IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 11, 2019, accepted September 30, 2019, date of publication October 7, 2019, date of current version October 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945787

Review: Build a Roadmap for Stepping Into the Field of Anti-Malware Research Smoothly

WEIJIE HAN[1,2], JINGFENG XUE[1], YONG WANG[1], SHIBING ZHU[2], AND ZIXIAO KONG[1]
[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[2]School of Space Information, Space Engineering University, Beijing 101416, China

Corresponding author: Yong Wang (wangyong@bit.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2016QY06X1205.

ABSTRACT In the era of cyberspace, malware is the main weapon for launching cyber-attacks and the critical rival for the security community. More and more researchers are investing in the wave of anti-malware research. In order to promote researchers to enter the field of anti-malware research more smoothly, it is necessary to provide a comprehensive roadmap of the related theory and techniques, so that new researchers can quickly obtain the desiring knowledge. To this end, this article systematically reviews the knowledge of malware in accordance with the most effective research route, that is, ''Why? → What? → How?'' First, we analyze the significance of conducting malware research and explains ''why?''; then, the concept, type, and harm of malware are summarized, and introduce ''what?''; finally, the focus is on ''how?'', i.e. malware detection and classification. In the presence of the increasing complexity of malware types and scales, this paper focuses on machine learning-based detection and classification methods in view of feature engineering and analysis environment. The abstract and contributions are summarized for each typical method so that researchers can quickly find the preferred references like a dictionary, and establish a comprehensive and clear framework for anti-malware research in a correct route.

INDEX TERMS Malware, machine learning, feature engineering, review, roadmap.

I. INTRODUCTION

In the age of the Internet, malware has caused serious damage to the network. To protect legitimate users from malware, researchers have designed different anti-virus software to build a security barrier. Unfortunately, due to the trend of economic interests, malware producers are constantly updating malware manufacturing technologies and modifying the structure and functions of disguised malware, leading to enormous growth in the volume of malware variants and the ability to evade the traditional detection. In order to mitigate the serious threats aroused by malware, it is urgent for analysts to establish an overall framework for anti-malware counterworking. However, it is not a turn-key process to obtain desiring knowledge for new researchers when conducting anti-malware research. A comprehensive reference guide may be the most appealing tool for them before stepping into the field of anti-malware research. To this end, this paper takes the Windows platform malware as the object, systematically reviews the malware concept, type, harm, evolution trend, and the commonly-used intelligent detection methods in recent years, and discusses the issues that need to tackle in the future research. By performing an extensive survey on malware and anti-malware literature, the paper aims to provide theoretical and methodological support for anti-malware research.

In summary, we make the following contributions in this review:

(1) Introduce malware according to the idea of ''Why? → What? → How?'' It is convenient for researchers to quickly and effectively establish awareness of malware.

(2) Depict a roadmap for conducting malware research, which can help researchers quickly and effectively step into the field of anti-malware research;

(3) Based on literature published in prestigious journals and top academic conferences after 2010, provide the latest and most systematic references for researchers to ensure the effectiveness and efficiency of the review;

(4) Briefly introduce the implementation process and innovation points of each typical method, and facilitate the researchers to quickly find the methods that can be referred to according to their research requirements;

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

(5) Classify the literature of malware research from different angles, which is convenient for researchers to quickly find the entry point for malware research.

II. WHY STUDY MALWARE?

Choosing the direction is the first step before conducting research. The choice of malware analysis as a research direction is reflected in the following aspects:

(1) Malware has become an acute threat to the current network environment

In the era of cyberspace, the network has become the coveted target of cyber attackers. Attackers often employ sophisticated malware to launch cyber-attacks. Even though the anti-virus community has spared no effort to build the protection fence, the volume of malware is increasing dramatically, and the threat posed by malware continues to rise. According to data released by SafetyDetective [89], malware infections have continued to swell over the past decade, with more than 810 million malware infections in 2018. And with the trend of economic interests, the volume of malware will continue to increase in the future. A safe network security environment can only be built by strengthening malware protection barrier continuously.

(2) The malware offensive and defensive arms race can promote the anti-malware research to be always at the forefront

Attack and protection of malware is an iterative evolution process. Malware creators have been exploring new technologies, writing new codes, and creating new threats; while the protection side has been analyzing the characteristics of new malware and adopting new technologies to ensure accurate and efficient detection of malware. Therefore, choosing malware as the research direction will not only make research work always at the forefront of network security, but also researchers can overlook the latest development trends of the network security struggle, and stimulate the continuous driving force in the research process.

III. CONCEPT, TYPE, AND HARM OF MALWARE

Any program that damages a user, computer, or network in some way is called malware Kramer and Bradfield [52]. According to the general knowledge of researchers, common malware mainly includes the following 10 types Panda Security Info Glossary[77]:

(1) *Computer Virus:* A computer virus is a set of computer instructions or codes that are attached to a computer program and activated after the host program runs. A computer virus can affect the normal use of the computer and self-replicate.

(2) *Worm:* A worm is a malicious program that is similar to a computer virus and capable of self-replication. The difference is that the worm does not need to be attached to another program, and it can be copied or executed without the host.

(3) *Backdoor:* A backdoor is a malicious program that stealthily installs itself into a computer to enable the attackers to bypass the security barrier of the computer and gain access to a program or system.

(4) *Botnet:* A botnet is a malicious program that enables an attacker to access the system stealthily like a backdoor. All infected computers will receive commands from the control command server to jointly attack the target.

(5) *Downloader:* A downloader is a malicious program that is usually employed to download other malware after being installed.

(6) *Launcher:* A launcher is a malicious program that configures itself or other malicious code snippets for instant or future secret operations. It aims to install some programs to hide malicious behavior from the user. The launcher usually contains the malware that is loaded to achieve the purpose of launching other malicious programs.

(7) *Kernel-Kit:* A kernel-kit is a malicious program designed to hide other malicious applications. Kernel suites are often combined with other malware (such as backdoors) into a toolkit that enables an attacker to remotely access and makes the software hard to find by the victim.

(8) *Spyware:* Spyware is a type of malicious program designed to steal confidential information from an infected computer and transmit it to the remote attacker without the user's permission.

(9) *Ransomware:* Ransomware is a type of malicious program that is implicitly installed on the victim's computer, encrypts files on the infected computer, and intimidates and extorts the victim.

(10) *Spamware:* Spamware is a type of malicious program that uses the system and network resources to deliver large amounts of spam. This type of malware benefits by selling spam delivery services to attackers.

The main harm of malware includes:

(1) *Degradation of Computer and Network Operation Performance:* The most direct harm of malware is that it affects the normal operation of computers and networks, resulting in a sharp or slow decline in their operational performance, ultimately causing damage to normal program operations.

(2) *Hardware Failure:* Some malware causes hardware failures by modifying its parameters or corrupting its core data and makes it work improperly. For example, the previous CIH virus caused the startup program to work improperly by destroying the data stored in the drive and the BIOS chip. The victim had to replace the BIOS chip to restart the computer Gratzer and Naccache [32].

(3) *Data Loss or Theft:* In the current information age, information has become the most valuable intangible asset. A large amount of malware aims to steal secret information, such as stealing personal privacy information from personal computers and then swindling victims. More malware aims at companies to steal valuable intelligence information from the company and gaining economic benefits; even more, targets a country and obtains intelligence information related to national security from administrative departments to achieve strategic goals.

(4) *Other Hidden Damage:* In addition to the above obvious harms, malware can also cause some hidden damage. For example, some Trojans and viruses do not cause any

TABLE 1. Advantages and disadvantages of static and dynamic analysis.

Analysis method	Advantages	Disadvantages
Static analysis	Low resource consumption, high analysis speed, high coverage	Weak detection of unknown malware and obfuscated malware
Dynamic analysis	Detect unknown malware and obfuscated malware	High resource consumption, miss malicious behavior outside the scope of analysis, difficult to detect evasive malware that hides malicious behavior at runtime.

apparent damage to the system after infecting the target system. Instead, they use the infected system as a transit station to use the Internet to send some command information or spam, and this information will be sent out hidden in normal network traffic, not easily detected.

IV. MAIN TASKS OF MALWARE DEFENSE AND COMMON ANALYSIS METHODS

Once malware emerges, malware and anti-malware have fallen into a never-ending struggle. To detect and defend against malware, we first need to analyze the unknown software and detect the malware, and then classify the malware into its corresponding family.

A. MAIN TASKS OF MALWARE DEFENSE

The two main tasks of malware defense are detection and classification. Malware detection aims at identifying malware from unknown samples, and the goal of classification is to group malware into their corresponding families. The features of the program extracted during the malware detection process can also be applied to malware classification. According to the different processes of feature extraction, malware detection and classification techniques are usually grouped into two categories: static analysis and dynamic analysis.

B. MALWARE ANALYSIS

1) STATIC ANALYSIS

In the process of static analysis, the program does not need to run actually. The researcher usually extracts information from the PE header, PE body, and binary code of the program, or disassembles the program, and extracts the opcode or other related information from the assembly code to characterize the program, so as to analyze the program's maliciousness Sung *et al.* [102]. Static analysis methods are more efficient but need to deal with the effects of packing and obfuscation Moser *et al.* [66].

2) DYNAMIC ANALYSIS

Compared with static analysis, dynamic analysis requires the actual running of a program to capture the behavioral characteristics during its execution. As a result, the dynamic analysis usually is considered to be a behavior-based analysis technique. The main dynamic features include the API sequence and various kinds of behaviors that the program interacts with the underlying OS resources during the runtime. In the process of dynamic analysis, a malware sample usually runs in a virtual environment so as to prevent causing damages to the host system.

3) HYBRID ANALYSIS

In addition, some researchers have combined static analysis with dynamic analysis to perform a hybrid analysis of malware by extracting both static and dynamic features from the malware and merging them to build a hybrid feature vector, and outline more comprehensive and accurate profiling of malware finally Roundy and Miller [87].

The advantages and disadvantages of the aforementioned analysis techniques are illustrated in Table 1.

V. MALWARE RESISTANCE MANEUVERS

In response to the development of malware detection technologies, the malware itself is constantly evolving and resisting by adopting different maneuvers to change its own characteristics or to cover hidden malicious behaviors, thereby avoiding detection. Obfuscation is the use of certain methods to change the program code while retaining its functionality, to reduce the possibility of being analyzed, and to counteract reverse engineering by confusing the original code. In general, the common ways of malware obfuscation include packing, polymorphism, oligomorphism, and metamorphism You and Yim [113]; Okane *et al.* [76].

A. PACKING

Packing is currently the most commonly used method of code obfuscation or compression. It first compresses and encrypts PE files, then restores the original state at runtime and loads them into memory for execution. Malware authors can change the characteristics of malware without having to change too many codes in this mode.

B. POLYMORPHISM

Polymorphism, also known as code sealing and code packing, uses encryption and data addition techniques to change the body of malware, and in order to keep changing, it can also change the encryption key every time it infects and change the decryption function to achieve continuous confusion. Identifying polymorphic malware from the wild is a daunting task for traditional anti-malware tools because polymorphic malware is constantly changing its own code and its size has grown dramatically.

C. OLIGOMORPHISM

Oligomorphism is also an obfuscation way to change its structure through encryption. This confusing method has a certain number of different decryptors to achieve the variation of its own decryption function. Both polymorphic and oligomorphic techniques change the code in real-time each time it runs, but its semantics remain the same. Therefore, detecting its maliciousness through semantics is an effective way to deal with these two ways of obfuscation.

D. METAMORPHISM

This kind of obfuscation does not use encryption techniques but changes the code structure primarily by changing the assembly code of the program. Metamorphism is usually implemented by the following four ways: 1) dead instruction or garbage instruction insertion, which inserts some instructions in the normal assembly code that do not perform any operations, such as NOP; 2) instruction reordering, this way aims to reorder the original instructions, and then use the jump instruction to restore and maintain the original semantics, thereby generating a different code structure different from the original features; 3) register reallocation, also known as variable renaming, which replaces the program identifier such as registers, tags and constant names, etc., in this way the original code is changed, but the program behavior does not change; 4) instruction substitution, also known as equivalent instruction replacement, this kind of metamorphic method uses the equivalent instruction sequence dictionary for the instruction sequence replacement. Therefore, the evolution and protection of malware are like a game of cats and mice, aiming at and playing against each other. The game situation is illustrated in Fig 1.

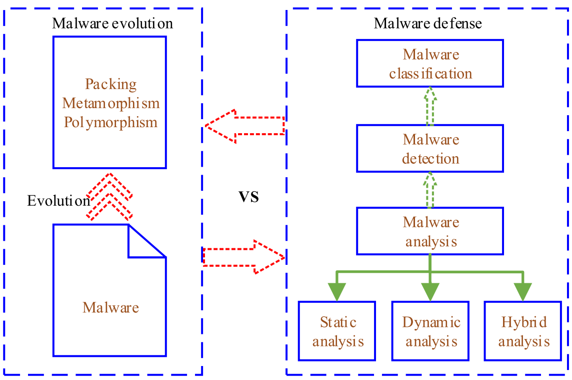


FIGURE 1. Malware offensive and defensive schematic chess.

VI. MALWARE DETECTION TECHNIQUES

Currently, the malware detection technology has evolved from traditional signature-based detection Moskovitch *et al.* [67], heuristic-based detection Bazrafshan *et al.* [12] to machine learning-based detection, and researchers have used machine-learning technologies to improve the level of automation and intelligence of malware detection. An intelligent malware detection process can generally be considered to consist of two phases: feature extraction and detection/classification. Therefore, malware detection completely depends on the process of feature extraction and detection/classification. Feature engineering is a key stage in automating machine learning. Among them, the realization of the acquisition feature is particularly critical. In order to facilitate researchers to easily find a research breakthrough that is suitable for their own research, we summarize the malware detection process based on machine learning, and then systematically introduce the detection methods based on different feature engineering. The implementation process and innovation points of each typical method are summarized, which is convenient for researchers to find the reference method that meets their requirements as quickly and efficiently as a dictionary.

A. BASIC PROCESS OF MALWARE DETECTION BASED ON MACHINE LEARNING

The machine learning-based malware detection process mainly includes two stages: training and detection, as shown in Fig 2. In the former stage, the analysts usually extract features from the samples set and then employ the features to train the automatic classifier; in the latter stage, the features will first be extracted from the samples to be detected, and then input into the trained classifier to obtain a decision result.

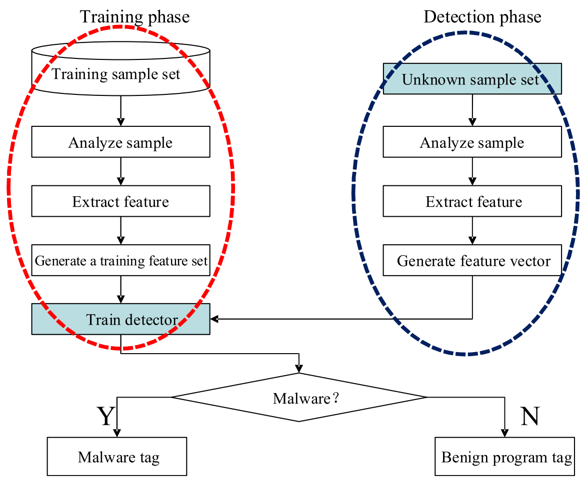


FIGURE 2. Basic process of malware detection based on machine learning.

B. COMMON MACHINE LEARNING CLASSIFICATION ALGORITHMS

Generally speaking, all typical machine learning algorithms can be applied into the field of malware detection, which include Naïve Bayes, Support Vector Machine, Decision Tree, Random Forest, K-Nearest Neighbor, Artificial Neural Network, and several boosting algorithms, such as GDBoost, AdaBoost and XGBoost.

In addition, because some researchers attempt to convert software programs into images and therefore convert malware

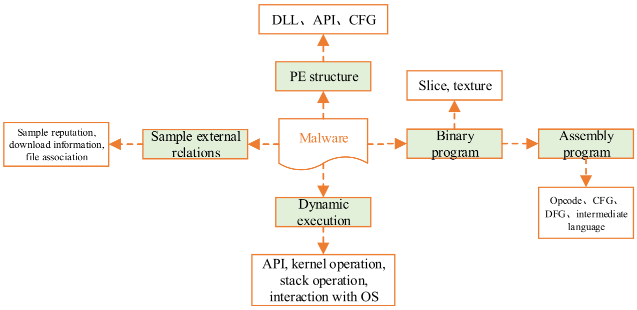


FIGURE 3. **Malware detection method system.**

detection problems into image classification problems, these researchers apply deep learning algorithms to the field of malware detection. Typical deep learning algorithms include Deep Belief Network, Convolution Neural Networks and Recurrent neural network Pouyanfar *et al.* [78]. At present, the above-mentioned machine learning algorithms have been widely and successfully applied in the field of network security [Egele *et al.* [23], Rieck [85]; Ye *et al.* [110]; Han *et al.* [34]. Because some relevant reviews have introduced the application of machine learning and deep learning algorithms in this filed in detail Ye *et al.* [110] Nguyen *et al.* [71], this article will not give description about this part anymore.

C. MALWARE DETECTION BASED ON DIFFERENT FEATURES

According to the machine learning-based malware detection process, since the classifier is mainly implemented by a general-purpose machine learning algorithm, the main factor affecting the effect of malware detection lies in the feature engineering. That is, selecting different features for analysis will determine different detection effects.

Based on a comprehensive analysis of the existing literature, malware detection methods can mainly be classified into the following categories based on the feature types, include: binary code-based detection (grayscale, slice, similarity), assembly instruction-based detection (opcode extracted from assembly program, stack in assembly instructions), PE structure-based detection, flow graph-based detection (CFG and DFG), dynamic link library-based detection, interaction behavior-based detection between program and operating system, file relationship-based detection, information entropy-based detection, hybrid feature-based detection, etc., as shown in Fig 3. For a person who starts research in this field, the first step is to choose what type of features to utilize to conduct malware detection research.

To facilitate the understanding and application of different types of features by researchers, we can divide these features into different levels and establish a hierarchical feature architecture, as illustrated in Fig 4.

The specific explanation and description of the feature level structure are as follows:

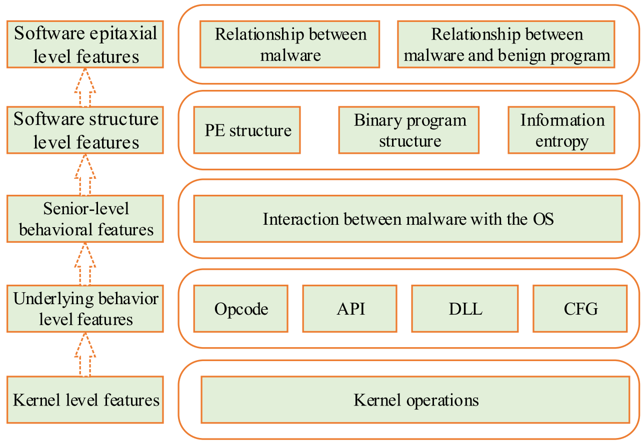


FIGURE 4. **Hierarchical architecture of different types of features.**

(1) *Kernel Level Features:* This level features mainly refer to the operation behaviors of the kernel object during the runtime of the program. The acquisition of this level features is more difficult, but it is more accurate for understanding the maliciousness of a program.

(2) *The Underlying Behavior Level Features:* This level of feature mainly refers to the behavioral characteristics of malware through assembly instructions (opcode), system call (API), dynamic link library (DLL) and control flow graph (CFG). These features are direct manifestations of program behaviors and are frequently applied to malware detection.

(3) *Senior-Level Behavior Features:* It mainly refers to the dynamic behaviors that the program interacts with the underlying OS resources, mainly including operations on the system files and the registry, and the network interaction behaviors Han *et al.* [33].

(4) *Software Structure Level Features:* This level focuses on the structural information of the software itself, including PE structure, binary code structure, and information entropy, which are a coarse-grained representation of software features.

(5) *Software Epitaxial Level Features:* This level mainly refers to the interrelationship between malware, and the relationship between malware and normal software. By mining the extension features of the software, we can also find its maliciousness.

1) PE STRUCTURE-BASED DETECTION

Because the PE structure is the standard format of the executable file under the Windows platform, the PE format can be employed to characterize the PE program, so some analysts explore to discover the clues of the maliciousness by mining the PE structure. The typical PE structure-based detection methods are summarized in Table 2. Compared with other methods, extracting features from the PE format is less complex, which is more suitable for beginners to quickly understand the basic process and principle of malware detection.

TABLE 2. Typical PE structure-based detection methods.

Method	Abstract & Contributions
Shafiq et al. 2009	This paper developed an extensible malware detection framework named PE-Miner that could automatically extract features from the PE format in real-time and detect unknown malware. PE-Miner mainly includes three parts: defining a computable structured feature set; preprocessing to remove redundancy; employing data-mining algorithm.
Treadwell et al. 2009	This paper designed a heuristic detection method for an obfuscated binary program that has been loaded into memory and not yet executed. The method performs a series of tests on the PE structures of a binary to find its packer or confusing trace and uses a risk scoring mechanism to evaluate its maliciousness.
Bai et al. 2014	This paper extracts 197 features from the PE format and employs feature selection techniques to decrease the volume and complexity of the feature set.

TABLE 3. Dynamic link library-based detection.

Method	Abstract & Contributions
Narouei et al. 2015	This paper proposed to extract DLL dependency trees from the PE structure statically, without actually executing the program. The process is as follows: 1. Extracting the dependency tree from the PE file; 2. Converting the extracted dependency tree into a string encoding format; 3. Extracting the closed frequent tree and constructing the feature vector; 4. Using the random forest classifier to perform cross-validation.

The detection method based on PE structure is simple and effective and is most suitable for beginners. However, because the PE structure is a standard format, the characteristics difference between normal software and malware may not be apparent, and the detection accuracy is difficult to guarantee. In addition, PE structural features are difficult to effectively characterize a program's semantic information.

2) DYNAMIC LINK LIBRARY-BASED DETECTION

Because the PE program needs to call the DLL during the actual running process, the calling relationship between the PE file and the DLL, and the association relationship between the DLLs called during the PE file execution process are also visual representations of the program behavior characteristics. The detection of malicious behavior can also be achieved by evaluating the relationship between the PE file and the DLL. The typical DLL-based detection methods are summarized in Table 3.

The dynamic link library-based detection method is simple to implement and high-efficiency. However, it is susceptible to obfuscation and difficult to obtain accurate and effective semantic information.

3) INFORMATION ENTROPY-BASED DETECTION

Entropy is an effective indicator for measuring information uncertainty. When a normal program is implanted with a malicious code segment, its entropy value changes before and after implantation. In addition, when a packed program is unpacked, its entropy value also changes. Based on the connotation of entropy and the content changes that may be involved in the malware detection process, some researchers have also proposed methods for malware detection based on entropy calculation. The typical information entropy-based detection methods are summarized in Table 4.

The information entropy-based detection is easier to implement and more efficient. However, the semantic information obtained in this way is limited and is susceptible to obfuscation.

4) BINARY CODE-BASED DETECTION

In the malware detection process, because of the difficulty of obtaining the source code of a program, its binary code is often analyzed directly. Researchers can extract byte n-grams from binary code; or convert binary to decimal to generate texture maps for image-based detection; or extract binary code slice for code slice matching; or generate binary code pattern for code pattern matching. The typical binary code-based detection methods are summarized in Table 5.

The binary code-based detection can directly perform static analysis on the binary code of one program and does not require operations such as disassembly, so the efficiency is high. However, binary code is poorly readable, it is difficult to understand program behavior characteristics, and is susceptible to obfuscation.

5) OPCODE BASED DETECTION

A common way of malware research is to perform the reverse engineering analysis of a binary program to obtain its assembly instructions. Compared with binary code, assembly instructions can more intuitively reflect the behavior of a program, which is more conducive to understanding the intent of the malware. Therefore, the detection method based on assembly instructions is the most common way to carry out malware detection. Even in the face of the latest ransomware, this analysis method also works properly Bolton *et al.*[6]; Hanqi *et al.* [114]. The typical assembly instruction-based detection methods are summarized in Table 6.

TABLE 4. Information entropy-based detection.

Method	Abstract & Contributions
Bat-Erdene et al. 2016	1. Describe a method for identifying the packed algorithm before the execution of the packed malware payload. Because malware is often encrypted and compressed, it is difficult to identify malicious parts of the program before unpacking. Therefore, the author first extracts the entropy pattern by unpacking each packed program, then detects and classifies each packing algorithm, and extracts unique symbol patterns by using entropy variables. These entropy variable patterns can be used to enhance packing algorithm classification. 2. By discovering the original entry point, the proposed entropy analysis method can be used to unpack the known and unknown packers without any unpacking tool. Therefore, the method can effectively deal with any type of packer and can recognize the packer algorithm even if it changes. 3. The method can identify the packing algorithm of the unknown packer without relying on any known features of the packer algorithm.
Radkani et al. 2018	1. Propose a novel detection method based on the similarity of entropy metrics. the entropy value is calculated based on the Opcode frequency; 2. Use entropy-based program difference metrics to represent the average difference between different malware variants of the same family, thereby representing the degree of deformation between morphing malicious programs.

TABLE 5. Binary code-based detection.

Method	Abstract & Contributions
Nataraj et al. 2011	1. Apply texture analysis methods to malware analysis; 2. Binary texture analysis methods can deal with packed malicious programs.
Fu 2016	Generate slices from malware programs and analyze the sensitivity of malware to the execution environment, so as to investigate ways to combat evasive malware.
Escalada et al. 2017	1. Automatically extract code patterns from native code; 2. The parameters are easy to configure and can be applied to different scenarios. 3. Process in parallel.
Cui et al. 2018	1. Convert binary malware to images to convert malware detection to image classification issues; 2. Detection of malware variants based on convolutional neural networks; 3. Employ the bat algorithm to cope with the data set imbalance problem.

The assembly instruction can better demonstrate the program semantics, and it is better than the binary code in the comprehensibility, but it is susceptible to the packing and metamorphism maneuvers.

6) API BASED DETECTION

Generally, malware needs to call API to perform malicious actions, so API calling information can be employed to characterize malware, and are most widely used for malware detection. The static methods for obtaining the system call API include: obtaining the system call by analyzing the program syntax; extracting the system call or function relationship by using the program state machine; obtaining the system call information by analyzing the PE file header, because the PE file header contains all the system calls information; the dynamic method of getting the system call information is to actually run the program and capture the behavior traces during the runtime to get the API sequence. The typical API-based detection methods are summarized in Table 7. The method of detecting malware based on API can be divided into multiple levels: the basic way is to directly use the API sequence, such as count the frequency of occurrence of API or n-grams as the feature value; or cut the basic API sequence into API sequence slices, and build more abstract semantic representation.

The API sequence usually is considered to be the best choice for characterizing a program, so the API based detection technique is most widely applied for malware detection. But the API extraction process requires a certain amount of effort, and only using API features is vulnerable to imitative malware.

7) CFG BASED DETECTION

Although Opcode and API can profile the program, they do not really indicate the execution intention of the program. Therefore, some researchers build control flow graphs based on the Opcode and API sequences to reflect the true execution intent of the program and apply graph matching to carry out the detection process. The typical control flow graph-based detection methods are summarized in Table 8.

The detection method based on CFG can more vividly represent the program behavior characteristics, but the implementation process is more complicated and difficult.

TABLE 6. Assembly instruction-based detection.

Method	Abstract & Contributions
Zhang et al. 2010	The paper proposed a novel detection model based on a negative selection algorithm with a penalty factor. The authors build a malware opcode instruction library based on the opcode frequency and the file frequency containing the opcode instructions and then filter the features by negative selection with a penalty factor.
Santos et al. 2011	This paper proposed to detect unknown malware families by only using a single class of samples. The method uses only one tagged sample set to construct a machine learning classifier by checking the appearance frequencies of opcodes to construct an opcode sequence.
Okane et al. 2013	This paper proposed to find an optimal opcode subset that can represent malware and to determine how long a program needs to be dynamically monitored to accurately classify it. This article represents the program with an Opcode density histogram that is dynamically acquired during different runtime phases. Based on a small number of Opcodes, malware can be detected at different runtimes.
Santos et al. 2013	This paper proposed to construct the feature vector based on the frequency of the Opcode sequence appearing in the malicious sample and the normal sample, and calculate the frequency difference of Opcode in the malicious sample and the normal sample. The method can detect unknown malware.
Zhao et al. 2013	1. Extract the opcode sequence according to the software control flow structure, and use the single-step feature filtering method to reduce the number of features; 2. Use machine learning algorithms to find classification rules between malware and benign software.
Ding et al. 2014	1. Extract the action code behavior of an executable program based on the control flow. The extracted Opcode sequence can fully represent the behavior characteristics of an executable file. 2. Compare the control flow and text-based Opcode sequence generation methods. The analysis and experimental results prove the better decision performance of control flow based detection than the text-based method.
Alam et al. 2015	1. Propose the malware analysis intermediate language to reduce the problem that assembly Opcode distribution is susceptible to the compiler, operating system, and optimizer; 2. Construct feature vectors based on annotation control flow graphs to improve the efficiency of malware detection.
Alexander et al. 2017	1. Generate a call graph to represent the program feature based on a static trace; 2. Evaluate the distance between the graphs based on a simulated annealing algorithm.
Carlin et al. 2017	1. Create a large malware run trace Opcode data set for free use; 2. Prove that Opcode sequences acquired during runtime can be better applied to malware classification tasks.
Raphe et al. 2017	1. Employing a method based on decision feature variance and a Markov coverage model to detect obfuscated unknown malware; 2. Generate two meta-Opcode spaces from the training set, including 25 features, and evaluate the performance of the machine learning model with varying feature lengths. 3. Verify the new samples using the meta-feature space and optimization model. The degree of deformation of the data set was verified through comprehensive experiments.
Khalilian et al. 2018	1. A frequency-based code pattern matching method is designed for deformed malware detection. The shared subgraph of the malware family is extracted from the operation code map by mining the frequent subgraph. 2. Conduct comparative analysis with the full-time subgraph and the maximum frequent subgraph method.
Zhang et al. 2019	1. The first-time attempt using the static method to detect the ransomware; 2. Based on the TF-IDF method, select the more decisive N-grams as the feature vector.

8) KERNEL OPERATION-BASED DETECTION

Malware need inevitably take relevant actions on the kernel during its actual running process. By monitoring the changes of the kernel during the runtime, the malicious tendency of the program can also be detected. The typical detection methods based on kernel operation behavior are summarized in Table 9.

The kernel-based detection method is a relatively low-level detection method that can accurately profile the malicious software. However, it is difficult to implement, and this detection process will encounter challenges when detecting rootkit-type malware.

9) INTERACTION BASED DETECTION BETWEEN THE PROGRAM WITH OS

When analyzing malware, in addition to paying attention to the behavior of the program itself, the interaction between the program and the underlying OS resources is also an important aspect worthy of analysis. These malicious behaviors can be assessed by evaluating these external behaviors. The typical methods in this area are summarized in Table 10.

This kind of method is system-centric and can intuitively reflect the behavioral characteristics of the program. However, this detection method usually

TABLE 7. API based detection.

Method	Abstract & Contributions
Ye et al. 2008	The paper proposed an integrated intelligent malware detection system, which consisted of three components: PE parser, rule generator, and data-mining classifier. PE parser extracts API calls from the PE file, then the system applies the association-based classification method to improve system effectiveness and efficiency. The proposed system has been implanted into Kingsoft anti-virus software.
Ye et al. 2010	1. Conduct in-depth analysis of the grey list and analyze its characteristics, prove that the gray list data is large and uneven generally, with high complexity and overlap; 2. Apply multiple post-processing techniques to construct an accurate and effective association classifier, and design a new hierarchical association classifier for unbalanced grey list prediction; 3. The proposed hierarchical association classifier has been integrated into the Kingsoft Anti-Virus system.
Liu et al. 2011	1. A two-layer malware detection framework is designed to combine the underlying classifier and the upper classifier to build a hybrid classifier. The Type-Function structure can effectively describe malware. 2. Verify the factors that influence detection including the imbalance in the number of benign software and malware; differences between different malware; and multiple correlations between malware.
Elhadi et al. 2012	This paper proposed a new malware analysis framework, which is based on API calls and the operating system resources used by the API to construct a hybrid call graph (the operating system resources are nodes, and the edges represent the reference relationships between nodes. Two attributes are included in the graph: API call and operating system resources. The graph tag is the API call itself or calling the operating system resource.) The implementation process is as follows: first run the PE file, extract the API call sequence; then merge the API call with the operating system resources to build the call graph; finally, calculate the API call and operating system resource allocation overhead matrix, and calculate the call graph distance to realize malware detection.
Saxe et al. 2012	1. Discover system call sequence relationships shared by large malware data sets and interactively visualize them; 2. Provide an intuitive understanding of the overall structural similarity and behavioral distribution of malware families for researchers.
Ding et al. 2013	1. Statically extract the API sequence from the sample and mine the association rules between the APIs; 2. Streamline and optimize the association rules to obtain the optimal rule set for characterizing the programs; 3. Implement malware detection based on multiple association rules.
Lu et al. 2013	1. Apply the anti-aliased iterative sequence ordering method to extract shared system calls between various family samples, which can effectively solve the rearrangement problem between independent system calls due to malware deformation. 2. Design a new behavioral representation, consisting of explicit handle dependencies and probabilistic ordering dependencies between system calls, which are simpler and more effective than behavioral graphs; 3. Scalable to process a large volume of malware efficiently, multiple detectors work together to fully capture variants of malware families distributed across multiple network domains.
Elhadi et al. 2014	1. Integrate the API call with the operating system resources to build an API call graph, which combines the static features and dynamic behavior information; 2. Use the approximation algorithm to perform graph similarity and graph edit distance calculation, and then find the largest shared subgraph in the integrated API call graph.
Salehi et al. 2014	1. Assume and verify that samples with similar behaviors will call similar APIs and variables to build the feature set. 2. Construct a global frequency table that reflects how many samples each feature exists in. The implementation process is as follows: firstly, the API call sequence is monitored and recorded; then, construct the global frequency table, and configure a threshold to select the ranking top-N API as the feature vector, and the frequency of the API appears in each sample as the eigenvalues to generate the feature matrix; finally, employ data-mining methods to realize automated detection.
Alazab 2015	1. Contour a systematic overview of the evolution of malware, find that malware variants are basically "new bottled old wine"; 2. Make a portrait of the program based on the sequence of system calls.
Ki et al. 2015	The paper proposed a DNA sequence alignment algorithm that extracts common API call sequence patterns for malicious functions from different categories of malware. The authors found that even if they belong to different categories, malware usually includes some specific malicious functions. By examining the existence of these specific functions or API call sequences in unknown samples, the method can detect known malware types and unknown malware. The proposed method has good applicability and can be deployed on any type of device.
Kirat et al.	1. Designed MalGene, a system that automatically extracts evasive features from evasive malware. The system uses

TABLE 7. *(Continued.)* **API based detection.**

2015	data mining and data stream analysis technologies to automate the feature extraction process and can be applied to large-scale sample sets. 2. A new bioinformatics-inspired approach was designed to find evasive behavior through system call sequence alignment. This method can realize deduplication, differential pruning, and can process branch sequences. 3. The method was evaluated on 2810 evasive samples, which automatically extracted the evasive features and grouped them into 78 similar evasive-technique types.
Naval et al. 2015	1. Use semantic-related paths to characterize program behavior to implement malware detection; 2. Use the asymptotically equal-partition attribute index to quantitatively analyze the semantic-related path to construct a new feature space. The feature space contains non-string features, which can combat system-call injection attacks.
Das et al. 2016	1. The paper proposed a hardware enhancement architecture named GuardOL, which employs system calls and their parameters to simulate program behavior and can detect attacks that leave a little bit of traces in system calls; 2. GuardOL can detect kernel-level malware during runtime.
Hellal et al. 2016	1. Propose a small and frequent subgraph mining algorithm, explore the compressed call graph, generate a minimum decision and behavior pattern for the malware family, generate only a limited number of features with the advantages of lower consumption of memory space and running time; 2. The method defines semantic features rather than grammatical features and can detect various variants with the same malicious behaviors.
Huda et al. 2016	This paper designed a hybrid detection framework with API call statistics as a feature, mixing SVM encapsulation and redundancy filtering heuristics. The novelty of the hybrid framework is that the filter's sorting score is injected during the package selection process, and the properties of the wrapper and filter are fused with the API call statistics to detect malware based on the nature of the infectious activity rather than the characteristics. Knowledge about the essential characteristics of malicious behavior is determined by the statistics of API calls, and knowledge is injected as a filter score into the backward elimination process of the wrapper to find the most important API. The main idea of this method is to find more important APIs as features for classification.
Lee et al. 2016	This paper introduced some measures to improve the classification reliability using local clustering coefficients, choose and manage the first malware for each family to efficiently classify large-scale malware. The proposed method can be used to analyze new malware, predict the trend of malware families and the same attacker, and automatically identify interesting malware.
Bidoki et al. 2017	1. Propose a model-independent method to realize multi-process detection. 2. Learn the execution strategy using an enhanced learning algorithm; 3. Use bioinformatics algorithms to strengthen the granularity of system call traces.
Ming et al. 2017	1. Propose the system call cut segment equivalence test method, cut out the corresponding code segment based on API call, and then use symbol execution to test equivalence; 2. Can detect the similarity or difference between multiple basic blocks, which can overcome the limitation of the block center to a certain extent, which is more accurate than dynamic analysis; 3. Due to indirect memory access and forgery control/data dependence, redundant instructions can contaminate the slice output, so it is easy to be spoofed and complicated to confuse binary program slices.
Salehi et al. 2017	This paper designed a novel strategy based on the new feature generation method to realize dynamic malicious code feature selection. The feature construction process is to first run the binary code in the controlled environment, and record the parameters and return values of each API call; then, construct the feature set based on the API call and the parameters and return values recorded during the running; finally, employ the Fisher score to ensure that the selected feature set has better decision performance. It is proved that features generated using the API and its parameters and return values can better represent the true purpose of the program.
Ding et al. 2018	This paper designed a common behavior graph for each malware family. The authors represent malware behaviors as a dependency graph based on API. In order to find out the dependencies between APIs, the authors use dynamic stain analysis technology to mark the parameters of API with the stain label and build the API dependency graph by tracking the spread of the stain data. Based on the dependency graph of the malicious sample, the author designed an algorithm to extract the general behavior graph, which is used to represent the behavior characteristics of a malware family. Finally, the malware is detected based on the graph matching algorithm of the maximum weight subgraph.
Lee et al. 2018	1. Apply local cluster coefficients to calculate the similarity between samples; 2. It can predict the trend of malware and the attacker's tendency for future operations.
Tajoddin et al. 2018	1. Introduce a polymorphic malware detection method based on program behavior-aware hidden Markov model, the Hidden Markov model topology is designed based on program states. 2. Employ malware operation sequences to dynamically detect polymorphic malware and significantly reduce complexity. 3. Only uses 26 operations and not increase with the volume of malicious samples, so it has better scalability.

TABLE 8. CFG based detection.

Method	Abstract & Contributions
Lee et al. 2010	The paper designs a code diagram to represent program semantics, which can effectively reduce the number of malware features, without worrying about the increase of features caused by the proliferation of malware. In addition, this method requires less space and can effectively reduce processing overhead.
Eskandari et al. 2012	1. Preprocess to eliminate the redundant assembly code, leaving only the useful instructions to build CFG. In addition, all APIs are saved to the API container with a unique identifier. In this way, the edges of the CFG can be marked with the API-ID, and finally, the API-CFG structure is generated. 2. Convert the API-CFG into a feature vector and use a feature selection algorithm to generate a feature subset to reduce its size.
Cesare et al. 2013	1. Use string-based features to display control flow graphs and quickly classify variants based on the similarity; 2. Design two string generation algorithms to achieve accurate and approximate matching of flow graphs. Accurate matching converts a graph variant to be a string to heuristically identify a deformed flow graph; the approximate matching uses a decompiled structured flow graph as a string feature. The authors assume that the similar stream graphs have similar strings and therefore measure string similarity based on the edit distance.
Cesare et al. 2014	1. Near real-time similarity search based on CFG set; 2. Similarity comparison using Levenshtein distance, NCD algorithm, and BLAST algorithm; 3. Constructing feature vector approximate atlas using fixed-size k subgraph; 4. The polynomial-time algorithm generates a q-gram feature vector of the decompiled CFG; 5. A distance metric method between two graph sets based on the minimum matching distance.
Nguyen et al. 2018	1. Designed an enhanced CFG called lazy-binding CFG that reflects the dynamic behavior of the program; 2. Convert the control flow graph to an adjacency matrix to implement a graphical representation of the control flow graph.

TABLE 9. Kernel operation-based detection.

Method	Abstract & Contributions
Song et al. 2012	1. Use the pushdown model to simulate the program; 2. Use a new logic called SCTPL to indicate malicious behavior; 3. Convert malware detection issues to model validation issues.
Shahzad et al. 2013	1. Propose a dynamic footprint detection framework based on the genetic footprint concept, the footprint includes a set of selected parameters including the kernel parameters reserved for each running process within the task structure, thereby defining the semantics and behavior of a running process; 2. The maliciousness of one program can be detected within 100ms after the malicious activity running, with an accuracy rate of 96%.
Rhee et al. 2014	(1) Detect kernel object hidden attacks, which hide data objects by manipulating pointers to them. (2) Design a new malware feature based on the unique mode of kernel data access that occurs during the attack process, which can effectively complement the code-based malware features; (3) Detect malware attacks by extracting and matching data access patterns, and detect variants with similar data access patterns.
Ghiasi et al. 2015	1. Discover maliciousness by monitoring the value of the kernel registers during runtime; 2. For a large number of sample records, the register value is large, and the feature space can be effectively reduced by selecting a register value with a higher frequency as a feature.
Burnap et al. 2018	1. Utilize behavior footprint information such as continuous machine activity to detect malware and discover fuzzy activity boundaries in unknown attacks. 2. Can find over-fitting phenomena during cross-validation; 3. Use self-organizing feature maps to process machine activity to identify fuzzy boundaries between machine activity and the categories.
Han et al. 2018	The existing malware detection method based on unlimited registration machine is difficult to detect targeted malware, and a new method based on the owner's unlimited registration machine is designed. The experiment proves that the new model can detect targeted attack malware.

runs malware in a virtual machine environment, which affects the detection effect when encountering evasive malware.

10) FILE RELATIONSHIP-BASED DETECTION

There are also some researchers who focus on the inter-relationship between malicious samples and explore the

TABLE 10. Interaction based detection between the program with OS.

Method	Abstract & Contributions
Lanzi et al. 2010	1. Collect a large number of system calls triggered by normal programs and analyze the differences between system calls. It is verified that the program-centric detector is based on the system call information, and the simple method based on the system call sequence may significantly improve the false alarm rate, which is not suitable for practical application; 2. A new system-centric malware detection method is designed, which is based on a model of access activities between normal programs and operating system resources.
Chandramohan et al. 2013	1. A malware behavior modeling method is designed to capture malicious interactions between malware and operating system key resources in a scalable manner. 2. The feature space dimension generated by the method is fixed, and the detection process does not increase with the increase of the number of malware samples, and the calculation time and memory overhead of the feature extraction and detection process are greatly reduced.
Fattori et al. 2015	1. A system-centric model is proposed to focus on the interaction between the program and the operating system; 2. Extend AccessMiner to implement the detector as a custom management program that can combat complex attacks; 3. Can be deployed in a virtual environment.
Mohaisen et al. 2015	1. Implement a fully automated malware analysis, classification, and clustering system that collects rich, low-grain behavior-based features; 2. The system has good scalability.
Mao et al. 2017	1. In-depth analysis of system-wide access behavior with a networked model; 2. An assessment of the importance of network structure based on integrity; 3. Malware detection based on importance metrics.
Stiborek et al. 2018	1. Represent malware features based on interactions between programs and system resources, using vocabulary-based methods derived from multi-instance learning domains and similarity measures for different resource types to reflect unique attributes of resources; 2. Design a fast approximate Louvain clustering approach to automatically define vocabulary to ensure that multi-instance methods can be extended to the domain of the large-scale data.

TABLE 11. File relationship-based detection.

Method	Abstract & Contributions
Tamersoy et al. 2014	1. Describe malware detection problems as large-scale graph mining and reasoning problems. The goal is to find out the relationship between unknown files and other files and determine their maliciousness through the association with known normal or malicious files. 2. Introduce the Aesop algorithm, which uses the position-sensitive hash to calculate the file similarity to construct the file relationship diagram, based on the maliciousness of the propagation reasoning file.
Ni et al. 2016	1. Improve the active learning method to detect malicious samples using semi-supervised learning model and graph mining algorithm; 2. Use the relationship between file content information and file samples, and apply graph mining algorithm to classify malware. Using the co-occurrence relationship between file samples to measure the similarity between each pair of samples, thereby identifying unknown samples; 3. selecting k nearest neighbors of each sample to construct a relationship diagram between files, To derive the probability that each file is malicious or normal, and design three file relationship features to sample representative files to query tags from experts; 4. Apply the tag propagation algorithm to spread tag information from tagged files to untagged files and use active learning methods in batch model and maximum network gain to improve efficiency and performance of the model.

malicious behavior intentions by constructing a relationship network between samples and conducting the quantitative evaluation. The typical methods are summarized in Table 11.

The file relationship-based detection method is an effective complement for malware detection that focuses on the program itself but may be limited by the way in which the data source is obtained. Usually, it is difficult for ordinary

TABLE 12. Hybrid features-based detection.

Method	Abstract & Contributions
Islam et al. 2013	This paper first introduced a classification method that combines static features and dynamic features to form a feature set. This approach improves on traditional methods based on unilateral features and reduces detection time. The static features include: function length-frequency vector, printable string information vector; the dynamic features include: statistical API occurrences. For the first time, static features and dynamic features are integrated into a simple feature vector.
Liu et al. 2013	1. Design an integrated single-class malware detection learning framework to extract mixed features from multiple semantic layers to enhance the understanding of the program; 2. The cost-sensitive twin classifier is designed and the generalization ability is enhanced by using the random subspace integration method.
Sheen et al. 2013	1. Analysis of malicious documents to extract decision features; 2. Parallel integration of classifiers to reduce the optimal subset for malware detection; 3. Harmony search algorithm to obtain effective streamlining.
Han et al. 2019a	1. Realize comprehensive profiling of malware from three perspectives: basic structure, underlying behavior, and high-level behavior, enrich feature space and improve detection accuracy; 2. Assess the maliciousness of three aspects: basic structure, underlying behavior, and high-level behavior by evaluating the malicious influence of the three aspects.
Han et al. 2019b	1. The semantic mapping model is designed to correlate the dynamic and static features of malware, which solves the shortcomings of the existing literature that ignore the relationship between the dynamic and static features. 2. By defining the behavior type, the reasonable interpretation of malware maliciousness is realized, which provides an explanatory framework for researchers to understand malware.

researchers to have such conditions without the necessary commercial support.

11) HYBRID FEATURE-BASED DETECTION

As the malware mechanism becomes more complex, the detection effect of relying solely on static features or dynamic features is difficult to meet the detection requirements. Therefore, some researchers have tried to integrate static and dynamic features to construct the hybrid feature set to realize hybrid detection. The typical research methods are summarized in Table 12.

The detection method based on the hybrid feature can realize the comprehensive depiction of the program, and the detection effect is the best. However, the workload required for the feature acquisition process is also relatively heavy.

12) COMPARISON BETWEEN DETECTION METHODS BASED ON DIFFERENT FEATURES

This section compares the advantages and disadvantages of malware detection methods based on different features to facilitate researchers to choose the suitable method. The comparison results are shown in Table 13.

D. MALWARE DETECTION BASED ON DIFFERENT ANALYSIS ENVIRONMENTS

According to the different principles of static analysis and dynamic analysis, the static analysis method usually analyzes the malware itself or its derivatives on the host, and the dynamic analysis needs to select different operating environments according to different application needs. In order to prevent malware from causing damages to the host, the virtual execution environment is usually suitable for running software. However, some malware can first detect whether the running environment is a virtual environment before running, and will not perform malicious operations once discovering the clue of a virtual environment. In response to this evasive malware, some researchers have proposed to build a bare-metal environment to stimulate the unobtrusive manifestation of malware, by comparing the behavior of the software in the bare-metal environment to realize detection.

1) VIRTUAL MACHINE-BASED DETECTION

In the process of dynamic analysis, virtual machines are often used to build virtual execution environments to run malware actually, capture malware footprints, and prevent malware from causing damages to the host system. Therefore, building different types of virtual machines to obtain different granularity and different types of behavioral characteristics information is also a common method for researchers to analyze malware. The typical methods for performing malware detection based on virtual machines are summarized in Table 14.

2) BARE-METAL BASED DETECTION

Current dynamic analysis methods generally use a virtual environment to run malware to prevent malware from damaging the underlying operating system. However, some malware will detect the characteristics of the environment before running, and if it is found to be in a virtual environment, it will stop running or not perform malicious behavior,

TABLE 13. Comparison between detection methods based on different features.

Method	Pros	Cons
PE structure-based detection	Because the PE structure is the standard format for Windows system executables, it is the easiest and most effective method to perform detection based on PE structural features. This method is best suitable for beginners.	Because the PE structure is a standard format, the difference in features between normal programs and malicious programs may not be obvious, and the detection accuracy is difficult to guarantee. In addition, it is difficult to understand the semantic information of a program only by analyzing the structural characteristics of PE.
Dynamic link library-based detection	Through static analysis, the DLL call information of the program is extracted from the PE structure, and the behavioral characteristics of the program can be understood to some extent. The method is simple to implement and high-efficiency.	Because extracting DLL call information from the PE structure is a static process, it is susceptible to obfuscation. And some malware may hide a lot of valuable DLL call information in static analysis mode. Therefore, this detection method is difficult to obtain accurate and effective semantic information.
Information entropy-based detection	The information entropy-based detection method discovers malware by comparing changes in the original program before and after the implantation of malicious code. This method is more like a black box test, the implementation process is easier and more efficient.	Because the details of the internals of the program and its characteristics cannot be obtained, the program semantic information obtained in this way is limited. Moreover, this type of detection is susceptible to obfuscation and may be misled easily.
Binary code-based detection	This kind of detection conducts a static analysis of the program's binary code directly, without the need for disassembly and other operations, can achieve high efficiency.	Binary code is poorly readable, it is not appropriate to understand program semantics, and it is difficult to understand the program's behavioural characteristics. In addition, during extracting code patterns from binary code, the extraction process may be affected by code obfuscation.
Opcode based detection	By generating assembly instructions for binary code and analyzing the behavioral characteristics of assembly instructions, the program semantics can be better understood in this kind of detection, and the comprehensibility is better than the binary code based detection method.	The disassembly process is susceptible to the way the program is packed and deformed. Up to now, there are no very effective shelling tools. In addition, the deformation method produces a large amount of obfuscated assembly instructions.
API based detection	API-based detection can use static or dynamic analysis to extract API sequences, and dynamic and static API sequences can be combined for detection. In addition, the API sequence is the most intuitive to reflect the behavior of the program, so it is the most widely used technique in the field of malware detection.	The process of statically extracting API sequences is susceptible to the confusion of the shelling and irrelevant APIs; the way to dynamically extract API sequences is more time consuming and will also be affected by the noise API. In addition, detection only based on API features is vulnerable to obfuscated malware.
CFG based detection	Control flow graph-based detection is usually based on the assembly instructions or API calls to build a control flow graph, and then convert the flow graph into understandable text information, or directly based on the graph to detect. This method can understand the behavior of the program more vividly than the detection method based on assembly instructions or API.	Because the flow graph is built on the basis of assembly instructions or API calls, both static analysis and dynamic analysis can be used. However, the implementation process based on flow graph detection is more complicated and more difficult. Therefore, an effective technique is needed to ensure the efficiency of the detection process.
Kernel operation based detection	Detecting malware based on the operating behavior of the kernel during the running process is a biased detection method. Compared with other methods, this kind of detection can accurately and objectively reflect the malicious behavior of the software.	Because it is needed to monitor the behavior of the kernel, it is a relatively low-level operation. Although it can provide researchers with deeper semantic information, it is more difficult to implement. In addition, this detection method encounters challenges when detecting rootkit-type malware.
Interaction based detection	The method is system-centric and analyzes the behavior characteristics of the program by capturing the interaction behavior between the program and the	This detection method typically runs malware in a virtual machine environment, but malware may not run in a virtual machine environment or generate unrelated behavior; it may

TABLE 13. *(Continued.)* **Comparison between detection methods based on different features.**

between the program with OS	operating system. These interaction behaviors also systematically reflect the behavior characteristics of the program, which is an effective detection method.	also produce misleading features that contaminate useful behavioral characteristics.
File relationship-based detection	This method detects the maliciousness of the program by observing the interconnection between normal and malicious programs, as well as among malicious programs. This method is an effective complement to the detection method that focuses on the analysis program itself, helping researchers to build awareness of the overall environment.	Because of the interlinkages between programs, it is difficult for general researchers to obtain the data needed to conduct such research. The conditions required for this method of detection are difficult to achieve.
Hybrid features based detection	By combining various features to characterize the program, a comprehensive portrait of the program can be realized, and the influence of the confusion and deception mode can be overcome to some extent, which helps to understand the behavior of the program comprehensively, accurately and systematically.	Because of the need to acquire multiple kinds of features, the implementation process is complex and time-consuming, and the efficiency is difficult to guarantee.

TABLE 14. **Virtual machine-based detection.**

Method	Abstract & Contributions
Dinaburg et al. 2008	1. Design a program execution and analysis framework named Ether for transparent malware analysis; 2. Ether is an externally transparent malware analyzer that uses hardware virtualization for analysis, providing both fine-grained (single instruction) and coarse-grained (system call) information.
Nguyen et al. 2009	1. Propose a more transparent and secure malware analysis framework with target-driven virtual machine and hardware virtualization scalability; 2. Implement a prototype system that extracts useful data that cannot be extracted by common virtual machine detection techniques; 3. The system is open-source and free. In addition to being used for malware analysis, the system can also be used for auditing, logging & playback, and many other purposes.
Jiang et al. 2010	1. Design an "out-of-the-box" mechanism to build rich semantic views and tackle semantic isolation problems; 2. Implement a detection framework based on semantic view comparison, which can be deployed in an "out-of-the-box" way to prevent tampering.
Yan et al. 2012	1. A new method of integrated hardware virtualization and dynamic binary conversion is designed to achieve transparent and efficient fine-grained analysis. 2. A prototype system V2E is designed and implemented. In the KVM environment, the recording component records the malware execution process in a transparent manner, and the playback component is implemented under the TEMU, largely by modifying the dynamic binary conversion logic of the TEMU. The existing TEMU plug-in can achieve transparency and higher analysis capabilities with minor modifications.
Roberts et al. 2013	This paper proposed a real-time system monitoring and analysis framework that uses virtual machine introspection (VMI) mechanisms to monitor the system without having to install any agents locally, in case the monitoring agents installed within the analysis system become the possible targets for malware. The proposed VMI framework is lightweight and can reconstruct the semantic gap through offline forensic analysis.
Ajay et al. 2018	1. Design and implement a continuous, real-time, virtual machine-based, client-assisted multi-level malware detection system that periodically checks the status of the active guest operating system; 2. Implement an intelligent cross-view analyzer prototype system, embedded in the actual system, intelligently collects status information to detect hidden, dead and suspicious processes, and uses time interval threshold techniques to predict early symptoms of malware execution; 3. Perform online malware detection and offline malware classification from a semantic perspective; 4. Firstly apply machine learning technology on the virtual machine for runtime detection of unknown malware from the VMI perspective.

thereby spoofing malware detection. To ensure that malware unleashes malicious manifestations, some researchers propose to build a real-world operating environment for malware to activate malware to fully perform its malicious behaviors. This real execution environment built on real hardware platforms is often referred to as bare-metal. Currently, the typical

TABLE 15. Bare-metal based detection.

Method	Abstract & Contributions
Kirat et al. 2011	1. BareBox can effectively visualize the true behavior of malware by actually executing malware in a local hardware environment; 2. BareBox can quickly restore the operating system running on the commercial hardware platform to the previously saved state, because it does not need to restart, so the recovery speed is faster than the existing solution, the recovery time is less than 4 seconds; 3. BareBox can resume the unstable state of the system and quickly replicate the same system state for repetitive security experiments in a bare-metal environment.
Kirat et al. 2014	1. Introduce BareCloud, a system for automated detection of evasive malware. The system executes the malware on a transparent bare metal system and any emulation-based and virtualization-based subsystems without any embedded monitoring components; 2. Propose a method for comparing malware behavioral portraits based on hierarchical similarity. This method can detect evasive behaviors due to the previous method based on set interaction.

TABLE 16. Comparison between methods based on different analysis environments.

Method	Pros	Cons
Virtual machine-based detection	By actually running malware in a virtual machine, the analysts can observe the true behavior of the malware and prevent the malware from directly causing damages to the host.	Some evasive malware can detect the virtual environment in which it is executed, so that it does not run, or does not actually perform malicious operations, thereby avoiding detection.
Bare-metal based detection	Because it runs in the actual hardware and software environment, malware can show real malicious behaviors and the analysts can observe the true behaviors of malware.	How to quickly and efficiently restore the original state after each run of malware is a problem that this kind of method needs to address. How to achieve fast recovery without restart is the key to ensuring the practicality of this detection method.

bare metal environment based detection methods are summarized in Table 15.

3) COMPARISON BETWEEN METHODS BASED ON DIFFERENT ANALYSIS ENVIRONMENTS

This section compares malware detection methods based on different analysis environments to facilitate researchers to choose the suitable method. The comparison results are shown in Table 16.

E. THE CHOICE OF MALWARE DETECTION FROM DIFFERENT ANGLES

Through the above analysis, malware detection can be performed based on different features. When conducting malware research, the corresponding research methods can be selected according to different angles. The choice of conducting malware detection from different angles can be summarized as shown in Fig 5. The specific division is illustrated as follows:

(1) According to the detection locations, the detection methods can be divided into various types including host-based detection, server-based detection, and cloud-based detection, that is, analyzing and detecting malware on the host side, or on the server-side, or in the cloud;

(2) According to the detection environments, the detection methods can be divided into various types including virtual machine-based detection, bare metal environment based detection, that is, malware is actually running to capture the behavior characteristics either in the virtual machine environment or in the bare metal environment;

(3) According to the detection objects, the detection methods can be divided into various types including program-centric and system-centric detection. That is, the program-centric detection method focuses on extracting features directly from the program itself, and the system-centric detection focuses on the interaction between the observed program and the operating system;

(4) According to the levels of extracted features, the detection methods can be divided into the following categories including detection based on kernel features, detection based on program and OS interaction behavior characteristics, detection based on program running behavior characteristics. These types of detection focus on extracting different types of features, from kernel to operating system-related information, to interactions between the underlying OS resources and the program.

VII. TYPICAL MALWARE CLASSIFICATION METHODS

Malware classification aims to group the malware into corresponding families, to grasp the overall characteristics of a malware family, and to quickly discover its unique features from a large number of malware variants. Malware classification is similar to detection. Researchers first need to extract features from malware samples and then select automated classifiers for classification. The typical malware classification method are summarized in Table 17.

TABLE 17. Typical malware classification methods.

Method	Abstract & Contributions
Bailey et al. 2007	1. Build malware fingerprints based on system state changes during execution, which is more suitable for describing program behavior characteristics than abstract code sequences; 2. A large amount of malware can be grouped into groups of similar behavior accurately and effectively.
Nataraj et al. 2011	1. Apply texture analysis methods to malware analysis; 2. Binary texture analysis methods can deal with packed malicious programs.
Ahmadi et al. 2016	1. Extract and evaluate different features directly from the content and structure of the packaged malware, without unpacking; 2. Extract features from the PE structure, it can achieve accurate classification; 3. The number of features used is limited, which can be applied to large-scale malware classification issue. 4. The feature fusion algorithm is designed to effectively correlate different categories of features. Each type of feature is associated with different aspects of malware, thus avoiding all features mixing and achieves a compromise between accuracy and number of features.
Hu et al. 2016	For the first time, malware-related intelligence information is used in the feature construction process to enrich the feature space of malware. This article introduced a machine learning framework based on multifaceted content features and intelligence information gathered from external sources such as anti-virus tools. The framework mainly consists of four parts: 1) preprocessing the malware data set, reconstructing the original PE file, and obtaining the tag information from the anti-virus software; 2) extracting two types of features, namely machine code command features and anti-virus tag features, and perform feature transformation; 3) synthesize the two above types of features into one feature vector; 4) automatic malware classification.
Lee et al. 2016	This paper introduced a method for detecting and classifying malware and using local clustering coefficient to improve classification reliability. The method selects the first representative malware for each malware family and classifies large-scale malware efficiently. It can be used to analyze new malware, predict the trend of malware families, identify interesting malware, predict the future operation of the same attacker, and promote to enhance the protective barrier against malware.
Raff et al. 2017	By extending the Lempel-Ziv Jaccard Distance, the author proposed a new SHWeL feature vector to represent malware features. The SHWeL vector improves the accuracy of LZJD, and its performance is better than the byte n-grams feature representation. In addition, the proposed model needs no domain knowledge and can be extended to classify byte sequences.
Le et al. 2018	1. A malware classifier that does not require domain knowledge; 2. Without the need for complex feature engineering, the proposed deep learning model can achieve accurate classification of 9 types of malware; 3. The one-dimensional representation of the original binary is similar to the image representation, but simpler and preserves the order of the bytecodes in the binary. This representation method is suitable for classification using a convolutional neural network model.

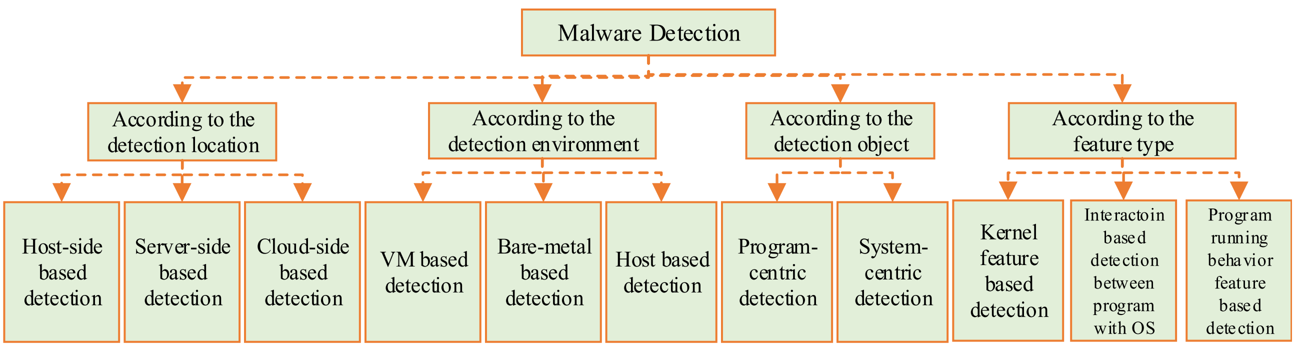


FIGURE 5. Different types of malware detection methods.

VIII. ADDITIONAL SUPPLEMENTS REQUIRED FOR MALWARE RESEARCH

In addition to focusing on malware detection methods, we also need to learn about other aspects in this field.

A. CHOICE OF DATASET

According to the published papers, there are three types of data sets currently used by the malware research community. The application of malware dataset is shown in Table 18.

TABLE 18. Common datasets for malware research.

Type of dataset	Data source	Application instances
Publicly available dataset	VX Heavens VirusShare	Nataraj et al. 2011; Zhang et al. 2010; Santos et al. 2011; Santos et al. 2013; Zhao et al. 2013; Carlin et al. 2017; Salehi et al. 2014; Alazab 2015; Naval et al. 2015; Das et al. 2016; Hellal et al. 2016; Huda et al. 2016; Bidoki et al. 2017; Ming et al. 2017; Salehi et al. 2017; Tajoddin et al. 2018; Shafiq et al. 2009; Bai et al. 2014; Nguyen et al. 2018; Mao et al. 2017; Shahzad et al. 2013; Ghiasi et al. 2015; Burnap et al. 2018; Bat-Erdene et al. 2016; Liu et al. 2013; Nguyen et al. 2009; Han et al. 2019a; Han et al. 2019b; Ajay et al. 2018
	Netlux Offensive Computing	Ding et al. 2014; Ding et al. 2013; Hellal et al. 2016 Cesare et al. 2013; Cesare et al. 2014 Bat-Erdene et al. 2016; Shahzad et al. 2013
	nexginrc	Elhadi et al. 2014
	Anubis	Cui et al. 2018; Lanzi et al. 2010 Chandramohan et al. 2013; Fattori et al. 2015
	Malicia-project	Ki et al. 2015 ; Narouei et al. 2015 ; Mao et al. 2017; Nguyen et al. 2018
	honeynet project	Huda et al. 2016
	http://malware.lu	Ding et al. 2018
Commercial dataset	Kingsoft Anti-Virus Laboratory	Ye et al. 2008; Ye et al. 2010
	Kaspersky Lab and Honey-Net	Liu et al. 2011
	Symantec's Norton Community Watch data	Tamersoy et al. 2014
	AMP ThreatGrid of CISCO Systems	Stiborek et al. 2018
	The Kaggle Malware dataset of Microsoft	Hu et al. 2016; Ahmadi et al. 2016
	MD:Pro (a paid-subscriber malware feed service)	Saxe et al. 2012
Artificially generated data	Generated by toolkits manually	Alam et al. 2015; Raphe et al. 2017; Khalilian et al. 2018; Lee et al. 2010; Lu et al. 2013; Jiang et al. 2010; Kirat et al. 2011; Kirat et al. 2014; Radkani et al. 2018; Zhang et al. 2019
	Malware Repository of APA, Malware Research Center at Shiraz University	Eskandari et al. 2012
	the BareCloud system	Kirat et al. 2015
	CA Labs of Australia	Islam et al. 2013

1) PUBLICLY AVAILABLE DATASETS

Most of the currently published papers use the publicly available data sets in the field of network security. These datasets are maintained by research enthusiasts in the world of cybersecurity and are constantly being updated for free use by researchers.

2) COMMERCIAL DATASETS

There are also some commercial projects that are supported by companies. These data sets are usually not publicly free for utilization.

3) ARTIFICIALLY GENERATED DATASETS

There are also some datasets in which the researcher uses special tools to generate manually or extract from the network traffic.

B. COMMON MACHINE LEARNING TOOLS

In the experimental phase of malware detection, some common machine learning tools can be used to assist in the experimental verification. Common machine learning tools include Python-based frameworks and Java-based frameworks.

1) MACHINE LEARNING TOOLS BASED ON PYTHON

Python is considered to be the most suitable programming language for machine learning. So, in the field of machine learning, researchers have developed a variety of machine learning and deep learning tools based on the Python language.

a: SCIKIT-LEARN

Scikit-learn is a simple and efficient Python-based data mining and data analysis tool based on NumPy, SciPy, and

TABLE 19. Shortcomings of similar reviews.

Author	Main content	Shortcomings
Egele et al. 2012	This paper summarizes the types and infection modes of malware. It mainly reviews the dynamic malware analysis technologies and elaborates the methods and tools for dynamic analysis through different analysis objects.	The static analysis method is not introduced.
Gardiner et al. 2016	This article mainly introduces the use of machine learning methods to discover the command and control channels in the process of malware attacks to discover evasive malware.	Mainly for malware that can evade detection, how to find the command and control channels.
Razak et al. 2016	This paper summarizes the papers on malware research published in the ISI Web of Science database between 2005 and 2015. The papers are statistically divided according to different continents, and the results, research trends and relationships of different intercontinental research institutions are compared.	The statistical analysis of the results of different intercontinental research institutions was carried out, and the research methods were not systematically sorted out.
Ye et al. 2017	This paper gives a comprehensive introduction to the research of malware. The author divides the malware research process into two stages: feature extraction and detection/classification, and synthesizes the features that can be extracted and the different types of detection/classification methods in summary. In addition, the challenges of malware detection based on machine learning are also discussed, and the trend of future research is predicted.	The concepts and methods related to malware research are summarized, and new results need to be further added to highlight the typical methods.
Souri et al. 2018	In this paper, malware detection methods are divided into two categories: feature-based detection and behavior-based detection. These methods are summarized and compared to help researchers establish a general understanding framework for malware detection.	The research method is divided only from the two aspects of feature and behavior, and a fine-grained introduction is not given.
Ucci et al. 2019	This paper summarizes the malware analysis methods from different angles, introduces the malware analysis methods according to different analysis	An overview of malware analysis methods from an analytical goal
	objectives, and summarizes the feature process and machine learning algorithms in the analysis process. In addition, the author proposes a new concept, malware analysis economics, intended to achieve a compromise between malware detection accuracy and overhead from an economic perspective.	perspective does not provide an easy-to-understand knowledge framework for initial researchers.

Matplotlib. Scikit-Learn provides a consistent and easy to use API set and random search framework. Its main advantage is that the algorithm is simple and fast. Scikit-learn mainly includes the following 6 basic functions: classification, regression, clustering, data dimensionality reduction, model selection, and data preprocessing.

b: KERAS

Keras is a high-level neural network API set that provides a Python deep learning library. For any beginner, Keras is the best choice for conducting machine learning applications because it provides a simpler way to construct neural networks than other libraries.

c: THEANO

Theano is one of the most mature Python deep learning libraries. Its main features include tight integration with NumPy, customize functions in symbolic languages, and efficient execution on GPU or CPU platforms.

2) MACHINE LEARNING TOOLS BASED ON JAVA

The most commonly used machine learning tool written in Java is WEKA. Weka integrates machine learning algorithms related to data mining tasks. These algorithms can be applied directly to the dataset, or you can call them by writing Java code yourself. Weka includes a variety of tools for data preprocessing, classification, regression, clustering, rules association, and visualization. In addition, new machine learning methods can also be developed based on Weka.

IX. ISSUES NEED TO TACKLE IN FUTURE MALWARE RESEARCH

The offense and defense of malware is a never-ending arms race in the field of cybersecurity. With the rapid development of web applications, unseen types of malware are continuously emerging, and old malware is constantly evolving. Issues that need to be addressed in the future include:

A. DETECTION OF NEW MALWARE

As the socio-economic situation continues to evolve, new malware will continue to emerge. For example, the ransomware that has been erupting in recent years has a similar form in real life, that is, by encrypting network information assets and then extorting asset owners for economic benefits. The ransomware has caused destructive damage in society, and the existing protective measures are still evolving and may have subsequent effects Al-Rimy Bander *et al.* [5] Homayoun *et al.* [39].

B. DETECTION OF MALWARE AGAINST CRITICAL NETWORK INFRASTRUCTURE

With the important influence on social development, network infrastructure has become an important target of cyber-attacks. Various advanced persistent threat attacks against network infrastructure occur frequently. How to effectively analyze such complex malware and discover advanced symptoms of persistent threats are key to protecting critical infrastructure in the cyberspace [Bolton *et al.* [6]; Li *et al.* [58].

C. MALWARE DETECTION BASED ON THE CLOUD COMPUTING ENVIRONMENT

The cloud computing environment provides a platform for ordinary researchers to customize their application services. Given the rich computing and storage resources available in the cloud, the cloud computing environment can perform malware analysis tasks that cannot be delivered by ordinary computing platforms. Therefore, the malware analysis strategy based on cloud computing environment has attracted considerable attention. How to fully utilize the cloud computing environment to carry out malware analysis tasks while ensuring the security of cloud resources is a problem that researchers must tackleYadav [108] and Zou *et al.* [117].

D. MALWARE DETECTION BASED ON EDGE COMPUTING

With the widespread application of edge computing in the network, analysis tasks for large amounts of malware can also be handled by local devices without recourse for the cloud service, and the analysis process will be done at the local edge computing side. The combination of edge computing and malware analysis will effectively improve the security of local devices. Therefore, edge computing-based malware analysis will also be a direction in the field of malware research [He *et al.* [37]; Kozik [51]; Ren *et al.* [83].

E. ANTI-MALWARE RESEARCH BASED ON THE TRUST MECHANISM

The trust mechanism is an effective security measure for the industrial Internet of Things (IoT) to detect compromised and malicious nodes. This mechanism calculates the trust values of the nodes based on their behaviors, which can tackle the typical security issues faced by industrial IoT. From the perspective of behavioral evaluation, the trust evaluation scheme can be applied in the anti-malware research field, especially with the widespread application and development of IoT. We can evaluate the trust values of the unknown applications emerging in the distributed nodes in IoT and identify the malware with lower consumption and higher performance [Huang *et al.* [41]; Liu *et al.* [61]; Wang *et al.* [107].

X. COMPARISON WITH SIMILAR REVIEWS

There have been some similar review articles on malware research, as shown in Table 19. The differences between this review and the published literature are as follows:

(1) This article does not propose any new concept, but according to the general roadmap for carrying out research "Why? $\rightarrow$ What? $\rightarrow$ How?", so that the beginners can quickly enter the malware research field guided by the article;

(2) This paper has categorized the malware research methods from different angles, which can help researchers quickly find the entry point suitable for their own research;

(3) This paper gives a systematic introduction of the influential papers published after 2010, ensuring the novelty and comprehensiveness of the content. Researchers can quickly find the content they are interested in, just like a dictionary, and improve research efficiency.

XI. CONCLUSION

In the cyberspace environment, malware offense and defense is an ever-lasting arms race. To help initial researchers quickly and effectively establish a framework for malware awareness, this article conducts an extensive survey on this field based on papers published in SCI journals and important international academic conferences after 2010, according to an easy-to-understand roadmap. The theories and techniques on malware and anti-malware are summarized and categorized comprehensively. Instructed by this article, the new researchers can step into the route of malware research quickly and smoothly.

CONFLICTS OF INTERESTS

The authors declare that they have no competing interests.

REFERENCES

[1] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proc. 6th ACM Conf. Data Appl. Secur. Privacy*, New Orleans, LA, USA, Mar. 2016, pp. 183–194. doi: 10.1145/2857705.2857713.

[2] M. A. A. Kumara and C. D. Jaidhar, "Automated multi-level malware detection system based on reconstructed semantic view of executables using machine learning techniques at VMM," *Future Gener. Comput. Syst.*, vol. 79, pp. 431–446, 2018. doi: 10.1016/j.future.2017.06.002.

[3] S. Alam, R. N. 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. doi: 10.1016/j.cose.2014.10.011.

[4] M. Alazab, "Profiling and classifying the behavior of malicious codes," *J. Syst. Softw.*, vol. 100, pp. 91–102, Feb. 2015. doi: 10.1016/j.jss.2014.10.031.

[5] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Comput. Secur.*, vol. 74, pp. 144–166, May 2018. doi: 10.1016/j.cose.2018.01.001.

[6] D. Alexander Bolton and M. Christine Anderson-Cook, "APT malware static trace analysis through bigrams and graph edit distance," *Stat. Anal. Data Mining Asa Data Sci. J.*, vol. 10, no. 3, pp. 182–193, 2017. doi: 10.1002/sam.11346.

[7] J. Bai, J. Wang, and G. Zou, "A malware detection scheme based on mining format information," *Sci. World J.*, vol. 2014, Jun. 2014, Art. no. 260905. doi: 10.1155/2014/260905.

[8] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of Internet malware," in *Proc. 10th Int. Conf. Recent Adv. Intrusion Detection*, Gold Goast, QLD, Australia, Sep. 2007, pp. 178–197. doi: 10.1007/978-3-540-74320-0_10.

[9] M. Bat-Erdene, H. Park, H. Li, H. Lee, and M.-S. Choi, "Entropy analysis to classify unknown packing algorithms for malware detection," *Int. J. Inf. Secur.*, vol. 16, no. 3, pp. 227–248, 2016. doi: 10.1007/s10207-016-0330-4.

[10] S. M. Bidoki, S. Jalili, and A. Tajoddin, "PbMMD: A novel policy based multi-process malware detection," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 57–70, Apr. 2017. doi: 10.1016/j.engappai.2016.12.008.

[11] P. Burnap, R. French, F. Turner, and K. Jones, "Malware classification using self organising feature maps and machine activity data," *Comput. Secur.*, vol. 73, pp. 399–410, Mar. 2018. doi: 10.1016/j.cose.2017.11.016.

[12] 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.*, Shiraz, Iran, May 2013, pp. 113–120. doi: 10.1109/IKT.2013.6620049.

[13] D. Carlin, A. Cowan, P. O'Kane, and S. Sezer, "The effects of traditional anti-virus labels on malware detection using dynamic runtime opcodes," *IEEE Access*, vol. 5, pp. 17742–17752, 2017. doi: 10.1109/ACCESS.2017.2749538.

[14] S. Cesare, Y. Xiang, and W. Zhou, "Malwise—An effective and efficient classification system for packed and polymorphic malware," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1193–1206, Jun. 2013. doi: 10.1109/TC.2012.65.

[15] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware variant detection," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 4, pp. 307–317, Jul./Aug. 2014. doi: 10.1109/TDSC.2013.40.

[16] 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. Automated Softw. Eng. (ASE)*, Silicon Valley, CA, USA, Jan. 2014, pp. 312–322. doi: 10.1109/ASE.2013.6693090.

[17] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018. doi: 10.1109/TII.2018.2822680.

[18] 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. doi: 10.1109/TIFS.2015.2491300.

[19] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: Malware analysis via hardware virtualization extensions," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, Oct. 2008, pp. 51–62. doi: 10.1145/1455770.1455779.

[20] Y. Ding, W. Dai, S. Yan, and Y. Zhang, "Control flow-based opcode behavior analysis for Malware detection," *Comput. Secur.*, vol. 44, pp. 65–74, Jul. 2014. doi: 10.1016/j.cose.2014.04.003.

[21] Y. Ding, X. Xia, S. Chen, and Y. Li, "A malware detection method based on family behavior graph," *Comput. Secur.*, vol. 73, pp. 73–86, Mar. 2018. doi: 10.1016/j.cose.2017.10.007.

[22] Y. Ding, X. Yuan, K. Tang, X. Xiao, and Y. Zhang, "A fast malware detection algorithm based on objective-oriented association mining," *Comput. Secur.*, vol. 39, pp. 315–324, Nov. 2013. doi: 10.1016/j.cose.2013.08.008.

[23] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, p. 6, 2012. doi: 10.1145/2089125.2089126.

[24] A. A. E. Elhadi, M. A. Maarof, B. I. A. Barry, and H. Hamza, "Enhancing the detection of metamorphic malware using call graphs," *Comput. Secur.*, vol. 46, pp. 62–78, Oct. 2014. doi: 10.1016/j.cose.2014.07.004.

[25] A. A. E. Elhadi, M. A. Maarof, and A. H. Osman, "Malware detection based on hybrid signature behaviour application programming interface call graph," *Amer. J. Appl. Sci.*, vol. 9, vol. 3, pp. 283–288, 2012. doi: 10.3844/ajassp.2012.283.288.

[26] J. Escalada, F. Ortin, and T. Scully, "An efficient platform for the automatic extraction of patterns in native code," *Sci. Program.*, vol. 2017, Feb. 2017, Art. no. 3273891. doi: 10.1155/2017/3273891.

[27] M. Eskandari and S. Hashemi, "A graph mining approach for detecting unknown malwares," *J. Vis. Lang. Comput.*, vol. 23, no. 3, pp. 154–162, 2012. doi: 10.1016/j.jvlc.2012.02.002.

[28] A. Fattori, A. Lanzi, D. Balzarotti, and E. Kirda, "Hypervisor-based malware protection with AccessMiner," *Comput. Secur.*, vol. 52, pp. 33–50, Jul. 2015. doi: 10.1016/j.cose.2015.03.007.

[29] X. Fu, "On detecting environment sensitivity using slicing," *Theor. Comput. Sci.*, vol. 656, pp. 27–45, Dec. 2016. doi: 10.1016/j.tcs.2016.09.004.

[30] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C&C detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, p. 59, 2016. doi: 10.1145/3003816.

[31] M. Ghiasi, A. Sami, and Z. Salehi, "Dynamic VSA: A framework for malware detection based on register contents," *Eng. Appl. Artif. Intell.*, vol. 44, pp. 111–122, Sep. 2015. doi: 10.1016/j.engappai.2015.05.008.

[32] V. Gratzer and D. Naccache, "Alien vs. Quine," *IEEE Security Privacy*, vol. 5, no. 2, pp. 26–31, Mar./Apr. 2007. doi: 10.1109/MSP.2007.28.

[33] L. Han, S. Liu, S. Han, W. Jia, and J. Lei, "Owner based malware discrimination," *Future Gener. Comput. Syst.*, vol. 80, pp. 496–504, Mar. 2018. doi: 10.1016/j.future.2016.05.020.

[34] W. Han, J. Xue, Y. Wang, Z. Liu, and Z. Kong, "MalInsight: A systematic profiling based malware detection framework," *J. Netw. Comput. Appl.*, vol. 125, pp. 236–250, Jan. 2019. doi: 10.1016/j.jnca.2018.10.022.

[35] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, and L. Mao, "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," *Comput. Secur.*, vol. 83, pp. 208–233, Jun. 2019. doi: 10.1016/j.cose.2019.02.007.

[36] W. Han, J. Xue, and H. Yan, "Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine," *IET Inf. Secur.*, vol. 13, no. 2, pp. 109–116, Mar. 2019. doi: 10.1049/iet-ifs.2018.5186.

[37] G. He, L. Zhang, B. Xu, and H. Zhu, "Detecting repackaged Android malware based on mobile edge computing," in *Proc. 6th Int. Conf. Adv. Cloud Big Data (CBD)*, Lanzhou, China, Aug. 2018, pp. 360–365. doi: 10.1109/CBD.2018.00071.

[38] A. Hellal and L. B. Romdhane, "Minimal contrast frequent pattern mining for malware detection," *Comput. Secur.*, vol. 62, pp. 19–32, Sep. 2016. doi: 10.1016/j.cose.2016.06.004.

[39] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K.-K. R. Choo, and D. E. Newton, "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Gener. Comput. Syst.*, vol. 90, pp. 94–104, Jan. 2019. doi: 10.1016/j.future.2018.07.045.

[40] X. Hu, J. Jang, T. Wang, Z. Ashraf, M. P. Stoecklin, and D. Kirat, "Scalable malware classification with multifaceted content features and threat intelligence," *IBM J. Res. Develop.*, vol. 60, vol. 4, pp. 6:1–6:11, Jul./Aug. 2016. doi: 10.1147/JRD.2016.2559378.

[41] M. Huang, W. Liu, T. Wang, Q. Deng, A. Liu, M. Xie, M. Ma, and G. Zhang, "A game-based economic model for price decision making in cyber-physical-social systems," *IEEE Access*, vol. 7, pp. 111559–111576, 2019. doi: 10.1109/ACCESS.2019.2934515.

[42] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Gener. Comput. Syst.*, vol. 55, pp. 376–390, Feb. 2016. doi: 10.1016/j.future.2014.06.001.

[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, 2013. doi: 10.1016/j.jnca.2012.10.004.

[44] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection and monitoring through VMM-based 'out-of-the-box' semantic view reconstruction," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 2, p. 12, 2010. doi: 10.1145/1698750.1698752.

[45] M. E. Karim, A. Walenstein, A. Lakhotia, and L. Parida, "Malware phylogeny generation using permutations of code," *J. Comput. Virol.*, vol. 1, nos. 1–2, pp. 13–23, 2005. doi: 10.1007/s11416-005-0002-9.

[46] A. Khalilian, A. Nourazar, M. Vahidi-Asl, and H. Haghighi, "G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families," *Expert Syst. Appl.*, vol. 112, pp. 15–33, Dec. 2018. doi: 10.1016/j.eswa.2018.06.012.

[47] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 6, 2015, Art. no. 659101. doi: 10.1155/2015/659101.

[48] D. Kirat, G. Vigna, and C. Kruegel, "BareCloud: Bare-metal analysis-based evasive malware detection," in *Proc. 23rd USENIX Conf. Secur. Symp.*, San Diego, CA, USA, 2014, pp. 287–301.

[49] D. Kirat and G. Vigna, "MalGene: Automatic extraction of malware analysis evasion signature," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.* Denver, CO, USA, 2015, pp. 769–780. doi: 10.1145/2810103.2813642.

[50] D. Kirat, G. Vigna, and C. Kruegel, "BareBox: Efficient malware analysis on bare-metal," in *Proc. 27th Annu. Comput. Secur. Appl. Conf.*, Orlando, FL, USA, Dec. 2011, pp. 403–412. doi: 10.1145/2076732.2076790.

[51] R. Kozik, M. Choras, M. Ficco, and F. Palmieri, "A scalable distributed machine learning approach for attack detection in edge computing environments," *J. Parallel Distrib. Comput.*, vol. 119, pp. 18–26, Sep. 2018. doi: 10.1016/j.jpdc.2018.03.006.

[52] S. Kramer and J. C. Bradfield, "A general definition of malware," *J. Comput. Virol.*, vol. 6, no. 2, pp. 105–114, 2010. doi: 10.1007/s11416-009-0137-1.

[53] 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)*, Chicago, IL, USA, Oct. 2010, pp. 399–412. doi: 10.1145/1866307.1866353.

[54] Q. Le, O. Boydell, B. M. Namee, and M. Scanlon, "Deep learning at the shallow end: Malware classification for non-domain experts," *Digit. Invest.*, vol. 26, pp. S118–S126, Jul. 2018. doi: 10.1016/j.diin.2018.04.024.

[55] J. Lee, K. Jeong, and H. Lee, "Detecting metamorphic malwares using code graphs," in *Proc. ACM Symp. Appl. Comput.*, Sierre, Switzerland, Mar. 2010, pp. 1970–1977. doi: 10.1145/1774088.1774505.

[56] T. Lee, B. Choi, Y. Shin, and J. Kwak, "Automatic malware mutant detection and group classification based on the n-gram and clustering coefficient," *J. Supercomput.*, vol. 74, vol. 8, pp. 3489–3503, 2018. 74. doi: 10.1007/s11227-015-1594-6.

[57] T. Lee and J. Kwak, "Effective and reliable malware group classification for a massive malware environment," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 5, 2016, Art. no. 4601847. doi: 10.1155/2016/4601847.

[58] Y. Li, W. Dai, J. Bai, X. Gan, J. Wang, and X. Wang, "An intelligence-driven security-aware defense mechanism for advanced persistent threats," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 646–661, Mar. 2019.

[59] J. Liu, J. Song, Q. Miao, and Y. Cao, "FENOC: An ensemble one-class learning framework for malware detection," in *Proc. 9th Int. Conf. Comput. Intell. Secur.*, Leshan, China, Dec. 2013, pp. 523–527. doi: 10.1109/CIS.2013.116.

[60] T. Liu, X. Guan, Y. Qu, and Y. Sun, "A layered classification for malicious function identification and malware detection," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 11, pp. 1169–1179, 2012. doi: 10.1002/cpe.1896.

[61] Y. Liu, A. Liu, X. Liu, and M. Ma, "A trust-based active detection for cyber-physical security in industrial environments," *IEEE Trans. Ind. Informat.*, to be published. doi: 10.1109/TII.2019.2931394.

[62] H. Lu, X. Wang, B. Zhao, F. Wang, and J. Su, "ENDMal: An anti-obfuscation and collaborative malware detection system using syscall sequences," *Math. Comput. Model.*, vol. 58, nos. 5–6, pp. 1140–1154, 2013. doi: 10.1016/j.mcm.2013.03.008.

[63] W. Mao, Z. Cai, D. Towsley, Q. Feng, and X. Guan, "Security importance assessment for system objects and malware detection," *Comput. Secur.*, vol. 68, pp. 47–68, Jul. 2017. doi: 10.1016/j.cose.2017.02.009.

[64] J. Ming, D. Xu, Y. Jiang, and D. Wu, "BinSim: Trace-based semantic binary diffing via system call sliced segment equivalence checking," in *Proc. 26th USENIX Secur. Symp.*, Vancouver, BC, Canada, Aug. 2017, pp. 253–270.

[65] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: High-fidelity, behavior-based automated malware analysis and classification," *Comput. Secur.*, vol. 52, pp. 251–266, Jul. 2015. doi: 10.1016/j.cose.2015.04.001.

[66] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Proc. 23th Annu. Comput. Secur. Appl. Conf.*, Miami Beach, FL, USA, Dec. 2007, pp. 421–430. doi: 10.1109/ACSAC.2007.21.

[67] R. Moskovitch, C. Feher, and Y. Elovici, "A chronological evaluation of unknown malcode detection," in *Proc. Pacific–Asia Workshop Intell. Secur. Inform.*, Bangkok, Thailand, Apr. 2009, pp. 112–117. doi: 10.1007/978-3-642-01393-5_12.

[68] M. Narouei, M. Ahmadi, G. Giacinto, H. Takabi, and A. Sami, "DLLMiner: Structural mining for malware detection," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3311–3322, 2015. 2015. doi: 10.1002/sec.1255.

[69] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, Chicago, IL, USA, Oct. 2011, pp. 21–30. doi: 10.1145/2046684.2046689.

[70] 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. doi: 10.1109/TIFS.2015.2469253.

[71] A. M. Nguyen, N. Schear, H. Jung, A. Godiyal, S. T. King, and H. D. Nguyen, "MAVMM: Lightweight and purpose built VMM for malware analysis," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Honolulu, HI, USA, Dec. 2009, pp. 441–450. doi: 10.1109/ACSAC.2009.48.

[72] M. H. Nguyen, D. Le Nguyen, X. M. Nguyen, and T. T. Quan, "Auto-detection of sophisticated malware using lazy-binding control flow graph and deep learning," *Comput. Secur.*, vol. 76, pp. 128–155, Jul. 2018. 2018. doi: 10.1016/j.cose.2018.02.006.

[73] G. Nguyen, S. Dlugolinsky, M. Bobák, V. Tran, A. L. García, I. Heredia, P. Malík, and L. Hluchý, "Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 77–124, 2019. doi: 10.1007/s10462-018-09679-z.

[74] M. Ni, T. Li, Q. Li, H. Zhang, and Y. Ye, "FindMal: A file-to-file social network based malware detection framework," *Knowl.-Based Syst.*, vol. 112, pp. 142–151, Nov. 2016. doi: 10.1016/j.knosys.2016.09.004.

[75] P. OKane, S. Sezer, and K. McLaughlin, "Obfuscation: The hidden malware," *IEEE Security Privacy*, vol. 9, no. 5, pp. 41–47, Sep./Oct. 2011. doi: 10.1109/MSP.2011.98.

[76] P. Okane, S. Sezer, K. Mclaughlin, and E. G. Im, "Malware detection: Program run length against detection rate," *IET Softw.*, vol. 8, no. 1, pp. 42–51, Feb. 2014. doi: 10.1049/iet-sen.2013.0020.

[77] (2019). *Panda Security Info Glossary*. Accessed: Jun. 28, 2019. [Online]. Available: https://www.pandasecurity.com/en/security-info/glossary/

[78] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, 2019, Art. no. 92. doi: 10.1145/3234150.

[79] E. Radkani, S. Hashemi, A. Keshavarz-Haddad, and M. A. Haeri, "An entropy-based distance measure for analyzing and detecting metamorphic malware," *Appl. Intell.*, vol. 48, no. 6, pp. 1536–1546, 2018. 48. doi: 10.1007/s10489-017-1045-6.

[80] E. Raff and C. Nicholas, "Malware classification and class imbalance via stochastic hashed LZJD," *Proc. AISec*, Dallas, TX, USA, Nov. 2017, pp. 111–120. doi: 10.1145/3128572.3140446.

[81] J. Raphel and P. Vinod, "Heterogeneous opcode space for metamorphic malware detection," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 537–558, 2017. doi: 10.1007/s13369-016-2264-6.

[82] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of 'malware': Bibliometric analysis of malware study," *J. Netw. Comput. Appl.*, vol. 75, pp. 58–76, Nov. 2016. doi: 10.1016/j.jnca.2016.08.022.

[83] J. Ren, Y. Guo, D. Zhang, Q. Liu, and Y. Zhang, "Distributed and efficient object detection in edge computing: Challenges and solutions," *IEEE Netw.*, vol. 32, no. 6, pp. 137–143, Nov./Dec. 2018. doi: 10.1109/MNET.2018.1700415.

[84] J. Rhee, R. Riley, Z. Lin, X. Jiang, and D. Xu, "Data-centric OS kernel malware characterization," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 1, pp. 72–87, Jan. 2014. doi: 10.1109/TIFS.2013.2291964.

[85] K. Rieck, "Off the beaten path: Machine learning for offensive security," in *Proc. ACM Workshop Artif. Intell. Secur.*, Berlin, Germany, Nov. 2013, pp. 1–2. doi: 10.1145/2517312.2517313.

[86] A. Roberts, R. McClatchey, S. Liaquat, N. Edwards, and M. Wray, "POSTER: Introducing pathogen: A real-time virtualmachine introspection framework," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Berlin, Germany, Nov. 2013, pp. 1429–1432. doi: 10.1145/2508859.2512518.

[87] K. A. Roundy and B. P. Miller, "Hybrid analysis and control of malware," in *Proc. Int. Workshop Recent Adv. Intrusion Detection (RAID)*, in Lecture Notes in Computer Science, vol. 6307. Berlin, Germany: Springer, 2010. doi: 10.1007/978-3-642-15512-3_17.

[88] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult, "A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1145–1172, 2nd Quart., 2017. doi: 10.1109/COMST.2016.2636078.

[89] (2019). *SafetyDetective*. Accessed: Mar. 6, 2019. [Online]. Available: https://www.safetydetective.com/blog/malware-statistics/

[90] Z. Salehi, A. Sami, and M. Ghiasi, "Using feature generation from API calls for malware detection," *Comput. Fraud Secur.*, vol. 2014, no. 9, pp. 9–18, 2014. doi: 10.1016/S1361-3723(14)70531-7.

[91] Z. Salehi, A. Sami, and M. Ghiasi, "MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values," *Eng. Appl. Artif. Intell.*, vol. 59, pp. 93–102, Mar. 2017. doi: 10.1016/j.engappai.2016.12.016.

[92] I. Santos, F. Brezo, B. Sanz, C. Laorden, and P. G. Bringas, "Using opcode sequences in single-class learning to detect unknown malware," *IET Inf. Secur.*, vol. 5, no. 4, pp. 220–227, Dec. 2011. doi: 10.1049/iet-ifs.2010.0180.

[93] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013. doi: 10.1016/j.ins.2011.08.020.

[94] J. Saxe, D. Mentis, and C. Greamo, "Visualization of shared system call sequence relationships in large malware corpora," in *Proc. 9th Int. Symp. Visualizat. Cyber Secur.*, Seattle, WA, USA, Oct. 2012, pp. 33–40. doi: 10.1145/2379690.2379695.

[95] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," *Inf. Secur. Tech. Rep.*, vol. 14, no. 1, pp. 16–29, 2009. doi: 10.1016/j.istr.2009.03.003.

[96] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PE-Miner: Mining structural information to detect malicious executables in realtime," in *Recent Advances in Intrusion Detection* (Lecture Notes in Computer Science), vol. 5758, E. Kirda, S. Jha, and D. Balzarotti, Eds. Berlin, Germany: Springer, 2009. doi: 10.1007/978-3-642-04342-0_7.

[97] F. Shahzad, M. Shahzad, and M. Farooq, "In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS," *Inf. Sci.*, vol. 231, pp. 45–63, May 2013. doi: 10.1016/j.ins.2011.09.016.

[98] S. Sheen, R. Anitha, and P. Sirisha, "Malware detection by pruning of parallel ensembles using harmony search," *Pattern Recognit. Lett.*, vol. 34, no. 14, pp. 1679–1686, 2013. doi: 10.1016/j.patrec.2013.05.006.

[99] F. Song and T. Touili, "Pushdown model checking for malware detection," *Int. J. Softw. Tools Technol. Transf.*, vol. 16, no. 2, pp. 147–173, 2012. doi: 10.1007/s10009-013-0290-1.

[100] 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, Jan. 2018, Art. no. 3. doi: 10.1186/s13673-018-0125-x.

[101] J. Stiborek, T. Pevný, and M. Rehák, "Multiple instance learning for malware classification," *Expert Syst. Appl.*, vol. 93, pp. 346–357, Mar. 2018. doi: 10.1016/j.eswa.2017.10.036.

[102] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static analyzer of vicious executables (SAVE)," in *Proc. 20th Annu. Comput. Secur. Appl. Conf.*, Tucson, AZ, USA, USA, Dec. 2004, pp. 326–334. doi: 10.1109/CSAC.2004.37.

[103] A. Tajoddin and S. Jalili, "HM^{3}alD: Polymorphic Malware detection using program behavior-aware hidden Markov model," *Appl. Sci.*, vol. 8, no. 7, p. 1044, Jun. 2018. doi: 10.3390/app8071044.

[104] A. Tamersoy, K. Roundy, and D. H. Chau, "Guilt by association: Large scale malware detection by mining file-relation graphs," in *Proc. 20th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2014, pp. 1524–1533. doi: 10.1145/2623330.2623342.

[105] S. Treadwell and M. Zhou, "A heuristic approach for detection of obfuscated malware," in *Proc. IEEE Int. Conf. Intell. Secur. Inform.*, Dallas, TX, USA, Jun. 2009, pp. 291–299. doi: 10.1109/ISI.2009.5137328.

[106] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Comput. Secur.*, vol. 81, pp. 123–147, Mar. 2019. doi: 10.1016/j.cose.2018.11.001.

[107] T. Wang, H. Luo, W. Jia, A. Liu, and M. Xie, "MTES: An intelligent trust evaluation scheme in sensor-cloud enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, to be published. doi: 10.1109/TII.2019.2930286.

[108] R. M. Yadav, "Effective analysis of malware detection in cloud computing," *Comput. Secur.*, vol. 83, pp. 14–21, Jun. 2019. doi: 10.1016/j.cose.2018.12.005.

[109] L.-K. Yan, M. Jayachandra, M. Zhang, and H. Yin, "V2E: Combining hardware virtualization and softwareemulation for transparent and extensible malware analysis," in *Proc. 8th ACM SIGPLAN/SIGOPS Conf. Virtual Execution Environ.*, London, U.K., Mar. 2012, pp. 227–238. doi: 10.1145/2151024.2151053.

[110] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, Oct. 2017, Art. no. 41. doi: 10.1145/3073559.

[111] Y. Ye, T. Li, K. Huang, Q. Jiang, and Y. Chen, "Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list," *J. Intell. Inf. Syst.*, vol. 35, pp. 1–20, Aug. 2010. doi: 10.1007/s10844-009-0086-7.

[112] 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, pp. 323–334, Nov. 2008. doi: 10.1007/s11416-008-0082-4.

[113] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Proc. Int. Conf. Broadband, Wireless Comput., Commun. Appl.*, Fukuoka, Japan, Nov. 2010, pp. 297–300. doi: 10.1109/BWCCA.2010.85.

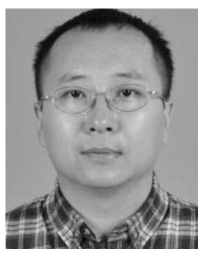
[114] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Gener. Comput. Syst.*, vol. 90, pp. 211–221, Jan. 2019. doi: 10.1016/j.future.2018.07.052.

[115] P. Zhang, W. Wang, and Y. Tan, "A malware detection model based on a negative selection algorithm with penalty factor," *Sci. China Inf. Sci.*, vol. 53, no. 12, pp. 2461–2471, Dec. 2010. doi: 10.1007/s11432-010-4123-5.

[116] Z. Zhao, J. Wang, and J. Bai, "Malware detection method based on the control-flow construct feature of software," *IET Inf. Secur.*, vol. 8, no. 1, pp. 18–24, Jan. 2014. doi: 10.1049/iet-ifs.2012.0289.

[117] D. Zou, J. Zhao, W. Li, Y. Wu, W. Qiang, H. Jin, Y. Wu, and Y. Yang, "A multigranularity forensics and analysis method on privacy leakage in cloud environment," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1484–1494, Apr. 2019. doi: 10.1109/JIOT.2018.2838569.

WEIJIE HAN received the B.E. and M.E. degrees from the Academy of Equipment Command and Technology, in 2003 and 2006, respectively (now named Space Engineering University). He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology. He was a Teacher with the Academy of Equipment Command and Technology, from 2006 to 2016. His research interest includes network security.



JINGFENG XUE received the B.E, the M.E., and Ph.D. degrees from the Beijing Institute of Technology, where he is currently a Professor and a Doctoral Supervisor with the School of Computer Science and Technology. His research interest includes network security.

YONG WANG received the Ph.D. degree from the Beijing Institute of Technology, where she is currently an Associate Professor with the School of Computer Science and Technology. Her current research interests include cyber security and machine learning. She is a Fellow of the China Computer Federation.

SHIBING ZHU received the B.E. degree from the Academy of Equipment Command and Technology (now named Space Engineering University), in 1992, the M.E. degree from the National University of Defense Technology, in 1997, and the Ph.D. degree from the Wuhan University of Technology, in 2009. He is currently a Professor with Space Engineering University, China. His research interests include communication networks and information security.

ZIXIAO KONG is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology. Her current research interest includes network security.