IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2019, accepted December 22, 2019, date of publication January 3, 2020, date of current version January 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963724

A Comprehensive Review on Malware Detection Approaches

ÖMER ASLAN[1,2] AND REFIK SAMET[1], (Member, IEEE)
[1]Computer Engineering Department, Ankara University, 06830 Ankara, Turkey
[2]Computer Engineering Department, Siirt University, 56100 Siirt, Turkey

Corresponding author: Ömer Aslan (omer.aslan@siirt.edu.tr)

ABSTRACT According to the recent studies, malicious software (malware) is increasing at an alarming rate, and some malware can hide in the system by using different obfuscation techniques. In order to protect computer systems and the Internet from the malware, the malware needs to be detected before it affects a large number of systems. Recently, there have been made several studies on malware detection approaches. However, the detection of malware still remains problematic. Signature-based and heuristic-based detection approaches are fast and efficient to detect known malware, but especially signature-based detection approach has failed to detect unknown malware. On the other hand, behavior-based, model checking-based, and cloud-based approaches perform well for unknown and complicated malware; and deep learning-based, mobile devices-based, and IoT-based approaches also emerge to detect some portion of known and unknown malware. However, no approach can detect all malware in the wild. This shows that to build an effective method to detect malware is a very challenging task, and there is a huge gap for new studies and methods. This paper presents a detailed review on malware detection approaches and recent detection methods which use these approaches. Paper goal is to help researchers to have a general idea of the malware detection approaches, pros and cons of each detection approach, and methods that are used in these approaches.

INDEX TERMS Cyber security, malware classification, malware detection approaches, malware features.

I. INTRODUCTION

In recent years, almost every member of the society has been using the Internet for daily life. This is because it is almost impossible to do anything without the Internet including social interactions, online banking, health related transaction, and marketing. Since the Internet has been growing rapidly, criminals have started to commit crimes on the Internet rather than in real world. Criminals are generally using malicious software to launch cyber-attacks to the victim machines. Any software which intentionally executes malicious payloads on victim machines (computers, smart phones, computer networks, etc.) is considered as malware. There are different types of malware including virus, worm, Trojan horse, rootkit, and ransomware. Each malware type and family is designed to affect original victim machine in different ways such as damaging the targeted system, allowing remote code execution, stealing confidential data, etc. These days, the classification of malware is getting harder because some malware instances can present the characteristics of multiple classes at the same time.

In the early days, malware was written for simple purposes, thus, it was easier to detect. This kind of malware can be defined as traditional (simple) malware. However, these days, the malware which can run in kernel mode, and is more destructive and harder to detect than traditional malware can be defined as new generation malware (next-generation). This kind of malware can easily bypass protection software that is running in kernel mode such as firewalls, antivirus software, etc. Generally, traditional malware consists of one process and does not use complicated techniques to hide itself. On the other hand, new generation malware uses multiple different existing or new processes at the same time, and uses some obfuscated techniques to hide itself and become persistent in the system. New generation malware can launch more destructive attacks such as targeted and persistent which have never been seen before, and more than one type of malware is used during the attacks. The comparison of traditional versus new generation of malware can be seen in Table 1.

These days, the number, sophistication, and cost of malware inflicted on the world economy have been increasing

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

TABLE 1. Traditional versus new generation malware.

Comparison Parameter	Traditional	New Generation
Implementation level	simple coded	hard coded
State of behaviors	static	dynamic
Proliferation	each copy is similar	each copy is different
Through spreading	uses .exe extension	uses also different extensions
Permanence in the system	temporal	persistent
Interaction with processes	a few processes	multiple processes
Use concealment techniques	none	yes
Attack type	general	targeted
Defensive challenge	easy	difficult
Targeted devices	general computers	many different devices

incrementally. According to scientific and business reports, approximately 1 million malware files are created every day, and cybercrime will damage the world economy by approximately $6 trillion annually by 2021 [1]. Recent studies show that mobile malware is on the rise. According to the McAfee mobile threat report, there is a huge increase in backdoors, fake applications and banking Trojans for mobile devices [2]. Besides, the malware attacks related to the social media, healthcare industry, cloud computing, internet of things (IoT), and cryptocurrencies are also on the rise. According to cybersecurity ventures, ransomware malware will cost around $11.5 billion globally at the end of 2019 [1].

To protect legitimate users and companies from malware, malware need to be detected. Malware detection is the process of determining whether a given program has malicious intent or not. In early days, signature-based detection approach was used widely to detect malware. However, this approach has some limitations such as it cannot detect unknown and new generation malware. In process of time, researchers proposed new approaches including behavioral-, heuristic-, and model checking-based detection. With these approaches, datamining and machine learning (ML) algorithms are also started to be used widely in malware detection. Recently, new approaches have been proposed such as deep learning-, cloud-, mobile devices-, and IoT-based detection. For known and some of unknown malware, heuristic detection approach performs well. On the other hand, for unknown and complicated malware; behavior-, model checking-, and cloud-based approaches perform better. Deep learning-, mobile devices-, and IoT-based approaches also emerge to detect some portion of known and unknown malware. It has not been proved exactly that one detection approach is more effective than the others. This is because each method has its own advantages and disadvantages, and in different situation one method can detect better than another. Even though several new methods have been proposed by using different malware detection approaches, no method could detect all new generation and sophisticated malware. This shows that building an effective method to detect malware is a very challenging task, and there is a huge demand for new studies and methods.

This paper presents the literature review in order to investigate the current situation of malware detection approaches. The paper makes the following contributions:

- Explains new technological trends for malware creation and new approaches to detect malware.
- Investigates the probability of detecting malware.
- Presents a summary of the current studies on malware detection.
- Explains important approaches and methods for malware detection.
- Discusses current challenges and proposes new assumptions for malware detection approaches.
- Provides a systematic overview of malware detection approaches and methods for further studies.

The rest of the paper is organized as follows: Section II demonstrates problem definition. Malware detection techniques and algorithms are explained in section III, and malware detection approaches are explained in section IV. Evaluation on malware detection approaches are presented in section V. Finally, the conclusion and future works are given in section VI.

II. PROBLEM DEFINITION

This section investigates the problem of malware and possibility of detection. It can be said that it is impossible to design an algorithm which can detect all malware. This is because the problem of detecting the malware has shown *NP-complete* in many studies. This is important because before starting to build an effective detection system, it is a good practice and experience for researcher to understand the scope, limitation, and possibility of malware detector. The possibility of detection malware is remaining problematic because theoretically it is a hard problem, and practically malware creators using complicated techniques such as obfuscation to make detecting process very challenging.

A. DIFFICULTY OF PROBLEM IN THEORY

Since the first malware that appeared in the wild was a virus, most of the studies had been done theoretically were based on the detection of virus. According to early studies, the detection of virus is impossible [3]–[5] and *NP-complete* [6]–[9]. According to F. Cohen, the detection of computer virus is an undecidable because detection process itself contains a contradiction [3], [5], [6]. If the detection problem is seen as a decision-making problem, D (decision-maker) will decide whether P is a virus or not. According to Cohen, it cannot be decided whether P is a virus because if P is a virus, it will be marked by D as a virus and will not be able to make changes to other programs, as it will not act as a virus. If D decision maker did not identify P as a virus, P will interact with other programs to spread and become infected.

This decision process involves contradiction, and therefore it is not possible to identify *P* as a virus. According to M. Chess and R. White, there is no program that detects all viruses without false positives (*FPs*) because viruses are polymorphic and can be exist in different forms [5]. According to M. Adleman detecting a virus is quite intractable and almost impossible [7]. This is because according to Gödel numberings of the partial recursive functions, it is not possible to create detecting mechanism. To reliably identifying a bounded-length mutating virus is *NP-complete* explained in [8]. According to the author, virus detector for certain virus strain can be used to solve the satisfiability problem. Since satisfiability problem is known to be *NP-complete*, so the detection of the malware is *NP-complete*. Zuo *et al*. claim that there exist computer viruses whose detecting procedures have sufficiently large time complexity, and there are undecidable viruses which have no minimal detecting procedure [9].

B. DIFFICULTY OF PROBLEM IN PRACTICE

The new generation malware uses the common obfuscation techniques such as encryption, oligomorphic, polymorphic, metamorphic, stealth, and packing methods to make detection process more difficult. This kind of malware can easily bypass protection software that is running in kernel mode such as firewalls, antivirus software, etc. and some malware instances can also present the characteristics of multiple classes at the same time. This makes practically almost impossible to detect all malware with single detection approach. The definition of common obfuscation techniques explain as follows:

- **Encryption:** In encryption, malware uses encryption to hide malicious code block in its entire code [10]. Hence, malware becomes invisible in the host.
- **Oligomorphic:** In oligomorphic method, a different key is used when encrypting and decrypting malware payload [11]. Thus, it is more difficult to detect malware which uses oligomorphic method than encryption.
- **Polymorphic**: In polymorphic method, malware uses a different key to encrypt and decrypt [12] likewise the key used in oligomorphic method. However, the encrypted payload portion contains several copies of the decoder and can be encrypted in layered [13]. Thus, it is more difficult to detect polymorphic malware when compared to oligomorphic malware.
- **Metamorphic:** Metamorphic method does not use encryption. Instead, it uses dynamic code hiding which the opcode changes on each iteration when the malicious process is executed [14]. It is very difficult to detect such malware because each new copy has a completely different signature.
- **Stealth:** Stealth method also called code protection, implements a number of counter techniques to prevent it from being analyzed correctly [11]. For instance, it can make changes on the system and keep it hidden from detection systems.
- **Packaging**: Packaging is an obfuscation technique to compress malware to prevent detection, or hiding the actual code by using encryption [15], [16]. Due to this technique, malware can easily bypass firewall and antivirus software. Packaged malware need to be unpacked before being analyzed. The packers can be divided into 4 different groups include compressors, crypters, protectors, and bundlers.

In this section, the limitations of malware detecting systems have been summarized. Current studies demonstrate that it is almost impossible to write an algorithm to detect all malware. This is because the computational complexity of malware is not clear, and the detection of malware problem is proved to be *NP-complete*. Besides, the use of new techniques (obfuscation and packing) during malware creation also makes detection process more challenging.

III. MALWARE DETECTION TECHNIQUES AND ALGORITHMS

In recent years, datamining and ML algorithms have been used extensively for malware detection. Malware detection is the process of investigating the content of the program and deciding whether the analyzed program malware or benign. The malware detection process includes 3 stages: Malware analysis, feature extraction, and classification.

A. MALWARE ANALYSIS

In order to understand the content and behaviors of malware, it needs to be analyzed. Malware analysis is the process of determining the functionality of malware and answers to following questions [17], [18]. How malware works, which machines and programs are affected, which data is being damaged and stolen, etc. There are mainly two techniques to analyze malware: static and dynamic [17]. Static analysis examines the malware without running the actual code [19]. On the other hand, dynamic analysis examines the malware behaviors while running its code. Malware analysis starts with basic static analysis and finishes with advanced dynamic analysis. The malware is analyzed by using reverse engineering [20] and some other malware analysis tools to represent the malware in different format. Reverse engineering process can be seen in Figure 1.

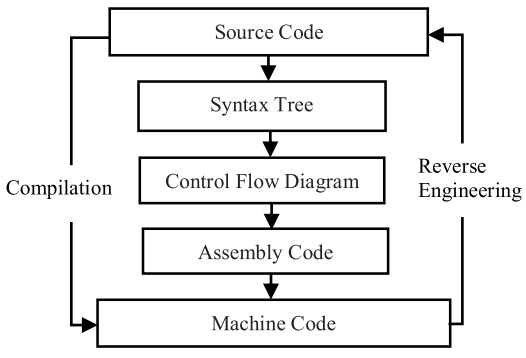

FIGURE 1. A flow chart of reverse engineering process.

B. MALWARE FEATURE EXTRACTION

Malware features are extracted by using data mining techniques. Data mining is the process of extracting new meaningful information from large datasets or databases which has been unknown before this process. In recent years, by using datamining new models and datasets have been created [21]. There are different models such as n-gram, and graph model to create malware dataset and features.

1) THE n-gram MODEL

The n-gram is a feature extraction technique which has been used widely in many areas as well as malware detection. The n-gram can use both static and dynamic attributes to create features. To create features from behaviors, n-gram group the system calls or application programing interfaces (APIs) in a consecutive order by specified n ($n = 2, n = 3, n = 4, n = 6$, etc.) values. Although the n-gram model has been used widely in malware detection, it has some drawbacks when determining features. This is because every sequential static and dynamic attributes are not related to one another. This makes classification and clustering more challenging for later processes. Besides, n-gram generates enormous feature space which increases the analysis time and decreases the model performance. For these reasons, there is a huge demand to find out new models to achieve better performance than n-gram.

2) GRAPH-BASED MODEL

The graph-based model is one of the commonly used techniques to generate features as well. System calls made in this method are converted into graph $G\ (V, E)$ such that V represents nodes which identify system calls and the E represents edges which identify the relationship among the system calls. Since the size of the graph increases over time, sub-diagrams can be used to describe the graph. The sub-diagram is defined in many studies as *NP-Complete*. This means that it requires a lot of time to define each sub-diagram. After the whole diagram is expressed with fewer nodes and edges, the programs are identified as malicious or benign.

3) MALWARE DATASET

As in other research areas, there are not many datasets published previously which are accepted and widely used for malware detection. In addition, most of the existing datasets are not accessible for research, and in most cases the datasets accessed are not in the appropriate formats for data mining processes and ML algorithms. The datasets used in malware analysis can be listed as follows: NSL-KDD, Drebin, Microsoft malware classification challenge, ClaMP (classification of Malware with PE headers), AAGM, and EMBER dataset.

- **NSL-KDD dataset (2009):** It is an updated version of the KDD'99 dataset which consists of approximately 125,000 records and 41 features [22]. It shows the network related attacks which are used for intrusion detection system.
- **Drebin dataset (2014):** This dataset is created for smart phones to examine the effectiveness of the existing anti-virus software [23]. It consists of 5560 malware across 20 families and 123,453 benign samples.
- **Microsoft malware classification challenge dataset (2015):** It has been published by Microsoft and consists of 20,000 malware [24]. Malware has been analyzed using the IDA packet disassembler and the output should be processed using data mining prior to ML.
- **ClaMP (Classification of Malware with PE headers) dataset (2016):** It consists of 5184 records and has 55 properties [25]. The dataset uses API arrays, contains examples of malicious and benign software with their features.
- **AAGM dataset (2017):** It is a network-based dataset for android malware [26]. It consists of 400 malware and 1500 benign samples from 12 families [26].
- **EMBER dataset (2018):** It consists of 1 million records and holds malware and benign features [27].

These datasets can be used for researches who want to get some experience before proposing a new malware detection approach.

C. MALWARE CLASSIFICATION

Machine learning (ML) is a set of algorithm that correctly estimates the outcomes of the applications without being explicitly programmed. The purpose of the ML is to convert the input data into acceptable value intervals by using statistical analysis. By using ML, many operations can be performed on related data such as classification, regression and clustering. ML algorithms have been used in malware detection for many years [28]. Well-known ML algorithms are Bayesian network (BN), naive Bayes (NB), C4.5 decision tree variant (J48), logistic model trees (LMT), random forest tree (RF), k-nearest neighbor (KNN), multilayer perceptron (MLP), simple logistic regression (SLR), support vector machine (SVM), and sequential minimal optimization (SMO). These algorithms are used especially in behavior-based detection and some of other detection approaches. Although each algorithm has its own advantages and disadvantages, it cannot be concluded that one algorithm is more efficient than another. However, an algorithm can perform better than other algorithms in terms of the distribution of the data, number of features, and dependencies between properties.

IV. MALWARE DETECTION APPROACHES

In recent years, there has been a rapid increase in the number of academic studies on malware detection. In the early days, signature-based detection method was widely used. This method works fast and efficiently against the known malware, but does not perform well against the zero-day malware [21], [29]. In the process of time, researchers have started to use techniques such as behavior-, heuristic-, and

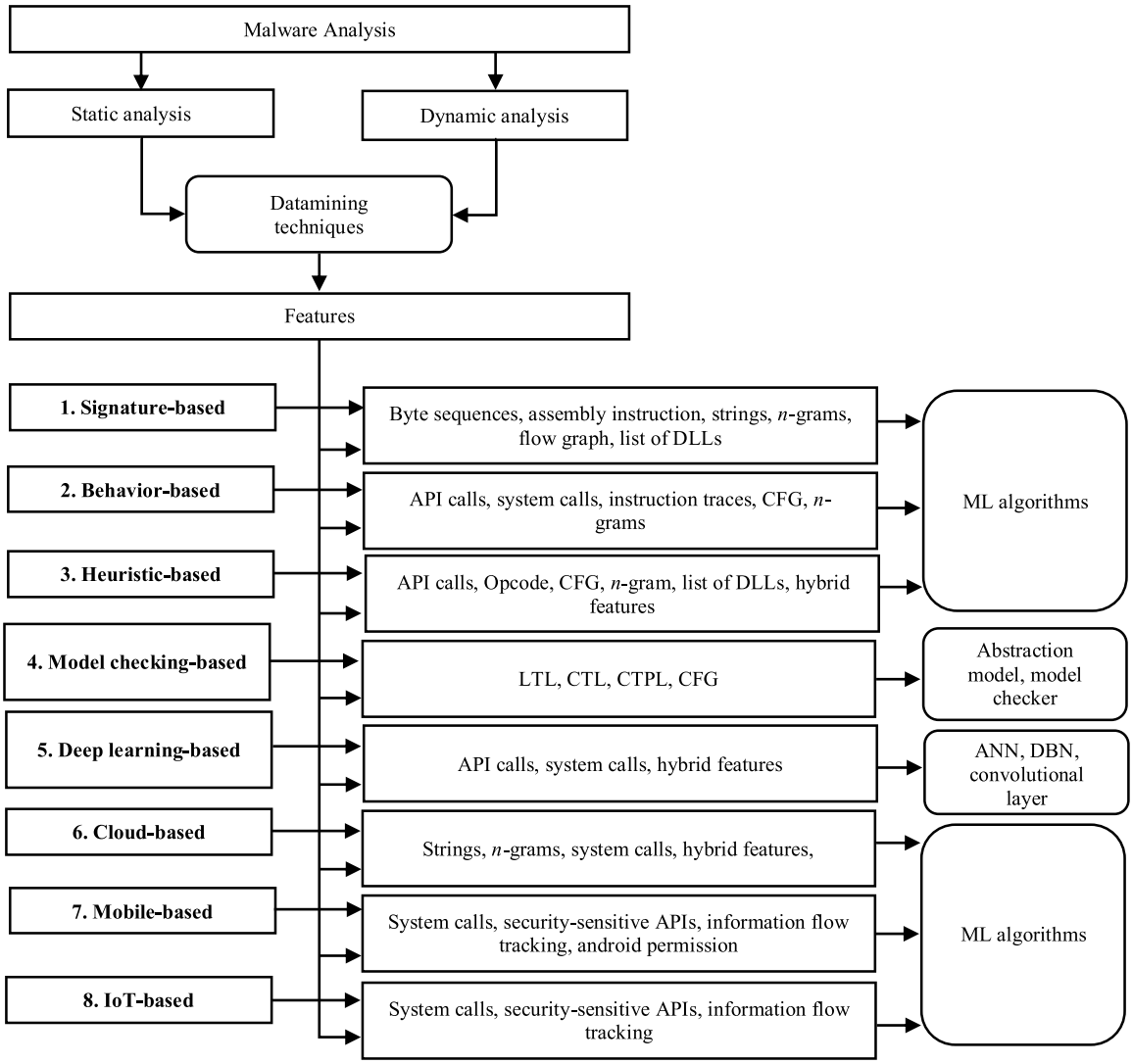

FIGURE 2. A flow chart of malware detection approaches and features.

model checking-based detection; and new techniques such as deep learning-, cloud-, mobile devices-, and IoT-based detection. Overview of malware detection approaches, features, and used techniques can be seen in Figure 2.

In each approach, feature extracting method is different one from another. It could not have been proven one detection method works better than another because each method has its own advantages and disadvantages. By using behavior-, heuristic-, and model checking-based detection approaches; huge number of malware can be detected with a few behaviors and specifications. In addition, new malware can be detected by using these approaches as well. However, they cannot detect all malware. There is great necessity to find the method which effectively detects more complex and unknown malware. Before explaining each detection approach in details, some well-known methods in each detection approach and their related works are summarized in Table 2. Then, detailed literature review is presented, and the pros and cons of each study are explained.

A. SIGNATURE-BASED MALWARE DETECTION

Signature is a malware feature which encapsulates the program structure and identifies each malware uniquely. Signature- based detection approach is widely used within commercial antivirus. This approach is fast and efficient to detect known malware, but insufficient to detect unknown malware. In addition, malware belonging to the same family can easily escape the signature-based detection by using obfuscation techniques. General view of signature-based detection schema can be seen in Figure 3.

1) SIGNATURE GENERATION PROCESS

During the signature generation, first features are extracted from executables (Figure 3). Then, signature generation engine generates a signatures and stores them into signature database. When sample program needs to be marked as malware or benign, signature of the related sample is extracted as the same way before and compared with signatures on the database. Based on the comparison, sample program is

TABLE 2. Summary of related works on malware detection approaches.

Paper	Features Representation/ Representation method	Goal/Success	Year
Signature-based approach			
Schultz *et al.*[30]	Printable strings, DLL and API system calls	Detecting new malware	2001
Karnik *et al.* [31]	Assembly instructions, cosine similarity	Identify different forms of malware	2007
Cha *et al.* [32]	Vector transformations of file hashes	Capture multiple malware with one signature	2011
Baldangombo *et al.* [33]	File header information, DLL and API names	Detection of known malware with a success rate of 99%	2013
Aashima and Bajaj [34]	Hybrid based text mining technique to extract instructions	Can predict types of malware with low overhead	2016
Behavior-based approach			
Moser *et al.* [35]	Conditional branching instructions	Identify multiple execution paths	2007
Wagener et al. [36]	System calls, Hellinger distance, phylogenetic tree	Identify new malware and different forms of malware	2008
Park *et al.* [37]	Creating system call diagrams	Identify different forms of malware	2013
Das *et al.* [38]	System call frequencies, *n*-gram	Identify new malware and different forms of malware	2016
Monire *et al.* [39]	Executive history XML file, nonsparse matrix	Identify traditional and new malware	2016
Heuristic-based approach			
Zhang *et al.* [40]	The *n*-gram byte sequences	Identify traditional and different forms of malware	2007
Griffin *et al.* [41]	Byte sequences	Identify different forms of malware	2009
Anderson *et al.* [42]	The *n*-gram, Markov chain and diagrams	Identify different forms of malware with 96.41%	2011
Islam *et al.* [43]	Printable strings, API method frequencies	Identify new and different forms of malware with 97%	2013
Naval *et al.* [44]	Diagram of system calls and relations	Detect code insertion attacks	2015
Model checking-based approach			
Singh and Lakhotia [45]	LTL (linear logic) formulas	Identify new and different forms of malware	2003
Kinder *et al.* [46]	Calculation tree validation logic	Identify different forms of malware	2005
Beaucams and Marion [47]	LTL formulas	Identify new and different forms of malware	2009
Song *et al.* [48]	Determining the stack behavior of the program using pushdown systems	Identify new malware	2012
Cimitile *et al.* [49]	Elective Mu-Calculus Logic and phylogenetic trees	Identify new and different forms of malware	2017
Deep learning-based approach			
Saxe and Berlin [50]	Contextual byte features, PE import features, Score calibration model	95% *DR* with 0.1% *FP*	2015
Huang and W. Stokes [51]	Raw bytes, gradient based attack	Show the inefficiency of deep learning	2016
Dali *et al.* [52]	Hybrid features, DeepFlow technique, DBN model	High detection F1 score of 95.05%, which outperformed traditional ML based learning	2017
Yanfang *et al.* [53]	Greedy layer-wise training operations	Improved overall performance when compared with traditional learning techniques	2018
Cloud-based approach			
Martignoni *et al.* [54]	API traces, system calls, dynamic features	To get multiple execution traces of the same malware	2009
Hao *et al.* [55]	Multiple hash functions, reversible sketch structure, bucket cross-filtering method	Outperform other existing systems with less time and communication consumption	2015
Xiao *et al.* [56]	Application traces, Dyna architecture, Q-learning strategy	To increase detection accuracy, reduce the detection delay	2017
Mobile devices-based approach			
Takamasa *et al.* [57]	System calls	To detect unknown malware	2011
Shabtai *et al.*[58]	APIs, permissions	To detect new malware	2012
Narayanan *et al.* [59]	Security-sensitive behaviors, context information from dependence	Outperforms two state-of-the-art techniques on a benchmark dataset achieving 99.23% f-measure	2017
IoT-based approach			
Amin *et al.* [60]	Monitors the energy consumption patterns of different processes	Outperformed KNN, neural networks, SVM and RF	2018

marked as malware or benign. There are many different techniques to create a signature such as string scanning, top-and-tail scanning, entry point scanning, and integrity checking.

- **String Scanning:** Compares the byte sequence in the analyzed file with the byte sequences previously saved in the database. Byte signatures have been

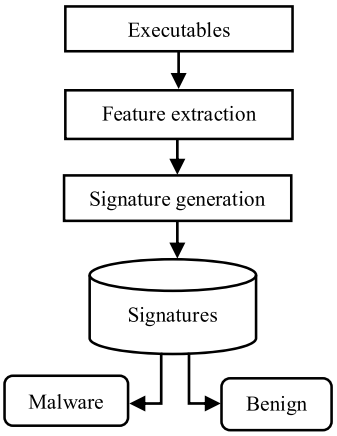

FIGURE 3. Signature-based malware detection schema.

TABLE 3. Example ClamAV byte signature.

90FF1683EE0483EB0175F6

TABLE 4. "90FF1683EE0483EB0175F6" Assembly byte sequence.

```
Start: 0x401A2E length: 0xC
90 nop
FF 16 call dword ptr [esi]
83 EE 04 sub esi, 4
83 EB 01 sub ebx, 1
75 F6 jnz short loc_401A30
```

TABLE 5. Display of byte signatures in Yara format.

```
Rules
{
   strings:
   signature = {66 90 FF 16 83 EE 04 83 EB 01 75 F6}
   condition:
   signature
}
```

used extensively by antivirus scanners for many years. They are often used to detect malware which belongs to the same family with different signatures [61]. Table 3 shows the ClamAV byte signature [62].

"90FF1683EE0483EB0175F6" is the hexadecimal representation of the relevant code section and it is shown in assembly language as in Table 4. The same byte signature is shown in Yara format in Table 5 [62].

- **Top-and-Tail Scanning:** Instead of the whole file, only the top and end points of the file are taken and certain signatures are created [11]. It is a very convenient signature method to detect viruses that attach themselves to the beginning and end of files.
- **Entry Point Scanning:** The entry point of a file indicates where the first run starts when that file starts to run. Malware usually changes the entry point of a program, so that malicious code being executed before the actual code [11]. Therefore, certain malware can be detected by extracting the signature from the sequences at the program entry points.
- **Integrity Checking (Hash Signatures):** Integrity check generates a cryptographic checksum such as MD5 and SHA-256 for each file in a system at regular intervals, and it is used to identify possible changes that may be caused by malware.

Different signature generation techniques have been summarized. Even though these techniques are quite fast and efficient to generate a signature, they are not resistant to malware obfuscating techniques. For example, malware can easily change the strings and program entry point in its instruction set. By this, generated signature may mislead the detecting schema. To extract more powerful and general signatures, different techniques and features can be used. Detailed review of signature-based malware detection approach and its methods are summarized as follows:

2) RELATED WORKS FOR SIGNATURE-BASED DETECTION

F. Zolkipli and Jantan proposed a new malware detection framework which is based on s-based detection, genetic algorithm (GA), and signature generator [63]. Even though the authors claim that this method can detect unknown malware, there is not enough information given in the paper for proposed framework such as test results, number of malware analyzed, and comparison of proposed method with other existing studies. Tang *et al.* proposed a bioinformatics technique to generate accurate exploit-based signatures for polymorphic worms [64]. The technique involves three steps: multiple sequence alignment to reward consecutive substring extractions, noise elimination to remove noise effects, and signature transformation to make the simplified regular expression signature compatible with current IDSs.

The authors claim that suggested schema is noise-tolerant, and more accurate and precise than those generated by some other exploit-based signature generation schemas. This is because it extracts more polymorphic worm characters like one-byte invariants and distance restrictions between invariant bytes. However, proposed schema is limited to polymorphic worm and cannot be generalized to other malware types.

Borojerdi and Abadi proposed a MalHunter detection system which is a new method based on sequence clustering and alignment [65]. It generates signatures automatically based on malware behaviors for polymorphic malware. The novel method works as follows: First, from different malware samples, behavior sequences are generated. Then, based on similar behavioral sequences, different groups are generated and stored in the database. To detect malware sample, behavior sequences are gathered and compared with sequences which have been generated earlier and stored in the database. Based on the comparison, the sample is marked as malware or benign. The test results showed that by choosing the cluster radius 0.4 and similarity threshold 0.05, they

achieved detection rate of 90.83% with a *FPR* of 0.80%. The authors claim that proposed schema is resistant to obfuscation techniques, and it can be used for the generic detection of all types of polymorphic malware rather than being limited to a specific malware type. The authors also claim that the suggested system outperformed state-of-the-art signature generation methods including Tang *et al.* [64], Newsome *et al.* [66], and Perdisci *et al.* [67] previously reported in the literature. The proposed method is limited to polymorphic malware and it has been tested on only hundreds of malware which is not enough to determine the performance of proposed method.

Automatic string signatures generation (Hancock) is explained in [41]. According to the paper, proposed schema can automatically generate high-quality string signatures with minimal *FPs* and maximal malware coverage. The proposed method uses a set of library code identification techniques, and diversity-based heuristics techniques to ensure the contexts in which a signature is embedded in containing malware files similar to one another [41]. Although the authors claim that Hancock can automatically generate string signatures with a *FPR* below 0.1%, this *FPR* will be changed based on benign samples that are analyzed. This is because benign set is constantly growing, and getting some satisfying result on some part of benign cannot be generalized to whole set. Thus, these problems need to be solved for further studies. Santos *et al.* proposed *n*-grams-based file signatures to detect malware [68]. First, for known files *n*-grams are extracted for every file and used as a file signature. Then, for any unknown instance, *n*-grams are generated, and by using measuring function and k-nearest neighbor algorithm [69], file is marked malware or benign. Paper demonstrated that *n*-grams-based signatures can detect unknown malware to a certain degree.

Efficient signature based malware detection on mobile devices is proposed in [70]. First, signature has been created. Second, hash table has been used to store the hash values of signatures to increase scanning speed. Finally, signature matching algorithm is used to compare the signatures. To eliminate the mismatches, the probability of occurrence of signature bytes in non-malicious content has been used. According to the authors, the results have shown that suggested schema performs well when compared to the Clam-AV scanner, and provides huge memory savings while maintaining fast scanning speed. The proposed system was only compared with Clam-AV scanner, which is not enough for overall evaluation. Zheng *et al.* presented the DroidAnalytics, an Android malware analytic system which can automatically collect malware, generate signatures for applications, identify malicious code segment, and associate the malware under study with various malware in the database [71]. In proposed system three-level signature generation schema has been used to identify each application. The authors assert that proposed signature methodology provides significant advantages over traditional cryptographic hash like MD5-based signature, and resistant to packing and mutations. The proposed system has not been compared with other studies in the literature, and the evaluation metrics are not very high and are not explained in detail.

3) EVALUATION OF SIGNATURE-BASED DETECTION

In the literature review, signature-based detection methods have been summarized. Signature-based detection schema has been used for antivirus vendors for many years and it is quite fast and effective to detect known malware. This approach is generally used to detect malware which belongs to the same family. However, it fails to detect new generation malware which uses obfuscation and polymorphic techniques. Besides, it is prone to many *FPs* and extracting signature takes a lot of man-power.

Although previous signature-based methods have achieved some success, they are not enough to detect new generation malware. To build an effective signature, the following key points are taken into consideration:

- Signature should be as short as possible and can represent many malware with single signature,
- Effective automatic signature generation mechanism must be built,
- During the signature generation, datamining and ML techniques need to be used more,
- Signature should be resistant to packing and obfuscation techniques.

B. BEHAVIOR-BASED MALWARE DETECTION

Behavior-based malware detection approach observes the program behaviors with monitoring tools and determines whether the program is malware or benign. Although the program codes are being changed, the behavior of the program will be similar; thus, majority of new malware can be detected with this method [29]. On the other hand, some malware binaries do not run properly under protected environment (virtual machine, sandbox environment). Hence, malware samples are may be incorrectly marked as benign.

1) BEHAVIOR DETECTION PROCESS

When establishing a behavior-based detection system, behaviors are obtained by using one the following procedure:

- Automatic analysis by using sandbox [18];
- Monitoring of system calls [36], [38];
- Monitoring of file changes [18];
- Comparison of registry snapshots [29];
- Monitoring network activities [17];
- Process monitoring [18].

In behavior-based detection, first, behaviors are determined by using one of the technique used above and the dataset is created by subtracting the features using datamining. Then, specific features from the dataset are obtained and classification done by using ML algorithms. General view of behavior-based schema can be seen in Figure 4.

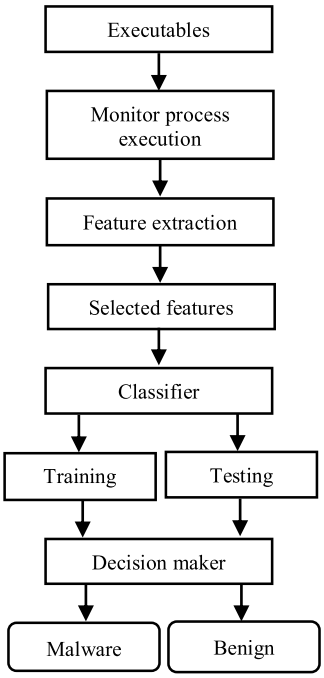

FIGURE 4. Behavior-based malware detection schema.

2) RELATED WORKS FOR BEHAVIOR-BASED DETECTION

The program similarities using system calls were described in [36]. Wagener *et al*. proposed a flexible and automated technique to extract malware behaviors from the system calls. The alignment technique has been used to identify similarities, and Hellinger distance has been calculated to compute associated distances. According to the paper, obfuscated malware variants that show similar behaviors can be detected. The authors assert that the classification process can be improved using a phylogenetic tree that represents the common functionalities of malware. The missing aspects of the article can be address as the following:

- Lack of knowledge about the malware dataset is shown,
- Statistical evaluation of performance is not provided,
- Comparison of proposed method against other methods are not given.

Besides, it is not clear how phylogenetic tree can improve the performance.

The behavior-based detection approach is proposed by Fukushima *et al.* in [72]. The proposed method can detect both unknown and encrypted malware on Windows OS. The proposed framework checks not only specific behaviors that malware performs, but also normal behaviors that malware usually does not perform. According to authors, *DR* was approximately 60% to 67% without any *FP*. The *DR* is very low, to increase the *DR,* more malicious behaviors can be identified, and to prove the effectiveness of new method, test set will be extended. Semantics-aware malware detection is proposed in [73]. The authors are determined that certain malicious behaviors such as a decryption loop in a polymorphic virus appears in all variants of a certain malware. According to the authors, experimental evaluation demonstrated that algorithm can detect all variants of certain malware with no*FPs*, and is resilient to obfuscation transformations. However, the algorithm has some limitation for obfuscation transformations. For instance, it cannot handle instruction replacement very well, and fails to detect malware which uses this technique. Handling instruction replacement problem and different ordering of memory updates can improve the performance.

The behavior detection methods which limit the number of features are represented in [74], [75], [38]. Lanzi *et al.* [74] proposed a system-centric behavior model. In proposed model, the interaction of the malware programs with system resources (directory, file, registry, etc.) is different from the benign. The behavior sequences of the program to be marked were compared with the behavior sequences of the two groups (malware, benign). The authors claim that the proposed system detected a significant fraction of malware with a few *FP*. The proposed method could not detect all malicious activities such as malware which does not attempt to hide its presence or to gain control of the OS, and which uses only computer network for transmission. To include network-related policies, and rules for malware programs that ignore other applications and the OS can improve the performance. M. Chandramohan *et al.* suggested Bounded Feature Space Behavior Modeling (BOFM) which limits the number of features to detect malware [75]. In this model, system calls were transformed into high-level behaviors and features were created using the behaviors. The feature vector was created and ML algorithms were applied to the feature vector to determine whether the given program is malware or benign. BOFM is fixed dimension which means it does not grow in proportion with the number of malware samples. This makes BOFM efficient and scalable in practice. Also, by using BOFM a better detection accuracy, lower computation times, and memory usage were obtained. This method ignored the frequency of system calls. Executing the same system call repeatedly can cause DoS attacks. Considering the frequency of system calls can improve *DR* and accuracy. A hardware-enhanced architecture which uses a processor and field-programmable gate array (FPGA) is proposed in [38]. The authors represented a frequency-centralized model (FCM) to extract the system calls and construct the features from the behaviors. Features obtained from the benign and malware samples were used for training the ML classifier to detect the malware. The paper claims that the suggested system achieved a high classification accuracy, fast *DR*, low power consumption, and can detect new malware samples. Besides, proposed method supports early prediction which can detect malware while malware is still running. However, malware can perform various behaviors, and there is no uniform policy to specify number of behaviors and features to be extracted before triggering the early prediction. Furthermore, the proposed method performance has only been compared with BOFM and *n*-gram which is not enough to determine the efficiency of the proposed algorithm.

Liu *et al.* used MapReduce to group malware behaviors and detect malware [76]. According to the authors, most of the studies done so far were process-oriented, and determined a process as a malware only by its invoking system calls. However, now most of the malware, which is defined as complex malware, consists of several processes and is transmitted to the system by driver or by DLL [77]. In such cases, malware performs actions on victim machine by using more than one process instead of its own processes. When only one process is analyzed, malware can be marked as benign. The paper emphasized persistent behaviors by using Auto-Start Extensibility Points (ASEP), and based on these behaviors it differentiated malware from benign. The experimental results showed that the DR improved on previous research by 28%. However, there are some limitations of proposed method. The limitations of this method can be address as follows:

- Some malware binaries do not require persistent behavior ASEP,
- Persistent malware behaviors can be completed without using system calls,
- The cost of data transmission has not been measured.

Besides, the proposed method results have not been compared with other studies in the literature. Eliminating above limitations can improve the method performance.

A supervised ML model is proposed in [78]. The model applied a kernel base SVM that used weighting measures, which calculates the frequency of each library call to detect Mac OS X malware. The *DR* was calculated as 91% with 3.9% *FP* rate. Test results indicated that increasing sample size increased the detection accuracy, but decreased the *FPR*. Combining static and dynamic features, using other techniques such as fuzzy classification and deep learning can increase the performance.

A graph-based detection schema was defined in [79], [37]. Kolbitsch *et al*. [79] proposed a graph-based detection, in which the system calls are converted into a behavior graph, where nodes represented system calls and edges indicated transitions among system calls that showed the data dependency. The program graph to be marked is extracted and compared with the existing graph to determine whether the given program is malware. Even though the proposed model performed well for the known malware, it has difficulties in detecting unknown malware. A graph-based method which specifies the common behaviors of malware and benign samples is represented in [37]. In proposed system, kernel objects were determined by system calls and behaviors were determined according to these objects. According to the authors, the proposed method is scalable and can detect unknown malware with high *DR*, and with low *FP* rates. In addition, the proposed model is highly scalable regardless of new instances added and robust against system call attacks. However, the proposed method can observe only partial behavior of an executable. To explore more possible execution paths would improve the accuracy of this method.

Graph-based malware detection using dynamic analysis is proposed in [42]. The proposed schema works on graphs which are constructed from dynamically collected instruction traces of the target executable. Markov chains have been used in which the vertices are the instructions and the transition probabilities are estimated by the data contained in the trace. They constructed similarity matrix which is combination of graph kernels between the instruction trace graphs. They performed classification by using SVM on similarity matrix. The results showed that there is a significant improvement over signature-based and other machine learning-based detection techniques. For the test case, modified version of Ether framework has been used. There are some limitations of Ether system including:

- Ether is not completely invisible which means that some intelligent malware can detect it and does not show their real behaviors,
- Ethernet card can be emulated by the underlying Xen system and string settings can be changed by malware,
- Ether is quite slow for malware analysis.

Using different framework can increase the performance.

Mojtaba and Hashemi proposed a graph mining method for detecting unknown malware binaries [80]. First, the paper extracted control flow graph (CFG) from programs and combined it with extracted API calls to have more information about executable files. This new representation model was called API-CFG. Then, the CFGs were converted to a set of feature vectors. Finally, the classification was performed by ML algorithms. According to the authors, the proposed method classified unseen benign and malicious code with high accuracy, and outperformed *n*-grams based detection method. However, the paper did not evaluate the performance for obfuscated malware, and also did not compare the results with known methods. To compare performance with other graph mining approaches may generate more trustworthy results.

3) EVALUATION OF BEHAVIOR-BASED DETECTION

In literature review, behavior-based detection approach and related methods have been summarized. Detection schema based on behaviors consists of 3 steps:

- Determine behaviors (datamining can be used),
- Extract features from behaviors (datamining is used),
- Apply classification (machine learning is used).

Data mining techniques such as *n*-gram, *n*-tuple, bag, graph model, etc. have been used to determine the features from behaviors; Hellinger distance, cosine coefficient, chi-square, etc. (probability and statistical method) distance algorithms are used to specify similarities among features. The difficulties in defining a behavior, the large number of extracted features (when using *n*-grams, etc.), and the difficulties in identifying the similarities and differences among the extracted properties have prevented the creation of an effective detection system. Besides, some malware does not run properly within the virtual machines/sandboxes, and

advanced code obfuscating techniques prevent malware from being analyzed correctly. The use of new methods and techniques along with the use of ML and data mining algorithms in malware detection has begun to play a major role when generating features meaningfully. There is huge demand for more scientific studies to cover shortcomings of existing methods. This study has summarized the existing researches and makes suggestions to fill the gap.

C. HEURISTIC-BASED MALWARE DETECTION

In recent years, heuristic based detection approach has been used frequently [81]. It is a complex detection method which uses experiences and different techniques such as rules and ML techniques [10]. Although it has a high accuracy rate to detect zero-day malware to a certain degree, it cannot detect complicated malware. Heuristic-based detection schema can be seen in Figure 5.

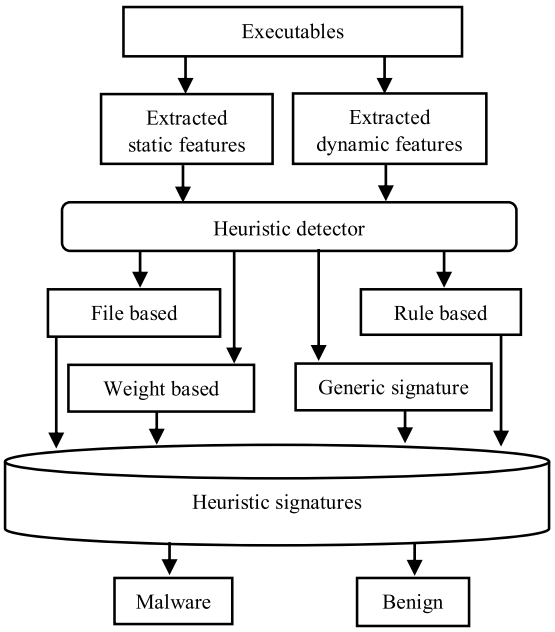

FIGURE 5. Heuristic-based malware detection schema.

1) RELATED WORKS FOR HEURISTIC-BASED DETECTION

Arnold and Tesauro proposed an automatically generated Win32 heuristic virus detection in [82]. They automatically construct multiple neural network classifiers which can detect unknown Win32 viruses. Generally, heuristic schema has high *FP* rate, but the authors claim that by combining the individual classifier outputs using a voting procedure, the risk of *FP* is reduced to an arbitrarily low level. The study is limited to Win32 virus, and can be extended to other malware. More malware needs to be examined for this method. Expert-designed heuristic features can improve the performance. Yanfang *et al*. proposed post-processing techniques of associative classification for malware detection [83]. The proposed system greatly reduced the number of generated rules by using rule pruning, rule ranking, and rule selection. This way, the technique did not need to deal with a large database of rules, which also accelerates the detection time and accuracy rate. According to the paper, the proposed system outperformed popular antivirus software tools such as McAfee, VirusScan and Norton AntiVirus; and outperformed data-mining-based detection systems including naive Bayes, support vector machine (SVM), and decision tree techniques. To collect more API calls which can provide more information about malware and identify complex relationships among the API calls may improve the performance.

Since traditional signature-based anti-virus systems fail to detect polymorphic, metamorphic, and previously unknown malicious executables; heuristic-based malware detection is explained in [84], [85]. Yanfang *et al*. proposed intelligent malware detection system (IMDS) [84]. The IMDS used objective-oriented association (OOA) mining that works based on windows API calls. The method consists of 3 parts: PE (portable executables) parser, OOA rule generator, and rule based classifier. PE parser extracted Windows API execution calls from PE. OOA Fast_FP-Growth algorithm used API calls and generated association rules. Finally, based on the association rules, OOA mining algorithms performed and executables marked malicious or benign. The paper claims that the proposed system performed better than other techniques including anti-virus software such as Norton AntiVirus, McAfee VirusScan and KAV, as well as the systems using data mining techniques such as naive Bayes, SVM and decision tree. To overcome the disadvantages of signature-, and behavioral-based malware detection approaches, B. Zahra, *et al*. proposed heuristic type of method which can detect malware that cannot be detected by previous two approaches [85]. Authors applied learning algorithm to generate a pattern which was similar to signature. Based on the signature, new suspicious programs were marked malware or benign. The paper mentioned API system calls, operational code (Opcode), *n*-grams, control flow graph (CFG), and hybrid features that are used extensively in heuristic approach [85].

A statistical analysis of opcode frequency distributions to identify and differentiate modern (polymorphic and metamorphic) malware is explained in [86]. A total of 67 malware executables were sampled statically disassembled and their statistical opcode frequency distributions were compared with the aggregate statistics of 20 non-malicious samples. Test results showed that there is a statistically significant difference in opcode distribution between malware and benign. To get more reliable results, more samples need to be analyzed and suggested method results' need to be compared with other well-known heuristic methods. A detection system that combines static and dynamic features has been suggested in [43]. According to the paper, combining static and dynamic features improve the method performance. By combining these features, the feature vector was constructed and classified using ML classifiers. The paper claims that the detection rate of the proposed system is satisfactory and increased when compared to their first study. However, the probability of

detecting unknown malware is still low and *FPR* is high. Using more distinctive features and train the model with more malware may improve the method performance for unknown malware.

Naval *et al*. [44] suggested a dynamic malware detection system, which collects system calls and constructs a graph that finds the semantically relevant paths among them. To find all semantically-relevant paths in a graph is a *NP-complete* problem. Thus, to reduce the time complexity, the authors measured the most relevant paths, which specify malware behaviors that cannot be found in benign samples. The authors claim that the proposed method outperforms its counterparts because it can detect malware even using system-call injection attacks at a high percentage, which the similar methods cannot detect. The paper has some limitations such as performance overhead during path computation, it is vulnerable to call-injection attacks, and cannot identify all semantically-relevant paths efficiently. Eliminating these limitations may improve the performance.

2) EVALUATION OF HEURISTIC-BASED DETECTION

The literature review of heuristic-based malware detection has been explained. Heuristic-based schema can use both strings and some behaviors to generate rules, and based on that rules it generates signature. It uses API calls, CFG, *n*-grams, Opcode, and hybrid features when generates a signature [85]. Although the heuristic-based detection can detect various forms of known and unknown malware, it is insufficient to detect all new generation of malware. In addition, heuristic-based approaches are prone to high *FPR*.

D. MODEL CHECKING-BASED MALWARE DETECTION

Although model checking is originally developed to verify the correctness of system against specifications, it has been used to detect malware as well. In this detection approach, malware behaviors are manually extracted and behavior groups are coded using linear temporal logic (LTL) to display a specific feature [10]. Program behaviors are created by looking at the flow relationship of one or more system calls and define behaviors by using properties such as hiding, spreading, and injecting. By comparing these behaviors, it is determined whether the program is malware or benign. Model checking-based detection can detect some new malware to a certain degree, but cannot detect all new generation of malware. Model checking-based detection schema can be seen in Figure 6.

1) RELATED WORKS FOR MODEL CHECKING-BASED DETECTION

Kinder *et al.* proposed a flexible method to detect malicious code patterns in executables by model checking [46]. They introduced the specification language CTPL (computation tree predicate logic) which extends the well-known logic CTL (computation tree logic), and describes an efficient model checking algorithm. According to the authors, test results demonstrated that proposed method can detect many worm

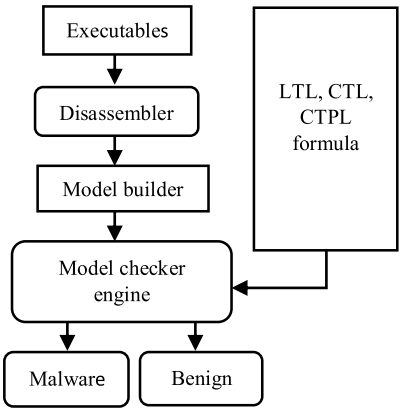

FIGURE 6. Model checking-based malware detection schema.

variants with a single specification. Proposed method has some limitations as follows:

- Can only detect worm variants,
- Some part of process has to be done manually,
- Performance of proposed method is low.

To get better results, CTPL can be extended to detect other malware. In addition, more accurate data integrity constructions and efficient data structures can be used to improve the method performance.

Holzer *et al*. presented verification technology to specify and detect malware [87]. They explained malware detection tool chain which integrates the process of specification development, and enables future automated malware analysis with specification extraction. In this method, malicious behavior is formalized using the expressive specification language CTPL based on classic CTL, and extracts a finite state model from the disassembled executable. Authors claim that a model checking-based approach can capture the semantics of malware more accurately than traditional methods, and consequently achieve higher *DR*. There is not enough information about proposed method and its test results. To get more reliable results, more malware and benign samples need to be analyzed, behavioral dependencies should be clear to extract more accurate specifications, and the whole process can be automated.

Kinder *et al*. proposed a proactive malware detector which works based on model checking and can detect worm variants without signature updates [88]. They described a tool that extracts an annotated control flow graph from the binary and automatically verifies against a formal malware specification. For this, they introduced the new specification language CTPL, which balances the high expressive power needed for malware signatures with efficient model checking algorithms. Test results showed that suggested method can recognize variants of existing malware with a low risk of *FP*. The suggested approach is an early stage and subject to some limitations:

- Model extraction process is syntactic and does not include data flow analysis,

- Malicious behaviors are split across several procedures and cannot be identified unless procedures are inlined, which decreases the method performance,
- The macro does not cover all instruction sets of the x86 architecture.

Eliminating or decreasing these deficiencies will surely improve the performance.

Beaucamps *et al*. represented rewriting and model checking which capture high-level malware behaviors when detecting malware [89]. Proposed method uses a rewriting-based abstraction mechanism which produces abstracted forms of program traces, independent of the program implementation. It can handle similar behaviors in a generic way and thus to be robust with respect to its variants. The authors claim that this method can be useful for both static and dynamic analysis. This approach is at an early stage and in the study only theoretical results are presented. To see the method efficiency, the proposed method needs to be tested.

Song and Touili proposed a pushdown model-checking method for malware detection [90]. Proposed schema works as follows:

- Binary code translates to pushdown systems (PDS),
- The paper introduced a stack computation tree predicate logic (SCTPL) to represent the malicious behaviors,
- It provides an algorithm to model-check pushdown systems against SCTPL specifications.

Proposed method reduced the model-checking problem to checking the emptiness of Symbolic Alternating Büchi Pushdown Systems. The authors claim that they obtained encouraging experimental results. However, suggested method works if the data in the stack cannot be changed by direct memory access. Identification of android malware families with model checking is represented in [91]. To show the effectiveness of suggested system most common malware family in Android environment the DroidKungFu and the Opfake families have been analyzed. The suggested algorithm can analyze and verify the java bytecode that is produced when the source code is compiled. A preliminary investigation has been also conducted to assess the validity of the proposed method. The authors mentioned that test results are promising, and they can identify malicious payloads with a very high accuracy in a reasonable time. The paper has analyzed only a few malware families, to extend the analysis and evaluate more malware families will produce more reliable results. Also, investigating the payload family tree can give clues about phylogenies of malware which will result in better classification.

2) EVALUATION OF MODEL CHECKING-BASED DETECTION

The literature review of model checking-based detection schema has been summarized. This approach is generally used for program verification and not used sufficiently for malware detection. Although it is effective to detect some new malware variants, it is still insufficient to detect all complex malware. Besides, it is a complex and resource-intensive approach which can provide only a limited view of the malware. To identify behavioral dependencies more accurately; extract more accurate specifications; and using effective LTL, CTL, CTPL formulas can improve the performance. Model checking-based detection approach can be evaluated at the early stage, so, to see the effectiveness of the approach, more studies need to be done.

E. DEEP LEARNING-BASED MALWARE DETECTION

Deep Learning is a subfield of ML that inherited from artificial neural networks (ANN) which learn from examples. It is a new approach and widely used for image processing, driverless cars, and voice control; but it is not used sufficiently in malware detection. Although it is quite effective and reduces feature space drastically, it is not resistant to evasion attacks. Deep learning-based schema can be seen in Figure 7.

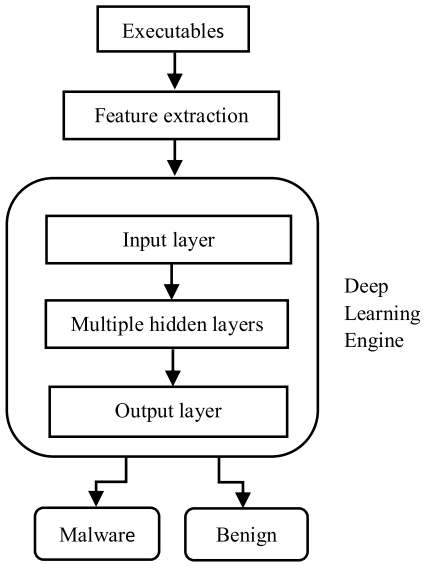

FIGURE 7. Deep learning-based malware detection schema.

1) RELATED WORKS FOR DEEP LEARNING-BASED DETECTION

Large-scale malware classification using random projections and neural networks is presented in [92]. In the suggested system, dimensionality of the original input space had reduced by a factor of 45 (179K/4K). Using suggested architecture, several very large-scale neural network systems with over 2.6 million labeled samples were trained and achieved classification results with a two-class error rate of 0.49% for a single neural network and 0.42% for an ensemble of neural networks. Authors emphasized that using more hidden layer could not improve the accuracy. For example, using one-layer neural network performed better than two and three-layer neural network. Droid-Sec which uses deep learning- based detection is proposed in [93]. It used both static and dynamic analysis and extracted more than 200 features. They used unsupervised pre-training phase and the supervised back-propagation phase. In the pre-training phase, they adopted

the deep belief network (DBN) [94] that utilizes the built restricted Boltzmann machines (RBM) which is beneficial for better characterizing Android apps. In the back-propagation phase, the pre-trained neural network fine-tuned with labeled value in a supervised manner. This way, the whole deep learning model is built completely. According to test results, 96% accuracy has been measured which outperformed SVM, C4.5, LR, and naïve Bayes. To analyze more apps and to automate the analysis processes can be useful to build more reliable detector.

Deep neural network based malware detection using two dimensional binary program features explained is in [50]. Proposed framework consists of 3 main parts:

- In the first part, 4 different types of complementary features from the benign and malicious binaries are extracted,
- In the second part, deep neural network which consists of an input layer, two hidden layers and an output layer has been used,
- In the third part, score calibrator, which translates the outputs of the neural network, is used and the probability of the file being malware is measured.

According to the authors, suggested system achieves a 95% *DR* at 0.1% *FPR* over an experimental dataset of over 400,000 software binaries. Even though proposed approach achieved high accuracy rate on the standard cross-validation, the performance decreased sharply when split validation was used. This can be eliminated by using deobfuscation the binary before feature extraction. Besides, the number of benign samples is too small when compared with the number of malware analyzed. To get accurate estimation more benign samples need to be analyzed.

Huang and W. Stokes proposed a new multi-task deep learning (multi-task neural network- MtNet) architecture for malware classification [51]. The proposed model is trained with data extracted from dynamic analysis of malicious and benign files. The system is trained on 4.5 million files and tested on a holdout test set of 2 million files. The paper claims that MtNet has made a big improvement compared to a shallow neural architecture. Multi-task learning encourages the hidden layers to learn a more generalized representation at lower levels in the neural architecture. Besides, MtNet architecture also employs rectified linear unit (ReLU) activation functions and dropout for the hidden layers. ReLU activation functions cut the number of epochs needed for training a binary malware classifier in half while dropout leads to significant reductions in the test error rate. The main challenge of this study is that it is almost impossible to increase the model performance by adding extra layers. Besides, MtNet is susceptible to attacks and can be evaded. Overcoming these challenges may improve the model performance.

2) EVALUATION OF DEEP LEARNING-BASED DETECTION

Although deep learning-based malware detection methods seem new and powerful to detect malware, it can be also fooled by evasion attacks. Grosse *et al*. investigated the viability of adversarial crafting against deep neural networks [95]. The authors mentioned that crafted inputs lead to deceive ML models which results misclassifications. For evaluation, DREBIN dataset has been used. They achieved misclassification rates of up to 80% against neural network, which shows that adversarial crafting is indeed a real threat in security critical domains. Kolosnjaji *et al.* investigated the vulnerabilities of malware detection methods that use deep networks to learn from raw bytes [96]. They proposed a gradient-based attack that is capable of evading a recently-proposed deep network by only changing few specific bytes at the end of each malware sample, while preserving its intrusive functionality. According to their test results, adversarial malware binaries evade the targeted network with high probability, even though less than 1% of their bytes are modified.

The literature review of deep learning-based malware detection has been summarized. Even though it is powerful, effective and reduces feature space drastically, it is not resistant to evasion attacks. Besides, building a hidden layer takes time and adding extra hidden layers rarely increases the model performance. Deep learning-based malware detection approach is quite in the early stages, so more studies need to be done to identify this approach more correctly.

F. CLOUD-BASED MALWARE DETECTION

Cloud computing has been rapidly developing because it provides a lot of advantages including easy accessibility, on-request storage, and decreasing costs. Since cloud has been so popular, it has also been used to detect malware. Cloud-based malware detection enhances the detection performance for Pcs and mobile devices with much bigger malware databases and intensive computational resources. Cloud-based detection uses different types of detection agents over the cloud servers and offers security as a service. A user can upload any type of file and receive a report whether uploaded file is malware or not. Cloud-based detection schema can be seen in Figure 8.

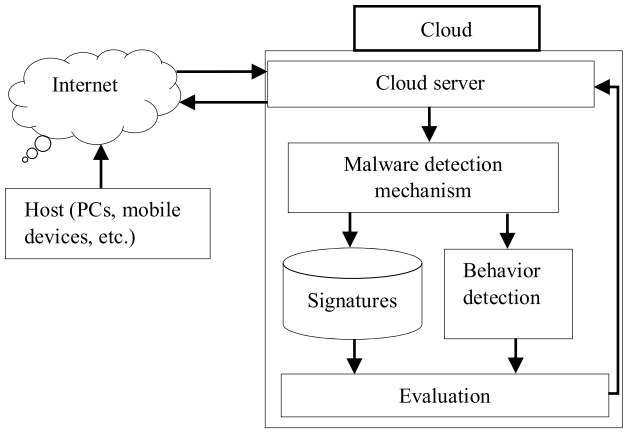

FIGURE 8. Cloud-based malware detection schema.

Even though cloud-based detection approach has many advantages, there are some issues with this detecting schema. Some of disadvantages can be the following:

(1) User needs to upload file contents to the cloud which can disclose some sensitive data such as location, password, and credit card information,

(2) The cloud detection mechanism has some over-head over other detection mechanism, so communication between the client and server must be optimized, especially for the IoT and mobile devices.

(3) The lack of real time monitoring for all files within all locations.

1) RELATED WORKS FOR CLOUD-BASED DETECTION

Sang Kil *et al.* proposed a design and implementation of a novel anti-malware system called SplitScreen [32]. It is a distributed malware detection schema which uses an additional screening step prior to the signature matching phase found in existing approach. The SplitScreen's two-phase scanning enables fast and memory efficient malware detection that can be decomposed into a client/server process that reduces the amount of storage. Proposed method implemented as an extension of ClamAV which improves scanning throughput using today's signature sets by over 2x by using half the memory. According to the authors, the speedup and memory savings of SplitScreen improve further as the number of signatures increases. The proposed method is scalable on a wide range of low-end consumer and handheld devices. Since single server is used in the cloud, it will be better to optimize the server performance, and load some works on client side.

Yanfang *et al*. presented cloud-based schema which combines file content and file relations to improve malware detection results and develops a file verdict system [97]. The system incorporated into the Comodo's Anti-malware products, and empirical studies were conducted on large daily datasets collected by Comodo cloud security center. The authors claim that their experimental results demonstrated that the accuracy and efficiency of Valkyrie system outperform other popular anti-malware software tools such as Kaspersky AntiVirus and McAfee VirusScan, as well as other alternative data mining based detection systems. However, since file relations and file content have different properties, combining these 2 features directly can decrease the quality of information including correlation and consistency issues. Using different approaches as well as Joint-Embedding approach can help to solve the correlation and consistency problem.

Martignoni *et al.* presented a framework that enhances the capabilities of existing dynamic behavior-based detectors. The proposed framework enables sophisticated behavior based analysis of suspicious programs in multiple realistic and heterogeneous environments in the cloud [54]. The suggested schema forces sample programs to execute in a distributed environment including security lab and potential victim machines. The evaluation results demonstrated that the analysis of multiple execution traces of the same malware sample in multiple end-users' environments can improve the results of the analysis with very small overhead. On the other hand, suggested framework raises the privacy and security issues, and is prone to various forms of detection and evasion attacks. Solving security related issues and implement resistant framework against evasion attacks will increase the framework performance.

A cloud-based anti-malware system called CloudEyes, which provides efficient and trusted security services for resource-constrained IoT devices presented is in [98]. For the client side, CloudEyes implemented a lightweight scanning agent that utilizes the digest of signature fragments to dramatically reduce the range of accurate matching. For the cloud server side, CloudEyes presented suspicious bucket cross filtering, a novel signature detection mechanism based on the reversible sketch structure, which provides retrospective and accurate orientations of malicious signature fragments. Furthermore, by transmitting sketch coordinates and the modular hashing, CloudEyes guarantees both the data privacy and low-cost communications by transmitting sketch coordinates and the modular hashing. Authors claim that the mechanisms in CloudEyes are effective and practical which can outperform other existing systems with less time and communication consumption. On the other hand, the detection rate and accuracy can be further improved. Also, some methods can be used such as Winnowing Block Shingling and Winnowing Multi-Hashing to reduce the size of the data in order to optimize the storage and matching performances during signature initialization.

Xiao, Liang, *et al.* investigated the cloud-based malware detection game, in which mobile devices offload their application traces to security servers via base stations or access points in dynamic networks [56]. They designed a malware detection scheme with Q-learning for a mobile device to derive the optimal offloading rate without knowing the trace generation and the radio bandwidth model of other mobile devices. The Dyna architecture is used to improve performance, and post-decision state learning-based scheme is used to accelerate the reinforcement learning process.

According to the authors, test results showed that the proposed schemes improve the detection accuracy, reduce the detection delay, and increase the utility of a mobile device in the dynamic malware detection game when compared with the benchmark strategy. Since many different parties communicate with each other during the detection process, some overhead can mitigate the performance including the network transmission delay, detection delay for mobile device, the cloud processing time, and the local detection delay. Reducing these delays will improve the performance.

Yadav R. Mahesh presented malware detection system for cloud environment [99]. The proposed work consists of 2 modules, clustering and classification. In clustering module, the input dataset is gathered into clusters with the utilization of Weighted Fuzzy C-means clustering (MFCM) algorithm. In classification module, the centroid from the clusters is given to the intermittent Auto Associative Neural Network

which is used to characterize whether the information is intruded or not. The authors claim that proposed classifier successfully identifies the malware with high detection precision thereby outperforming existing classifiers. The proposed system performance needs to be improved more in the future.

2) EVALUATION OF CLOUD-BASED DETECTION

The literature review of the cloud-based malware detection has been summarized. Recently, since cloud computing is becoming very popular, cloud-based malware detection has become popular as well. Cloud-based malware detection enhances the detection performance for Pcs and mobile devices with much bigger malware databases, and intensive computational resources. Other advantages of cloud-based detection are installations, configurations and updating regularly. However, there are some disadvantages of cloud- based detection. In order to work properly the Internet connection must be always fast, but in some cases internet connection is slow. Furthermore, real time monitoring for all files is not possible in the cloud.

G. MOBILE DEVICES-BASED MALWARE DETECTION

In mobile devices world, Android platform has become the market leader. According to recent studies, new malicious app for Android is introduced every 10s. Because of that researchers have focused on Android platform rather than other platforms for malware detection. Numerous malware detection methods have been proposed for smartphones especially for Android platform. Generally, these methods use datamining and ML algorithms to detect malware. A number of different features such as system calls, security-sensitive APIs, information flows, and control flow structures are used. Even if current studies have made improvement in detecting traditional and new generation malware for mobile devices; detecting of complex malware, and scaling the detection techniques for a large bundle of apps still remain a challenging task. Mobile devices-based detection schema can be seen in Figure 9.

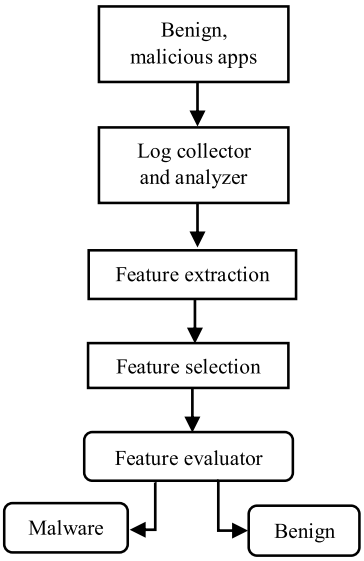

FIGURE 9. Mobile devices-based malware detection schema.

1) RELATED WORKS FOR MOBILE DEVICES-BASED MALWARE DETECTION

Isohara *et al.* proposed a kernel-base behavior analysis for Android malware inspection [57]. The system consists of a log collector and a log analysis application. The log collector records all system calls and filters events with the target application, and the log analyzer matches activities with signatures described by regular expressions to detect a malicious activity. They evaluated 230 applications in total. According to the authors, system can effectively detect malicious behaviors of the unknown applications. 230 apps are not enough to measure the efficiency of the suggested system, so more apps need to be analyzed. Besides, there is no enough information about *DR*, accuracy, and *FP*.

A new framework to obtain and analyze smartphone application activity is presented in [100]. The 2 types of datasets have been used: those from artificial malware created for test purposes, and those from real malware found in the wild. Simple *2*-means clustering algorithm is chosen to distinguish benign applications and their correspondent malware version. The authors specified that API call analysis, information flow tracking, and network monitoring technique contribute to a deeper analysis of the malware, and provide malware behaviors and more accurate results. The authors identified that open(), read(), access(), chmod(), and chown() are the most used system calls by malware. The authors mentioned that the proposed method has shown to be an effective means of isolating the malware and alerting the users to downloaded malware. However, test cases have been done generally on self-written malware and a few real malware which is not enough for real evaluation. Thus, more real malware needs to be analyzed. Moreover, there is no enough information about metrics which represent the framework performance such as *DR*, accuracy, and *FP*. In addition, the authors did not mention how they handle zero-day malware.

Host-based malware detection system for Android is presented in [58], [101]. Andromaly—a behavioral malware detection framework for Android devices is represented in [58]. The proposed framework used a host-based malware detection system that continuously monitors various features and events obtained from the mobile device and then applies ML anomaly detectors to classify the collected data as normal or malicious. They evaluated several combinations of anomaly detection algorithms, feature selection techniques and the number of top features to find the combination that yields the best performance when detecting new malware on Android. The authors claim that proposed framework is effective for both mobile devices in general and on Android in particular. However, experiments have been done on artificially-created malware rather than real malware. Saracino *et al*. proposed MADAM, a novel multi-level host-based malware detection system for Android devices that

simultaneously analyzes and correlates features at 4 levels: kernel, application, user, and package to detect malicious behaviors [101]. In this study, the actions of each malware are examined and misbehavior classes are generated from malware behaviors, which encompass most of the known malware behaviors. According to the authors, MADAM detects and effectively blocks more than 96% of malicious apps among the 2800 apps. MADAM is subject to mimicry attacks which inserting malicious code into benign apps to misleading the detection system. Besides, the paper did not mention how they handled unknown malware.

Li *et al.* introduced significant permission identification (SigPID) method to detect android malware [102]. Instead of extracting and analyzing all Android permissions, three levels of pruning by mining the permission data have been developed which identifies the most significant permissions to distinguishing malware and benign. SigPID then utilizes ML classification algorithms to classify different families of malware and benign apps. According to the authors' findings, only 22 permissions are significant out of 135 when over 2000 malware analyzed. The test results indicated that when a SVM is used as the classifier, they could achieve over 90% of precision, recall, accuracy, and f-measure; which are about the same as those produced by the baseline technique. When proposed schema is compared with other state-of-the-art methods, SigPID is more effective by detecting 93.62% of malware in the dataset and 91.4% new malware samples. To use SigPID features with static features can improve the performance. A review on feature selection in mobile malware detection is presented in [103]. In the paper, 100 studies were examined based on features selection techniques. They categorized features into 4 groups including: static, dynamic, hybrid features and applications metadata. The authors identified that the most common and distinctive static features are Android permission, network address, strings, and hardware components; dynamic features are system calls, network traffic, system components, and user interaction; hybrid features are permissions and Java code, system calls, and AndroidManifest.xml; metadata features are category, description, permissions, contact email, number of screenshots, and version. The authors emphasized that some of examined papers introduced novel methods, however due to lack of malware sample, authors could not test their systems thoroughly.

Malware detection using graph kernel for Android is presented in [59], [104]. Narayanan *et al*. proposed CASANDRA context-aware, adaptive and scalable android malware detector through online learning [59]. The authors proposed a novel graph kernel, which facilitates capturing apps security-sensitive behaviors along with their context information from dependence. The authors mentioned that CASANDRA has specific advantages: it is adaptive to the evolution in malware features over time, and explains the significant features that led to an apps classification as being malware or benign. According to the authors, CASANDRA outperforms two state-of-the-art methods on a benchmark dataset achieving 99.23% f-measure. Furthermore, when evaluated with more than 87.000 apps collected in-the-wild, CASANDRA achieves 89.92% accuracy, which has outperformed existing methods by more than 25% in their typical batch learning setting and more than 7% when they are continuously retained. The authors, did not mention how they handle malware which uses obfuscation techniques and unknown malware. To improve the model performance different graph kernel, and API dependencies such as information flows and permission dependencies can be used.

Narayanan *et al.* proposed a MKLDROID, a unified framework for Android that systematically integrates multiple views of apps for performing comprehensive malware detection and malicious code localization [104]. The MKLDROID uses a graph kernel to capture structural and contextual information from apps' dependency graphs when identifies malicious code patterns. Then, it employs multiple kernel learning (MKL) to find a weighted combination of the views which yields the best detection accuracy. Through large-scale experiments on several datasets wild apps, authors claim that MKLDROID outperforms three state-of the-art methods consistently, in terms of accuracy. In addition, malicious code localization experiments on a dataset of repackaged malware, MKLDROID was able to identify all the malware classes with 94% average recall. On the other hand, MKLDROID, cannot detect all sorts of malicious behaviors and cannot be resistant to obfuscating techniques. Furthermore, MKLDROID can be fooled by adversarial attacks. MKLDROID used only user-awareness contextual information to separate malware from benign. However, other types of contextual information such as probing and device-specific privileges could be used.

2) EVALUATION OF MOBILE DEVICES-BASED MALWARE DETECTION

The literature review of the mobile devices-based detection approach has been summarized. It can use both static and dynamic features. Although the proposed methods seem effective when detecting traditional malware, it needs to be improved to detect up-to-date malware. Besides, it is not scalable for large bundle of apps. In mobile area, the malware detection is still in the earlier stages, and there need to be more studies on this area to fill the gaps.

H. IoT-BASED MALWARE DETECTION

Internet of Things (IoT) architecture generally consists of a wide range of Internet-connected smart devices such as home appliances, network cameras, and sensors. The IoT and mobile devices have started to dominate the Internet more than PCs. Since mobile and IoT devices are becoming more popular among users day by day, they are also becoming more favorite targets for attackers. Because of that the malware detection schema landscape is changing from computers to IoT and mobile devices. IoT-based detection schema can be seen in Figure 10.

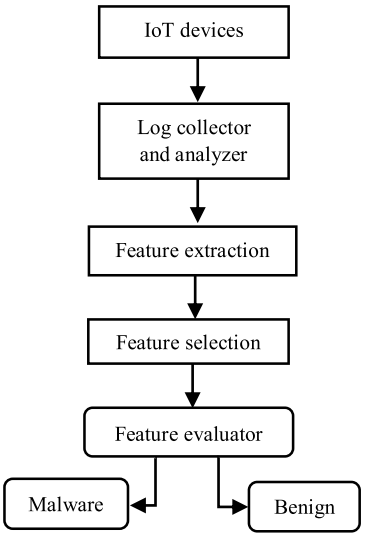

FIGURE 10. IoT-based malware detection schema.

1) RELATED WORKS FOR IoT-BASED MALWARE DETECTION

Malware detection approach for IoT devices is represented in [105], [60]. Novel light-weight technique for detecting DDos malware in IoT environments is explained in [105]. They extracted the malware images such as one-channel gray-scale image from a malware binary, then utilized a light-weight convolutional neural network for classifying their families. According to the paper, experimental results showed that the proposed system can achieve 94.0% accuracy for the classification of benign and DDoS malware, and 81.8% accuracy for the classification of benign and two main malware families. Even though proposed method is fast and lightweight, it is vulnerable to complex code obfuscation techniques. The author mentioned that this problem can be partially reduced by using more complex static features, such as Opcode sequences and API calls to a certain degree. Detecting crypto-ransomware in IoT networks based on energy consumption footprint for Android devices is represented in [60]. The proposed system use ML algorithms and specifically monitors the energy consumption patterns of different processes to classify ransomware from malware applications. According to the authors, proposed technique outperformed KNN, neural networks, SVM and RF, in terms of accuracy rate, recall rate, precision rate and f-measure. The proposed method description is not clear. Besides, there is no information about which ransomware family was analyzed and how they handled unknown ransomware. Also, the paper did not mention any limitations and challenging tasks.

2) EVALUATION OF IoT-BASED MALWARE DETECTION

The literature review of the IoT-based detection approach has been summarized. Although the proposed methods seem effective when detecting traditional malware, it needs to be improved to detect up-to-date malware. Besides, the malware detection is still in the earlier stages for IoT, and there need to be more studies on this area to fill the gaps.

TABLE 6. Comparison of malware detection approaches.

Malware Detection Approach	Detect unknown malware	Resistant to obfuscation	Well-known approach	New approach
Signature-based	×	×	✓	×
Behavior-based	✓	✓	✓	×
Heuristic-based	✓	×	✓	×
Model checking-based	✓	✓	✓	×
Deep learning-based	✓	×	×	✓
Cloud-based	✓	×	×	✓
Mobile devices-based	✓	×	×	✓
IoT-based	✓	×	×	✓

V. EVALUATION ON MALWARE DETECTION APPROACHES

In previous section, malware detection approaches were analyzed based on the main idea, algorithm types, and feature extraction methods, etc. This section summarizes detection approaches and their methods, provides advantages and disadvantages of each detection approach, and provides some suggestions to build a more effective detection schema. The comparison of malware detection approaches, and advantages, disadvantages of each malware detection approach can be seen in Table 6 and Table 7, appropriately.

Signature-, behavior-, heuristic-, and model checking-based approaches are well-known and have been used for malware detection more than a decade. These approaches are using reverse engineering, datamining, and ML techniques to detect malware.

Signature-based detection approach is fast and effective to detect known malware. During the signature generation; static features such as byte sequences, assembly instructions, strings, Opcode, and list of DLLs are used. Signature detection schema has been used for many years and decreases overhead and execution time. However, it cannot detect new generation of malware (Table 6), it is vulnerable to obfuscation and polymorphic techniques, and omitting feature selection. To build an effective signature-based detection schema: some dynamic features can be used to avoid obfuscation; feature selection phase can be added; and new technologies such as deep learning, active learning, and ML can be used to increase the detection rate.

Behavior-based detection approach is used to determine the functionality of malware. Thus, even if malware instruction sequence and signature may change, the functionality of malware will be more or less the same. So, it can detect new malware, and different variants of the same malware [106]. It is also effective against obfuscation and polymorphic techniques (Table 7). However, it produces high *FPs*. Besides, some behaviors are similar in malware and benign samples, so grouping these behaviors is difficult, and some malware does not run in protected environment and mistakenly marked as benign. To specify all behaviors correctly, multiple

TABLE 7. Pros and cons of each malware detection approach.

Malware Detection Approach	Pros	Cons
Signature-Based	Fast and efficient for known malware	Insufficient to detect new generation malware
	Used for many years	Prone to many *FPs*
	Effective to detect malware which belongs to the same family	Extracting signature takes time
		Vulnerable to obfuscation and polymorphic techniques
Behavior-Based	Determines the malware functionality	Produces high *FPs*
	Effective to detect new malware	Some behaviors are similar in malware and benign samples
	Effective to detect different variants of the same malware	Impossible to specify all behaviors
	Effective against obfuscation and polymorphic techniques	Difficult to group behavior as malicious and normal
Heuristic-Based	Can detect some previously unknown malware	Numerous rules and training phases
	Can use both static and dynamic features	Vulnerable to metamorphic techniques
Model Checking-Based	Effective to detect malware that belongs to the same family	Complex and resource-intensive technique
	Effective against obfuscation and polymorphic techniques	Obtains a limited view of the malware
		Cannot detect all new generation of malware
Deep Learning-Based	Powerful and effective	Not resistant to evasion attacks
	Reduces feature space drastically	Building a hidden layer takes time
Cloud-Based	Enhances the detection performance for PCs and mobile devices	Lacks real-time monitoring
	Bigger malware databases and intensive computational resources	Can disclose some sensitive data such as password, and location
	Easily accessible, manageable, and updates regularly	Over-head between client and server
Mobile devices-Based	Effective to detect traditional and new generation malware	Cannot detect complex malware
	Can use both static and dynamic features	Cannot scale large bundle of apps
IoT-Based	Can use both static and dynamic features	Cannot detect complex malware

execution paths can be gathered using different machines on clouds. This can decrease the number of malware mistakenly marked as benign.

Heuristic-based detection approach can use both static and dynamic features such as API calls, Opcode, CFG, *n*-gram, list of DLLs, and hybrid features. It can detect some previously unknown malware, but it is vulnerable to metamorphic techniques, and numerous rules and training phases [107] make this detection approach complicated (Table 7). Decreasing the number of rules, and building a more efficient learning phase can improve the method performance.

Model checking-based approach is powerful, can detect unknown malware, and is resistant against obfuscation and polymorphic techniques (Table 7). However, it can obtain a limited view of the malware, not resistant to evasion attacks, and cannot detect all new generation of malware. To identify more accurate formulas, and using effective model checker may improve performance.

Recently; deep learning-, cloud-, mobile devices-, and IoT-based approaches have started to be used in malware detection (VI). Deep learning-based detection approaches are effective to detect new malware and reduce features space sharply [108], [109], but it is not resistant to some evasion attacks. On the other hand, cloud-based detection approaches increase *DR*, decrease *FPs*, and provide bigger malware databases and powerful computational resources [110]. The overhead between client and server, and lack of real monitoring are still a challenging tasks in cloud environment. Mobile devices- and IoT-based detection approaches can use both static and dynamic features, and improve detection rates on traditional and new generation of malware [111]. But, it has difficulties to detect complex malware, and is not scalable for large bundle of apps. To integrate the mobile and IoT-based approach with cloud-based can improve the*DR* and scale better for large bundle of apps.

Even though each detection method has its own advantages and works better for different datasets, no detection method could detect all malware. Malware detection rate versus complexity of malware can be seen in Figure 11. When complexity of malware (unknown malware, new generation of malware, obfuscated malware) increases, the detection rate decreases for all detection approaches. It can be seen that signature types of detection approaches such as signature-, heuristic-, and most of the time mobile devices- and IoT-based schemas show lower performance than other approaches such as behavior-, model checking-, cloud-, and deep learning-based approaches (Figure 11).

This is because the later approaches are more effective to detect unknown and obfuscated malware. Behavior-based detection approach performs pretty well, while signature based detection approach shows lowest performance (Figure 11). Model checking- and cloud-based detection approaches perform slightly better than deep-learning-, heuristic-, mobile devices-, and IoT-based detection approaches. Combining malware detection approaches can provide better detection mechanism. For example, combining behavior-based with model checking-based approaches, and using deep learning and cloud at the same time will surely provide better detection mechanism. Besides, using new technologies such as block chain and big data may give more opportunity to build a more effective detector.

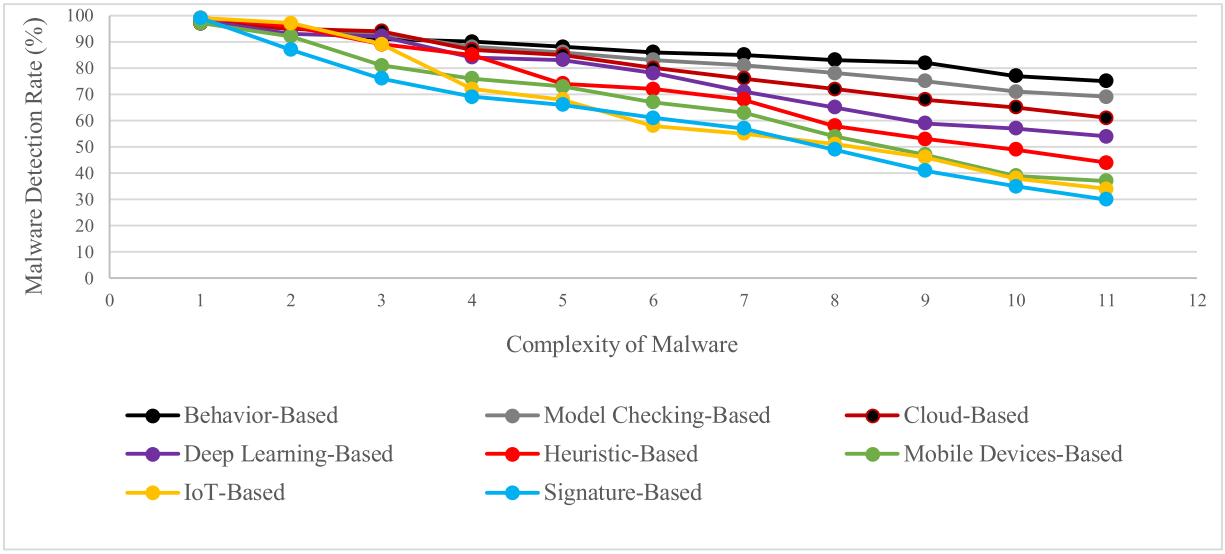

FIGURE 11. Malware detection rate versus complexity of malware based on previous studies.

Although malware detectors are being improved every day, the following research challenges still remain an open issue in malware detection approaches:

- New generation malware uses some obfuscation and packing techniques to hide itself. By using these techniques malware can prevent itself from being correctly analyzed and avoid detection. Signature-based detection approach is not resistant to malware obfuscation. Even if behavior-, and model checking-based approaches are effective to most of obfuscation techniques, they cannot be resistant to all obfuscation techniques.
- Real-time monitoring and detection are a challenging tasks. Most of the studies have been done so far to detect malware by using dataset and are not appropriate for real-time monitoring.
- Most of the malware detection approaches are prone to *FPs* and *FNs*. Some features and signatures can be very close in malware and benign samples which raises *FPs* and *FNs*.
- No detection method can affectively detect all unknown malware.
- Generally, learning algorithms are prone to bias, and overfitting. This leads to decreases *DRs* and increases *FPs*.
- There is no well-known and accepted dataset which can be used to evaluate the malware detection approaches performance. This is because each malware detection method uses different malware and dataset.

VI. CONCLUSION

Even though several new methods have been proposed by using these different malware detection approaches, no method could detect all new generation and sophisticated malware. For the known malware signature- and heuristic-based detection approaches perform well. On the other hand, for an unknown and complicated malware behavior-, model checking-, and cloud-based approaches perform better. Deep learning-, mobile-, and IoT-based approaches have also emerged to detect some portion of known and unknown malware. However, some portion of malware could not be detected by using these approaches. This shows that to build an effective method to detect malware is a very challenging task, and there is a huge gap to fill in new studies and methods. Even though the trends in malware creation and detection approaches are changing rapidly, this study still can be considered as a key reference for the computer scientist and developers who work in this field. As a future work, new approach and method need to be proposed. To do that combining malware detection approaches can be one of the solutions among many. For instance, combining behavior-based with model checking-based approaches, and using deep learning and cloud at the same time will surely provide better detection mechanism.

Recently, the number, severity, sophistication of malware attacks, and cost of malware inflicts on the world economy have been increasing exponentially. Attacks with these kinds of software have a disastrous effect and cause considerable material damage to individuals, private companies, and governments' assets. Thus, malware should be detected before damaging the important assets in the company. However, there are large gaps in the research area of malware detection and prevention. The aim of this study is to contribute to the research of malware. In this context, the paper has presented a detailed review of the state-of-the-art studies for malware detection approaches, and techniques and algorithms that are used for malware detection. The advantages and disadvantages of each malware detection approach have been explained. As well as datamining and ML, new technologies such as deep learning-, cloud-, mobile devices-, and IoT-based detection schemas have also become popular.

REFERENCES

[1] S. Morgan. 2019 cybersecurity almanac: 100 facts, figures, predictions and statistics. Cisco and Cybersecurity Ventures. Accessed: Nov. 10, 2019. [Online]. Available: https://cybersecurityventures.com/cybersecurity-almanac-2019

[2] R. Samani and G. Davis. (2019). *McAfee Mobile Threat Report Q1*. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf

[3] F. Cohen, "Computer viruses," Ph.D. dissertation, Univ. Southern California, Los Angeles, CA, USA, 1986.

[4] F. Cohen, "A formal definition of computer worms and some related results," *Comput. Secur.*, vol. 11, no. 7, pp. 641–652, Nov. 1992.

[5] D. M. Chess and S. R. White, "An undetectable computer virus," in *Proc. Virus Bull. Conf.*, vol. 5, 2000.

[6] F. Cohen, "Computer viruses: Theory and experiments," *Comput. Secur.*, vol. 6, no. 1, pp. 22–35, 1987.

[7] L. M. Adleman, "An abstract theory of computer viruses," in *Advances in Cryptology—CRYPTO*. New York, NY, USA: Springer-Verlag, 1990.

[8] D. Spinellis, "Reliable identification of bounded-length viruses is NP-complete," *IEEE Trans. Inf. Theory*, vol. 49, no. 1, pp. 280–284, Jan. 2003.

[9] Z. Zuo, Q. Zhu, and M. Zhou, "On the time complexity of computer viruses," *IEEE Trans. Inf. Theory*, vol. 51, no. 8, pp. 2962–2966, Aug. 2005.

[10] K. Alzarooni, "Malware variant detection," Ph.D. dissertation, Dept. Comput. Sci., Univ. College London, London, U.K., 2012.

[11] P. Szor, *The Art of Computer Virus Research and Defense*. Upper Saddle River, NJ, USA: Pearson Education, 2005.

[12] W. Stallings and L. Brown, *Computer Security: Principles and Practice*. Upper Saddle River, NJ, USA: Pearson Education, 2012.

[13] P. Szor and P. Ferrie, "Hunting for metamorphic," in *Proc. Virus Bull. Conf.*, 2001.

[14] S. Alam, R. Horspool, I. Traore, and I. Sogukpinar, "A framework for metamorphic malware analysis and real-time detection," *Comput. Secur.*, vol. 48, pp. 212–233, Feb. 2015.

[15] M. D. Preda. *Code Obfuscation and Malware Detection by Abstract Interpretation*. Accessed: Nov. 12, 2019. [Online]. Available: https://iris.univr.it/retrieve/handle/11562/337972/3306/main.pdf

[16] W. Yan, Z. Zhang, and N. Ansari, "Revealing packed malware," *IEEE Secur. Privacy Mag.*, vol. 6, no. 5, pp. 65–69, Sep. 2008.

[17] Y. Alosefer, "Analysing Web-based malware behaviour through client honeypots," Ph.D. dissertation, School Comput. Sci. Inform., Cardiff Univ., Cardiff, U.K., 2012.

[18] M. Sikorski and A. Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. San Francisco, CA, USA: No Starch Press, 2012.

[19] N. Idika and P. Mathur, "A survey of malware detection techniques," Purdue Univ., West Lafayette, IN, USA, Tech. Rep., vol. 48, 2007.

[20] E. Eilam, *Reversing: Secrets of Reverse Engineering*. Hoboken, NJ, USA: Wiley, 2011.

[21] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 3, 2018.

[22] M. Tavallaee, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, 2009, pp. 1–6.

[23] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of Android malware in your pocket," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, vol. 14, 2014, pp. 23–26.

[24] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," 2018, *arXiv:1802.10135*. [Online]. Available: https://arxiv.org/abs/1802.10135

[25] *Classification of Malware PE Headers*. Accessed: Nov. 14, 2019. [Online]. Available: https://github.com/urwithajit9/ClaMP

[26] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah, and A. A. Ghorbani, "Towards a network–based framework for Android malware detection and characterization," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017.

[27] S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," 2018, *arXiv:1804.04637*. [Online]. Available: https://arxiv.org/abs/1804.04637

[28] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *J. Inf. Secur.*, vol. 5, no. 2, pp. 56–64, 2014.

[29] O. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017.

[30] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proc. IEEE Symp. Secur. Privacy*, May 2001.

[31] A. Karnik, S. Goswami, and R. Guha, "Detecting obfuscated viruses using cosine similarity analysis," in *Proc. 1st Asia Int. Conf. Modeling Simulation (AMS)*, Mar. 2007.

[32] S. K. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, and D. G. Andersen, "SplitScreen: Enabling efficient, distributed malware detection," *J. Commun. Netw.*, vol. 13, no. 2, pp. 187–200, Apr. 2011.

[33] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A static malware detection system using data mining methods," 2013, *arXiv:1308.2831*. [Online]. Available: https://arxiv.org/abs/1308.2831

[34] A. Malhotra and K. Bajaj, "A hybrid pattern based text mining approach for malware detection using DBScan," *CSI Trans.*, vol. 4, nos. 2–4, pp. 141–149, Dec. 2016.

[35] A. Moser, C. Kruegel, and E. Kirda, "Exploring multiple execution paths for malware analysis," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007.

[36] G. Wagener, R. State, and A. Dulaunoy, "Malware behaviour analysis," *J. Comput. Virol.*, vol. 4, no. 4, pp. 279–287, Nov. 2008.

[37] Y. Park, D. S. Reeves, and M. Stamp, "Deriving common malware behavior through graph clustering," *Comput. Secur.*, vol. 39, pp. 419–430, Nov. 2013.

[38] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics–based online malware detection: Towards efficient real–time protection against malware," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 289–302, Feb. 2016.

[39] M. Norouzi, A. Souri, and S. M. Zamini, "A data mining classification approach for behavioral malware detection," *J. Comput. Netw. Commun.*, vol. 2016, p. 1, Mar. 2016.

[40] B. Zhang, J. Yin, J. Hao, D. Zhang, and S. Wang, "Malicious codes detection based on ensemble learning," in *Autonomic and Trusted Computing* (Lecture Notes in Computer Science), vol. 4610. Berlin, Germany: Springer, 2007, pp. 468–477.

[41] K. Griffin, S. Schneider, X. Hu, and T.-C. Chiueh, "Automatic generation of string signatures for malware detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Berlin, Germany: Springer, 2009.

[42] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *J. Comput. Virol.*, vol. 7, no. 4, pp. 247–258, Nov. 2011.

[43] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 646–656, Mar. 2013.

[44] S. Naval, V. Laxmi, M. Rajarajan, M. S. Gaur, and M. Conti, "Employing program semantics for malware detection," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 12, pp. 2591–2604, Dec. 2015.

[45] P. Singh and A. Lakhotia, "Static verification of worm and virus behavior in binary executables using model checking," in *Proc. IEEE Syst., Man Soc. Inf. Assurance Workshop*, Mar. 2003.

[46] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Detecting malicious code by model checking," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Berlin, Germany: Springer, 2005.

[47] P. Beaucamps and J. Marion, "On behavioral detection," in *Proc. EICAR*, vol. 9, 2009.

[48] F. Song and T. Touili, "Efficient malware detection using model-checking," in *Proc. Int. Symp. Formal Techn.* Berlin, Germany: Springer, 2012.

[49] A. Cimitile, F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, and G. Vaglini, "Model checking for mobile android malware evolution," in *Proc. IEEE/ACM 5th Int. FME Workshop Formal Methods Softw. Eng. (FormaliSE)*, May 2017.

[50] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015.

[51] W. Huang and J. W. Stokes, "MtNet: A multi-task neural network for dynamic malware classification," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Cham, Switzerland: Springer, 2016.

[52] D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017.

[53] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: A heterogeneous deep learning framework for intelligent malware detection," *Knowl. Inf. Syst.*, vol. 54, no. 2, pp. 265–285, Feb. 2018.
[54] L. Martignoni, R. Paleari, and D. Bruschi, "A framework for behavior-based malware analysis in the cloud," in *Proc. Int. Conf. Inf. Syst. Secur.* Berlin, Germany: Springer, 2009.
[55] H. Sun, X. Wang, J. Su, and P. Chen, "RScam: Cloud-based anti-malware via reversible sketch," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Cham, Switzerland: Springer, 2015.
[56] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud–based malware detection game for mobile devices with offloading," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2742–2750, Oct. 2017.
[57] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for Android malware detection," in *Proc. 7th Int. Conf. Comput. Intell. Secur.*, Dec. 2011.
[58] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: A behavioral malware detection framework for Android devices," *J. Intell. Inf. Syst.*, vol. 38, no. 1, pp. 161–190, Feb. 2012.
[59] A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, "Context–aware, adaptive, and scalable Android malware detection through online learning," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 3, pp. 157–175, Jun. 2017.
[60] A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K.-R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Hum. Comput.*, vol. 9, no. 4, pp. 1141–1152, Aug. 2018.
[61] K. Hahn, "Robust static analysis of portable executable malware," in *Proc. HTWK Leipzig*, 2014.
[62] *Hooked on Mnemonics Worked for Me*. Accessed: Nov. 15, 2019. [Online]. Available: http://hooked-on-mnemonics.blogspot.com/2011/01/intro-to-creating-anti-virus-signatures.html
[63] M. F. Zolkipli and A. Jantan, "A framework for malware detection using combination technique and signature generation," in *Proc. 2nd Int. Conf. Comput. Res. Develop.*, May 2010.
[64] Y. Tang, B. Xiao, and X. Lu, "Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms," *Comput. Secur.*, vol. 28, no. 8, pp. 827–842, Nov. 2009.
[65] H. Razeghi Borojerdi and M. Abadi, "MalHunter: Automatic generation of multiple behavioral signatures for polymorphic malware detection," in *Proc. ICCKE*. Mashhad, Iran: Ferdowsi Univ. Mashhad, vol. 1, Oct. 2013.
[66] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Proc. IEEE Symp. Secur. Privacy (Samp;P)*, Oakland, CA, USA, May 2005, pp. 226–241.
[67] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *Proc. 7th USENIX Conf. Netw. Syst. Design Implement.*, San Jose, CA, USA, 2010, pp. 391–404.
[68] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection," in *Proc. 11th Int. Conf. Enterprise Inf.*, vol. 9, 2009, pp. 317–320.
[69] E. Fix and J. L. Hodges, "Discriminatory analysis: Nonparametric discrimination: Small sample performance," Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. 11, 1952.
[70] D. Venugopal and G. Hu, "Efficient signature based malware detection on mobile devices," *Mobile Inf. Syst.*, vol. 4, no. 1, pp. 33–49, 2008.
[71] M. Zheng, M. Sun, and J. C. Lui, "Droid analytics: A signature based analytic system to collect, extract, analyze and associate android malware," in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jul. 2013.
[72] Y. Fukushima, A. Sakai, Y. Hori, and K. Sakurai, "A behavior based malware detection scheme for avoiding false positive," in *Proc. 6th IEEE Workshop Secure Netw. Protocols*, Oct. 2010, pp. 79–84.
[73] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics–aware malware detection," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2005.
[74] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "AccessMiner: Using system-centric models for malware protection," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 399–412.
[75] M. Chandramohan, H. B. K. Tan, L. C. Briand, L. K. Shar, and B. M. Padmanabhuni, "A scalable approach for malware detection through bounded feature space behavior modeling," in *Proc. 28th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, Nov. 2013, pp. 312–322.
[76] S.-T. Liu, H.-C. Huang, and Y.-M. Chen, "A system call analysis method with mapreduce for malware detection," in *Proc. IEEE 17th Int. Conf. Parallel Distrib. Syst.*, Dec. 2011, pp. 631–637,
[77] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel, "A view on current malware behaviors," in *Proc. USENIX Workshop*, 2009.
[78] H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K.-R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hack Tech.*, vol. 14, no. 3, pp. 213–223, Aug. 2018.
[79] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X.-Y. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," in *Proc. USENIX Secur. Symp.*, Aug. 2009, vol. 4, no. 1, pp. 351–366.
[80] M. Eskandari and S. Hashemi, "A graph mining approach for detecting unknown malwares," *J. Vis. Lang. Comput.*, vol. 23, no. 3, pp. 154–162, Jun. 2012.
[81] F. Adkins, L. Jones, M. Carlisle, and J. Upchurch, "Heuristic malware detection via basic block comparison," in *Proc. 8th Int. Conf. Malicious Unwanted Softw., Amer. (MALWARE)*, Oct. 2013.
[82] W. Arnold and G. Tesauro, "Automatically generated Win32 heuristic virus detection," in *Proc. Int. Virus Bull. Conf.*, vol. 200, 2000.
[83] Y. Ye, T. Li, Q. Jiang, and Y. Wang, "CIMDS: Adapting postprocessing techniques of associative classification for malware detection," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 3, pp. 298–307, May 2010.
[84] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent PE-malware detection system based on association mining," *J. Comput. Virol.*, vol. 4, no. 4, pp. 323–334, Nov. 2008.
[85] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *Proc. 5th Conf. Inf. Knowl. Technol.*, May 2013.
[86] D. Bilar, "Opcodes as predictor for malware," *Int. J. Electron. Secur. Digit. Forensics*, vol. 1, no. 2, p. 156, 2007.
[87] A. Holzer, J. Kinder, and H. Veith, "Using verification technology to specify and detect malware," in *Proc. Int. Conf. Comput. Aided Syst. Theory*. Berlin, Germany: Springer, 2007.
[88] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Proactive detection of computer worms using model checking," *IEEE Trans. Depend. Sec. Comput.*, vol. 7, no. 4, pp. 424–438, Oct. 2010.
[89] P. Beaucamps, I. Gnaedig, and J. Marion, "Abstraction-based malware analysis using rewriting and model checking," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2012.
[90] F. Song and T. Touili, "Pushdown model checking for malware detection," *Int. J. Softw. Tools Technol. Transf.*, vol. 16, no. 2, pp. 147–173, Apr. 2014.
[91] P. Battista, F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Identification of Android malware families with model checking," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016.
[92] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013.
[93] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid–Sec: Deep learning in Android malware detection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 371–372, 2014.
[94] Y. Bengio, "Learning deep architectures for AL," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[95] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," 2016, *arXiv:1606.04435*. [Online]. Available: https://arxiv.org/abs/1606.04435
[96] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018.
[97] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, "Combining file content and file relations for cloud based malware detection," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011.
[98] H. Sun, X. Wang, R. Buyya, and J. Su, "CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained Internet of Things (IoT) devices," *Softw. Pract. Exper.*, vol. 47, no. 3, pp. 421–441, Mar. 2017.
[99] R. M. Yadav, "Effective analysis of malware detection in cloud computing," *Comput. Secur.*, vol. 83, pp. 14–21, Jun. 2019.

[100] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for Android," in *Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices (SPSM)*, 2011.
[101] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and efficient behavior-based Android malware detection and prevention," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 1, pp. 83–97, Jan. 2018.
[102] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.
[103] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," *Digit. Invest.*, vol. 13, pp. 22–37, Jun. 2015.
[104] A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, "A multi-view context-aware approach to Android malware detection and malicious code localization," *Empirical Softw. Eng*, vol. 23, no. 3, pp. 1222–1274, Jun. 2018.
[105] J. Su, V. Danilo Vasconcellos, S. Prasad, S. Daniele, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2018.
[106] E. B. Karbab and M. Debbabi, "MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports," *Digit. Invest.*, vol. 28, pp. S77–S87, Apr. 2019.
[107] C. Choi, C. Esposito, M. Lee, and J. Choi, "Metamorphic malicious code behavior detection using probabilistic inference methods," *Cognit. Syst. Res.*, vol. 56, pp. 142–150, Aug. 2019.
[108] W. Wang, M. Zhao, and J. Wang, "Effective Android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3035–3043, Aug. 2019.
[109] H. Zhang, W. Zhang, Z. Lv, A. K. Sangaiah, T. Huang, and N. Chilamkurti, "MALDC: A depth detection method for malware based on behavior chains," in *Proc. World Wide Web*, 2019, pp. 1–20.
[110] Q. K. Ali Mirza, I. Awan, and M. Younas, "CloudIntell: An intelligent malware detection system," *Future Gener. Comput. Syst.*, vol. 86, pp. 1042–1053, Sep. 2018.
[111] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, "A combination method for Android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21235–21245, 2019.

ÖMER ASLAN received the B.Sc. degree in computer engineering from the University of Trakya, Turkey, in 2009, and the M.Sc. degree in information security from the University of Texas at San Antonio, USA, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with Ankara University, Turkey. He is a Research Assistant in cyber security with the University of Siirt, Turkey. He has published seven conference papers and one book chapter.

REFIK SAMET (Member, IEEE) is currently a Professor with the Computer Engineering Department, Ankara University, Turkey. He has worked and managed projects at TUBITAK, NATO, European Union, and University Scientific Research Units. He is working on reliability and fault-tolerance of real-time computer systems, information security, cyber security, malware analysis, and computer forensics. He has six patents, four books, and four book chapters. He has over 60 articles published in National and International journals and more than 60 conference papers. He is a member of scientific committee at many National and International science conferences and journals. He is a member of the IEEE Computer Society and IEEE Turkey Section.