated malware detection, behaviour updates, and real-time app installation monitoring. Based on the observed results, 96.35% and 94.55% detection accuracy are attained for obfuscated samples and obfuscated malwares respectively. DL-based recent android malware detection techniques are compared in Table 13.

6.3. Windows malware detection techniques

Cuckoo sandbox [32] is utilized for tracing the runtime behaviour of every malware in which every behaviour report is converted as a 20,000-bit vector with the help of uni-grams. A bit vector is used as the input for the DNN's signature creation. SVM is utilized for the classification of malware by accessing signatures. It utilizes 1800 malware application samples that have been experimented which attained a classification accuracy of 96.4%.

Hybrid Analysis for Detection of Malware (HADM) [279], a deep learning model, involves the extraction of features both statically and dynamically. Static analysis characteristics are extracted and transmitted as feature vectors, whereas dynamic analysis features are obtained as feature vectors as well as graphs. DNN is developed by training the feature vectors. Likewise, various graph kernels are incorporated with the graph-based feature vector. For classification purposes, kernel matrices are built and forwarded for ML classifiers like SVM. Thus, hierarchical Multiple Kernel Learning (MKL) is combined with deep learning, and accuracy is attained as 94.7%. MKL research work on windows

Table 13
Comparison of recent Android mobile malware detection techniques using DL.

Research paper	Features	Type of malware	Algorithms of classification
Tang et al. (2022) [278]	Multi-granularity opcode features, TFIDF algorithm, DL model, Resnet	Android	MGOPDroid
Arvind et al. (2022) [277]	ANN, SOM, t test analysis	Android	SOMDROID
Talha Kabakus et al. (2022) [276]	1DCNN, De-facto standard dataset, intents and API calls	Android	DroidMalwareDetector
Namrud et al. (2022) [275]	SOM, K means clustering, coherence and generalizability	Android	Deep layer clustering technique
Kim et al. (2022) [274]	1D CNN, gradient analysis, DREBIN and AMD malware datasets	Android	Multi-streams based deep leaning network
Gao et al. (2021) [163]	GCN, heterogeneous graph, App-API and API-API edges	Android	GDroid
Haq et al. (2021) [273]	CNN and BiLSTM, Androzoo and AMD datasets, DNN-GRU and LSTM-GRU,	Android	Mybrid DL enabled intelligent multi vector
Feng et al. (2021) [272]	Binary	Android	MobiTive
Francesco et al. (2020) [264]	Grey scale images	Android	DNN
Bakour et al. (2020) [265]	5 types of image datasets VisDroid	Android	RNN and Inception3
Yuan eta al (2016) [271]	Static and dynamic	Android	DroidDetector CNN
Saxe et al. (2015) [58]	Static features	Android	DNN
Wei et al. (2014) [270]	ICC flow graphs	Android	Amandroid
Yuan et al. (2014) [81]	More than 200 features by hybrid analysis	Android	Droid-Sec

malware is carried out before HADM by applying Gaussian kernel and Spectral kernel on the feature part [280]. A proposed CNN-based method for detecting Windows malware utilizes behaviour features of runtime executable files for both detecting and classifying unknown malwares [281]. 10-fold cross-validation is tested to evaluate the ability of this approach. Additionally, the malware executable files are handled effectively using the Relief Feature Selection technique. Results show a 97.968% of accuracy in windows malware detection.

A Deep Learning-based malware detection method that utilizes the combination of a visualization technique and CNN [282] is proposed that uses VGG16 network-based CNN. Both static and dynamic analysis approaches are incorporated for visualizing malware. Cuckoo Sandbox is utilized for dynamically analysing unknown samples and converting the results of dynamic analysis into visualization images using a designed algorithm. Further, the neural network is trained by these images and 2 different models are constructed. Two distinct detection models are tested with accuracy levels of 82.5% and 92.5%. Multiple dimensionality reduction methods like PCA and autoencoder are utilized for proposing distinct feature vectors for grouping [283]. Three models – HFVC, OEL, and BENN – include different dimensionality reduction methods and architectures. A public dataset of 138,047 benign and malware samples is used for examining the proposed method. The F1 score for both the OEL and BENN models was over 0.9.

Deep CNN and Xception CNN models are combined for malware visualization and classification in Windows and IoT environments [284]. Grayscale, RGB and Markov images are utilized by the deep CNNs making use of traditional learning and transfer learning techniques. In the initial step, all the malware images are generated from malware binaries. Markov probability matrix plays a key role in retaining global statistics of malware bytes during Markov image creation. Gabor filter-based technique is used for extracting textures and deep CNNs are trained on the ImageNet dataset which comprises 1.5 million images. Microsoft malware challenge dataset with the size of 500 GB is utilized for obtaining enhanced classification results in terms of accuracy, 99.05% for custom CNN and 99.22% for Xception CNN.

A static analysis deep unsupervised windows malware detection framework namely PROUD-MAL is proposed in the work [285]. The feature attention-based neural network (FANN) architecture serves as the pseudo labels for the feature attention blocks during training. A real-time malware dataset is collected with the deployment of low and high-interaction honeypots over an enterprise organizational network. Dataset with the size of 15,457 Portable Executable samples with the size of 25 GB is obtained after performing post-processing and cleaning. Among them, 8775 are malicious samples and 6681 are benign samples. Accuracy of PROUD-MAL is accomplished as above 98.09% on the created dataset which is regarded as having superiority to current convolutional ML techniques. Table 14 compares the important features of Deep Learning-based Windows malware detection techniques.

6.4. IoT malware detection techniques

Based on McKinsey Global Institute's report, 127 new devices are connected in IoT environment per second, and it is expected to reach above 64 Billion USD by the year 2025 which shows that the market of IoT remains at the top and it is forecasted to reach 11 Trillion USD in 2025 [286,287]. Due to the increase in the usage of IoT devices by multiple organizations over the world, the possibility of attacks also increases in recent days [288]. IoT Security's current state is given in-depth, along with the areas that need further attention [289]. Classical protection systems like anti-virus cannot act proactively to detect malware because of the enormous amount of malware development per day.

Malware detection method using deep eigenspace learning is implemented in the Internet of Battlefield Things (IoBT) environment through the sequence of opcodes [65]. Optimization is done by the implementation of eigenspace learning for differentiating legitimate and illegitimate apps. By applying ML, insertion attack of junk code is avoided. Neural networks and DT (Decision Tree) provide a detection accuracy of 94.93%. Besides, NB (Naive Bayes) (Naive bayes) and KNN (K-Nearest Neighbour) (K Nearest Neighbour) are also utilized with the accuracy of 95.90% detection of malwares.

Three different DL-based malware detection approaches are surveyed which employ CNN with the consideration of various features like byte sequence, colour images, and assembly sequences [290]. IoT malware samples of 15 000 and legitimate samples of 1000 are utilized for creating a training dataset. Assembly sequences are more accurate than byte sequences, according to implementation results. IoT systems attack models

Table 14
Comparison of Windows malware detection techniques using DL.

Research paper	Features	Type of malware	Algorithms of classification
Rizvi et al. (2022) [285]	FANN, static analysis	Windows	PROUD-MAL
Sharma et al. (2022) [284]	Markov images, transfer learning,	Windows & IoT	Deep CNN and Xception CNN
Aslam et al. (2021) [283]	PCA and auto encoder	Windows	HFVC, OEL, and BENN
Shiv darshan et al. (2019) [281]	Behaviour, Relief Feature Selection	Windows	CNN
Huang et al. (2018) [282]	Cuckoo sandbox	Windows	Visualization technique and CNN
Xu et al. (2016) [279]	Hybrid analysis	Windows	HADM, DNN
David et al. (2015) [32]	Behaviour report,	Windows	DeepSign, DNN+SVM

are investigated on the basis of ML techniques like supervised learning, unsupervised learning, and reinforcement learning.

While using SVM for detecting IoT-based android malware it attains an accuracy rate of 0.99 from the dataset created [291] that employs the Naive Bayes classifier [292]. An accuracy of 98% is attained by this classification method where a decision tree is also used. Further, a similar decision tree-based detection method obtained an F- score of 97% [293]. Similarly, while using Naïve Bayes the F- score is obtained as 51% and a 94% F- score is attained while using Logistic Regression. 98.2% accuracy is attained by implementing an IoT malware detection method that utilizes a KNN classifier and considers the fingerprint feature for extraction [44].

In order to minimize the damage to IoT devices, Dynamic analysis for IoT malware detection (DAIMD) is proposed by implementing a well-known as well as novel IoT malware detection technique [104]. CNN model is used by the DAIMD scheme for learning. Dynamic IoT malware analysis is performed in a nested cloud environment and then behaviours associated with memory, network, virtual file system, process, and system calls are extracted. Behaviour images are created by converting extracted behaviour, and then CNN is used to train and categorize those images. To tackle the problems with portraying the flaws caused by variances in platforms, the RGB picture approach [268] is advised for representing IoT features. The created image includes various information about malware such as binary, assembly, and visible string. Utilizing a combination of the self-attention mechanism and spatial pyramid pooling, enhanced CNN is used to identify the variants. Enhanced CNN overcomes the size issues related to IoT malware. The proposed method is compared with a few states of the art over 10 000 samples (i.e 25 families) and accuracy is achieved at 98.57% for IoT malwares based on experimental results.

Lightweight Convolutional Neural Networks (LCNN) based IoT malware classification approach is proposed in the work [267]. Multidimensional Markov images are obtained by the conversion of malware binaries without performing reverse and dynamic analysis processes. To classify malware images, the design of LCNN is intended into 2 processes-depth wise convolutions and channel shuffle. When comparing the LCNN model to the VGG16 model, the former is smaller (1MB) while the latter is larger (552.57MB). 95% accuracy is gained while applying the proposed model with the grey images of various datasets of IoT malware. 99.356% accuracy is attained with the Microsoft dataset.

Federated learning (FL) based Industrial IoT based android malware detection architecture; Fed-IIoT suggested by Taheri et al. [294], composed of two parts — participant side and server side. On the participant side, Generative Adversarial Networks (GAN) and federated GAN are used for triggering the data by implementing two poisoning attacks. The server side is involved in global model monitoring and robust training model construction. In addition, it proposed a defence algorithm namely avoiding anomaly in aggregation by a GAN network (A3GAN) which is formed on the basis of cumulating FL and GAN algorithms in order to detect server-side adversaries. The Byzantine defence algorithm is modified and adopted and two countermeasure solutions Byzantine Median (BM) and Byzantine Krum (BK) are applied for verifying effectiveness. Based on the results, accuracy is increased by 8% when compared with existing methods.

An effective feature subset selection approach is proposed to detect IoT Botnet cyber-attacks making use of meta-heuristic methods like scatter search (ScS) and K-Medoid sampling [295]. In order to achieve optimal detection of IoT botnets with reduced features, various classifiers are proposed like Averaged Two-dependence Estimator (A2DE) Bayesian, JChaid* decision tree, DL-based CNN, Deep MLP and unsupervised classification making use of hybrid K-means clustering and genetic algorithm (HGC). The UNSW-NB15 dataset is utilized for experimental purposes. While considering the results, 100% accuracy is attained and 0 false alarm rate for identifying network breaches.

A deep active learning-based IIoT malware detection framework is proposed by utilizing PSE (phase space embedding), SAE (sparse autoencoder) and LSTM (long–short term memory) network with an action-value function for training active learners [296]. By identifying the occurrence, this approach considers PSE or SAE as a policy when deciding how to proceed. It is a hypothetical framework where fusion strategies are evaluated to improve the performance of malware classification. On the basis of observed experimentation results, by using 50% of the training data, classification accuracy is achieved at 95.1% and adversarial classification accuracy is attained at 86.9%.

IoT malware detection and prevention framework is proposed by utilizing hybrid optimization techniques and deep learning techniques [297]. A cyber security warning system is developed with a large-scale dataset where the index system is first designed and then index factors are selected and computed for evaluating the situation. Grey Wolf Optimization algorithm (GWO) and Whale optimization algorithm are integrated for building the framework (WGWO). GWO enhances the feature selection to obtain appropriate features by removing noisy features. Various pre-processing methods are integrated by the smart initialization step for ensuring informative features. Tensorflow deep neural network is used for classification during experimentation. For testing, malware samples are gathered from the Mailing database. Results depict that 99% accuracy is obtained which is considered higher.

Deep Learning (DL) based Bidirectional-Gated Recurrent Unit-Convolutional Neural Network (Bi-GRU-CNN) model is proposed for detecting IoT malwares and classifying the families [298]. For the Bi-GRU-CNN model, binary file byte sequences are provided as an input feature in the ELF (Executable and Linkable) format. The performance of the model is evaluated by making use of RNN-based DL model combinations. According to the results, 100% accuracy is attained for IoT malware detection and 98% accuracy is achieved for IoT malware family classification.

A three-phase deep malware detection framework DMD-DWT-GAN is proposed for IoT-based smart agriculture (IoT-SA) systems by combining Discrete Wavelet Transform (DWT) and Generative Adversarial Network (GAN) [299]. A multi-resolution analysis is performed by applying DWT where the image is decomposed as Approximation coefficients (Ac) and Detail Coefficients (Dc).

Table 15
Comparison of recent IoT malware detection techniques using DL.

Research paper	Features	Type of malware	Algorithms of classification
Santosh et al. (2022) [299]	Multiresolution & in-depth analysis, LCNN, MalImg datasets	IoT	DMD-DWT-GAN
Chaganti et al. (2022) [298]	ELF, RNN based DL model combinations	IoT	Bi-GRU-CNN
Nagaraju et al. (2022) [297]	GWO & Whale optimization, Tensorflow DNN	IoT	Hybrid optimization (WGWO)
Khowaja et al. (2021) [296]	PSE, SAE, hypothetical framework, IIoT	IoT	Q-learning and LSTM
Panda et al. (2021) [295]	A2DE Bayesian, JChaid* DT, Deep CNN, Deep MLP, hybrid K-means clustering and HGC	IoT	Scatter search (ScS) and K-Medoid sampling
Taheri et al. (2021) [294]	Federated GAN, A3GAN, Byzantine defence algorithm	IoT	Fed-IIoT
Yuan et al. (2021) [267]	Multidimensional markov images, depth wise convolution and channel shuffle	IoT malware	LCNN
Li et al. (2021) [268] Jeon et al. (2020) [104] Duc Nguyen et al. (2019) [44]	Binary, assembly and visible string Behaviour images Fingerprint feature	IoT malware IoT malware IoT malware	Enhanced CNN DAIMD KNN
Nguyen et al. (2018) [290]	Byte sequence, colour images, and assembly sequences	IoT malware	CNN
Azmoodeh et al. (2018) [65]	Sequence of opcodes	IoT malware	Deep eigenspace learning

Malware families are involved in in-depth analysis by employing lightweight Convolutional Neural Network (LCNN) which is considered to be multi-class classifier. IoT malware and MalImg datasets are utilized for evaluating the performance of the proposed framework. Experimental results show that 99.99% accuracy has been attained for the two datasets. Table 15 contrasts the most recent methods used to identify IoT malware using Machine Learning.

6.5. APT detection techniques

An expensive and frequently used destructive attack on the target system is the Advanced Persistent Threat (APT) attack. For businesses, governments, and organizations' information security systems, this attack has become a challenge. Using Machine Learning or Deep Learning algorithms to examine clues and unusual patterns in network data, methods for detecting and stopping APT assaults have gained popularity in recent years.

Deep Belief Network (DBN) is utilized for malware detection and the performance of the system is compared with the analysis of classical neural networks [300]. The performance analysis of the malware detection system is done by the various ML classifiers like DT, SVM, and KNN. Issues of signature-based methods are overcome by DBN and also allowed precise detection with the use of autoencoders, which reduces the feature set's complexity. Concealed executable program files are revealed with the help of SVM, NB, and DT. To avoid the application of cryptographic functions for gaining money, the Maximal Frequent Pattern of live ransomware is exploited [46]. MLP algorithm is utilized for analysing the threat. It attains an accuracy rate of 99% with a time of 10 s. Some families of ransomware are not detectable by MLP which can be resolved using Dynamic Link Libraries which also minimizes the false +ve rate. RNN is utilized for detecting malwares at earlier stages by analysing the behaviours of static and dynamic data with the help of a sandbox [34]. Besides, RNN with Hidden Markov Models provides the maximum accuracy rate for forecasting malware. Malicious payloads are prohibited, reducing the risk of malware vulnerabilities. It accurately reduces the detection time by 94% i.e. within 5 s.

It is challenging to identify an APT attack because of how long they stay active on the network and the risk that their huge traffic volume would cause the system to crash. By automatically extracting and choosing the features in the neural network's hidden layers, a 6-layer deep learning model is suggested [301]. For the purpose of rapid detection and classification of APT attacks on the NSL-KDD dataset, machine learning techniques such as C5.0 decision trees, Bayesian networks, and deep learning are employed. Additionally, a 10-fold cross-validation technique is employed to test these models. Based on the experimental findings, 95.64%, 88.37%, and 98.85% accuracy are determined for the C5.0 decision tree, Bayesian network, and 6-layer deep learning models, respectively. When compared to earlier work involving APT attack detection on the NSL-KDD dataset, the 6-layer deep learning model exhibited the best execution and performance in terms of accuracy.

The Strange Behaviour Inspection (SBI) model is implemented for detecting APT attacks over the first potential victim [302]. It mainly concentrates on watching the footsteps of APT attacks that investigated and selected behavioural characteristics of the potential victim machine. The credential dumping approach is used to stop APT attackers from obtaining complete user and password lists and to stop data exfiltration by APT attackers. Additionally, it is used to keep an eye on dangerous activity on the CPU, file systems, RAM, and Microsoft Windows registry. Based on the experimental results it is observed that, the SBI model attained 99.8% detection accuracy and reduced the detection time to 2.7 min.

Stacked Autoencoder with Long Short-Term Memory (SAE-LSTM) and Convolutional Neural Networks with Long Short-Term Memory Network (CNN-LSTM) are employed in a hybrid deep learning technique to evaluate large amounts of network traffic in order to look for signs of APT assaults [303]. The proposed approach is assessed using the valid dataset "DAPT2020", which includes all APT stages. The experimental findings show that, for detecting malicious behaviour in each APT stage, the hybrid deep learning technique outperformed the individual deep learning model.

A combined deep learning model is used to detect APT assaults based on network traffic analysis [304]. In order to study and detect indications of APT assaults in network traffic, separate deep learning networks including multilayer perceptron (MLP), convolutional neural network (CNN), and long short-term memory (LSTM) are sought after, created, and integrated into combined deep learning networks. Extracting IP features based on flow and classifying APT attack IPs are the two main steps that the combined deep learning model executes in order to detect APT attack signals. Combined deep learning model demonstrated in

Table 16
Comparison of recent APTs detection techniques using DL.

Research paper	Features	Type of malware	Algorithms of classification
Xuan et al. (2022) [306]	GCN, GIN, behaviour profile,	APT	Deep graph network
Li et al. (2022) [305]	Edge game and edge AI, Edge Bayesian Stackelberg game	APT	Explainable Intelligence-Driven Defence Mechanism
Xuan et al. (2021) [304] Alrehaili et al. (2021) [303]	Network traffic analysis, MLP, CNN, LSTM SAE-LSTM, CNN-LSTM, DAPT2020	APT APT	Combined deep learning model Hybrid deep learning technique
Mohamed et al. (2021) [302]	Credential dumping technique, footsteps of APT attack, first potential victim	APT	SBI Model
Hassannataj et al. (2020) [301]	NSL-KDD dataset, C5.0 decision trees, Bayesian networks	APT	6-layer deep learning model
Rhode et al. (2018) [34]	Behaviours	APT	RNN with Hidden Markov Model

the experimental segment their improved capacity to guarantee accuracy on all measurements ranging from 93 to 98%.

An explainable intelligence-driven APT defence mechanism is proposed on the basis of an edge game and edge AI (Artificial Intelligence) approach [305]. Real-time APT detection, intelligence generation and explanatory functions analysis are supported by the proposed mechanism. Edge Bayesian Stackelberg game and threat intelligence-based defence strategy model is developed for balancing rapid response and resource allocation. Defendant and attacker interactions are modelled in it. APT defence game is provided with equilibrium and existence conditions. Optimal solutions for attackers and defenders are provided for various kinds of resource budgets. It provides edge device protection against attack models over dynamic games.

In order to detect APT malwares over workstations, a deep-graph network-based method is proposed by analysing the behaviour profile of malware [306]. It consists of two primary tasks. In the first task, behaviour profiles of malware are built on the basis of collecting and validating event IDs of kernel workstations. Results are obtained with the labels of normal, malicious, suspicious or unknown after the completion of building behaviour profiles. In the second task, malwares are detected on the basis of analysing behaviour profiles with the help of a Graph Convolutional Network (GCN) where the Graph Isomorphism Network (GIN) method is utilized. Results depicted 90.57% accuracy, 90.51% precision, and 90.57% recall attained during experimentation. The latest DL-based APT detection techniques are compared in Table 16.

6.6. Ransomware detection techniques

Ransomware represents a very significant threat since new families and variants are consistently being discovered on the internet and dark web. Because of the nature of the encryption techniques they employ, ransomware outbreaks are challenging to recover from. This rise in ransomware is also related to the expansion of artificial intelligence. After Machine Learning, Deep Learning can identify zero-day threats; hence there is great interest in researching Deep Learning-based ransomware detection methods. Deep Ransomware Threat Hunting and Intelligence System (DRTHIS) is implemented for segregating the samples into malicious or benign and to relate the family of ransomware with it [307]. To classify ransomware, CNN is combined with LSTM with the help of the Softmax algorithm and results are obtained as 97.2% TPR and 2.7% FPR.

An Industrial IoT-based ransomware detection model is developed by utilizing Asynchronous Peer-to-Peer Federated Learning (AP2PFL) and Deep Learning (DL) techniques [308]. This model comprises two modules which are Data Purifying Model (DPM) and Diagnostic and Decision Module (DDM). DPM involves data refinement and representation using Contractive Denoising Auto-Encoder (CDAE). DDM is utilized for identifying ransomware and its stages based on Deep Neural Network (DNN) and Batch Normalization (BN). Every edge gateway module is associated asynchronously with neighbours. This model handles homogeneous and heterogeneous data and combats evasion attacks. X-IIoTID, ISOT, and NSL-KDD datasets are used for validating the proposed experimental setup and robustness is evaluated by implementing black-and-white box evasion attack methods.

Deep Convolutional Generative Adversarial Network (DCGAN) and Transferred Generating Adversarial Network-Intrusion Detection System (TGAN) based dual generative adversarial networks detection framework is proposed to detect attacks of anonymous variant encrypted ransomware [309]. In TGAN, the transfer learning mechanism is utilized for enhancing adversarial sample generation ability and detection ability. In addition, the reconstruction loss function is established for increasing the ability of the discriminator. CICIDS2017, KDD99, SWaT and WADI ransomware datasets are utilized for experimentation. The performance of TGAN-IDS is evaluated with the metrics like detection accuracy, recall and F1-score. Table 17 provides a summary of recently proposed DL-based Ransomware detection techniques.

7. Inference

In this section, conclusions drawn by this comprehensive examination are presented.

- Malware detection methods with 100% efficiency in terms of accuracy, TPR, FPR, precision, and recall are a nightmare for developers because new generation malwares are coded in a complicated manner and advanced evasion techniques are employed against familiar protection mechanisms like firewalls and anti-viruses.
- Static analysis techniques cannot handle malwares incorporated with obfuscation techniques. It does not opt for the run time environment but it provides better results for familiar malware families.
- Behaviour and signature-based approaches fail to battle with obfuscation techniques implemented in malwares. So dynamic analysis is applied. Later, a combination of both static and dynamic analysis (hybrid analysis) techniques is employed for efficient results.
- Visualization techniques and image-based analysis techniques are incorporated to create the feature set based on images to reduce complexity. Further Deep Learning models like CNN and RNN are utilized alone or with a combination of ML classifiers.
- Over 50% of digitally connected people are using windows in this world. To avoid the increased number of violations in the Windows environment hybrid analysis methods are applied for feature extraction and Deep Learning models like CNN are employed for classifying malwares instead of ML classifiers.

Table 17
Comparison of recent Ransomware detection techniques using DL.

Type of method	Research paper	Features	Algorithms of classification
Network traffic monitoring based	Zhang et al. (2022) [309]	CICIDS2017, KDD99, SWaT, WADI, transfer learning	DCGAN & TGAN
Others	Al-Hawawreh et al. (2021) [308]	DPM, DDM, DNN, BN	AP2PFL
API calls based	Homayoun et al. (2019) [307]	CNN, LSTM, Softmax algorithm	DRTHIS

- The Internet of Things is expanding since every gadget is now interconnected. Even the companies that make these devices lack the necessary expertise. A combination of ML classifiers and DL models is encouraged for classification. By using both static and dynamic analysis, feature extraction parameters including heuristics, power usage, colour images, and byte sequences are also used.
- APTs are specifically designed for performing denial-of-service attacks. Predicting APTs before the beginning of the runtime environment is a tough and risky task. Recent detection methods of APTs utilize an automatic detection framework considering features based on a socket, IP address, open XML, real-time traffic, etc.
- Private confidential information is targeted that applies cryptographic functions in ransomware attacks. Recent ransomware detection techniques can be classified into different types based on their methods such as (i). Behaviour-based, (ii). IRP monitoring-based, (iii). NW traffic monitoring-based, (iv). Memory-based, (v). API calls-based, (vi). Android environment-based and other methods ensuing from the above.
- The android environment seems more unsafe than Mac OS where personal information is mainly targeted. Mobile malware detection methods utilize static, dynamic, and image-based analysis techniques. For feature extraction also ML classifiers are utilized and classification is performed by adopting DL models.
- From the overall observation of the current scenario, malware detection methods should be accompanied by (i). An intelligent system (ii). Automotive detection mechanisms and (iii). Earlier stage detection without any delay. These factors would implement a protection mechanism which would help combat new generation malwares effectively.

8. Conclusion

Being proactive rather than reactive is a key lesson that security gives us. Today, developing a malware detection system is challenging, especially when dealing with new generation malware. Advanced evasion strategies have enabled the evolution of new generations of malware, which had very significant effects. However, Deep Learning-based malware detection technology reduces the flaws of both conventional and traditional methods. This paper presents a systematic review of malware detection using Deep Learning techniques. On the basis of the evolution towards Deep Learning-based techniques, research taxonomy is proposed. Recent techniques for detecting malware on Android, iOS, IoT, Windows, APTs, and Ransomware are also explored and compared. Finally, by giving researchers a thorough understanding of malware analysis, this study will steer researchers in the appropriate path for developing mitigation approaches for both conventional and complex malware.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to thank VIT-AP University, the editors and the reviewers. The authors would like to thank Dr. S. V. Kota Reddy, Vice Chancellor, VIT-AP University for his continuous support and encouragement. A special thanks to Dr. Hari Seetha, Dr. Ganesh Reddy Karri, Devi Priya V.S and the team members of Center of Excellence, Cyber Security & Artificial Intelligence and Robotics (AIR), VIT-AP University. The author Mr. M. Gopinath would like to thank Miss. Winny Elizabeth Philip for continuously supporting and helping to carryout this research.

References

[1] R. Anderson, et al., Measuring the cost of cybercrime, in: The Economics of Information Security and Privacy, Springer, Berlin, Germany, 2013, pp. 265–300.

[2] https://ciso.economictimes.indiatimes.com/news/most-firms-see-rise-in-cyberattacks-during-pandemic-survey/75043660.

[3] https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-covid-19-report-reveals-pandemic-threat-evolution/.

[4] https://www.marketsandmarkets.com/Market-Reports/malware-analysis-market-108766513.html.

[5] https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2019-2020.pdf.

[6] Priyanka Dixit, Sanjay Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Comp. Sci. Rev. (ISSN: 1574-0137) 39 (2021) 100317.

[7] Omer Aslan, Refik Samet, A comprehensive review on malware detection approaches, IEEE Trans. 8 (2020) 6249–6271.

[8] Y. Li, K. Xiong, T. Chin, C. Hu, A machine learning framework for domain generation algorithm-based malware detection, IEEE Access 7 (2019) 32765–32782.

[9] E. Gandotra, D. Bansal, S. Sofat, Malware analysis and classification: a survey, J. Inf. Secur. 5 (2014) 56–64.

[10] N. Udayakumar, V.J. Saglani, A.V. Cupta, T. Subbulakshmi, Malware classification using machine learning algorithms, in: 2018 2nd International Conference on Trends in Electronics and Informatics, ICOEI, Tirunelveli, 2018, pp. 1–9.

[11] M. Alazab, S. Venkataraman, P. Watters, Towards understanding malware behaviour by the extraction of API calls, in: Proc. 2nd Cybercrime Trustworthy Comput. Workshop, Jul. 2010, Vol. 7, 2019, pp. 52–59, 46736.

[12] M. Tang, M. Alazab, Y. Luo, Big data for cybersecurity: Vulnerability disclosure trends and dependencies, IEEE Trans. Big Data 5 (3) (2019) 317–329.

[13] D. Gibert, C. Mateu, J. Planes, A hierarchical convolutional neural network for malware classification, in: The International Joint Conference on Neural Networks 2019, IEEE, 2019, pp. 1–8.

[14] M. Alazab, Profiling and classifying the behavior of malicious codes, J. Syst. Softw. 100 (2015) 91–102.

[15] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, J. Yearwood, Hybrids of support vector machine wrapper and filter based framework for malware detection, Future Gener. Comput. Syst. 55 (2016) 376–390.

[16] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, A. Alazab, Cybercrime: The case of obfuscated malware, in: C.K. Georgiadis, H. Jahankhani, E. Pimenidis, R. Bashroush, A. Al-Nemrat (Eds.), Global Security, Safety and Sustainability & e-Democracy, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 99, Springer, Berlin, Germany, 2012.

[17] E. Raff, J. Sylvester, C. Nicholas, Learning the PE header, malware detection with minimal domain knowledge, in: Proc. 10th ACMWorkshop Artif. Intell. Secur, ACM, New York, NY, USA, 2017, pp. 121–132.

[18] C. Rossow, et al., Prudent practices for designing malware experiments: Status quo and outlook, in: Proc. IEEE Symp. Secur. Privacy, SP, 2012, pp. 65–79.
[19] H.S. Anderson, A. Kharkar, B. Filar, P. Roth, Evading Machine Learning Malware Detection, Black Hat, New York, NY, USA, 2017.
[20] R. Verma, Security analytics: Adapting data science for security challenges, in: Proc. 4th ACM Int. Workshop Secur. Privacy Anal., ACM, New York, NY, USA, 2018, pp. 40–41.
[21] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[22] Sudhakar, S. Kumar, An emerging threat fileless malware: a survey and research challenges, Cybersecur 3 (2020) 1.
[23] Sibi Chakkaravarthy, D. Sangeetha, V. Vaidehi, A survey on malware analysis and mitigation techniques, Comp. Sci. Rev. 32 (2019) 1–23.
[24] Daniel Gibert, Carles Mateu, Jordi Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, J. Netw. Comput. Appl. (ISSN: 1084-8045) 153 (2020) 102526.
[25] N. Koroniotis, N. Moustafa, E. Sitnikova, Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions, IEEE Access 7 (2019) 61764–61785.
[26] Priyanka Dixit, Sanjay Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Comp. Sci. Rev. (ISSN: 1574-0137) 39 (2021) 100317.
[27] A. Davis, M. Wolff, Deep learning on disassembly data, 2015, URL. https://www.blackhat.com/docs/us-15/materials/us-15-Davis-Deep-Learning-On-Disassembly.pdf.
[28] R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, Malware classification with recurrent networks, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2015, p. 1916e1920, http://dx.doi.org/10.1109/ICASSP.2015.7178304.
[29] O.D. Gibert Llaurad, Convolutional Neural Networks for Malware Classification (Master's thesis), Universitat Politfiecnica de Catalunya, 2016.
[30] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto, Novel feature extraction, selection and fusion for effective malware family classification, in: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY '16, ACM, New York, NY, USA, 2016, p. 183e194.
[31] W. Hardy, L. Chen, S. Hou, Y. Ye, X. Li, Dl4md: A Deep Learning Framework for Intelligent Malware Detection, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens, 2016, p. 61e67.
[32] O.E. David, N.S. Netanyahu, Deepsign: deep learning for automatic malware signature generation and classification, in: 2015 International Joint Conference on Neural Networks, IJCNN, 2015, p. 1e8.
[33] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C. Nicholas, Malware detection by eating a whole exe, 2017, [Online]. Available: https://arxiv.org/abs/1710.09435.
[34] M. Rhode, P. Burnap, K. Jones, Early-stage malware prediction using recurrent neural networks, Comput. Secur. 77 (2018) 578–594.
[35] M. Krcál, O. vec, M. Bálek, O. Jaek, Deep convolutional malware classifiers can learn from raw executables and labels only, 2018, [Online]. Available: https://openreview.net/forum?id=HkHrmM1PM.
[36] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, P. de Geus, Malicious software classification using VGG16 deep neural network's bottleneck features, in: Information Technology-New Generations, Springer, Cham, Switzerland, 2018, pp. 51–59.
[37] A.F. Agarap, F.J.H. Pepito, Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification, 2017.
[38] W. Huang, J.W. Stokes, Mtnet: A multi-task neural network for dynamic malware classification, in: Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment, Springer, Cham, Switzerland, 2016, pp. 399–418.
[39] A. Feizollah, N.B. Anuar, R. Salleh, G. Suarez-Tangil, S. Furnell, Androdialysis: analysis of android intent effectiveness in malware detection, Comput. Secur. 65 (2017) 121–134.
[40] Zhiang Fang, Jun Yeonjoon Wang, Jiaxuan Geng, Xuan Khan, Feature Selection for Malware Detection Based on Reinforcement Learning, Vol. 7, IEEE, 2019, 176177-176187.
[41] I. Firdausi, C. lim, A. Erwin, A.S. Nugroho, Analysis of machine learning techniques used in behaviorbased malware detection, in: Proceedings of the 2nd International Conference Advances on Computing Control, and Telecommunications Technology, 2010, pp. 201–203.
[42] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, L. Mao, Maldae: detecting and explaining malware based on correlation and fusion of static and dynamic characteristics, Comput. Secur. 83 (2019a) 208–233.
[43] W. Han, J. Xue, Y. Wang, Z. Liu, Z. Kong, Malinsight: a systematic profiling based malware detection framework, J. Netw. Comput. Appl. 125 (2019b) 236–250, http://www.sciencedirect.com/science/article/pii/S1084804518303503.
[44] T. Duc Nguyen, S. Marchal, A.-R. Sadeghi, DÏoT: a self-learning system for detecting compromised IoT devices, in: Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst., IEEE, 2019.
[45] F. Wu, L. Xiao, J. Zhu, Bayesian model updating method based android malware detection for IoT services, in: 2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019, IEEE, 2019, pp. 61–66.
[46] M. Moradi, M. Zulkernine, A neural network based system for intrusion detection and classification of attacks, in: Proceedings of the IEEE International Conference on Advances in Intelligent Systems-Theory and Applications, 2004, pp. 15–18.
[47] H. Zhu, Y. Li, R. Li, J. Li, Z. You, H. Song, SEDMDroid: An enhanced stacking ensemble framework for android malware detection, IEEE Trans. Netw. Sci. Eng. 8 (2) (2021) 984–994.
[48] Gurumayum Akash Sharma, Khundrakpam Johnson Singh, Maisnam Debabrata Singh, A deep learning approach to image-based malware analysis, progress in computing, analytics and networking, in: Advances in Intelligent Systems and Computing 1119, 2020, pp. 327–339.
[49] A. Irshad, R. Maurya, M.K. Dutta, R. Burget, V. Uher, Feature optimization for run time analysis of malware in windows operating system using machine learning approach, in: 2019 42nd International Conference on Telecommunications and Signal Processing, TSP, Budapest, Hungary, 2019, pp. 255–260.
[50] Z.A. Genç, G. Lenzini, P.Y.A. Ryan, No random, no ransom: a key to stop cryptographic ransomware, in: C. Giuffrida, S. Bardin, G. Blanc (Eds.), DIMVA 2018, in: LNCS, vol. 10885, Springer, Cham, 2018, pp. 234–255.
[51] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, T. Yada, Efficient dynamic malware analysis based on network behavior using deep learning, in: Proc. IEEE Global Commun. Conf., GLOBECOM, 2016, pp. 1–7.
[52] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Proc. Australas. Joint Conf. Artif. Intell., Springer, Cham, Switzerland, 2016, pp. 137–149.
[53] E. Raff, et al., An investigation of byte n-gram features for malware classification, J. Comput. Virol. Hacking Tech. 14 (1) (2018) 1–20.
[54] H.S. Anderson, P. Roth, EMBER: An open dataset for training static PE malware machine learning models, 2018, https://arxiv.org/abs/1804.04637.
[55] https://arxiv.org/abs/1804.04637.
[56] https://www.unb.ca/cic/datasets/.
[57] https://www.sonicwall.com/2022-cyber-threat-report/sonicwall-cyber-threat-report-thank-you/.
[58] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in: Proc. 10th Int. Conf. Malicious Unwanted Softw. (Malware), 2015, pp. 11–20.
[59] TaeGuen Kim, BooJoong Kang, Mina Rho, Sakir Sezer, Eul Gyu Im, A multimodal deep learning method for android malware detection using various features, IEEE Trans. Inf. Forensics Secur. http://dx.doi.org/10.1109/TIFS.2018.2866319.
[60] Zhiang Fang, Junfeng Wang, Boya Li, Siqi Wu, Yingjie Zhou, Haiying Huang, Evading Anti-Malware Engines with Deep Reinforcement Learning, Vol 7, IEEE, 2019, pp. 48867–48879.
[61] A. Damodaran, F. Di Troia, C.A. Visaggio, T.H. Austin, M. Stamp, A comparison of static, dynamic, and hybrid analysis for malware detection, J. Comput. Virol. Hacking Tech. 13 (1) (2017) 1–12.
[62] Wei Zhong, Feng Gu, A multi-level deep learning system for malware detection, Expert Syst. Appl. 133 (2019) 151–162.
[63] R. Vinayakumar, Mamoun Alazab, K.P. Soman, Prabaharan Poornachandran, Sitalakshmi Venkatraman, Robust intelligent Malware detectionusing deep learning, IEEE Trans. 7 (2019) 46717–46738.
[64] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, Zero-day malware detection based on supervised learning algorithms of API call signatures, in: Proc. 9th Australas. Data Mining Conf., Vol. 121, Australian Computer Society, Ballarat, Australia, 2011, pp. 171–182.
[65] Azmoodeh, Choo, Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning, IEEE Trans. Sustain. Comput. (2018) http://dx.doi.org/10.1109/TSUSC.2018.2809665.
[66] L. Nataraj, A Signal Processing Approach to Malware Analysis, Univ. California, Santa Barbara, CA, USA, 2015.
[67] L. Nataraj, B.S. Manjunath, SPAM: Signal processing to analyze malware, 2016, [Online]. Available: https://arxiv.org/abs/1605.05280.
[68] L. Nataraj, D. Kirat, B.S. Manjunath, G. Vigna, Sarvam: Search and retrieval of malware, in: Proc. Annu. Comput. Secur. Conf. (ACSAC) Worshop Next Gener. Malware Attacks Defence, NGMAD, 2013, pp. 1–9.

[69] L. Nataraj, V. Yegneswaran, P. Porras, J. Zhang, A comparative assessment of malware classification using binary texture analysis and dynamic analysis, in: Proc. 4th ACM Workshop Secur. Artif. Intell., ACM, New York, NY, USA, pp. 21–30.

[70] L. Nataraj, G. Jacob, B.S. Manjunath, Detecting Packed Executables Based on Raw Binary Data, Tech. Rep., Univ. California, Santa Barbara, CA, USA, 2010.

[71] M. Farrokhmanesh, A. Hamzeh, A novel method for malware detection using audio signal processing techniques, in: Proc. Artif. Intell. Robot., IRANOPEN, 2016, pp. 85–91.

[72] D. Kirat, L. Nataraj, G. Vigna, B.S. Manjunath, SigMal: A static signal processing based malware triage, in: Proc. 29th Annu. Comput. Secur. Appl. Conf., ACM, New York, NY, USA, 2013, pp. 89–98.

[73] Yongkang Jiang, Shenghong Li, Yue Wu(B), Futai Zou, A novel image-based malware classification model using deep learning, in: 26th International Conference, ICONIP 2019 Sydney, NSW, Australia, December 12–15, 2019 Proceedings, Part II.

[74] D. Gibert, J. Bjar, C. Mateu, J. Planes, D. Solis, R. Vicens, Convolutional neural networks for classification of malware assembly code, in: Recent Advances in Artificial Intelligence Research and Development - Proceedings of the 20th International Conference of the Catalan Association for Artificial Intelligence, Deltebre, Terres de l'Ebre, Spain, October 25-27, 2017, 2017, pp. 221–226.

[75] D. Gibert, C. Mateu, J. Planes, An end-to-end deep learning architecture for classification of malware's binary content, in: V. Krkov, Y. Manolopoulos, B. Hammer, L. Iliadis, I. Maglogiannis (Eds.), Artificial Neural Networks a8nd Machine Learning ICANN 2018, Springer International Publishing, Cham, 2018, pp. 383–391.

[76] K. Kosmidis, C. Kalloniatis, Machine learning and images for malware detection and classification, 2017.

[77] D. Gibert, C. Mateu, J. Planes, R. Vicens, Using convolutional neural networks for classification of malware represented as images, J. Comput. Virol. Hacking Tech. (2018).

[78] D. Gibert, C. Mateu, J. Planes, R. Vicens, Classification of malware by using structural entropy on convolutional neural networks, in: IAAI Conference on Artificial Intelligence, 2018b, pp. 7759–7764.

[79] M. Dib, S. Torabi, E. Bou-Harb, C. Assi, A multi-dimensional deep learning framework for IoT malware classification and family attribution, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1165–1177.

[80] G.E. Dahl, J.W. Stokes, L. Deng, D. Yu, Large-scale malware classification using random projections and neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 3422–3426.

[81] Z. Yuan, Y. Lu, Z. Wang, Y. Xue, Droid sec: Deep learning in Android malware detection, ACM SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 371–372.

[82] Y. Bengio, Learning deep architectures for AL, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.

[83] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, Handb. Brain Theory Neural Netw. 3361 (10) (1995) 1995.

[84] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, p. 1097e1105.

[85] Quan Le, Oisín Boydell, Brian Mac Namee, Mark Scanlon, Deep learning at the shallow end: Malware classification for non-domain experts, Digit. Investig. 26 (2018) S118eS126, Proceedings of the Eighteenth Annual DFRWS USA.

[86] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735e1780.

[87] B. Athiwaratkun, J.W. Stokes, Malware classification with lstm and gru language models and a character-level CNN, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2017, pp. 2482–2486.

[88] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 12 (5) (2017) 21954–21961.

[89] V.V. Strelkov, A new similarity measure for histogram comparison and its application in time series analysis, Pattern Recognit. Lett. 29 (13) (2008) 1768–1774.

[90] B. Kang, H.S. Kim, T. Kim, et al., Fast malware family detection method using control flow graphs, in: Proceedings of 2011 ACM Symposium on Research in Applied Computation, ACM, 2011, pp. 287–292.

[91] L.E. Gonzalez, R.A. Vazquez, Malware classification using euclidean distance and artificial neural networks, in: 2013 12th Mexican International Conference on Artificial Intelligence, MICAI, IEEE, 2013, pp. 103–108.

[92] C. Annachhatre, T.H. Austin, M. Stamp, Hidden Markov models for malware classification, J. Comput. Virol. Hacking Tech. 11 (2) (2015) 59–73.

[93] K.S. Han, J.H. Lim, B. Kang, et al., Malware analysis using visualized images and entropy graphs, Int. J. Inf. Secur. 14 (1) (2015) 1–14.

[94] M.M. Alani, A.I. Awad, PAIRED: An explainable lightweight android malware detection system, IEEE Access 10 (2022) 73214–73228.

[95] K. Rieck, P. Trinius, C. Willems, T. Holz, Automatic analysis of malware behavior using machine learning, J. Comput. Secur. 19 (4) (2011) 639–668.

[96] S. Rasthofer, S. Arzt, E. Bodden, A machine-learning approach for classifying and categorizing android sources and sinks, in: Proceedings of the Network and Distributed System Security Symposium, 2014, pp. 23–26.

[97] G. Schwenk, K. Rieck, Adaptive detection of covert communication in HTTP requests, in: Proceedings of the 7th European Conference on Comput. Netw. Defence (EC2ND'11), 2011, pp. 25–32.

[98] N. Nissim, Y. Lapidot, A. Cohen, Y. Elovici, Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining, Knowl.-Based Syst. 153 (2018) (2018) 147–175.

[99] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, J. Vandewalle, Machine learning in side-channel analysis: A first study, J. Cryptogr. Eng. 1 (4) (2011) 293–302.

[100] J. Demme, et al., On the feasibility of online malware detection with performance counters, ACM SIGARCH Comput. Archit. News 41 (3) (2013) 559.

[101] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, M. Prvulovic, EDDIE: EM-based detection of deviations in program execution, in: Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA'17, 2017, pp. 333–346.

[102] T.N. Nguyen, Q.-D. Ngo, H.-T. Nguyen, G.L. Nguyen, An advanced computing approach for IoT-botnet detection in industrial internet of things, IEEE Trans. Ind. Inform. 18 (11) (2022) 8298–8306.

[103] M.'. Husainiamer, M.M. Saudi, A. Ahmad, Classification for iOS mobile malware inspired by phylogenetic: Proof of concept, in: 2020 IEEE Conference on Open Systems, ICOS, 2020, pp. 59–63.

[104] J. Jeon, J.H. Park, Y. Jeong, Dynamic analysis for IoT malware detection with convolution neural network model, IEEE Access 8 (2020) 96899–96911.

[105] A. Pekta, T. Acarman, Classification of malware families based on runtime behaviors, J. Inf. Secur. Appl. 37 (2017) 91–100.

[106] Microft: Sam cybersecurity engagement kit, Internet (2018) https://assets.microsoft.com/en-nz/cybersecurity-sam-engagement-kit.pdf.

[107] Y. Ye, T. Li, D.A. Adjeroh, S.S. Iyengar, A survey on Malware detection using data mining techniques, ACM Comput. Surv. 50 (3) (2017) 41.

[108] L. Nataraj, S. Karthikeyan, G. Jacob, B.S. Manjunath, Malware images: Visualization and automatic classification, in: Proc. 8th Int. Symp. Vis. Cyber Secur., ACM, New York, NY, USA, 2011, p. 4.

[109] J. Yan, Y. Qi, Q. Rao, Detecting malware with an ensemble method based on deep neural network, Secur. Commun. Netw. 16 (2018).

[110] T.M. Kebede, O. Djaneye-Boundjou, B.N. Narayanan, A. Ralescu, D. Kapp, Classification of malware programs using autoencoders based deep learning architecture and its application to the Microsoft Malware classification challenge (big 2015) dataset, in: 2017 IEEE National Aerospace and Electronics Conference, NAECON, IEEE, 2017, pp. 70–75, A Novel Image-Based Malware Classification Model Using Deep Learning 161.

[111] H.-J. Kim, Image-based malware classification using convolutional neural network, in: J.J. Park, V. Loia, G. Yi, Y. Sung (Eds.), CUTE/CSA -2017, in: LNEE, vol. 474, Springer, Singapore, 2018, pp. 1352–1357.

[112] F.C.C. Garcia, F.P. Muga, Random forest for malware classification, Cryptogr. Secur. (2016) arXiv.

[113] E. Raff, C. Nicholas, An alternative to NCD for large sequences, Lempel–Ziv Jaccard distance, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 1007–1015.

[114] J. Drew, M. Hahsler, T. Moore, Polymorphic malware detection using sequence classification methods and ensembles, EURASIP J. Inf. Secur. 2 (1) (2017).

[115] Digital: Trends, 2011, https://www.digitaltrends.com/android/smartphone-sales-exceed-those-of-pcs-for-first-time-applesmashes-record/.

[116] M.G. Ciobanu, F. Fasano, F. Martinelli, F. Mercaldo, A. Santone, A data life cycle modeling proposal by means of formal methods, in: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, ACM, 2019, pp. 670–672.

[117] F. Fasano, F. Martinelli, F. Mercaldo, A. Santone, Energy consumption metrics for mobile device dynamic malware detection, Procedia Comput. Sci. 159 (2019) 1045–1052.

[118] F. Martinelli, F. Mercaldo, A. Santone, Social network polluting contents detection through deep learning techniques, in: 2019 International Joint Conference on Neural Networks, IJCNN, IEEE, 2019, pp. 1–10.

[119] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, A.K. Sangaiah, Android malware detection based on system call sequences andLSTM, Multimedia Tools Appl. 78 (4) (2019) 3979–3999.

[120] V. Rastogi, Y. Chen, X. Jiang, Catch me if you can: evaluating android anti-malware against transformation attacks, IEEE Trans. Inf. Forensics Secur. 9 (1) (2014) 99–108.

[121] X. Jiang, Y. Zhou, Dissecting android malware: characterization and evolution, in: 2012 IEEE Symposium on Security and Privacy, IEEE, 2012, pp. 95–109.

[122] G. Canfora, F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, C.A. Visaggio, Leila: formal tool for identifying mobile malicious behaviour, IEEE Trans. Softw. Eng. 45 (12) (2018) 1230–1252.

[123] Google: Play, 2015, https://play.google.com/store.

[124] Apk: Tool, 2018, https://ibotpeaches.github.io/Apktool/.

[125] F. Fasano, F. Martinelli, F. Mercaldo, A. Santone, Investigating mobile applications quality in official and third-party marketplaces, in: Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, SCITEPRESS-Science and Technology Publications, Lda, 2019, pp. 169–178.

[126] F. Fasano, F. Martinelli, F. Mercaldo, A. Santone, Measuring mobile applications quality and security in higher education, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 5319–5321.

[127] M. Scalas, D. Maiorca, F. Mercaldo, C.A. Visaggio, F. Martinelli, G. Giacinto, On the effectiveness of system API-related information for android ransomware detection, Comput. Secur. 86 (2019) 168–182.

[128] Ah: Myth, 2018, https://github.com/AhMyth/AhMyth-Android-RAT.

[129] Droid: Jack, 2018, http://droidjack.net/.

[130] F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, A.K. Sangaiah, A. Cimitile, Evaluating model checking for cyber threats code obfuscation identification, J. Parallel Distrib. Comput. 119 (2018) 203–218.

[131] J. Oberheide, C. Mille, Dissecting the android bouncer, in: SummerCon, 2012.

[132] F. Mercaldo, V. Nardone, A. Santone, Ransomware inside out, in: 2016 11th International Conference on Availability, Reliability and Security, ARES, IEEE, 2016, pp. 628–637.

[133] F. Mercaldo, V. Nardone, A. Santone, C.A. Visaggio, Hey malware, i can find you!, in: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, IEEE, 2016, pp. 261–262.

[134] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, S. Ioannidis, Rage against the virtual machine: hindering dynamic analysis of android malware, in: Proceedings of the Seventh European Workshop on System Security, ACM, 2014, p. 5.

[135] Asma Razgallah, Raphaël Khoury, Sylvain Hallé, Kobra Khanmohammadi, A survey of malware detection in Android apps: Recommendations and perspectives for future research, Comp. Sci. Rev. (ISSN: 1574-0137) 39 (2021) 100358.

[136] Shivi Garg, Niyati Baliyan, Comparative analysis of android and iOS from security viewpoint, Comp. Sci. Rev. (ISSN: 1574-0137) 40 (2021) 100372.

[137] G. Canfora, F. Mercaldo, C.A. Visaggio, A classifier of malicious android applications, in: Proceedings of the 2nd International Workshop on Security of Mobile Applications, in Conjunction with the International Conference on Availability, Reliability and Security, 2013.

[138] A. Cimitile, F. Mercaldo, V. Nardone, A. Santone, C.A. Visaggio, Talos: no more ransomware victims with formal methods, Int. J. Inf. Secur. 17 (2017) 1–20.

[139] G. Canfora, A. Di Sorbo, F. Mercaldo, C.A. Visaggio, Obfuscation techniques against signature-based detection: a case study, in: 2015 Mobile Systems Technologies Workshop, MST, IEEE, 2015, pp. 21–26.

[140] F. Mercaldo, V. Nardone, A. Santone, C.A. Visaggio, Ransomware steals your phone. formal methods rescue it, in: International Conference on Formal Techniques for Distributed Objects, Components, and Systems, Springer, 2016, pp. 212–221.

[141] D. Octeau, P. McDaniel, S. Jha, A. Bartel, E. Bodden, J. Klein, Y. Le Traon, Effective inter-component communication mapping in android: an essential step towards holistic security analysis, in: Presented as Part of the 22nd USENIX Security Symposium (USENIX Security 13), 2013, pp. 543–558.

[142] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, P. McDaniel, Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps, ACM SIGPLAN Not. 49 (6) (2014) 259–269.

[143] M. Lindorfer, M. Neugschwandtner, C. Platzer, Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis, in: 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), Vol. 2, IEEE, 2015, pp. 422–433.

[144] M. Faiella, A. LaMarra, F. Martinelli, F. Mercaldo, A. Saracino, M. Sheikhalishahi, A distributed framework for collaborative and dynamic analysis of android malware, in: 2017 25th Euromicro International Conference on Parallel, Distributed and Network- Based Processing, PDP, IEEE, 2017, pp. 321–328.

[145] F. Martinelli, F. Mercaldo, A. Saracino, Bridemaid: An hybrid tool for accurate detection of android malware, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ACM, 2017, pp. 899–901.

[146] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss, Andromaly : a behavioral malware detection framework for android devices, J. Intell. Inf. Syst. 38 (1) (2012) 161–190.

[147] T. Blasing, A.D. Schmidt, L. Batyuk, S.A. Camtepe, S. Albayrak, An android application sandbox system for suspicious software detection, in: Proceedings of 5th International Conference on Malicious and Unwanted Software, 2010.

[148] B. Dixon, Y. Jiang, A. Jaiantilal, S. Mishra, Location based power analysis to detect malicious code in smartphones, in: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, 2011.

[149] M. Polino, A. Scorti, F. Maggi, S. Zanero, Jackdaw: Towards automatic reverse engineering of large datasets of binaries, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2015, pp. 121–143.

[150] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, A.N. Sheth, Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones, ACM Trans. Comput. Syst. (TOCS) 32 (2) (2014) 5.

[151] A. Shabtai, U. Kanonov, Y. Elovici, Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method, J. Syst. Softw. 83 (8) (2010) 1524–1537.

[152] Y. Zhou, Z. Wang, W. Zhou, X. Jiang, Hey, you, get off of my market: detecting malicious apps in official and alternative android markets, in: Proceedings of the Network and Distributed System Security Symposium, NDSS, 2012.

[153] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, W. Zou, Smartdroid: an automatic system for revealing UI-based trigger conditions in android applications, in: Proceedings of the 2nd ACMWorkshop on Security and Privacy in Smartphones and Mobile Devices, SPSM, New York, NY, USA, 2012, pp. 93–104.

[154] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. van der Veen, C. Platzer, Andrubis-1, 000, 000 apps later: a view on current android malware behaviors, in: Proceedings of the 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS, 2014.

[155] M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp, J. Hoffmann, Mobile-sandbox: combining static and dynamic analysis with machine-learning techniques, Int. J. Inf. Secur. 14 (2014) 141–153.

[156] A. Ferrante, E. Medvet, F. Mercaldo, J. Milosevic, C.A. Visaggio, Spotting the malicious moment: Characterizing malware behavior using dynamic features, in: 2016 11th International Conference on Availability, Reliability and Security, ARES, IEEE, 2016, pp. 372–381.

[157] H. Hashemi, A. Hamzeh, Visual malware detection using local malicious pattern, J. Comput. Virol. Hacking Tech. 15 (1) (2019) 1–14.

[158] M. Farrokhmanesh, A. Hamzeh, Music classification as a new approach for malware detection, J. Comput. Virol. Hacking Tech. 15 (2) (2019) 77–96.

[159] H. Rathore, S.K. Sahay, Towards robust android malware detection models using adversarial learning, in: 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2021, pp. 424–425.

[160] R. Surendran, T. Thomas, S. Emmanuel, On existence of common malicious system call codes in android malware families, IEEE Trans. Reliab. 70 (1) (2021) 248–260.

[161] Y. Hei, et al., Hawk: Rapid android malware detection through heterogeneous graph attention networks, IEEE Trans. Neural Netw. Learn. Syst. http://dx.doi.org/10.1109/TNNLS.2021.3105617.

[162] H. Bai, N. Xie, X. Di, Q. Ye, FAMD: A fast multifeature android malware detection framework, design, and implementation, IEEE Access 8 (2020) 194729–194740.

[163] Han Gao, Shaoyin Cheng, Weiming Zhang, GDroid: Android malware detection and classification with graph convolutional network, Comput. Secur. (ISSN: 0167-4048) 106 (2021) 102264.

[164] Satheesh Kumar Sasidharan, Ciza Thomas, ProDroid — An android malware detection framework based on profile hidden Markov model, Pervasive Mob. Comput. (ISSN: 1574-1192) 72 (2021) 101336.

[165] J. Xu, Y. Li, R.H. Deng, K. Xu, SDAC: A slow-aging solution for android malware detection using semantic distance based API clustering, IEEE Trans. Dependable Secure Comput. 19 (2) (2022) 1149–1163.

[166] Shaojie Yang, Yongjun Wang, Haoran Xu, Fangliang Xu, Mantun Chen, An android malware detection and classification approach based on contrastive learning, Comput. Secur. (ISSN: 0167-4048) 123 (2022) 102915.

[167] S. Seraj, S. Khodambashi, M. Pavlidis, et al., HamDroid: permission-based harmful android anti-malware detection using neural networks, Neural Comput. Appl. 34 (2022) 15165–15174.
[168] Hui-juan Zhu, Wei Gu, Liang-min Wang, Zhi-cheng Xu, Victor S. Sheng, Android malware detection based on multi-head squeeze-and-excitation residual network, Expert Syst. Appl. (ISSN: 0957-4174) 212 (2023) 118705.
[169] Shannon Williams, Mobile Malware and Exploitation Amongst Biggest Cyber Threats for 2020, Security Brief Asia, 2020, [online] Available: https://securitybrief.asia/story/mobile-malware-and-exploitation-amongst-biggest-cyber-threats-for-2020.
[170] Swati Khandelwal, 'Exodus' Surveillance Malware Found Targeting Apple iOS Users, The Hacker News, 2019, [online] Available: https://thehackernews.com/2019/04/exodus-ios-malware.html.
[171] Swati Khandelwal, Powerful FinSpy Spyware Found Targeting iOS and Android Users in Myanmar, 2019.
[172] D. Damopoulos, G. Kambourakis, S. Gritzalis, iSAM: an iPhone stealth airborne malware, in: IFIP International Information Security Conference, Springer, 2011, pp. 17–28.
[173] L. Garcıa, R.J. Rodrıguez, Apeek under the hood of iOSmalware, in: 2016 10th International Conference on Availability, Reliability and Security, ARES, 2016.
[174] A. Cimitile, F. Martinelli, F. Mercaldo, Machine learning meets iOS malware: Identifying malicious applications on apple environment, in: ICISSP, 2017, pp. 487–492.
[175] M. Szydlowski, M. Egele, C. Kruegel, G. Vigna, Challenges for dynamic analysis of iOS applications, in: Open Problems in Network Security, Springer, 2012, pp. 65–77.
[176] M. Lindorfer, B. Miller, M. Neugschwandtner, C. Platzer, Take a bite-finding the worm in the apple, in: 2013 9th International Conference on Information, Communications and Signal Processing, ICICS, IEEE, 2013, pp. 1–5.
[177] H.H. Pajouh, A. Dehghantanha, R. Khayami, et al., Intelligent OS X malware threat detection with code inspection, J. Comput. Virol. Hacking Tech. 14 (2018) 213–223.
[178] S. Bojjagani, V.N. Sastry, VAPTAi: A threat model for vulnerability assessment and penetration testing of android and iOS mobile banking apps, in: 2017 IEEE 3rd International Conference on Collaboration and Internet Computing, CIC, 2017, pp. 77–86.
[179] G. Zhou, M. Duan, Q. Xi, H. Wu, ChanDet: Detection model for potential channel of iOS applications, J. Phys. Conf. Ser. 1187 (4) (2019).
[180] Y. Lee, X. Wang, X. Liao, X. Wang, Understanding illicit UI in iOS apps through hidden UI analysis, IEEE Trans. Dependable Secure Comput. 18 (5) (2021) 2390–2402.
[181] Nir Nissim, et al., Novel active learning methods for enhanced PC malware detection in windows OS, Expert Syst. Appl. 41 (13) (2014) 5843–5857.
[182] P.V. Shijo, A. Salim, Integrated static and dynamic analysis for malware detection, Procedia Comput. Sci. 46 (2015) 804–811.
[183] Gandeva B. Satrya, Niken D.W. Cahyani, Ritchie F. Andreta, The detection of 8 type malware botnet using hybrid malware analysis in executable file windows operating systems, in: Proceedings of the 17th International Conference on Electronic Commerce 2015, ACM, 2015, p. 5.
[184] Tulika Mithal, Kshitij Shah, Dushyant Kumar Singh, Case studies on intelligent approaches for static malware analysis, in: Emerging Research in Computing, Information, Communication and Applications, Springer, Singapore, 2016, pp. 555–567.
[185] Bander Alsulami, et al., Lightweight behavioral malware detection for windows platforms, in: 2017 12th International Conference on Malicious and Unwanted Software, MALWARE, IEEE, 2017, pp. 75–81.
[186] Shamsul Huda, et al., A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection, Future Gener. Comput. Syst. 83 (2018) 193–207.
[187] H. Kim, J. Smith, K.G. Shin, Detecting energy-greedy anomalies and mobile malware variants, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, 2008.
[188] S. Dija, J. Ajana, V. Indu, M. Sabarinath, Cyber forensics: Discovering traces of malware on windows systems, in: 2020 IEEE Recent Advances in Intelligent Computational Systems, RAICS, 2020, pp. 141–146.
[189] R. Yang, et al., RATScope: Recording and reconstructing missing RAT semantic behaviors for forensic analysis on windows, IEEE Trans. Dependable Secure Comput. http://dx.doi.org/10.1109/TDSC.2020.3032570.
[190] S. Yousefi, F. Derakhshan, H. Karimipour, H.S. Aghdasi, An efficient route planning model for mobile agents on the internet of things using Markov decision process, Ad Hoc Netw. 98 (2020) 102053.
[191] M. Al-Asli, T.A. Ghaleb, Review of signature-based techniques in antivirus products, in: 2019 International Conference on Computer and Information Sciences, ICCIS, IEEE, 2019, pp. 1–6.
[192] H.H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, K.K.R. Choo, A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks, IEEE Trans. Emerg. Top. Comput. 7 (2) (2019) 314–323.
[193] S. Sharmeen, S. Huda, J.H. Abawajy, W. Nagy Ismail, M.M. Hassan, Malware threats and detection for industrial mobile-IoT networks, IEEE Access 6 (2018) 15941–15957.
[194] A. Lohachab, B. Karambir, L.A. Lohachab, Critical analysis of ddos-an emerging security threat over IoT networks, J. Commun. Inf. Netw. 3 (3) (2018) 57–78.
[195] J. Su, V. Danilo Vasconcellos, S. Prasad, S. Daniele, Y. Feng, K. Sakurai, Lightweight classification of IoT malware based on image recognition, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC, Tokyo, 2018, pp. 664–669.
[196] S. Papafotikas, A. Kakarountas, A machine-learning clustering approach for intrusion detection to IoT devices, in: 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), IEEE, 2019, pp. 1–6.
[197] L. Xiao, X. Wan, X. Lu, Y. Zhang, D. Wu, IoT security techniques based on machine learning: how do IoT devices use AI to enhance security? IEEE Signal Process. Mag. 35 (5) (2018) 41–49.
[198] Y.-T. Lee, et al., Cross platform IoT-malware family classification based on printable strings, in: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 775–784.
[199] S.M.P. Dinakarrao, et al., Cognitive and scalable technique for securing IoT networks against malware epidemics, IEEE Access 8 (2020) 138508–138528.
[200] M.N. Aman, U. Javaid, B. Sikdar, IoT-Proctor: A secure and lightweight device patching framework for mitigating malware spread in IoT networks, IEEE Syst. J. http://dx.doi.org/10.1109/JSYST.2021.3070404.
[201] T. Trajanovski, N. Zhang, An automated and comprehensive framework for IoT botnet detection and analysis (IoT-BDA), IEEE Access 9 (2021) 124360–124383.
[202] J. Bhayo, R. Jafaq, A. Ahmed, S. Hameed, S.A. Shah, A time-efficient approach toward ddos attack detection in IoT network using SDN, IEEE Internet Things J. 9 (5) (2022) 3612–3630.
[203] R. Kalakoti, S. Nõmm, H. Bahsi, In-depth feature selection for the statistical machine learning-based botnet detection in IoT networks, IEEE Access 10 (2022) 94518–94535.
[204] A. Azmoodeh, A. Dehghantanha, M. Conti, K.K.R. Choo, Detecting crypto-ransomware in IoT networks based on energy consumption footprint, J. Ambient Intell. Humaniz. Comput. 9 (4) (2018) 1141–1152.
[205] I. Ghafira, et al., Detection of advanced persistent threat using machine-learning correlation analysis, 89 (2018) 349–359.
[206] S.T. Liu, Y.M. Chen, S.J. Lin, A novel search engine to uncover potential victims for APT investigations, in: C.-H. Hsu, X. Li, X. Shi, R. Zheng (Eds.), NPC 2013, in: LNCS, vol. 8147, Springer, Heidelberg, 2013, pp. 405–416.
[207] M. Balduzzi, V. Ciangaglini, R. McArdle, Targeted attacks detection with spunge, in: 2013 Eleventh Annual Conference on Privacy, Security and Trust, IEEE, 2013, pp. 185–194.
[208] Z. Ma, Q. Li, X. Meng, Discovering suspicious APT families through a large-scale domain graph in information-centric IoT, IEEE Access 7 (2019) 13917–13926.
[209] X. Liu, L. Li, Z. Ma, X. Lin, J. Cao, Design of APT attack defence system based on dynamic deception, in: 2019 IEEE 5th International Conference on Computer and Communications, ICCC, Chengdu, China, 2019, pp. 1655–1659.
[210] H. Sun, C. Shen, C. Weng, A flexible framework for malicious open XML document detection based on APT attacks, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 2005–2006.
[211] R. Coulter, J. Zhang, L. Pan, Y. Xiang, Unmasking windows advanced persistent threat execution, in: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 268–276.
[212] Y. Su, Research on APT attack based on game model, in: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC, 2020, pp. 295–299.
[213] W. Alghamdi, M. Schukat, Practical implementation of APTs on PTP time synchronisation networks, in: 2020 31st Irish Signals and Systems Conference, ISSC, 2020, pp. 1–5.
[214] Y. Qi, R. Jiang, Y. Jia, A. Li, An APT attack analysis framework based on self-define rules and mapreduce, in: 2020 IEEE Fifth International Conference on Data Science in Cyberspace, DSC, 2020, pp. 61–66.
[215] S.-P. Hong, C.-H. Lim, H.J. Lee, APT attack response system through AM-HIDS, in: 2021 23rd International Conference on Advanced Communication Technology, ICACT, 2021, pp. 271–274.

[216] L.-X. Yang, K. Huang, X. Yang, Y. Zhang, Y. Xiang, Y.Y. Tang, Defence against advanced persistent threat through data backup and recovery, IEEE Trans. Netw. Sci. Eng. 8 (3) (2021) 2001–2013.

[217] T. Halabi, O.A. Wahab, R. Al Mallah, M. Zulkernine, Protecting the internet of vehicles against advanced persistent threats: A Bayesian stackelberg game, IEEE Trans. Reliab. 70 (3) (2021) 970–985.

[218] Jaafer Al-Saraireh, Ala' Masarweh, A novel approach for detecting advanced persistent threats, Egypt. Inform. J. (ISSN: 1110-8665) (2022).

[219] N. Scaife, H. Carter, P. Traynor, K.R.B. Butler, CryptoLock (and drop it): stopping ransomware attacks on user data, in: 2016 IEEE 36th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2016.

[220] T. Dargahi, A. Dehghantanha, P.N. Bahrami, M. Conti, G. Bianchi, L. Benedetto, A cyber-kill-chain based taxonomy of crypto-ransomware features, J. Comput. Virol. Hacking Tech. 15 (4) (2019) 277–305.

[221] A. Kharraz, W. Robertson, E. Kirda, Protecting against ransomware: a new line of research or restating classic ideas? IEEE Secur. Priv. 16 (3) (2018) 103–107.

[222] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, E. Kirda, UNVEIL: a largescale, automated approach to detecting ransomware, in: 25th USENIX Security Symposium (USENIX Security 2016), USENIX Association, Austin, TX, 2016, pp. 757–772.

[223] J.A. Gomez-Hernandez, L. Álvarez-Gonzaalez, P. Garcıa-Teodoro, R-Locker: thwarting ransomware action through a honeyfile-based approach, Comput. Secur. 73 (2018) 389–398.

[224] B.A.S. Al-rimy, M.A. Maarof, Y.A. Prasetyo, S.Z.M. Shaid, A.F.M. Ariffin, Zero-day aware decision fusion-based model for crypto-ransomware early detection, Int. J. Integr. Eng. 10 (6) (2018) 82–88.

[225] T. Honda, K. Mukaiyama, T. Shirai, T. Ohki, M. Nishigaki, Ransomware detection considering user's document editing, in: 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications, AINA, IEEE, 2018.

[226] S. Jung, Y. Won, Ransomware detection method based on context-aware entropy analysis, Soft Comput. 22 (20) (2018) 6731–6740.

[227] S. Mehnaz, A. Mudgerikar, E. Bertino, Rwguard: a real-time detection system against cryptographic ransomware, in: M. Bailey, T. Holz, M. Stamatogiannakis, S. Ioannidis (Eds.), RAID 2018, in: LNCS, vol. 11050, Springer, Cham, 2018, pp. 114–136.

[228] A. Continella, et al., ShieldFS: a self-healing, ransomware-aware filesystem, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, ACM, New York, 2016, pp. 336–347.

[229] G. Bottazzi, G.F. Italiano, D. Spera, Preventing ransomware attacks through file system filter drivers, in: Second Italian Conference on Cyber Security, Milan, Italy, 2018.

[230] D. Morato, E. Berrueta, E. Magana, M. Izal, Ransomware early detection by the analysis of file sharing traffic, J. Netw. Comput. Appl. 124 (2018) 14–32.

[231] K. Cabaj, M. Gregorczyk, W. Mazurczyk, Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics, Comput. Electr. Eng. 66 (2018) 353–368.

[232] K. Cabaj, W. Mazurczyk, Using software-defined networking for ransomware mitigation: the case of cryptowall, IEEE Netw. 30 (6) (2016) 14–20.

[233] D.F. Netto, K.M. Shony, E.R. Lalson, An integrated approach for detecting ransomware using static and dynamic analysis, in: 2018 International CET Conference on Control, Communication, and Computing (IC4), IEEE, 2018.

[234] O.M.K. Alhawi, J. Baldwin, A. Dehghantanha, Leveraging machine learning techniques for windows ransomware network traffic detection, in: A. Dehghantanha, M. Conti, T. Dargahi (Eds.), Cyber Threat Intelligence, in: AIS, vol. 70, Springer, Cham, 2018, pp. 93–106.

[235] J.-Y. Paik, J.-H. Choi, R. Jin, J. Wang, E.-S. Cho, A storage-level detection mechanism against crypto-ransomware, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, ACM Press, 2018.

[236] S.H. Baek, Y. Jung, A. Mohaisen, S. Lee, D. Nyang, SSD-insider: internal defence of the solid-state drive against ransomware with perfect data recovery, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2018.

[237] N.B. Harikrishnan, K.P. Soman, Detecting ransomware using GURLS, in: 2018 Second International Conference on Advances in Electronics, Computers and Communications, ICAECC, IEEE, 2018.

[238] A. Ferrante, M. Malek, F. Martinelli, F. Mercaldo, J. Milosevic, Extinguishing ransomware - a hybrid approach to android ransomware detection, in: A. Imine, J.M. Fernandez, J.-Y. Marion, L. Logrippo, J. Garcia-Alfaro (Eds.), FPS 2017, in: LNCS, vol. 10723, Springer, Cham, 2018, pp. 242–258.

[239] M. Scalas, D. Maiorca, F. Mercaldo, C.A. Visaggio, F. Martinelli, G. Giacinto, R-PackDroid: practical on-device detection of Android ransomware, 2018, CoRR, abs/1805.09563.

[240] S. Song, B. Kim, S. Lee, The effective ransomware prevention technique using process monitoring on Android platform, Mob. Inf. Syst. 2016 (2016) 1–9.

[241] J. Baldwin, A. Dehghantanha, Leveraging support vector machine for opcode density based detection of crypto-ransomware, in: A. Dehghantanha, M. Conti, T. Dargahi (Eds.), Cyber Threat Intelligence, in: AIS, vol. 70, Springer, Cham, 2018, pp. 107–136.

[242] A. Adamov, A. Carlsson, Reinforcement learning for anti-ransomware testing, in: 2020 IEEE East-West Design & Test Symposium, EWDTS, 2020, pp. 1–5.

[243] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence, IEEE Trans. Emerg. Top. Comput. 8 (2) (2020) 341–351.

[244] U. Urooj, M. Aizaini Bin Maarof, B. Ali Saleh Al-rimy, A proposed adaptive pre-encryption crypto-ransomware early detection model, in: 2021 3rd International Cyber Resilience Conference, CRC, 2021, pp. 1–6.

[245] D. Min, Y. Ko, R. Walker, J. Lee, Y. Kim, A content-based ransomware detection and backup solid-state drive for ransomware defence, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. http://dx.doi.org/10.1109/TCAD.2021.3099084.

[246] F. Khan, C. Ncube, L.K. Ramasamy, S. Kadry, Y. Nam, A digital DNA sequencing engine for ransomware detection using machine learning, IEEE Access 8 (2020) 119710–119719.

[247] S. Sibi Chakkaravarthy, D. Sangeetha, M.V. Cruz, V. Vaidehi, B. Raman, Design of intrusion detection honeypot using social leopard algorithm to detect IoT ransomware attacks, IEEE Access 8 (2020) 169944–169956.

[248] M. Wazid, A.K. Das, S. Shetty, BSFR-SH: Blockchain-enabled security framework against ransomware attacks for smart healthcare, IEEE Trans. Consum. Electron. (2022).

[249] Ahmad O. Almashhadani, Domhnall Carlin, Mustafa Kaiiali, Sakir Sezer, MFMCNS: a multi-feature and multi-classifier network-based system for ransomworm detection, Comput. Secur. (ISSN: 0167-4048) 121 (2022) 102860.

[250] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, Mikel Izal, Crypto-ransomware detection using machine learning models in file-sharing network scenarios with encrypted traffic, Expert Syst. Appl. (ISSN: 0957-4174) 209 (2022) 118299, http://dx.doi.org/10.1016/j.eswa.2022.118299.

[251] Masoudeh Keshavarzi, Hamid Reza Ghaffary, An ontology-driven framework for knowledge representation of digital extortion attacks, Comput. Hum. Behav. (ISSN: 0747-5632) 139 (2023) 107520.

[252] Liu Liu, Baosheng Wang, Automatic malware detection using deep learning based on static analysis, in: ICPCSEE 2017, Part I, CCIS 727, 2017, pp. 500–507.

[253] Y. Tang, Deep learning using linear support vector machines, 2013.

[254] K. Grosse, N. Papernot, P. Manoharan, M. Backes, P. McDaniel, Adversarial perturbations against deep neural networks for malware classification, 2016, arXiv:1606.04435.

[255] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, F. Roli, Adversarial malware binaries: Evading deep learning for malware detection in executables, in: Proc. 26th Eur. Signal Process. Conf., EUSIPCO, 2018.

[256] P. Prasse, L. Machlica, T. Pevn, J. Havelka, T. Scheffer, Malware detection by analysing encrypted network traffic with neural networks, in: M. Ceci, J. Hollmn, L. Todorovski, C. Vens, S. Deroski (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, 2017, pp. 73–88.

[257] M. AL-Hawawreh, N. Moustafa, E. Sitnikova, Identification of malicious activities in industrial internet of things based on deep learning models, J. Inf. Secur. Appl. 41 (2018) 1–11.

[258] N. Kumar, S. Mukhopadhyay, M. Gupta, A. Handa, K.S. Shukla, Malware classification using early-stage behavioral analysis, in: 2019 14th Asia Joint Conference on Information Security (AsiaJCIS), 2019, pp. 16–23.

[259] M. Rhode, L. Tuson, P. Burnap, K. Jones, Lab to soc: robust features for dynamic malware detection, in: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Industry Track, 2019, pp. 13–16.

[260] X. Huang, L. Ma, W. Yang, et al., A method for windows malware detection based on deep learning, J. Signal Process. Syst. 93 (2021) 265–273.

[261] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, Malware detection with deep neural network using process behavior, in: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Vol. 2, 2016, p. 577e582.

[262] R. Ronen, M. Radu, C.E. Feuerstein, E. Yomtov, M. Ahmadi, Microsoft Malware classification challenge, Cryptogr. Secur. (2018) arXiv.

[263] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[264] Francesco Mercaldo, Antonella Santone, Deep learning for image-based mobile malware detection, J. Comput. Virol. Hacking Tech. (2020).
[265] K. Bakour, H.M. Ünver, VisDroid: Android malware classification based on local and global image features, a bag of visual words and machine learning techniques, Neural Comput. Appl. (2020).
[266] I. Almomani, A. Alkhayer, W. El-Shafai, An automated vision-based deep learning model for efficient detection of android malware attacks, IEEE Access 10 (2022) 2700–2720.
[267] B. Yuan, J. Wang, P. Wu, X. Qing, IoT Malware classification based on lightweight convolutional neural networks, IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3100063.
[268] Q. Li, J. Mi, W. Li, J. Wang, M. Cheng, CNN-based malware variants detection method for internet of things, IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3075694.
[269] https://gs.statcounter.com/osmarketshare/mobile/worldwide.
[270] F. Wei, S. Roy, X. Ou, et al., Amandroid: a precise and general inter-component data flow analysis framework for security vetting of android apps, in: Proceedings of the 2014ACMSIGSACConference on Computer and Communications Security, ACM, 2014, pp. 1329–1341.
[271] Z. Yuan, Y. Lu, Y. Xue, Droiddetector: android malware characterization and detection using deep learning, Tsinghua Sci. Technol. 21 (01) (2016) 114–123.
[272] R. Feng, S. Chen, X. Xie, G. Meng, S.-W. Lin, Y. Liu, A performance-sensitive malware detection system using deep learning on mobile devices, IEEE Trans. Inf. Forensics Secur. 16 (2021) 1563–1578, http://dx.doi.org/10.1109/TIFS.2020.3025436.
[273] I.U. Haq, T.A. Khan, A. Akhunzada, A dynamic robust DL-based model for android malware detection, IEEE Access 9 (2021) 74510–74521.
[274] H.-I. Kim, M. Kang, S.-J. Cho, S.-I. Choi, Efficient deep learning network with multi-streams for android malware family classification, IEEE Access 10 (2022) 5518–5532.
[275] Z. Namrud, S. Kpodjedo, A. Bali, C. Talhi, Deep-layer clustering to identify permission usage patterns of android app categories, IEEE Access 10 (2022) 24240–24254.
[276] Abdullah Talha Kabakus, DroidMalwareDetector: A novel android malware detection framework based on convolutional neural network, Expert Syst. Appl. (ISSN: 0957-4174) 206 (2022) 117833.
[277] Arvind Mahindru, Amrit Sangal, SOMDROID: android malware detection by artificial neural network trained using unsupervised learning, Evol. Intell. 15 (2022) http://dx.doi.org/10.1007/s12065-020-00518-1.
[278] Junwei Tang, Ruixuan Li, Yu Jiang, Xiwu Gu, Yuhua Li, Android malware obfuscation variants detection method based on multi-granularity opcode features, Future Gener. Comput. Syst. (ISSN: 0167-739X) 129 (2022) 141–151.
[279] L. Xu, D. Zhang, N. Jayasena, J. Cavazos, HADM: Hybrid analysis for detection of malware, in: Y. Bi, S. Kapoor, Bhatia R. (Eds.), Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016. IntelliSys 2016, in: Lecture Notes in Networks and Systems, vol. 16, Springer, Cham, 2018.
[280] B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane, Graph-based malware detection using dynamic analysis, J. Comput. Virol. 7 (4) (2011) 247–258.
[281] S.L. SD, C.D. J, Windows malware detector using convolutional neural network based on visualization images, IEEE Trans. Emerg. Top. Comput.
[282] X. Huang, L. Ma, W. Yang, et al., A method for windows malware detection based on deep learning, J. Signal Process. Syst. (2020).
[283] W. Aslam, M.M. Fraz, S.K. Rizvi, S. Saleem, Optimizing features for malware-benign clustering using windows portable executables, in: 2021 International Conference on Artificial Intelligence, ICAI, 2021, pp. 28–32.
[284] O. Sharma, A. Sharma, A. Kalia, Windows and IoT malware visualization and classification with deep CNN and Xception CNN using Markov images, J. Intell. Inf. Syst. (2022).
[285] S.K.J. Rizvi, W. Aslam, M. Shahzad, et al., PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable, Complex Intell. Syst. 8 (2022) 673–685.
[286] C. Petrov, Internet of things statistics from 2019 to justify the rise of IoT, 2019, https://techjury.net/stats-about/internet-of-things-statistics/. Accessed 25 Oct 2019.
[287] L. Columbus, IoT market predicted to double by 2021, reaching $520b, 2018, https://www.forbes.com/sites/louiscolumbus/2018/08/16/iot-market-predicted-to-double-by-2021-reaching-520b/#768bbd9d1f94. Accessed 13 Dec 2019.
[288] J. Sakhnini, H. Karimipour, A. Dehghantanha, R.M. Parizi, G. Srivastava, Security aspects of internet of things aided smart grids: a bibliometric survey, Internet Things (2019) 100111.
[289] M. Binti Mohamad Noor, W.H. Hassan, Current research on internet of things (IoT) security: a survey, Comput. Netw. 148 (2019) 283–294.
[290] K.D.T. Nguyen, T.M. Tuan, S.H. Le, A.P. Viet, M. Ogawa, N. Le Minh, Comparison of three deep learning-based approaches for IoT malware detection, in: Proceedings of 2018 10th International Conference on Knowledge and SystemsEngineering, KSE 2018, IEEE, 2018, pp. 382–388.
[291] H.S. Ham, H.H. Kim, M.S. Kim, M.J. Choi, Linear SVM-based android malware detection for reliable IoT services, J. Appl. Math. 2014 (2014) 594501.
[292] R. Kumar, X. Zhang, W. Wang, R.U. Khan, J. Kumar, A. Sharif, A multimodal malware detection technique for android IoT devices using various features, IEEE Access 7 (2019) 64411–64430.
[293] Z. Markel, M. Bilzor, Building a machine learning classifier for malware detection, in: WATeR 2014 - Proceedings of the 2014 2nd Workshop on Anti-Malware Testing Research, IEEE, 2015.
[294] R. Taheri, M. Shojafar, M. Alazab, R. Tafazolli, Fed-IIoT: A robust federated malware detection architecture in industrial IoT, IEEE Trans. Ind. Inform. 17 (12) (2021) 8442–8452.
[295] M. Panda, A.A.A. Mousa, A.E. Hassanien, Developing an efficient feature engineering and machine learning model for detecting IoT-botnet cyber attacks, IEEE Access 9 (2021) 91038–91052, http://dx.doi.org/10.1109/ACCESS.2021.3092054.
[296] S.A. Khowaja, P. Khuwaja, Q-learning and LSTM based deep active learning strategy for malware defence in industrial IoT applications, Multimed. Tools Appl. 80 (2021) 14637–14663.
[297] Regonda Nagaraju, Jupeth Toriano Pentang, Shokhjakhon Abdufattokhov, Ricardo Fernando CosioBorda, N. Mageswari, G. Uganya, Attack prevention in IoT through hybrid optimization mechanism and deep learning framework, Measurement: Sensors (ISSN: 2665-9174) 24 (2022) 100431.
[298] Rajasekhar Chaganti, Vinayakumar Ravi, Tuan D. Pham, Deep learning based cross architecture internet of things malware detection and classification, Comput. Secur. (ISSN: 0167-4048) 120 (2022) 102779.
[299] Santosh K. Smmarwar, Govind P. Gupta, Sanjay Kumar, Deep malware detection framework for IoT-based smart agriculture, Comput. Electr. Eng. (ISSN: 0045-7906) 104 (Part A) (2022) 108410.
[300] G.E. Hinton, Deep belief networks, Scholarpedia 4 (5) (2009) 5947.
[301] J. Hassannataj Joloudari, M. Haderbadi, A. Mashmool, M. Ghasemigol, S.S. Band, A. Mosavi, Early detection of the advanced persistent threat attack using performance analysis of deep learning, IEEE Access 8 (2020) 186125–186137.
[302] N. Mohamed, B. Belaton, SBI model for the detection of advanced persistent threat based on strange behavior of using credential dumping technique, IEEE Access 9 (2021) 42919–42932.
[303] Meaad Alrehaili, Adel Alshamrani, Ala Eshmawi, A hybrid deep learning approach for advanced persistent threat attack detection, in: The 5th International Conference on Future Networks & Distributed Systems (ICFNDS 2021), Association for Computing Machinery, New York, NY, USA, 2022, pp. 78–86.
[304] C. Do Xuan, M.H. Dao, A novel approach for APT attack detection based on combined deep learning model, Neural Comput. Appl. 33 (2021) 13251–13264.
[305] H. Li, J. Wu, H. Xu, G. Li, M. Guizani, Explainable intelligence-driven defence mechanism against advanced persistent threats: A joint edge game and AI approach, IEEE Trans. Dependable Secure Comput. 19 (2) (2022) 757–775.
[306] C. Do Xuan, D. Huong, A new approach for APT malware detection based on deep graph network for endpoint systems, Appl. Intell. 52 (2022) 14005–14024.
[307] S. Homayoun, et al., DRTHIS: deep ransomware threat hunting and intelligence system at the fog layer, Future Gener. Comput. Syst. 90 (2019) 94–104.
[308] M. Al-Hawawreh, E. Sitnikova, N4. Aboutorab, Asynchronous peer-to-peer federated capability-based targeted ransomware detection model for industrial IoT, IEEE Access 9 (2021) 148738–148755.
[309] X. Zhang, J. Wang, S. Zhu, Dual generative adversarial networks based unknown encryption ransomware attack detection, IEEE Access 10 (2022) 900–913.