IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

A Theoretical Feature-wise Study of Malware Detection Techniques

Om Prakash Samantray[1*], Satya Narayana Tripathy[2], Susant Kumar Das [3]

[1]Department of Computer Science, Berhampur University, Odisha, India
[2]Department of Computer Science, Berhampur University, Odisha, India
[3]Department of Computer Science, Berhampur University, Odisha, India

[*]*Corresponding Author: om.prakash02420@gmail.com, Tel.: +91-9705172290*

Available online at: www.ijcseonline.org

Accepted: 18/Dec/2018, Published: 31/Dec/2018

***Abstract*—** Malware is the acronym of Malicious Software. It has become a big threat in today's computing world. The threat is increasing with a greater pace as the use of Internet in our day to day activities is growing extensively. The number of malware creators and websites distributing malware is increasing at an alarming rate which attracts researchers and developers to develop a better security solution for it. Developing an efficient malware detection technique is still an ongoing research. Understanding malware, features of malware, analysis methods and detection techniques are the prerequisites of malware research. In this paper, we have studied a few past research works based on API calls, N-Grams, Opcodes features used in malware detection. A detailed fundamental concept of malware detection is also presented in this paper. Use of Data mining algorithms in malware detection, different types of malware detection and analysis methods along with their pros and cons are also presented here. Aim of this paper is to gain prerequisite knowledge of malware research and concepts of malware detection techniques.

***Keywords*—**Malware detection, API call Sequence, Malware feature, Opcode sequence, n-grams, Data mining

I. INTRODUCTION

The definition of malware or malicious software is as follows. Malicious software is a program designed to intrude and damage a computer system & information without the owner's knowledge and permission, which is a serious threat to the security of computer systems from last few decades.

A. Types of Malware

There are various types of malware including virus, worm, Trojan horse, botnets, rootkit, adware, scareware, spyware, ransomware, backdoor, Key loggers, rogue security software and browser hijacker.

1) Virus: The malicious code which attaches itself to other programs or software and then replicates itself is termed as virus.

2) Worm: A stand-alone program which replicates itself and destroys data and files on the computer. Unlike viruses, worms do not require any extra involvement of user to replicate and execute.

3) Trojan horse: It is commonly known as Trojan which is malicious software that disguises itself as a useful file or program and embedded by its designer in an application or system.

4) Bot: Bots are software programs which are designed to perform specific operations automatically and can be controlled remotely. Botnets are special form of bots which can be used in networks to cause distributed Denial of Service attacks. Bots can be used as spambots that may produce advertisements on websites, damage server data as web spiders and so on. In order to prevent bots, websites use CAPTCHA tests which verify users as human.

5) Rootkit: The malicious software which is designed to remotely control and access a computer without being detected security programs or users is known as rootkit. Adware: Adware is the acronym for advertising-supported software which automatically delivers advertisements.

6) Scareware: Scareware is a malicious program masked as trial or free anti-virus software or some other free online malicious trick. It gets installed in the system when the user downloads fake security software or visits a malicious website or opens attachments.

7) Spyware: Spyware is a malware which monitors user activity and gathers personal information like frequently visited pages, financial data, credit card no, email address, account information, keystrokes and many more. It enters a system when free and potentially dangerous software is downloaded and installed without the user's knowledge.

8) Ransomware: Ransomware is a type of malware that holds or blocks a computer system until a demanded ransom or sum of money is paid.

9) Backdoors: Backdoors are similar to trojans or worms, except that they open a "backdoor" onto a computer system,

© 2018, IJCSE All Rights Reserved

providing a network connection for intruders or other malware to enter or for viruses or SPAM to be sent.

10) Keyloggers: Records everything you type on your PC in order to gather your log-in credentials and other sensitive information and sends it on to the source of the keylogging program.

11) Rogue security software: This kind of software deceives or misleads users. It appears to be a good program to remove Malware infections, but all the while it is the malware.

12) Browser Hijacker: This kind of malware redirects the normal browser search activities and gives the results the malware developers want us to see. The major intention of browser hijacker is to make money off the user's web surfing.

B. Malware creators

The people or organizations that create malware are sometimes called as vandals, blackmailers, swindlers and cyber-criminals and spammers. Most of the malicious programs are designed in order to earn money in an illegal manner. In older days, pranksters used to create malware in order to avoid boredom and to increase popularity. But, later malware were used for criminal purposes like stealing financial information, personal information, spying, destroying confidential files and many other criminal activities. Malware creators may be internal or external to organizations. An internal threat is an insider or a trusted developer of an organization capable of inserting malicious code into software before its release to the market. All other persons or organizations may insert the malicious code after releases of the product are called as external threat.

C. Malware Propagation

Malware may attack a computer or mobile device in a number of ways such as through infected email attachments, file sharing, instant messaging, use of third-party software during social networking, use of pirated software and use of USB & other removable media. After entering into the system, malware may damage the system's boot sector, installed software, data files and even the system BIOS which leads to abnormal behaviour of the system.

The main purpose of all malware creators is to insert and distribute their malware across as many computers or mobile devices as possible. This can be done either by social engineering or by infecting a system without the user's knowledge. These methods are often used concurrently and usually include processes to evade antivirus programs installed in those systems.

1) Concealment strategies are used by malware creators to avoid being detected by anti-malware software. As a result of these concealment strategies some malware are changed for each propagation and transmission. Some of the malware encrypt themselves and their malevolent activities which makes it difficult to extract their signature for malware detection. A few of the concealment strategies are given below.

a) Code Obfuscation: In this technique, developers set out actions like adding unnecessary jumps, dead-code insertion, use of garbage commands, register reassignment, instruction substitution, subroutine reordering and code integration/transposition which prevent signature based detection techniques to detect their malware.

b) Code encryption: This is a defensive mechanism which encrypts malware or their malicious activities using encryption algorithm and encryption key. During execution, the malware copies itself and generates and creates a new encrypted version of the malware which contains encryption algorithm and the new key. So, even the encryption key and the encrypted code are changing constantly, but they may get detected because the decoding algorithm is fixed.

c) Oligomorphic strategy: This strategy uses encryption as a defensive mechanism to encrypt malware but uses different decryptor for the new generations. A set of decryptors is maintained by each malware and randomly one decryptor is selected for decryption.

d) Polymorphic strategy: In this strategy millions of decryptors can be generated by changing instructions in the next variant of the malware to avoid signature based detection. In each execution, a new decryptor is created and joined with the encrypted malware body to create a new variant of the malware. Although a large number of different decryptors can be created, but still signature based technique can detect the malwares by identifying the original program with emulation technique.

e) Metamorphic strategy: Metamorphic malware change themselves so that the new instance has no similarity to the original one. Here, instead of creating new decryptor, a new instance or body is created without changing its actions. The malware does not contain any coding engine and automatic changes occur in the malware source code in each transmission.

D. Malware Symptoms

Malware may differ in the way of propagation and infection but, they all can produce similar symptoms. Malware infected computers may exhibit any of the following symptoms:

- Slow computer processing speed.
- Slow web browser speed.
- Network connection problems.
- Increased CPU usage.
- Appearance of strange programs, files or icons.
- Programs running, terminating or reconfiguring themselves.
- System Freezing or crashing.

© 2018, IJCSE All Rights Reserved

- Automatic Modification or deletion of files.
- Emails/messages being sent automatically without user's consent.

Section I presents definition of malware, types of malware, malware creators, symptoms, concealment strategies. Section II contains different malware analysis and detection methods. Section III presents a theoretical literature survey of malware detection strategies based on different malware features. Section IV presents a summarized discussion of the literature survey and section V concludes the paper.

II. MALWARE ANALYSIS AND DETECTION METHODS

A. Malware Analysis

The process of determining purpose, functionality, associated risks and attack lifecycle of malware is known as malware analysis. Malware analysis is an essential step required to develop effective malware detection techniques.

1) Static analysis: Analysing malicious code without executing it is called static analysis. The detection patterns used in static analysis include byte-sequence, n-grams, system call analysis, syntactic library call, data flow graph, control flow graph and opcode (operational code) frequency distribution etc. Before performing static analysis the executable is unpacked, disassembled and decrypted using different disassembler/debugger tools like OllyDbg, IDAPro, HXD, Hexdump and Netwide command to understand the structure of the malware. The disassembler tools display malware code as assembly instructions using which we can observe the intensions, functionalities and patterns to identify the attacker. Packed executables which are difficult to disassemble can be analysed using memory dumper tools like LordPE and OllyDump to obtain protected code located in the system's memory and dump it to a file[1].
Binary obfuscation techniques, which transform the malware binaries into self-compressed and uniquely structured binary files, are designed to resist reverse engineering and thus make the static analysis very expensive and unreliable[1]. The extensive use of evasion techniques by malware creators to spoil static analysis process has become the motivation to develop dynamic analysis technique.

2) Dynamic Analysis: The process of analysing the behaviour of malicious code while executing it is called dynamic analysis. Dynamic analysis is done in a controlled environment using virtual machine, emulator, simulator, sandbox etc. Before executing the malware sample, the appropriate monitoring tools like Process Monitor, and Capture BAT (for file system and registry monitoring), Process Explorer and Process Hackerreplace (for process monitoring), Wireshark (for network monitoring) and Regshot (for system change detection) are installed and activated [1]. Various techniques that can be applied to perform dynamic analysis include function call monitoring, function parameter analysis, information flow tracking, instruction traces and autostart extensibility points etc. [2].
Dynamic analysis is more effective than static analysis because it reveals the malwares' natural behaviour which is difficult to find in static analysis. On the other hand, it is resource consuming and time intensive because it requires an appropriate controlled environment to execute and analyse the file. The virtual environment used in dynamic analysis may be different from the real system environment hence, sometimes malware may act artificially which differs from the original behaviour. Online automated tools used in dynamic analysis are CW-Sandbox, Norman Sandbox, TT-Analyser, Ether, Anubis and Threat expert. The reports of dynamic analysis generated by the tools provide details of malware behaviour and actions performed by them. Then the analysis system represents the report outcomes in an organized way which is later used for classification either by feature vectors or similarity vectors.

3) Hybrid Analysis: This technique is proposed to overcome the limitations of static and dynamic analysis techniques. It firstly analyses the signature specification of any malware code & then combines it with the other behavioural parameters for enhancement of complete malware analysis. Due to this approach hybrid analysis overcomes the limitations of both static and dynamic analysis [3].

B. Malware Detection:

Malware detection techniques are used to detect the malware and prevent the computer system from being infected, protecting it from potential information loss and system compromise. They can be categorized as shown in figure 1. Each of these detection techniques may use one of the three analysis approaches: static, dynamic and hybrid.

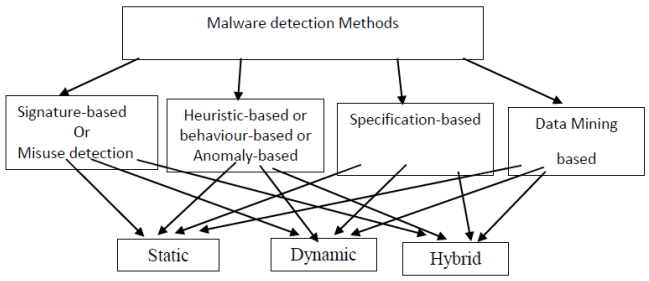

Figure 1. Types of malware detection & Analysis Methods

1) Signature-Based Detection

It is also called as Misuse detection. It maintains the signature database and compares the patterns against database to detect malware. The signatures are created by examining the disassembled code of malware binary. Disassembled code is analysed and features are extracted. These features are used to construct the signature of particular malware family. A library of known code

© 2018, IJCSE All Rights Reserved

signatures is updated and refreshed constantly by the antivirus software vendor so this technique can detect the known instances of malware accurately. The main advantages of this technique is that it can detect known instances of malware accurately, less amount of resources are required to detect the malware and it mainly focus on signature of attack. The major drawback is that it can't detect the new, unknown instances of malware as no signature is available for such type of malware.

2) Heuristic-Based Detection

It is also called as behaviour-based or anomaly-based detection. The main purpose is to analyse the behaviour of known or unknown malwares. Behavioural parameter includes various factors such as source or destination address of malware, types of attachments, and other countable statistical features. It usually occurs in two phases: Training phase and detection phase. During training phase the behaviour of system is observed in the absence of attack and machine learning technique is used to create a profile of such normal behaviour. In detection phase this profile is compared against the current behaviour and differences are considered as potential attacks [4].

The advantage of this technique is that it can detect known as well as new, unknown instances of malware and it focuses on the behaviour of system to detect unknown attack or zero-day attack. The disadvantage of this technique is that it needs to update the data describing the system behaviour and the statistics in normal profile but it tends to be large. It needs more resources like CPU time, memory and disk space. Other limitations include high false positive rate and difficulty in selecting features to be learned in the training phase.

3) Specification-Based Detection

It is derivative of behaviour-based detection that tries to overcome the typical high false alarm rate associated with it. Specification based detection relies on program specifications that describe the intended behaviour of security critical programs [4]. It involves monitoring program executions and detecting deviation of their behaviour from the specification, rather than detecting the occurrence of specific attack patterns. This technique is similar to anomaly detection but the difference is that instead of relying on machine learning techniques, it will be based on manually developed specifications that capture legitimate system behaviour [4]. The advantage of this technique is that it can detect known and unknown instances of malware and level of false positive is low but level of false negative is high and not as effective as behaviour based detection in detecting new attacks; especially in network probing and denial of service attacks. Development of detailed specification is time consuming.

4) Data mining based detection

From last decade data mining has been the main focus of many malware researcher for detecting the new, unknown malwares; hence, data mining can be considered as the fourth proposed malware detection technique. Probably the Data Mining is now been dominated by Machine Learning techniques. In 2001 Schultz et al. [5] first introduced the idea of applying the data mining and machine learning method for the detection of new, unknown malware based on their respective binary codes. Machine learning algorithms are used for detecting patterns or relations in data, which are further used to develop a classifier [6]. The common method of applying the data mining technique for malware detection is to start with generating a feature sets. These feature sets include instruction sequence, API/System call sequence, hexadecimal byte code sequence (n-gram) etc. The numbers of extracted features are very high so various text categorization techniques are applied to select consistent features and generate the training and test feature sets. Then classification algorithms are applied on the consistent training feature set to generate and train the classifier and test feature set is examined by using these trained classifiers. The performance of each classifier is evaluated by identifying the rate of False Positive, False Negative, True Positive, True Negative and calculate the TPR, FPR, Recall, precision and F1-measure. The advantage of data mining based detection is that detection rate is high as compared to signature based detection method [5]. It detects the known as well as unknown, new instances of malware.

III. LITERATURE SURVEY

Heuristic malware detection methods use data mining and machine learning (ML) techniques to learn the behaviour of an executable file. These ML techniques require some features like API (Application Programming Interface) calls, CFG (Control Flow Graph), N-Gram, Opcodes and Mixed features. This section presents a survey of malware detection mechanisms based on these features.

A. API calls

The behaviour of a piece of code like malware can easily be reflected using API sequences because; almost all programs send their requests to the Operating System using API calls. Hofmeyr et al. [7] were among the first ones who observed API call sequences as a feature of a malware. They presented an anomaly detection method which used the system call sequences as feature. They used Hamming distance with a specific threshold for matching system call sequences and to identify anomalies. Usually, large Hamming distance value reported as anomalies. Yuxin Ding et al. [8] proposed an API-based Object oriented association mining method for malware detection. They have used many strategies to improve the rule quality, API selection criteria to remove the rules which are redundant which in turn increases the running speed of the OOA malware detection.

Jeong and Lee [9] used system call sequences for both malicious and benign executables to build a topological graph which is called code graph. This graph is extracted for every binary program and is compared with the code graph of

© 2018, IJCSE All Rights Reserved

malicious and benign programs. Based on this comparison, a program is classified as malware or benign. Ye et al. [10] proposed an interpretable classifier based on the analysis of API calls by a PE file for detecting malware from large and imbalanced gray-list. They have studied around 8,000,000 malware and benign files with 100,000 samples from the gray-list collected from lab of King Soft Corporation and built effective associative classifier based on several post processing methods like rule pruning and rule reordering. Then, to make the classifier less sensitive to the imbalance dataset and improve its performance, they developed the Hierarchical Associative Classifier (HAC).

Wespi et al. [11] proposed an improved version with variable length system call sequences. A detection method based on the frequency of system calls has been proposed by Sato et al. [12]. Manzoor et al. [13] collect some Windows malicious executable from VX Heavens and their API call sequences are monitored by API Monitor. The DCA (Dendritic Cell Algorithm) [14] is applied for detection. Later, Ahmed et al. [15] use statistical features which extracted from both spatial (arguments) and temporal (sequences) information available in Windows API calls for malware detection. All these methods use system calls or API calls to monitor program behaviour. Seifert et al. [16] compared three popular event-based techniques that can monitor program behaviour: user mode API hooking, kernel mode API hooking, and kernel mode call backs.

M. K. Shankarapani et al. [17] presented two techniques such as, Static Analyser for Vicious Executables (SAVE) and Malware Examiner using disassembled Code (MEDiC). MEDiC uses assembly calls for analysis and SAVE uses API calls for analysis. They presented that assembly can be superior to API calls as it allows a more detailed comparison of executables. On the other hand, API calls can be superior to Assembly for its speed and its smaller signature. They proved that both these proposed techniques can provide a better detection performance against obfuscated malware.

Dolly Uppal et. Al. [18] have used a feature selection algorithm based on Fisher Score to select distinct APIs. Then they have applied machine learning techniques like SVM, Naïve Bayes, J-48, Random Forest, KNN, ANN and Voted Perceptron on the selected feature vector. They claimed that SVM shows the highest accuracy of 98.4 and ANN shows lowest accuracy of 78.2 among these 7 selected algorithms.

Ehab M. Alkhateeb [19] has proposed a dynamic malware detection method based on API similarity. He used API filtration method to remove all duplicate and repeated API calls and then created different patterns for different malware. The patterns were then matched to find groups of malware having similar patterns. He proposed API similarity algorithm to find similarity among the files by finding distance between strings present in those files.

B. OpCode

An Opcode (Operational Code) is the part of a ML instruction that identifies the operation to be executed. More specifically, instructions of a program are defined as a pair composed of an operational code and an operand or a list of operands. The most significant research on Opcode has been done by Bilar [20]. He showed the ability of single Opcode to use as a feature in malware detection. Santos et al. [21] presented various type of malware detection techniques based on Opcode sequences. In their first work, they presented an approach focused on detecting obfuscated malware variants using the appearance frequency of Opcode sequences in order to build a representation of executable files. To do so, they had applied the disassembly process on exe files and built an opcode profile containing a list of Opcodes from the generated assembly files and then they computed the relevance of each Opcode based on the frequency of appearance of each of them in both malware and benign datasets using mutual information [22]. Finally they used Weighted Term Frequency (WTF) [23] to make suitable feature vector extracted from executables. They calculated the Cosine similarity measure between new instance feature vector and malware variants feature vector for detection. Later, in their next work, Santos et al. [23] presented a new feature extraction method based on Opcode sequences and trained several machine learning classifiers by embedding the extracted features.

As we know, the machine learning based classifiers requires high number of samples for each of the concept classes they try to detect and it is quite difficult to obtain this amount of labelled data in real world. So, Santos et al., in their next research, proposed several methods to eliminate this limitation such as Collective classification [24], Single class learning [25], and Semi supervised learning [26]. Runwal et al. [27] proposed a new approach based on Opcodes and used this method for detecting unknown and also metamorphic malware families based on a simple graph similarity measurement. They extracted Opcodes from both file types (i.e. malware and benign), count the number of each pair Opcodes appeared in them and based on the numbers, make a graph of Opcodes and after that can predict the maliciousness of a new executable by calculating the similarity of graph obtained from the executable and both file types and finally the file will be classified either as benign or malware. More similar work was done by Shabtai et al. [28] who tried to detect unknown malicious codes by applying classification techniques on Opcode patterns. They created a dataset of malicious and benign executables for the Windows operating system. After disassembling the executables, they calculated the normalized term frequency (TF) and TF Inverse Document Frequency (TF-IDF) representations as a feature for each file. Finally, they used several classical classification techniques such as Support Vector Machine (SVM), Logistic

© 2018, IJCSE All Rights Reserved

Regression (LR), Artificial Neural Networks (ANN) etc. to evaluate the proposed feature selection method.

Hamid Divandari et. Al. [29] presented an opcode based method which used Markov Blanket algorithm as a feature selection method to reduce the number of features. They have 5 classes of malware such as: Worms, Backdoors, Trojan horses, Viruses and Rootkits. For each class of malware, one special Hidden Markov Model was developed and trained.

Cheng Wang et. al. [30] proposed a model to extract opcodes profiles of sample files through static analysis and selected the sequences having higher entropy value as representation of malware instance. Then they have used a Fast Density-Based Clustering algorithm for clustering similar malware instance which determine whether an unknown instance is malware or benign according to the cluster.

Jixin Zhang et.al. [31] have used image detection mechanism in which images are reconstructed by opcodes sequences of length two. They have used histogram normalization, dilation and erosion to enhance the contrast between malware variant images and benign images. Then convolutional neural network (CNN) is used as a learning approach to recognize malware variant images. A much similar work was done by Tingting Wang, Ning Xu in the paper [32].

C. N-grams

N-grams are all substrings of a larger string with a length of N [28]. For example, the string "TOMATO" can be segmented into several 3-grams: "TOM", "OMA", "MAT" and "ATO". Tesauro et al. [33] were the first who try to use N-Grams as a feature for malware detection domain. They used N-Grams to detect Boot Sector Viruses using Artificial Neural Networks (ANN). A Boot Sector Virus is a malware variant which infects DOS Boot Sector or Master Boot Record (MBR). When a system has infected, the MBR is usually ruined and the computer boot order is changed The N-Grams was selected from most frequent sections in malware and benign executables. They used a specific feature reduction algorithm such that each malware must consist of at least four N-Grams from existing N-Grams set. Tesauro et al. [34], in their next study, used N-Grams to build several classifiers based on ANN and also used a specific voting strategy to achieve final results. In that research a simple threshold value was used to reduce the number of N-Grams.

Abou-Assaleh et al. [35] presented a framework that uses the Common N-Gram method and the K-Nearest-Neighbour (KNN) classifier for malware detection. For both classes (i.e. malicious and benign) a delegate profile was built. A new instance was matched with the profiles of both classes and was assigned to the most similar one. Kolter and Maloof [36] used byte N-Gram representation to detect unknown malware. Though the vector of N-Gram features was binary, presenting the attendance or non-attendance of a feature in the file. In an extension of their previous study, Kolter and Maloof [37] classified malware into several families based on the functions in their respective payload attempting to approximate their capability to detect malicious codes based on their subject dates.

Wang et al. [38] proposed a method which uses data mining as detection category to classify various file types based upon their file prints. An n-gram analysis method was used and the distribution of n-grams in a file was used as its file print. The distribution was given by byte value frequency distribution and standard deviation. These file prints represented the normal profile of the files and were compared against file prints taken at a later time using simplified Mahalanobis distance. A large distance indicated a different n-gram distribution and hence maliciousness. Schultz et al. [5] proposed a static misuse detection method using data mining as detection category where strings data were used to fit a naive-Bayes classifier while n-grams were used to train a multi naive Bayes classifier with a voting strategy. Dataset partitioning and 6-Naive-Bayes classifier trained on each partition of data. They used different feature classifiers that do not pose a fair comparison among the classifiers. Naive-Bayes using strings gave the best accuracy in their model. Extending the same idea, Schultz et al. [39] created MEF, Malicious Email Filter, that integrated the scheme described in [5] into a Unix email server where a large dataset containing 3301 malicious and benign program was used to train and test a Naive-Bayes classifier. For feature reduction, the dataset was partitioned into 16 subsets. Each subset is differently trained on a different classifier and a voting strategy was used to obtain final outcome. InSeon Yoo [40] proposed a static misuse detection using data mining where they used Self Organizing Maps (SOM). N-grams are extracted from the infected programs and SOM's were trained on this data. They claimed that each Virus has its own DNA like character that changes the SOM projection of the program that it infects. The method looks for change in the SOM projection as a result of Virus infection. Hence, it is able to detect Polymorphic and metamorphic malwares.

Zhang Fuyong & ZhaoTiezhu [41] proposed a malware classification method based on n-grams attribute similarity. They have extracted all n-grams of byte codes from training samples and selected the most relevant as attributes. They used similarity analysis method to determine the sample either as malware or benign. They have presented a comparative study of their method with 4 other machine learning methods such as: Naïve Bayes, Bayesian Networks, Support Vector Machine and C4.5 Decision Tree. They claimed that their method outperforms the other four methods.

D. Control Flow Graph (CFG)

CFG is a directed graph, where each node represents a statement of the program and each edge represents control flow between the statements. Statements may be assignments, copy statements, branches etc. In [42], authors performed a

© 2018, IJCSE All Rights Reserved

set of normalization operation after disassembling an executable program for reducing effects of mutation techniques and unveiling the flow connections between benign and malicious code. Then they generated corresponding CFG for the program. CFG compared against the CFG of a normalized malware in order to know whether CFG contains a sub graph which is isomorphic to CFG of the normalized one. Thus, the problem of detecting malware is changed to the sub-graph isomorphism problem. Zhao [43] proposed a detection method based on features of the control flow graph for PE files. At first, he created CFG for each executable file. Then, he used features which extracted from CFG as the train data. These features are information about nodes, edges and sub graphs. After feature selection, some data mining algorithm have been used for classification based on these features such as Decision Tree, Bagging and Random Forest. Bonfante et al. [44] built CFG based on six types of nodes such as, jmp, jcc, call, ret, inst and end. Then, they reduced these nodes in this way: for any node of kind inst or jmp, they removed the node from the graph and linked all its predecessors to its unique successor. After reduction, they used this graph as a signature for each file.

B. Anderson et al. [45] introduced a novel malware detection algorithm based on the analysis of graphs constructed from dynamically collected instruction traces of the target executable. These graphs represent Markov chains, where the vertices are the instructions and the transition probabilities are estimated by the data contained in the trace. They used a combination of graph kernels to create a similarity matrix between the instruction trace graphs. The resulting graph kernel measures similarity between graphs on both local and global levels. Finally, the similarity matrix is sent to a support vector machine to perform classification. They used the data representation to perform classification in graph space rather than using n-gram data. Ming Xu et al. [46] have used disassembled codes of program, the caller–callee relationships of functions and the opcode information about functions to create the function-call graph and graph colouring techniques were used to measure the similarity metric between two function-call graphs. The similarity metric was used to identify the malware variants from known malwares.

Shahid Alam [47] presented a method named, named Annotated Control Flow Graph (ACFG) to efficiently detect metamorphic malware. ACFG is built by annotating CFG of a binary program and is used for graph and pattern matching to analyse and detect metamorphic malware. They claimed that ACFG is more accurate than CFG.

E. Mixed Features

Mikhail Zolotukhin and Timo Hamaainen [48] have analysed executable files to get opcodes sequences and then applied n-gram models to find essential features from those sequences. They used a clustering algorithm based on the iterative usage of support vector machines to build a benign software behaviour model which is used to detect malicious executables within new files. They claimed that their scheme can detect unseen malware with a greater accuracy.

Ding Yuxin et. Al. [49] have constructed the opcode running tree to simulate the dynamic execution of a program, and extracted opcode n-grams to represent the features of an executable. They have used three classifiers such as: KNN, decision tree (C4.5) and support vector machine to classify malicious and benign files.

Yuxin Ding et. al. [50] used the n-gram model to generate the opcode n-grams of different lengths and then used information gain and document frequency to select opcode-based features. They have applied Deep Belief Networks to detect malware which was claimed as a better malware detector as compared to other classification techniques like SVM, decision tree and KNN.

IV. DISCUSSION

Malware detection is a continuous research. Though several techniques have been evolved, we can't completely eliminate malware from this internet based digital world. But definitely we can minimize its harmful effect by continuously working on it, to develop new, robust and effective detection techniques with higher accuracy rate. Data mining and Machine learning techniques play a vital role in this regard. In addition to this, best malware features should also be extracted from different types of malware before the machine learning model is implemented. Zero-day attack is the biggest headache for malware researchers because the new malware may possess new feature and behaviour that, the current anti-malware may not understand. Hence, a strong future malware prediction system which will build a model not only on present behaviour and feature of malware but also on future behaviour and features. Artificial Intelligence techniques along with machine learning may help the researchers to develop such a system.

V. CONCLUSION

In this paper, we have presented basic definition of malware, its types and its propagation strategy. Further we have presented advantages and disadvantages of different ways the malware can be analysed and detected. Then in the literature survey we have surveyed a number of malware detection strategies based on APIs, OPCODE, n-grams, Control Flow Graph and mixed features. The main aim and objective of this paper is to understand the basics of malware detection which will be very helpful in further research. In the next work, we will focus on practical approaches of malware analysis and how the static & dynamic analysis methods can be merged to model an efficient and robust malware detection technique. Different classification algorithms of machine learning can be applied with different-sized malware datasets. A comparison analysis of the above process will also be included in our next work.

© 2018, IJCSE All Rights Reserved

REFERENCES

[1] Ekta Gandotra, Divya Bansal, Sanjeev Sofat, *"Malware Analysis and Classification: A Survey",* Journal of Information Security, April 2014, pp: 56-64

[2] Egele, M., Scholte, T., Kirda, E. and Kruegel, C. *, "A Survey on Automated Dynamic Malware-Analysis Techniques and Tools",* Journal in ACM Computing Surveys, 44,2012, Article No. 6.

[3] Kirti Mathur, Saroj Hiranwal, *"A Survey on Techniques in Detection and Analyzing Malware Executables",* International Journal of Advanced Research in Computer Science and Software Engineering, April 2013, Volume 3, Issue 4, ISSN: 2277 128X.

[4] Robiah Y, Siti Rahayu S., Mohd Zaki M, Shahrin S., Faizal M. A., Marliza R., *"A New Generic Taxonomy on Hybrid Malware Detection Technique, (IJCSIS)International Journal of Computer Science and Information Security",* Vol. 5, No. 1, 2009.

[5] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo, *"Data Mining Methods for Detection of New Malicious Executables",* in Proceedings of the Symposium on Security and Privacy, 2001, pp. 38-49.

[6] Raja Khurram Shahzad, Niklas Lavesson, Henric Johnson, *"Accurate Adware Detection using Opcode Sequence extraction",* in Proc. of the 6th International Conference on Availability, Reliability and Security (ARES11),Prague, Czech Republic. IEEE, 2011, pp. 189-195.

[7] S. Hofmeyr, S. Forrest, and A. Somayaji, *"Intrusion detection using sequences of system calls."* Journal of Computer Security, , 1998, pp. 151–180.

[8] Yuxin Ding , Xuebing Yuan, Ke Tang, Xiao Xiao, Yibin Zhang," *A fast malware detection algorithm based on objective-oriented association mining",* computers & s e c u rity 3 9, 2013, pp: 3 1 5 - 3 2 4, Elsevier.

[9] K. Jeong and H. Lee, *"Code graph for malware detection. In Information Networking."* ICOIN. International Conference on,. Jan 2008.

[10] 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".* Journal of Intelligent Information Systems, 35(1),2010, pp.1-20.

[11] Wespi, A., Dacier, M., Debar, H.*: Intrusion detection using variable- length audit trail patterns.* In: Proceedings of the Recent Advances in Intrusion Detection, 2000 , pp. 110–129. Springer, France

[12] Sato, I., Okazaki, Y., Goto, S.: *An improved intrusion detection method based on process profiling.* IPSJ J. 43,2002, 3316–3326 (2002).

[13] Manzoor, S., Shafiq, M.Z., Tabish, S.M., Farooq, M.: *A sense of 'danger' for windows processes.* In: ICARIS. LNCS, vol. 5666,2009, pp. 220–233. Springer, Heidelberg .

[14] Greensmith, J., Aickelin, U.: *The deterministic dendritic cell algorithm.* In: Proceedings of the ICARIS. LNCS, vol. 5132,2008, pp. 291– 303. Springer, Heidelberg.

[15] Ahmed, F., Hameed, H., Shafiq, M.Z., Farooq, M.: *Using spatio-temporal information in API calls ith machine learning algorithms for malware detection.* In: Proceedings of the ACM Conference on Computer and Communications Security, 2009, pp. 55–62.

[16] Seifert, C., Steenson, R., Welch, I., Komisarczuk, P., Endicott-Popovsky, B.*: Capture-a behavioral analysis tool for applications and documents.* Digit. Investig. 4(Suppl. 1), 2007, S23–S30 .

[17] Madhu K. Shankarapani · Subbu Ramamoorthy , Ram S. Movva · Srinivas Mukkamala, *"Malware detection using assembly and API call sequences",* J Comput Virol ,2011, Springer, pp:107–119

[18] Dolly Uppal, Rakhi Sinha, vishakha Mehra and Vinesh Jain, *"Exploring Behavioural Aspects of API calls for Malware Identification and Categorization",* 6th int. conf. on computational intelligene and comm. Networks,IEEE,2014,pp:824-828.

[19] Ehab M. Alkhateeb, "Dynamic Malware Detection using API Similarity", *International Conference on Computer and Information Technology* ,IEEE, 2017, pp:297-301

[20] D. Bilar, "OpCodes as predictor for malware," *International Journal of Electronic Security and Digital Forensics,* vol. 1, no. 2, 2007, pp. 156.

[21] I. Santos, F. Brezo, J. Nieves, and Y. Penya, *"Idea: OpCode-sequence based malware detection,",* Engineering Secure Software and System , 2010.

[22] C. Peng, H. Long and F. Ding, *"Feature selection based on mutual information: cri-teria of max-dependency, max-relevance, and minredundancy.,"* in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005.

[23] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, *"OpCode sequences as representation of executables for data-mining-based unknown malware detection,"* Information Sciences, Aug. 2011.

[24] I. Santos, C. Laorden, and P. Bringas, *"Collective classification for unknown malware detection,"* Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, 2011.

[25] I. Santos, F. Brezo, B. Sanz, C. Laorden, and P. G. Bringas, *"Using opCode sequences in single-class learning to detect unknown malware,"* IET Information Security, vol. 5, no. 4, 2011, p. 220.

[26] I. Santos, B. Sanz, and C. Laorden, *"OpCode-sequence-based semi-supervised unknown malware detection,",* Computational Intelligence in Security for Information Systems , 2011.

[27] N. Runwal, R. M. Low, and M. Stamp, *"OpCode graph similarity and metamorphic detection,"* Journal in Computer Virology, vol. 8, no. 1–2, Apr. 2012, pp. 37–52.

[28] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, *"Detecting unknown malicious code by applying classification techniques on OpCode patterns*," Security Informatics, vol. 1, no. 1, p. 1, 2012.

[29] Cheng Wang et.al. ," *A malware variants detection methodology with an opcode based feature method and a fast density based clustering algorithm",* 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery ,IEEE,2016,pp:481-487.

[30] Yuxin Ding et. al., *Application of Deep Belief Networks for Opcode Based Malware Detection,* International Joint Conference on Neural Networks (IJCNN),IEEE,2016,pp:3901-3908.

[31] Jixin Zhang et.al. *"IRMD: Malware variant Detection using opcode Image Recognition",* 22nd International Conference on Parallel and Distributed Systems, IEEE, 2016,pp:1175-1180.

[32] Tingting Wang, Ning Xu, *"Malware Variants Detection Based on Opcode Image Recognition in Small Training Set",* 2nd International Conference on Cloud Computing and Big Data Analysis,IEEE,2017, pp:328-332.

[33] G. B. S. Gerald, J. Tesauro, Jeffrey O. Kephart, *"Neural Network for Computer Virus Recognition."* IEEE Expert, 1996.

[34] W. A. and G. Tesauro, *"Automatically Generated Win32 Heuristic Virus Detection,"* in Virus Bulletin Conference, 2000.

[35] T. Abou-assaleh, N. Cercone, V. Keselj, and R. Sweidan, *"N-gram-based Detection of New Malicious Code,"* Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.

[36] M. M. Kolter JZ, *"Learning to detect malicious executables in the wild."* in roc of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.

[37] J. Z. Kolter and M. A. Maloof, *"Learning to Detect and Classify Malicious Executables in the Wild,"* vol. 7, pp. 2721–2744, 2006.

© 2018, IJCSE All Rights Reserved

[38] K.Wang W. Li and, S. Stolfo, , and B. Herzog. *"Fileprints: Identifying File Types by n-gram Analysis."* In 6th IEEE Information Assurance Workshop, 2005.
[39] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, Manasi Bhattacharya, and Salvatore J. Stolfo. *"MEF: Malicious Email Filter: A UNIX mail Filter That Detects Malicious Windows Executables."* pp. 245–252, 2001.
[40] InSeon Yoo. *"Visualizing windows executable viruses using self-organizing maps."* In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 82–89, 2004.
[41] Zhang Fuyong & ZhaoTiezhu, *"Malware Detection and Classification Based on ngrams Attribute Similarity",* International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), IEEE, 2017,pp: 793-796.
[42] D. Bruschi, L. Martignoni and M. Monga *"Detecting self-mutating malware using control-flow graph matching,"* In: Büschkes, R. And Laskov, P. (eds) Detection of Intrusions and Malware & Vulnerability Assessment, volume 4064 of LNCS, pp 129–143. Springer, Berlin. 2006.
[43] Z. Zhao, *"A virus detection scheme based on features of Control Flow Graph."* 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), pages 943- 947, 2011.
[44] G. Bonfante, M. Kaczmarek, J.Y. Marion. *''Control Flow Graphs as Malware Signatures.''* WTCV, May, 2007.
[45] Blake Anderson, Daniel Quist, Joshua Neil, Curtis Storlie, Terran Lane *"Graph-based malware detection using dynamic analysis"* J Comput Virol (2011) 7:247–258, Springer-Verlag France 2011.
[46] Ming Xu, Lingfei Wu, Shuhui Qi , Jian Xu, Haiping Zhang , Yizhi Ren, Ning Zheng , *"A Similarity metric method of obfuscated malware using function-call graph",* J Comput Virol Hack Tech (2013) 9,pp :35–47, Springer-Verlag, France, 2013.
[47] Shahid Alam et. al., *"Annotated Control Flow Graph for Metamorphic Malware Detection",* Security in Computer Systems and Networks , The Computer Journal, 2014.
[48] Mikhail Zolotukhin and Timo Hamaainen , *"Detection of Zero-day Malware Based on the Analysis of Opcode Sequences*", 11th Annual IEEE CCNC - Security, Privacy and Content Protection, 2014,pp:386-391.
[49] Ding Yuxin et. Al., *Malicious Code Detection Using Opcode Running Tree Representation*, Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing,IEEE,2014,pp:616-621.
[50] Yuxin Ding et. al., *Application of Deep Belief Networks for Opcode Based Malware Detection*, International Joint Conference on Neural Networks (IJCNN),IEEE,2016,pp:3901-3908.

Authors Profile

Mr. Om Prakash Smantray pursed Bachelor of Engineering in Information Technology from Berhampur University, Odisha, India in 2006 and Master of Technology in CSE from BPUT Odisha in year 2010. He is currently pursuing Ph.D. in Department of Computer Science, in Berhampur University, Odisha, India since 2015. He is a life-member of ISTE since 2017. His main research work focuses on Information Security, Data Mining, Machine Learning and Big Data Analytics, He has 10 years of teaching experience and 3 years of Research Experience.

Dr. Satya Narayan Tripathy received his M.C.A. and Ph.D. degrees in Computer Science from Berhampur University, Berhampur, Odisha, India in the years 1998 and 2010, respectively. He has been teaching in the Department of Computer Science, Berhampur University since 2011. Currently, he is a Lecturer in the Department of Computer Science, Berhampur University. Dr. Tripathy serves on the advisory boards of several organizations and conferences. He is a Life Member of Computer Society of India (LMCSI), Life Member of Orissa Information Technology Society (LMOITS) and Member of several professional bodies. His research interests include computer network security, wireless ad hoc network, network security in wireless communication and data mining.

Dr. Susant Kumar Das received his Ph.D. degree from Berhampur University, Odisha, India in 2006. Dr. Das is currently a Reader at the Department of Computer Science. He is a life member of IEEE, ISTE, SGAT, OITS and member of several professional bodies. His research interests include Data Communication & Computer Networks, Computer Security, Internet & Web Technologies, Database Management Systems and Mobile Ad- Hoc Networking & Applications.

© 2018, IJCSE All Rights Reserved