A Survey on Feature Extraction Methods of Heuristic Malware Detection

Nuannuan Li¹, Zheng Zhang¹, Xin Che²*, Zhimin Guo¹, Junfei Cai¹
¹State Grid Henan Electric Power Research Institute, Zhengzhou, China
²Hangzhou UWNTEK Automatic System Co.Ltd, Hangzhou, China
*chexin1417@163.com

Abstract. In the age of the Internet, while the network is sending a lot of information to people, hackers also transmit a lot of malicious code through the network. Hackers use these malicious codes to steal sensitive information from infected people and damage machines and devices to achieve their evil goals. Therefore, it is very important to accurately detect malware to protect users from loss. There are now three major methods of detecting malware, which are signature-based, behavior-based, and heuristic-based methods. However, with the rapid increase in the types and number of malware, signature-based method can’t detect unknown malware and behavior-based cannot guarantee the False Positive Ratio. So these detection methods can no longer meet the needs. Therefore, some researchers proposed some heuristic-based detection methods. In this article we overview the methods used to extract features and the features extracted in heuristic detection and discuss the advantages and disadvantages of the features.

Keywords: Malware, heuristic detection, feature extraction

1. Introduction
Malware software, also named malware, is used to represent all unwanted computer programs[1]. Malware is used as a weapon by attackers to achieve their malicious purpose. Malware is known to be used for various security attacks, such as stealing sensitive user’s information, electronic accounts and passwords. These attacks will seriously threaten the user's information security and cause serious economic losses. And under the drive of economic interests, the types and numbers of malware are rapidly increasing in recent years. Traditional detection methods, such as signature based and behavioral based detection methods already cannot satisfy the need, and a more intelligent detection method is urgently needed to protect the security of users' computers and networks. Therefore, a heuristic based detection method was proposed. Feature extraction has important significance for heuristic detection. Because the extracted features represent sample files, the heuristic detection classifier will determine whether the sample file is benign or malware based on the extracted features. The remainder of this article is structured as follows: Section II introduces the classification of malware; Section III over-
views the analysis methods of malware samples and the features extracted; Section IV compares the advantages and disadvantages of extracted features; Section V introduces some open issues; Section VI summarizes the survey.

2. Overview Of Malware

This section outlines the most common types of malware. According to the different behavior of malicious code, they can be divided into the following categories in Fig.1.

Viruses: Virus is a piece of code used to destroy data stored in a computer and is capable of self-replication. A virus infection is usually divided into two steps: insertion and execution [2]. There are three behavior features: 1. Self-replicating. 2. Attaching its code into legitimate. 3. Active when host program runs.

Worms: A worm is a malicious program that can reproduce itself and have complete functionality. Worms spread using local area network connections or Internet connections and do not need to connect to other executables or document files. When the worm finds a target host, it will copy itself to this host and continue to search other hosts that can be infected. Replication will continue until a self-timing mechanism stops the process. There are two behavior features: 1. Self-replicating. 2. Using network to spread.

Trojans: The Trojan seems to be a normal program on the surface, but it is secretly doing harmful activities. An activated Trojan may perform one or more destructive tasks, such as stealing identity information, destroying data, or providing remote access. There are three behavior features: 1. Hiding itself. 2. Disguising as a legitimating program. 3. Giving access to remote hijacker to control user system.

Backdoor: The backdoor is an entry point to the system, allowing the user to gain access to the system or program without going through a standard security program. Backdoors can also be used by programmers to debug software and require special permissions. This means that the backdoor is not dedicated to malware, but people can use it to complete some malicious behavior. There are two behavior features: 1. Hiding itself. 2. Giving access to remote hijacker to control user system.

Spyware: Spyware can steal victim information and gain system control authority without the victim's awareness. It can be installed on the user's computer in some hidden way, for example as part of a virus. There is a behavior feature: Keeping track of user's activity without their knowledge.
3. Feature extraction methods

In this section, we will discuss the two ways of extracting malware features and the extracted features in heuristic detection in Fig. 2.

3.1. Static analysis

Static analysis means that the sample program is not run dynamically when analyzing the sample file [3]. The goal of static analysis can be a sample binary file or source code. In order to evade detection, the malware developer may confuse the code or encrypt the code, which makes it difficult to be statically analyzed [4]. Therefore, before performing static analysis, it is necessary to unpack or decode operations. After the doing this, some features can be extracted, for example, printable strings, N-grams, Control flow graph and opcodes et al.

3.1.1. N-grams

N-grams are substrings of a larger string with length n. Although most malware uses obfuscation techniques to counter anti-virus detection techniques, n-gram exists in most malware [5].

Sornil et al [5] extracted n-gram based sequential features from content of the files. They determined the sequential n-gram mode, and calculated and reduced the mode statistics through the sequential floating forward selection method.

Boujouni et al [6] proposed an improved version of the method for detecting unknown malware based on n-gram and support vector domain description. According to the information acquisition indicators they proposed, they extracted n-grams from the sample file set.

3.1.2. Printable Strings

Printable strings are included in the body and body parts of the executable file. The attacker’s intent and target may be hidden in these printable characters. Strings are the most obvious feature and it is difficult for malware developer to evade string-based detection. These strings can be part of the filename, the signature of the file author or information on system resources used [7].

Islam et al [8] extracted function length and printable characters from a set of malware samples, and used these two features to detect malicious code in an experimental test. In their experiment, they
entered about 1400 unpacked malware and 151 normal files into different classification algorithms, and finally achieved an overall classification accuracy of over 98%.

3.1.3. OpCode
An OpCode (short for Operational Code) is the subdivision of a machine language instruction that identifies the operation to be executed. The executable program consists of several assembly instructions. An instruction consists of an opcode and an operand or a list of operand such as “push ebp” and “mov ebp, esp”.

Divandari et al[9] proposed a new method for detecting malware by opcode sequences. They extracted the opcode sequence of the sample file and used it as features. Because there are too many features, they use Markov Blanket algorithm to reduce the number of features. Hidden Markov Model was trained by the selected features and used to detect malwares.

Yewale et al[10] proposed a method to detect malware by counting the frequency of opcodes in sample files. They collected opcodes from 100 benign and malware files and found 20 most frequent opcodes. The success rate was about 96.67 per cent.

The first advantage of static analysis is security. Because malicious samples are not run during static analysis, the analyst's computer is not attacked by malware. Another advantage is fast analysis speed, high efficiency, low cost, suitable for large-scale analysis, and can cover almost all code paths [11]. But like the two sides of a coin, static analysis has its drawbacks. Due to the continuous development and application of code obfuscation technology, static analysis of malware becomes more and more difficult [12], and some simple code obfuscation techniques can have a serious impact on the results of static analysis.

3.2. Dynamic Analysis
The biggest difference between dynamic analysis and static analysis is that dynamic analysis will run malware in a secure environment to analyze the behavior of malware. Dynamic analysis method extracts features that can represent sample files by monitoring system API calling, file system writing, and registry operating.

3.2.1. API calls
The API is a series of functions and protocols used to create a program[14]. Almost all programs use Windows API calls to send requests to the operating system. Therefore, researchers can analyze the behavior of the sample file by calling the Windows api sequence by analyzing the sample file[15].

Therefore, in order to achieve some specific functions, there is a correlation between some Windows api functions, and this correlation can be used as a feature of a sample file for malicious code detection.

Pektaş et al[13] used the API call sequence and n-gram of the sample file as features to detect malicious code. They use expert voting algorithms to extract the API call sequence used to implement malicious behavior. Their method fused both the results about mining and searching n-gram API call sequences and OS state changes.

3.2.2. File system
For the file system, two features can be extracted, one is the path and the subpath, the other is the extension [16].

4. COMPARISON THE EXTRACTED FEATURES
In this section, we will compare the extracted features through tables and analyze their advantages and disadvantages in Tab.1.
### 5. SUMMARY
In this article, we first discuss the classification of malware. After that, we review the feature extraction methods and the features extracted. Finally, through the form of a table, the advantages and disadvantages of the extracted features are compared.

### Acknowledgments
This work is supported by the Science and Technology Program of SGCC (5700-202024193A-0-0-00).

### Reference
[1] Yanfang Y, Li T, Adjeroh D and S Sitharama I 2017 A survey on malware detection using data mining techniques ACM Computing Surveys 50 pp1-40
[2] Bishop M A 2002 The Art and Science of Computer Security
[3] Raymond W L, Karl N L, Ronald A O 1995 MCF: A malicious code filter. Computers & Security 14 pp 541-566
[4] Maryann G, Uma M G and Benjamin X 2016 Classifying Windows Malware with Static Analysis
[5] Sornil O and Liangboonprakong C 2013 Malware Classification Using N-grams Sequential Pattern Features JIPM 4 pp59-67
[6] Mohamed EI B, Mohamed J and Noureddine Z 2015 New malware detection framework based on N-grams and support vector domain description 2015 11th international conference on information assurance and security pp123-128
[7] Schultz M G, Eskin E, Zadok F and Stolfo S J 2000 Data mining methods for detection of new malicious executables In Proceedings IEEE Symposium on Security and Privacy S&P pp 38-49
[8] Islam R, Tian R, Batten L and Versteeg S 2010 Classification of malware based on string and function feature selection Second Cybercrime and Trustworthy Computing Workshop Ballarat pp 9-17

| Analysis method | Reference | Feature | Advantages | Disadvantages |
|-----------------|-----------|---------|------------|---------------|
| Static          | [5]       | N-grams | The larger n-gram size yields the higher accuracy. The 4-gram achieves 96.64% in accuracy. | Time Complexity. |
|                 | [6]       |         | Detects unknown malware with good accuracy. | Did not evaluate false negatives. |
|                 | [8]       | Strings | Detects different families malware with good accuracy. | Did not evaluate false negatives. |
|                 | [9]       | OpCode  | Trained model showed good practical functionality for detecting malwares from benign files. | Did not compare the precision and sensitivity of its algorithm with other techniques. |
|                 | [10]      |         | Trained model showed high accuracy and low false positive rate. | Needed number of dataset to train the classifier. |
| Dynamic         | [14]      | API     | Detects polymorphic and unknown malware. | Large size of graph for comparison. |
|                 | [15]      | File system | The classification time is linear in the number of samples. | The training time is quadratic in the number of samples. |

**Tab.1** Comparison of The Extracted Features
[9] Divandari H, Pechaz B and Jahan M V 2015 Malware detection using Markov Blanket based on opcode sequences *International Congress on Technology, Communication and Knowledge* pp 564-569

[10] Yewale A and Singh M 2016 Malware detection based on opcode frequency *International Conference on Advanced Communication Control and Computing Technologies* pp 646-649

[11] Hu X Large Scale 2011 Malware Analysis Detection and Signature Generation

[12] Hofmeyr S A, Forrest S and Somayaji A 1998 Intrusion detection using sequences of system calls *Journal of computer security* 6 pp151-180

[13] Pektaş A and Acarman T 2017 Malware classification based on API calls and behaviour analysis *IET Information Security* 12 pp107-117

[14] Alkhateeb EMS 2017 Dynamic malware detection using api similarity *IEEE International Conference on Computer and Information Technology* pp 297-301

[15] Hofmeyr S A, Forrest S and Somayaji A 1998 Intrusion detection using sequences of system calls. *Journal of computer security* 6 pp151-180

[16] Cabau G, Buhu M and Oprisa C P 2016 Malware classification based on dynamic behavior *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* pp 315-318