A Study on Malware Taxonomy and Malware Detection Techniques

Satya Narayan Tripathy[1], S. K. Das[2], Brojo Kishore Mishra[3], Om Prakash Samantray[4]
[1,2,4]Department of Computer Science, Berhampur University, Berhampur, India.
[3]C. V. Raman College of Engineering, Bhubaneswar, India.

***Abstract*— Malware is one of the most serious security threats on the Internet today. The threat is increasing in a greater pace with the intensive use of networks and Internet in our day-to-day activities. The most recent reports emphasize that the invention of malicious software is rapidly increasing. Over the last decade, a number of studies have been made on malware and their countermeasures. Researchers and manufacturers are making great efforts to invent effective malware detection methods to produce anti-malware systems for better protection of computers and networks. In this paper, a detailed study has been conducted on malware taxonomy and the approaches made by the researchers to improve anti-malware or malware detection systems, giving emphasis on signature-based and data mining based techniques in malware detection. Thus, it provides an up-to-date comparative reference to the researchers and developers of malware detection systems.**

Keywords—Malware, anti-malware, malware detection systems, data mining, Signature-based.

I. INTRODUCTION

Software which is specifically designed to disrupt or damage a computer system is known as malware. Malware is short for "malicious software" - computer programs designed to infiltrate and damage computers without the users consent. It is code or software that is specifically designed to damage, disrupt, steal, or in general inflict some other "bad" or illegitimate action on data, hosts, or networks. Malware creators or malware writers started off writing malware in the early 1980's. Until the late 1990's most of the malwares were just pranks made up in order to annoy users and to see how far a malware could spread but, in the late 1990's and early 2000's, as the internet had become everyone's tool for communication, marketing, business and banking, the malware writers and hackers began to put their talent to more professional and sometimes criminal use. Today many experts believe the amount of malicious software being released on the web might actually surpass the release of valid software.

The term malware includes viruses, worms, Trojan Horses, rootkits, spyware, adware, keyloggers, botnet and more. To get an overview of the difference between all these types of threats and the way they work, it makes sense to divide them into groups [1].

A. Viruses and worms - the contagious threat

Viruses and worms are defined by their behaviour – malicious software designed to spread without the user's knowledge. A virus infects legitimate software and when this software is used by the computer owner it spreads the virus – so viruses need you to act before they can spread. Computer worms, on the other hand, spread without user action. Both viruses and worms can carry a so-called "payload" – malicious code designed to do damage. A virus is a type of malware that propagates by inserting a copy of itself into and becoming part of another program. It spreads from one computer to another, leaving infections as it travels. Almost all viruses are attached to an executable file, when the file is executed; the viral code is executed as well. Viruses spread when the software or document they are attached to is transferred from one computer to another using the network, a disk, file sharing, or infected e-mail attachments. Unlike viruses, worms are standalone softwares and do not require a host program or human help to propagate. To spread, worms either exploit vulnerability on the target system or use some kind of social engineering to trick users into executing them [1].

B. Trojans, rootkits and adware – the masked threat

Trojans and rootkits are grouped together as they both seek to conceal attacks on computers. Trojan Horses are malignant pieces of software pretending to be benign applications. Users therefore download them thinking they will get a useful piece of software and instead end up with a malware infected computer. Rootkits are a masking technique for malware, but do not contain damaging software. Rootkit techniques were invented by virus writers to conceal malware, so it could go unnoticed by antivirus detection and removal programs. Trojan is named after the wooden horse the Greeks used to infiltrate Troy. It is a harmful piece of software that looks legitimate. Users are typically tricked into loading and executing it on their systems. After it is activated, it can achieve any number of attacks on the host, from irritating the user (popping up windows or changing desktops) to damaging the host (deleting files, stealing data, or activating and spreading other malware, such as viruses). Trojans are also known to create back doors to give malicious users access to the system. Unlike viruses and worms, Trojans do not reproduce by infecting other files nor do they self-replicate. Trojans must spread through user interaction such as opening an e-mail attachment or downloading and running a file from the Internet. Adware or Advertising-supported software automatically plays, displays or downloads advertisements to a computer after malicious software is installed or application is used. This kind of code is also embedded into free software. The most common source of adware programs are free games, Peer to peer clients like Kazaa, Bearshare etc [1].

C. *Spyware and keyloggers – the financial threat*

Spyware and keyloggers are malware used in malicious attacks like identity theft, phishing and social engineering - threats designed to steal money from unknowing computer users, businesses and banks. Spyware is a collective term used for software which monitors or gathers personal information about the user like ,the pages frequently visited, email address, credit card no, key pressed by user etc. It enters a system when free or trial software is downloaded and installed without the user's knowledge. It changes the settings of yours browser and adds abdominal browser toolbars [2].

II. MALWARE ANALYSIS TECHNIQUE

Malware analysis is necessary to develop effective malware detection technique. It is the process of analyzing the purpose and functionality of a malware, so the goal of malware analysis is to understand how a specific piece of malware works so that defenses can be built to protect the organization's network. There are three types of malware analysis which achieve the same goal of explaining, how malware works, their effects on the system but the tools, time and skills required to perform the analysis are very different.

A. *Static analysis*

Analysis of the infected file without executing it is known as static analysis. It is also known as code analysis. It is the process of analyzing the program by examining it i.e. software code of malware is observed to gain the knowledge of how malware's functions work. In this technique reverse engineering is performed by using disassemble tool, decompile tool, debugger, source code analyzer tools such as IDA Pro and Ollydbg in order to understand structure of malware [3]. In this approach, we extract low-level information such as Control Flow Graphs (CFGs), Data-Flow Graphs (DFGs) and System call analysis. This information can be gathered by disassembling or decompiling the infected file using different tools as mentioned earlier. Before program is executed, static information is found in the executable including header data and the sequence of bytes is used to determine whether it is malicious. Disassembly technique is one of the techniques of static analysis. With static analysis executable file is disassembled using disassembling tools like XXD, Hex dump, NetWide command, to get the assembly language program file. From this file the opcode is extracted as a feature to statically analyze the application behaviour to detect the malware [5]. Sometimes analyzing the infected file in a different environment to avoid auto execution of the malware is better. Using static analysis we get fast, safe and low false positives and we trace all paths, which helps in terms of getting a lot of information to analyze. On the other hand static analysis may fail in analyzing unknown malware that uses code obfuscation techniques.

B. *Dynamic analysis*

It is also called as behavioral analysis. Analysis of infected file during its execution is known as dynamic analysis [4]. Infected files are analyzed in simulated environment like a virtual machine, simulator, emulator, sandbox etc. [1]. After that malware researchers use SysAnalyzer, Process Explorer, ProcMon, RegShot, and other tools to identify the general behaviour of file [3]. In dynamic analysis the file is detected after executing it in real environment, during execution of file its system interaction, its behaviour and effect on the machine are monitored. The advantage of dynamic analysis is that it accurately analyses the known as well as unknown, new malware. It's easy to detect unknown malware also it can analyze the obfuscated, polymorphic malware by observing their behaviour but this analysis technique is more time consuming. It requires as much time as to prepare the environment for malware analysis such as virtual machine environment or sandboxes [5]. Dynamic analysis fails to detect activities of interest if the target changes its behavior depending on trigger conditions such as existence of a specific file or specific day as only a single execution path may be examined for each attempt.

C. *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 behavioral parameters for enhancement of complete malware analysis. Due to this approach hybrid analysis overcomes the limitations of both static and dynamic analysis [1].

III. MALWARE DETECTION METHODS

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 into signature-based detection, heuristic-based detection, specification-based and data mining based detection as shown in figure 1.

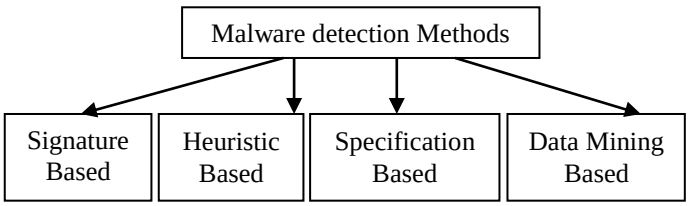

Figure 1: Types of malware detection Methods

A. *Signature-Based Detection*

It is also called as Misuse detection. It maintains the database of signature and detects malware by comparing pattern against the database. The signatures are created by examining the disassembled code of malware binary. Disassembled code is analyzed and features are extracted. These features are used in constructing the signature of particular malware family. A library of known code 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.

B. Heuristic-Based Detection

It is also called as behaviour or anomaly-based detection. The main purpose is to analyze the behaviour of known or unknown malwares. Behavioral 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 [6]. 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. 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 need more resources like CPU time, memory and disk space and level of false positive is high.

C. 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 [6]. 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 [6]. 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.

D. Data mining based detection

From last decade data mining has been the main focus of many malware researcher for detecting the new, unknown malwares; they have added data mining as a fourth proposed malware detection technique. In 2001 Schultz [7] 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. Then different studies have been conducted for detection of different malwares. Data mining helps in analyzing the data, with automated statistical analysis techniques, by identifying meaningful patterns or correlations. The results from this analysis can be summarized into useful information and can be used for prediction. Machine learning algorithms are used for detecting patterns or relations in data, which are further used to develop a classifier [8]. 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 survey of various feature selection technique & classification technique used for data mining is presented in [9]. The advantage of data mining based detection is that detection rate is high as compared to signature based detection method [7]. It detects the known as well as unknown, new instances of malware.

IV. MALWARE DETECTION TECHNIQUES

Signature based and behavior based malware detection methods have some disadvantages. Hence, heuristic malware detection methods are proposed to overcome these disadvantages. Heuristic malware detection methods use data mining and machine learning techniques to learn the behavior of an executable file. e.g. as the first attempt, Naïve Bayes and Multi Naïve Bayes were presented by Schultz et al. [10] to classify malware and benign files. Perceptibly, these ML techniques require some features representing the input instance in the way that can be used for classification. Some of the features used for malware detection are, API (Application Programming Interface) calls, CFG (Control Flow Graph), N-Gram, Opcode and Hybrid features.

Almost all programs send their requests to the Operating System using API calls. Hence, the behavior of a piece of code like malware can easily be reflected using API sequences. Hofmeyr et al. [11] were among the first ones who regarded API call sequences as a feature of a malware. They introduced an anomaly detection method based on system call sequences. Normal behavior profiles were made using short sequences of system calls. Hamming distance was used for matching sequences; also a threshold used to determine anomalies. Typically, large Hamming distance value reported as anomalies. Later, an extensive research on malware detection using API calls was done by Bergeron et al. [12], Sekar et al. [13], Sung et al. [14], etc. In 2007, based on the analysis of Windows API execution sequences called by Portable Executable (PE) files, Ye et al. [15] proposed Intelligent Malware Detection System (IMDS) using Object Oriented Association (OOA) mining based classification. To generate efficient OOA rules for classification, an OOA-Fast-FP Growth algorithm is adapted. In spite of its good performance in malware detection, IMDS has few demerits such as, Handling the large set of generated rules to build the classifier and finding effective rules to classify new file samples. To overcome these problems, Ye et al. [16] used post processing techniques of associative classification. At first they applied Chi-squared testing [17] and insignificant rule pruning, followed by database coverage based on the Chi-square measure rule ranking mechanism and Pessimistic error estimation. They finally, performed prediction by selecting the best first rule. They incorporate IDCPF [16] into existing IMDS system and called the new system CIMDS. It

was the first attempt on using post processing techniques of associative classification in malware detection. Jeong and Lee [18] used system call sequences for both malicious and benign executables to build a topological graph which is called code graph. For every binary program this graph is extracted and is compared with the code graph of malicious and benign programs. Based on this comparison, a program is classified as malware or benign. Due to these large sized graphs, Lee et al. [19] classified API calls to 128 groups, so the code graph reduced. Ye et al. [20] 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 different post processing techniques including 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).

Forrest et al. [21] powered system calls to discriminate between benign and UNIX processes. Hofmeyret al. [22] build normal behavior of UNIX processes in terms of short sequences of system calls. The Hamming distance is used to determine how closely a system call sequence resembles another. A threshold must be set to determine whether a process is anomalous. Wepsi et al. [23] 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. [24]. Manzoor et al. [25] collect some Windows malicious executables from VX Heavens [26] and their API call sequences are monitored by API Monitor [27]. The DCA (Dendritic Cell Algorithm) [28-30] is applied for detection. Later, Ahmed et al. [31] 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 behavior. However, the system call or API call sequences can be manipulated by a crafty attacker to circumvent detection [32–34]. Seifert et al. [35] compared three popular event-based techniques that can monitor program behavior: user mode API hooking, kernel mode API hooking, and kernel mode callbacks. Z. Fuyong et al. [36] proposed a novel classification algorithm based on the idea of positive selection, which is one of the important algorithms in Artificial Immune Systems (AIS), inspired by positive selection of T-cells. The proposed algorithm is applied to learn and classify program behavior based on I/O Request Packets (IRP). Their experiments proved that the proposed algorithm outperforms Artificial Negative Selection classifier (ANSC), Naïve Bayes, Bayesian Networks, Support Vector Machine, and Decision Tree. This algorithm can also be used in general purpose classification problems not just two-class but multi-class problems.

M. K. Shankarapani et al. [37] presented detection algorithms that can help the antivirus community to ensure a variant of a known malware can still be detected without the need of creating a signature; a similarity analysis (based on specific quantitative measures) is performed to produce a matrix of similarity scores that can be utilized to determine the likelihood that a piece of code under inspection contains a particular malware. They presented two techniques such as, Static Analyzer for Vicious Executables (SAVE) and Malware Examiner using disassembled Code (MEDiC). MEDiC uses assembly calls for analysis and SAVE uses API calls (Static API call sequence and Static API call set) for analysis. They showed 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. Their two proposed techniques are implemented in SAVE and MEDiC and experimentally proved that both these proposed techniques can provide a better detection performance against obfuscated malware.

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 [38]. He showed the ability of single Opcode to use as a feature in malware detection. To this end, he proved that Opcode can be used as a powerful representation for executable files. Santos et al.[39] 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 datasets (i.e. malware and benign dataset) using mutual information[40]. Finally they used Weighted Term Frequency (WTF) [41] to make suitable feature vector extracted from executables. They used this feature vector in order to detect obfuscated malware variants and to this end they calculated the Cosine similarity measure between two feature vectors (i.e. new instance feature vector and malware variants feature vector). Afterward, in the next work, Santos et al. [41] presented a new feature extraction method based on Opcode sequences [41] 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 labeled data in real world. So, Santos et al., in their next research, proposed several methods to eliminate this limitation such as Collective classification [42], Single class learning [43], and Semi supervised learning [44]. Runwal et al. [45] 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 as a class which is more similar to Shabtai et al. [46] 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 Regression (LR), Artificial Neural Networks (ANN) etc. to evaluate the proposed feature selection method.

N-Grams are all substrings of a larger string with a length of N [46]. For example, the string "VIRUS", can be segmented into several 3-grams: "VIR", "IRU", "RUS" and so on. Over the past decade, several researches have been motivated on the detection of unknown malware based on its binary code content. Schultz et al. [10] were the first who introduced the idea of applying ML techniques for detection of diverse malwares based on their own binary codes. Three different feature extraction methods were engaged: features mined from the PE section, expressive plain-text strings that are encoded in executables, and byte sequence features. Tesauro et al. [48] 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 change. 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. [49], 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. [47], presented a framework that uses the Common N-Gram method and the K-Nearest-Neighbor (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. Kotler and Maloof [50] used byte N-Gram representation to detect unknown malware. Though the vector of N-Gram features was binary, presenting the attendance or nonattendance of a feature in the file. In an extension of their previous study, Kolter and Maloof [51] 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. Cai et al. [52] conducted several experiments in which they evaluated the mixtures of seven feature selection techniques, three classifiers, and byte N-Gram size. Recently, Moskovitch et al. [53] published the results of a research which used an imbalance data set characterized by byte N-Grams. Moreover, a research of the imbalance problem was illustrated.

Control Flow Graph (CFG) is a graph that represents the control flow of programs and are widely used in the analysis of software and have been studied for many years [54], [55], [56]. CFG is a directed graph, where each node represents a statement of the program and each edge represents control flow between the statements (i.e. what happens after what). Statements may be assignments, copy statements, branches etc. In Figure 4 we can see an example of a generated CFG for Chernobyl malware. In [57], authors performed a 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 generate 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 [58] 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 [59], Bagging [60] and Random Forest [61]. Bonfante et al. [62] used CFG as a signature for malware detection. As we mentioned, CFG is composed of nodes and edges and as we know each assembler consists of four types of instruction: non-conditional jumps (jmp), conditional jumps (jcc), function calls (call) and function returns (ret). They abstract any contiguous sequence of instructions in a node named "inst", and after that the end of the program comes in a node named "end". So, they defined six types of node: jmp, jcc, call, ret, inst and end. They build CFG based on these types. Then, they reduce 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. [63] 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.

Due to the rapid production of malware and the desperate need for human effort to extract some kinds of signature, the signature based approach is a tedious solution; thus, an intelligent malware detection system is required to deal with new malware threats. Most of intelligent detection systems utilize some data mining methods in order to distinguish malware from normal programs. Rey et al. [64] proposed a data mining techniques for malware detection based on an automation of signature extraction for viruses. Viruses were executed in secured environment to infect decoy programs. Candidate signature of variable length is produced by

analyzing the infected region in these programs that remains invariant from on program to another. Signatures with lowest estimated false positive probabilities were choosen as best signatures. Then a simple approximation formula can be used to estimate the probability of a long sequence by combining the measured frequencies of the shorter sequences from which it is composed. To measure algorithm's effectiveness, candidate signatures are generated and their estimated and actual probabilities are compared. Then Tesauro et al. [65] extended the n-grams analysis to detect boot sector viruses using neural networks. The n-grams were selected based upon the frequencies of occurrence in viral and benign programs. Then they continued their work and used n-grams as features to build multiple neural network classification and adopted a voting strategy to predict the final outcome. Wang et al. [66] proposed a method which uses data mining as detection category to classify various file types based upon their fileprints. An n-gram analysis method was used and the distribution of n-grams in a file was used as its fileprint. The distribution was given by byte value frequency distribution and standard deviation. These fileprints represented the normal profile of the files and were compared against fileprints taken at a later time using simplified Mahalanobis distance. A large distance indicated a different n-gram distribution and hence maliciousness. Schultz et al. [68] 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. [69] created MEF, Malicious Email Filter, that integrated the scheme described in [68] 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 [67] 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.

Shahzad et al. [70] carried out a forensic analysis of Linux executable and linkable format (ELF) files to find out different features that have the potential to discriminate malicious executables from benign ones. They selected features' set of 383 features that are extracted from ELF headers and quantified the classification potential of features using information gain and then removed redundant features by employing pre-processing filters. Finally, they performed evaluation among classical rule-based machine learning classifiers—RIPPER, PART, C4.5 Rules, and decision tree J48—and bio inspired classifiers—cAnt Miner, UCS, XCS, and GAssist—to select the best classifier for their system. They have evaluated their approach on an available collection of 709 Linux malware samples from vx heavens and offensive computing. Their experiments show that ELF-Miner provides more than 99% detection accuracy with less than 0.1% false alarm rate. M. Eskandari et al. [71] presented a novel hybrid approach, HDM-Analyzer which takes advantages of dynamic and static analysis methods for rising speed while preserving the accuracy in a reasonable level. HDM-Analyzer is able to predict the majority of decision making points by utilizing the statistical information which is gathered by dynamic analysis; therefore, there is no execution overhead. The importance was given to the process of incorporating the accuracy advantage of dynamic analysis into static analysis in order to augment the accuracy of static analysis. In fact, the execution overhead has been tolerated in learning phase; thus, it does not impose on feature extraction phase which is performed in scanning operation. They experimentally demonstrated that HDM-Analyzer attains better overall accuracy and time complexity than static and dynamic analysis methods.

V. CONCLUSION

In this survey a series of malware detection techniques have been presented. We have presented signature based detection and data mining based detection techniques using different types of features like API, N-Grams, Opcode and control flow graph. Data Mining is a vast area used in variety of applications that requires data analysis. Now a day's data mining techniques plays an important role in malware detection systems. Different data mining techniques like Classification, Clustering and Association rules are frequently used to acquire information about malware as well as intrusions by observing network data. Still the research on data mining techniques in malware detection is going on and this survey might help the researchers for some extent. Similarly, Detection of malware's changing their signatures frequently has posed many open research issues. Challenge lies in the development of good disassembler, similarity analysis algorithm so that the variants of malware's can be detected in shorter time there by reducing the computation overhead.

REFERENCES

[1] 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, ISSN: 2277 128X, Volume 3, Issue 4, April 2013.

[2] Vinod P. V.Laxmi,M.S.Gaur: Survey on Malware Detection Methods, 3rd Hackers‟ Workshop on Computer and Internet Security, Department of Computer Science and Engineering, Prabhu Goel Research Centre for Computer & Internet security, IIT, Kanpur, pp-74-79, March,2009.

[3] Pham Van Hung, An approach to fast malware classification with machine learning technique, Keio University, 5322 Endo Fujisawa Kanagawa 252-0882 JAPAN, 2011.

[4] Ammar Ahmed E. Elhadi, Mohd Aizaini Maarof and Ahmed Hamza Osman, Malware detection Based on Hybrid Signature Behaviour Application Programming Interface Call Graph, American Journal of Applied Sciences 9 (3): 283-288, 2012, ISSN 1546-9239, 2012, Science Publications.

[5] Ravindar Reddy Ravula. Classification of Malware using Reverse Engineering and Data mining Techniques, Thesis-Master of Science, University of Akron, August 2011.

[6] 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.

[7] 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. 3849.

[8] 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.

[9] Sunita Beniwal, Jitender Arora, Classification and Feature Selection Techniques in Data Mining, International Journal of Engineering Research & Technology (IJERT),Vol. 1 Issue 6, August – 2012, ISSN: 2278-0181.

[10] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo, " Data mining methods for detection of new malicious executables." In IEEE Symposium on Security and Privacy, pages 38-49. IEEE COMPUTER SOCIETY, 2001.

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

[12] J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, and N. Tawbi, "Static detection of malicious code in executable programs." Int. J. Of Req. Eng., 2001.

[13] R. Sekar, M. Bendre, P. Bollineni, and D. Dhurjati, "A Fast Automaton- Based Approach for Detecting Anomalous Program Behaviors." In IEEE Symposium on Security and Privacy, 2001.

[14] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static Analyzer of Vicious Executables." In 20th Annual Computer Security Applications Conference, pp. 326–334, 2004.

[15] Y. Ye, D. Wang, T. Li, and D. Ye, "IMDS: Intelligent malware detection system," in Proc. ACM Int. Conf. Knowl. Discovery Data Mining, pp. 1043–1047, 2007.

[16] Y. Ye, T. Li, Q. Jiang, and Y. Wang, "CIMDS: adapting postprocessing techniques of associative classification for malware detection," IEEE Trans. Syst., Man, Cybern. C, vol. 40, no. 3, pp. 298-307, 2010.

[17] W. Snedecor and W. Cochran, "Statistical Methods", 8th ed. Iowa City, IA: Iowa State Univ. Press, 1989.

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

[19] J. Lee, K. Jeong, and H. Lee, "Detecting metamorphic malwares using code graphs" In Proceedings of the ACM symposium on Applied Computing, ser. New York, NY, USA: ACM, pp. 1970-1977, 2010.

[20] 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), pp.1-20. 2010.

[21] Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self forUnix processes. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 120–128 (1996).

[22] Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. J. Comput. Secur. 6(3), 151–180 (1998).

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

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

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

[26] VX Heavens Virus Collection. http://vx.netlux.org/vl.php

[27] API Monitor. http://www.rohitab.com/apimonitor.

[28] Aickelin, U., Bentley, P., Cayzer, S., Kim, J., McLeod, J.: Danger theory: the link between AIS and IDS? In: Proceedings of the ICARIS. LNCS, vol. 2787, pp. 147–155, Springer, Heidelberg (2003).

[29] Greensmith, J., Aickelin, U., Cayzer, S.: Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: Proceedings of the ICARIS. LNCS, vol. 3627, pp. 153–167, pringer, Heidelberg (2005).

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

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

[32] Parampalli, C., Sekar, R., Johnson, R.: A practical mimicry attack against powerful system-callmonitors. In: Proceedings of theACM Symposium on Information, Computer andCommunications Security (AsiaCCS), pp. 156–167, Japan (2008).

[33] Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), pp. 255–264. ACM Press, New York (2002).

[34] Oberheide, J.: Detecting and evading CWSandbox. http://jon. oberheide.org/blog/2008/01/15/detecting-and-evading- wsandbox/

[35] 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), S23–S30 (2007).

[36] Zhang Fuyong ,Qi Deyu: Run-time malware detection based on positive selection. Springer 267–277(2011).

[37] Madhu K. Shankarapani ,Subbu Ramamoorthy ,Ram S. Movva ,Srinivas Mukkamala: Malware detection using assembly and API call sequences. Springer J Comput Virol (2011) 7:107–119.

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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, p. 220, 2011.

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

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

[46] 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.

[47] T. Abou-assaleh, N. Cercone, V. Ke□, and R. Sweidan, "N-gram-based Detection of New Malicious Code," no. 1, 2004.

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

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

[50] 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.

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

[52] T. J. Cai DM, M. Gokhale, "Comparison of feature selection and classification algorithms in identifying malicious executables," in Computational Statistics and Data Analysis, 2007.

[53] E. Y. Moskovitch, D. Stopel, C. Feher, N. Nissim and N. Japkowicz, "Unknown malcode detection and the imbalance problem," journal in Computer Virology, 2009.

[54] P. Jalote, "An Integrated Approach to Software Engineering", Springer, New York, NY, 2005.

[55] T. McCabe, "A complexity measure", IEEE Transactions on Software Engineering SE-2(4): 308–320, 1976.

[56] L. Tan, "TheWorst Case Execution Time Tool Challenge", The External Test, Technical report, 2006.

[57] 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.

[58] 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.

[59] T. M. Mitchell, "Machine learning and data mining," Commun. ACM, vol. 42, no. 11, 1999.

[60] L. Breiman. "Bagging Predictors." Machine Learning, 24(2):123–140, 1996.

[61] L. Breiman. "Random Forests." Machine Learning, 45(1):5–32, 2001.

[62] G. Bonfante, M. Kaczmarek, J.Y. Marion. ''Control Flow Graphs as Malware Signatures.'' WTCV, May, 2007.

[63] 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.

[64] Je_rey O. Kephart and Bill Arnold. "Automatic Extraction of Computer Virus Signatures." In Proceedings of the 4th Virus Bulletin Internation conference, pp. 178–184, 1994.

[65] Gerald J. Tesauro, Je_rey O. Kephart, and Gregory B. Sorkin. "Neural Network for Computer Virus Recognition." IEEE Expert, 11(4):5–6, 1996.

[66] 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.

[67] 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.

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

[69] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, Manasi Bhattacharyya, and Salvatore J. Stolfo. "MEF: Malicious Email Filter: A UNIX mail Filter That Detects Malicious Windows Executables." pp. 245–252, 2001.

[70] Farrukh Shahzad , Muddassar Farooq "ELF-Miner: using structural knowledge and data mining methods to detect new (Linux) malicious executables." Knowl Inf Syst, pp: 589–612, Springer-Verlag London Limited 2011.

[71] Mojtaba Eskandari , Zeinab Khorshidpour, Sattar Hashemi "HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection" J Comput Virol Hack Tech (2013) 9:77–93 Springer-Verlag France 2013.