International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2021/210724 Volume 8, Issue 6, November – December (2021)

SURVEY ARTICLE

A Survey on Malware Classification Using Machine Learning and Deep Learning

Manish Goyal
Department of Computer Science and Engineering, IK Gujral Punjab Technical University, Kapurthala, Punjab, India
manishgoyalpup@gmail.com

Raman Kumar
Department of Computer Science and Engineering, IK Gujral Punjab Technical University, Kapurthala, Punjab, India
raman.kumarptu@gmail.com

Received: 01 October 2021 / Revised: 12 November 2021 / Accepted: 17 November 2021 / Published: 30 December 2021

Abstract – In today's era, there is fast development in the field of Information Technology. It is a matter of great concern for cyber professionals to maintain security and privacy. Studies revealed that the number of new malware is increasing tremendously. It is a never-ending cycle between the world of attack and the defense of malicious software. Antivirus companies are always putting their efforts to develop signatures of malicious software and attackers are always in try to overcome those signatures. For the detection of malware machine learning are highly efficient. The process of detection of malware is split into two categories first is feature extraction and the second is malware classification. The effectiveness of classification algorithms depends on the feature extracted. In this paper, firstly an in-depth study of the features is provided that can be used to differentiate malware. Thereafter describe the various stages of machine learning and deep learning that researchers use in their research work and the pros and cons they face that can assist new researchers while selecting an algorithm for their research work.

Index Terms – Malware Detection, Static Analysis, Dynamic Analysis, Security, Features of Malware, Machine Learning, Deep Learning.

1. INTRODUCTION

There has been a mushroom growth of malware which is articulated by various encyclopedias such as; in 2014 panda reported 84 million new variants [1]. Similarly, in the 3rdquarter of 2020 McAfee reported new MacOS malware surged 420% [2]. At the stage of inception, the computer virus was developed just for fun. The malicious code that was evolved by teenagers to play pranks with their friends has now turned into a serious malware threat. Malware writers have started using their brains professionally to do unlawful activities such as stealing money, crashing system, burglarizing very important information, etc.

In Dec 1999, the San Diego Supercomputer Center (SDSC) experimented by installing Red Hat Linux 5.2 without any security patches on a computer with an internet connection [3]. The computer was attacked in just 8 hours of installation and in 21 days the computer was attacked 20 times and compromised 40 days after installation.

Anti-virus companies mainly use signature-based detection techniques (it is a technique in which detection of malware is done based on features extracted from previously known malware) to capture malware, but using this technique only known malware can be detected. Zero-day malware (new and unseen malware) can't be detected using this approach. Moreover, malware writers practice evasion techniques like encryption and obfuscation to prevent them from being detected at an early stage. After knowing the catastrophic effects of malware, it is necessary to protect systems from malware.

1.1. Background Motivation

In 2012 Egele et.al. [4] Surveyed the default strategies and tools for malware detection. They first describe the malware and its variants and then the vectors of infection. After that, the malware analysis techniques used are described, namely parameter analysis of function, monitoring of function calls, information flow tracking instruction trace, and automated extensible points. Malware analysis is defined in the context of the user/kernel space, emulator and virtual machine, etc. The researchers explained a lot of tools that run malware samples like Anubis, CWSandbox, Norman Sandbox, Joebox, WiLDCAT, etc. According to their observation, most of the dynamic tools analyze system call and API they are required to interact with the system. Some tools observe the sensitivity of processed data. This information can be used to determine if the sample is malware.

In 2013 Bazrafshan et. al. [5] discussed 3 methods namely, behavior-based, signature-based and heuristic-based malware detection. He first explained these methods and then the strategies to hide the malware. They mainly focus on

SURVEY ARTICLE

describing heuristic malware detection methods using features such as opcodes, API calls, N-grams and discuss their pros and cons.

In 2014 Gandotra et.al.[6] surveyed various papers on malware analysis using machine learning. They categorized work done by authors into static, dynamic, and hybrid techniques. In 2015 LeDoux &Lakhotia[7] surveyed malware and machine learning. They explained the pipeline process of malware detection, challenges in Malware Analysis, machine learning concepts like supervised, unsupervised, semi-supervised, and ensemble learning, and features of malware detection. They mainly focused on pointing problems in malware analysis and machine learning concepts to tackle those problems.

In 2016 Basu et.al.[8] surveyed data mining techniques in the context of malware detection. The researchers have explained the work of various authors, their malware sample sources, and techniques used by authors and also distinguished the work of various authors based on features used by authors like API calls, byte sequence, PE header, etc. After that, they proposed a malware detection approach in which they used the Xptruss tool to detect API calls and apply Naïve Bayes for testing and training.

In 2017 Ye et.al.[9] explained data mining methods based on both static and dynamic representations and some novel representations. Researchers have divided the malware detection process into two phases the first phase is feature extraction and the second phase is classification/ clustering. They concluded that data mining frameworks could be designed to detect malware with a low false-positive rate.

In 2017 Ucci et.al. [10] outlined 7 different objectives to detect malware and grouped malware features according to their specific type. They also noticed some issues in the dataset. They feel that mainly the dataset used is not balanced. Therefore they proposed 3 characteristics for benchmark datasets namely labeling according to objective, balanced, and maintaining them over time. They also introduced the concept of malware analysis economics. They have identified a tradeoff between performance metrics and economical costs.

In 2019 Berman et.al. [11] explained two approaches towards dynamic analysis; one is by analyzing the difference between defined points and the second by observing runtime behavior. They also discussed the advantages and disadvantages of both malware detection techniques namely, signature-based and behavior-based.

In 2019 Gibert et.al. [12] compared shallow learning i.e. ANN and deep learning. They provide an overview of different methods of deep learning like RNN, CNN, etc. They also described Deep Learning evaluation strategies using various metrics.

Many authors have done handsome surveys on malware with machine learning techniques but this paper aims to survey based on features used by various researchers along with machine learning techniques and highlighting advantages and limitations faced by them.

In this paper, the following section contains types of malware. Section 3 includes features of malware detection which are extracted from various samples are explained in this section. Section 4 contains a study of machine learning and its types. Section 5 explains deep learning and lists the work done by various authors for malware classification using deep learning. Section 6 shows the results of the survey of various papers and at last Section 7 concludes the paper.

2. MALWARE

Malware (malicious software) is a software program that enters into a user's computer his permission and has intentions to cause damage or to steal sensitive information.

Virus: These viruses enter the system illegally and then cause infections or get replicated on the system by attaching themselves to other executable. One of the most primitive forms of the virus was 'Creeper' which was an experimental outcome of a program written by Bob Thomas at BBN in 1971[9]. It was not active malicious software as it didn't harm any data. 'Elk cloner' was written in 1982 by Rich Skrenta which spread by the floppy disk and it attached itself to Apple II operating system [7].

Worms: These are types of a virus but worms can also spread over the internet and can replicate themselves without human intervention on to various machines. Unlike viruses, worms are self-contained and they don't need to be a part of other programs to propagate themselves. The term worm was taken from the novel 'The Shockwave Rider' written by John Brunner in 1970. The first functional worm was created in 1978[13]. In January 2004 the scandalous worm named Mydoom was used by attackers to cause severe damage. This worm launches DoS Attack on www.sco.com, due to which the website was inaccessible for several months [14].

Trojan: Trojan appears to be useful files but is malicious files that are just used as a trick to get entry into the system. There is a story behind the word Trojan horse, in which the Greeks invaded human power hidden inside a huge wooden horse into the independent city of Troy and won the war. Metaphorically, a Trojan horse is any ploy used to enter a system. In August 2006 Scandinavian bank Nordea's client started to receive emails to install antispam products which were a trick by cybercriminals to install Trojan named Haxdoor to their system. This trojan key logged the victim's system and send information to the cyber criminal's server which caused the theft of over $1million [14].

SURVEY ARTICLE

Backdoor: It is a technique in which one can bypass security mechanisms undetectably. Software writers sometimes develop backdoors to facilitate software testing. Default passwords kept for first-time use of hardware or software is an example of a backdoor. In February 2005 a businessman from Florida lost $90,000 from his account because of his computer infected by a backdoor [14].

Rootkits: These provide attackers, higher-level access to data rather than authorized access, i.e. it can provide administrative rights of the victim to the attacker. Rootkits are designed to remain hidden i.e. it masks their (and/or other software) existence on the host PC. This achieves sustained and maintained privileged access, as well as hiding this access.

Keylogger: It keeps a record of the keys pressed by the user or can also take screenshots and send them to the attacker thus stealing banking or other important information of a user. Banking sites provides virtual keyboards to prevent keylogging attack. There are many examples of key loggers across the world. This malicious program has caused major damage. It can be used by combining with other malicious programs. In February 2006, 55 people were arrested by Brazilian police, for stealing an approximate amount of $4.7 million from 20 clients across 6 countries[14].

Spyware: Generally, referred to as Potentially Unwanted Program (PUP). It is an unwanted program intended to steal Internet usage data, delicate information (such as user passwords or bank account information) without the user's knowledge. The term spyware was firstly coined in 1995 but got popularity in 2000[15].

Adware: It is a subclass of spyware. But the major difference is that adware doesn't harm the computer intentionally. Its main purpose is to capture a user's interest so that similar kinds of advertisements, emails and pop-ups can be displayed on the victim's system.

R.A.T: It provides remote control access to the victim's system without his consent. It is like a team viewer but with wrong intentions.

Ransomware: A type of malicious software in which an attacker encrypts all the files of the victims and asks for a ransom to decrypt those files.

3. MALWARE FEATURES

There are two phases of malware detection namely feature extraction and classification/clustering. Features of malware mean the input which can be provided to machine learning algorithms. These define the set of all possible concepts which can be modeled using machine learning. Malware features can be classified into two categories namely static and dynamic. Static features are those features that can be extracted without the execution of malware. Such features are extracted by studying binaries of malware [16]. While dynamic features [4] are those which are extracted after executing a binary file. These features are extracted through dynamic analysis. Classification/ clustering of malware is done using machine learning techniques. In this paper, only classification techniques are explained. This section listed some of the features of malware detection.

3.1. N-Gram

It is one of the common static features which consists of a consecutive sequence containing n bytes which are taken from hexdump (a tool that converts a binary file into corresponding hexadecimal portrayal) output. Authors Lin et.al.[17], Ahmadi et.al.[18]& Anderson et.al.[19] worked upon 1Gram, Lin et.al.[17], Ahmadi et.al.[18], Sexton et.al.[20] Kolter et.al.[21] worked upon 2 Gram, Ahmadi et.al.[18], Kolter et.al.[21] & Khodamoradi et.al.[22]worked upon 4 Gram, etc. 4 Gram is the most commonly used feature which means to take a combination of 4 bytes. N-Gram can be used in overlapping or Non-overlapping. In case of overlapping the byte used once can be used again in the next gram but in a non-overlapping byte is not used twice. An example of the overlapping and non-overlapping concept of N-Gram with a value of N=2 is shown in Table 1.

Byte Code	Overlapping	Non-overlapping
0f4b 0e3f fe0f efef	0f4b 0e3f 0e3f fe0f fe0f efef	0f4b 0e3f fe0f efef

Table 1 Overlapping and Non-Overlapping Byte Sequence

3.2. Opcodes

Different machine-level operations performed by programmable executable (PE) are called opcodes. These opcodes can be obtained from the assembly code [19]. Sexton et. al. [20] worked upon branch instructions, count, and memory instruction count. Belaoued et. al.[23]used math, stack, logic, NOP, and other instruction counts in their research work. Ahmadi et. al.[18] used data to define instruction proportions in their research. Darabian et. al.[24]a used sequence of opcodes for the detection of malware.

3.3. Strings

Strings were firstly noticed by Schultz et.al.[25]and are defined as consecutive printable characters. They noticed that PE-format headers consist of plain text which could be used to extract information. Moreover, non-PE executables also have encoded strings that could be used along with their dataset. This definition of strings is more refined by Ye et. al. [9] which says that only "interpretable" strings which make some semantic sense can be used. Ito &Mimura [26] uses

SURVEY ARTICLE

ASCII strings with Natural Language Processing for malware detection.

3.4. Call Graph

The call graph shows how data is exchanged between different functions. The call graph is a directed graph that depicts the relation between various procedures of a program. The call graph doesn't give information on how control of program flows instead it provides information on calling of various procedures. Call graphs are employed in [27]–[30] to depict the relation of procedures. An example of a call graph is shown in Figure 1 [7].

```
void doActivity(){
//Activity
doEducationalActivity();
}
void doSportsActivity() {
doEducationalActivity();
if(need cultural)
doCulturalActivity();
}
void doEducationalActivity() {
doActivity();
if(need Sports)
doSportsActivity();
}
void doCulturalActivity() {
//Do Cultural Activity
}
int main() {
doActivity();
}
```

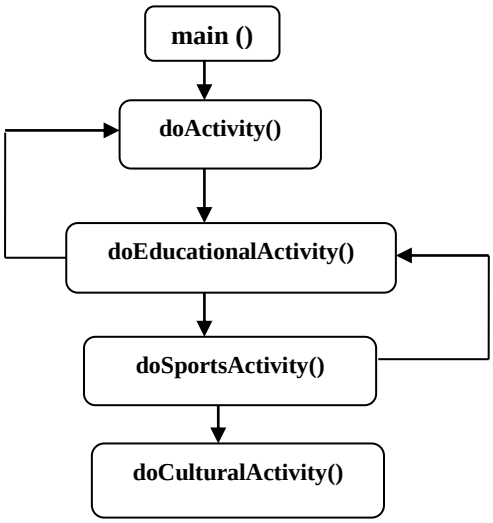


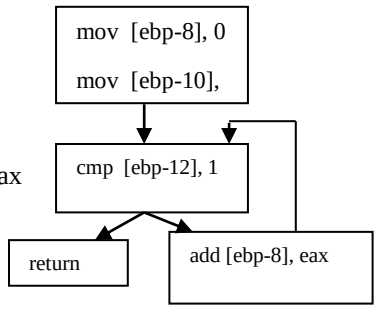
Figure 1: Example of a Call Graph

3.5. Control Flow Graph

The control flow graph is a more refined variety of call graphs [31]. It depicts flow within a single function. The control flow graph reflects how the flow of control is accomplished through the function. It shows all possible way outs of a flow of a program. Chionis et.al.[32] used control flow graph in their research. Figure 2 depicts an example of a control flow graph 2[6].

```
0 : mov   [ebp-8], 0
1 :mov    [ebp-10], 0
2 :mov    [ebp-12], 0
3 :jmp    6
4 :add    [ebp-10], eax
5 :add    [ebp-8], 1
6 :cmp    [ebp-12], 1
7 :jle    4
8 :return
```

(a)



(b)

Figure 2: Control Flow Graph Example

3.6. PE File Characteristics

PE file is a data structure, designed by Microsoft for enveloping information based on Win 32 systems [33]. Even scrutinizing the PE header statically can yield handsome information [34]. Structural information is mainly present in the PE header. Moreover, the PE header also stores the information required by the loader to load the program into memory [7].

Hence, the PE header contains information about size, selection, symbols, used compiler, and so forth. PE file header can be useful to detect malware at the triage phase [7]. Asquith [35] used some PE file characteristics like resource icon's checksum, section attribute, PE header checksum, section names and sizes, import table location and size, and entry point offset in their study.

While other researchers like Yonts[36] studied several symbols, a pointer to the symbol table, PE characteristics flags, and section count and Bai[37] carried their work on section count, resource's directory table, symbols in export table count, and items in .reloc section count. Wadkar et.al. [38] used PE file characteristics to study the evolution of malware families over time.

SURVEY ARTICLE

3.7. Memory Access

A large amount of data passes through the main memory like a window registry key, configuration, and network activity. Thus, important information about malware can be extracted by analyzing how memory is accessed. Author Egele et.al. [39] used memory-based features by tracking values of the stack (read/write from/to) dynamically and heap in combination with calls to import library functions, system calls, and return values stored in %rax registers.

Other authors like Kong et.al.[29]carried their research in memory read-write operations count and several I/O read-write operations in association with other features of API and opcodes.

3.8. CPU Register

How CPU registers are used can also indicate valuable information about malware. Kong et.al.[29]excelled in their research using several flags changed in a register, register read, and write count while Egele et. al. [39] used return values stored in %rax registers and Ahmadi et. al. [18] used registers usage frequency and register value respectively.

3.9. Raised Exceptions

It is a common trick used by malware writers is to throw malware code in exception and to raise an exception in a program. So, to understand malware evasion techniques it's better to scrutinize exceptions raised [35], [40].

3.10. Network

The interaction of PE with the network also reveals huge information about malware. Malware like a key logger steals a user's personal information and sends it over the network to its attacker. Sometimes, attackers also send instructions to their malware program over the internet like in the case of botnets. Some malware tries to generate traffic over the network by floating false packets. So, a study of the network can reveal some information about malware. Many authors survey dynamic analysis to extract information about networks [32], [41]–[45]. Vadrevu et al., 2013 [46] used download domain, download history, destination IP, download request, and queried URL's while [28], [41] used connection count and [47] worked upon Unique IP count, protocol type, connection count, HTTP request/response type count, and request-response packet size. M. Belaoued et. al. [23] worked on network traffic, domain names and IP addresses, which are used by malware to have remote access.

3.11. Sandbox Submissions

Sandboxes are used to execute files in an isolated environment to detect any malicious activity. Many sandboxes are available online. Malware writers take leverage of sandbox and execute their malware code on this sandbox first to test the effectiveness of their code and to check if their code can evade the most commonly used antivirus[28], [41]. So, studying these sandbox submissions can provide important information about malware.

3.12. API Calls

Application Programming Interface (API) acts as a mediator between software and operating system. Some tasks can be directly performed by the operating system only like writing a file to the disk and the system library contains a library of such functions. When a program invokes a call to the system library is referred to as an API call [7]. For example, when there is a need to copy a file, CopyFileW API will be called. API calls are a very important feature for malware detection and are used by many researchers. If we can preserve a sequence of API calls, it is called an API trace [48]. This API trace can detect high-level behavior of malware [49]–[51] such as "walk through directories" or "copies itself to disk". Jinrong Bai et.al. [37] used referred API calls and referred DDLs counts along with PE file characteristics in their research. Bojan Kolosnjaji et.al. [52] used Kernel API calls in their research. They used a system call sequence and build a neural network by using recurrent and convolution layers. Many other authors like [30], [35], [53], [54];[23], [55] also worked on system call sequences while other authors like [27], [42] used system call dependencies. Table 2 represents the summary of features extracted in malware detection.

4. MACHINE LEARNING

The interaction of PE with the network also reveals huge information about malware. Malware like a key logger steals a user's personal information and sends it over the network to its attacker. Sometimes, attackers also send instructions to their malware program Arthur Samuel in 1959 [75] in his paper, firstly coined the term Machine Learning, in which he builds a game of checkers on computers. The basic motive behind machine learning is to train a computer how to act like humans or animals in some situations. In the case of machine learning, there is no fixed equation, rather a computer learns from previous situations. For machine learning, a large amount of data is fed to the computer and it co-relates input and output from this data and works further based on this data. It includes two main modes- the first is training and the second is testing. In training mode, a large set of data is provided to the computer, and the computer analyzes this data and builds its equation of model of this data. In the testing mode, a small subset of the sample is used to test what the computer has learned in the training phase. The accuracy of results increases with the increment of training data. Machine learning algorithms don't directly work on the provided data. It will first extract features from the provided data. These features act as input to machine learning classifiers [6]. For example, if we have to classify different kinds of animals its features may include size, color, presence of trunk, size of the

SURVEY ARTICLE

neck, etc. An elephant would have features "size = big, color= grey, presence of trunk = yes, size of neck= small". A giraffe would have features "size= big, color =yellow, brown, presence of trunk= no, size of neck= long". If we talk about machine learning in the case of malware, it is a very powerful tool. Using machine learning we can detect malware and can also label the types of malware, i.e. if it is a virus, worm, rootkit, or any other kind. The features used for the detection of malware can be call graphs, API calls, strings, etc.

Author and Year	Static	Dynamic
M.G. Schultz et al. 2001 [56]	Byte Sequence, Strings	X
J. Zico Kolter et al. 2006[57]	n gram	X
F. Ahmed et al. 2009[58]	x	Arguments & Sequences Extracted from API
D. H. Chau et al. 2010[59]	File System	X
Firdausi et al. 2010[60]	File System	Win Reg, Sys Call Monitoring
B. Anderson et al. 2011[61]	Byte Sequence	API calls
Santos et al. 2011[62]	Byte Sequence	X
B. Anderson et al. 2012[63]	Binary File, Disassembled Binary, Control Flow Graph	Instruction trace system call trace
J. Bai et al. 2014[64]	PE file characteristic	X
K. Bojan et al. 2016[65]	x	Kernel API calls
M. Ehsan et al. 2017[66]	x	duration, network features, API frequencies
H. William et al. 2017[67]	x	Window API calls
M. Asha Jerlin et. al. 2018[68]		API call sequences
N. Maleki et. al. 2019[69]	PE header and Section header	
Wadkar et al., 2020[70]	PE file characteristic	X
Jagsir Singh et. al. 2020[71]	X	API calls
Ajay Kumar et.al. 2020 [72]	PE file	X
JeyaprakashHemalatha et.al. 2021[73]	Binary Image	X
Omar N. Elayan et.al. 2021[74]	X	API calls + Permissions

Table 2 Summary of Feature Extracted in Malware Detection

4.1. Naïve Bayes Classification

Naïve Bayes classifiers are a family of the classifier which work on probability and are based on the Bayes theorem[76]. Naïve Bayes classifiers take an assumption that features of every set are independent of one and another, say values of features V_1, V_2, V_3, …… V_n is conditionally independent of a given class C. Even if features have a dependency on each other it is said to be a dependency in the classification of something. That is why it is called Naïve. To measure the probability (P) Naïve Bayes classifier works according to equation 1.

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y) \ldots P(x_n|y)P(y)}{P(x_1)P(x_2) \ldots P(x_n)} \quad (1)$$

Where y is class variable and x_1, $x_{2\ldots}$,x_n is the feature vector of size n.

4.2. k-Nearest Neighbor (k-NN)

The basic fundamental behind this algorithm is that two objects which belong to the same class have some commonalities which can be detected based on some distance metric[77]. It maps the unknown class to the known class by simply calculating the Euclidean, Manhattan, Minkowski, or Hamming distance among its k nearest neighbors, and then

SURVEY ARTICLE

voting for maximum class. k here represents the number of nearest neighbors which is the main factor and its value is generally small odd numbers like 1, 3, 5, or 7. Consider one has to find a class for point A1. To do so firstly find k nearest neighbors of A1, then voting is done for class. Then the class having maximum votes will be added with point A1. The main advantage of using k-NN is its simplicity and easiness to use and understanding but it becomes significantly slow when there is a large number of entries.

4.3. SVM

SVM [78] is a binary classifier that finds out a hyperplane to make a clear separation between two classes with the largest margin. SVM is mainly classified into two types: Linear SVM and polynomial SVM. In the case of linear SVM, it is possible to distinguish between classes with a linear hyperplane. But sometimes the dataset is dispersed in such a manner that it becomes difficult to distinguish between classes with a linear hyperplane then the kernel method is used for separation of classes. In this method, a polynomial function is used for the separation of classes. It is known as polynomial SVM.

4.4. Decision Tree

This classifier is represented like a tree [79], its internal nodes represent some conditions and leaf nodes represent the label of class. This tree-like structure helps to make decisions using the divide and conquer technique. While going from root to leaf nodes the feature values are captured for an unknown variable. Consider an unknown variable A1 whose class has to find then starting from the root node, one has to choose the path according to features of A1 and move down in a tree until a leaf node is encountered and on reaching leaf node label of class (which is contained by that particular leaf node) is given to A1.

4.5. Random Forest

Random Forest [64] has become popular recently on different problems of classifications and is an upgraded version of Decision Tree. As the name suggests random forest comprises numerous trees that work in a group. There are two types of randomness, selection of input variable and bootstrap sampling are used to collaborate with different trees in the forest. For the process of classification in Random forest, a class with maximum votes out of all trees is chosen. Having different trees using randomness provides better prediction rather than an individual tree.

4.6. Gradient Boosting

Gradient Boosting is somewhat similar to Random forest as they both focus to improve the weak classifier of the Decision Tree. Unlike Random Forest, Gradient Boosting doesn't use multiple decision trees but uses regression to estimate the relationship between independent and dependent variables.

4.7. XGBoost:

XG Boost algorithm is proposed by Tianqi Chen[81]Each base studying set of rules learns from its preceding base learner and decreases its error. Eventually, the very last learner shows minimal bias and variance in the course of the training process. In general, the XGBoostset of rules combines numerous base learners (decision trees) to the construction of sturdy aggregated models. To be expecting the very last output, the XGBoostset of rules combines the weights of the leaves of all trees T.

Table 3 represents the summary of algorithms and their advantages and limitations used in process of detection of malware.

Author	Data Set	Algorithm	Accuracy	Advantage	Limitation
M.G. Schultz et. al. 2001 [56]	3265 - Malware, 1001 - Benign	Naïve Bayes	97.11	1. Tested for new malicious executables	1. High false positive 2. Needs to do work in terms of time and accuracy 3. Used non-over-lapping byte sequence. 4. The collection is mainly of viruses only.
		Multi Naïve Bayes	96.88		
J. Zico Kolter et. al. 2006[57]	1651 - Malware, 1971 - Benign	Naïve Bayes	--	1. Used overlapping byte sequence. 2. ROC curves are shown.	1. Dataset is stored on a server, when there is a change in training data evaluation of a variety of methods becomes critical for a server
		Multi Naïve Bayes	--		

SURVEY ARTICLE

F. Ahmed et. al. 2009[58]	316 - Malware, 100 - Benign	J48 NB RIPPER	-- -- --	1. Used spatial-temporal information available in API calls which helps to provide high accuracy and also helps to detect from some of the evasion techniques	Processing and memory overheads.
Author	Data Set	Algorithm	Accuracy	Advantage	Limitation
D. H. Chau et. al. 2010[59]	903 Million Files	Graph Mining	--	1. Used a large dataset hence providing good results. 2. Results improved on iteration	1. All files are treated with of same weight which is not in a real-time situation 2. Execution time is high which can be reduced using parallelization techniques
Firdausi et al. 2010[60]	220 - Malware, 250 - Benign	KNN Naïve Bayes J48 Dec tree SVM MLP	92.9 92.3 96.8 94.9 94.2	Feature selection and feature reduction are used which used training time.	It reduced the performance of the system
. Anderson et al. 2011[61]	1615 - Malware, 615 - Benign	Similarity Graph	--	1. Computational Complexity would restrictive progressively setting. 2. Used kernel learning framework which provides a logical way to measure different aspects of program trace	1. Ethers are not completely invisible 2. String setting can be easily changed so that can't be detected 3 Slow for analysis
Santos et al. 2011[62]	1000 - Malware, 1000 - Benign	Learning with local & Global Consistency (LLGC)	--	Reduce Req labels	Less Accuracy than supervised learning
B. Anderson et al. 2012[63]	780 - Malware, 776 - Benign	Multiple Kernel-based learning	98.07	1. Combines both static & dynamic approaches. 2. A future data source can be easily added	Intel pin program is used which is not as transparent tracing tool like ether framework
J. Bai et al. 2014[64]	10421 - Malware, 8592 - Benign	Decision Tree Random Forest	-- --	1. Better Accuracy 2. Timely Detection	PE header of malware can be forged by malware writers to evade detection
K. Bojan et al. 2016[65]	4753 - Malware	Tensor Flow Supervised Learning	--	1. Better results than hidden Markov and SVM 2. Takes benefits of two types of layering of ANN	After insertion of noise malware can evade detectors that are not considered by authors.

SURVEY ARTICLE

M. Ehsan et al. 2017[66]	23146 - Malware	Supervised ANN	--	Classification reduces the effort of analysis	Computationally expensive
H. William et al. 2017[67]	22500 - Malware, 22500 - Benign	Deep Learning	93.68	Worked on real industrial application	Sparsity constraints can be imposed on AutoEncoder
		CG_SVM	88.24		
		CG_ANN	87.88		
		CG_NB	77.94		
		CG_DT	87.42		
Author	Data Set	Algorithm	Accuracy	Advantage	Limitation
M. Asha Jerlin et. al. 2018[68]	-	Multidimensional Naïve Bayes Classification (MDNBS)	--	High detection rate, and low time and computational complexity.	They used a normalized dataset that is not available in real-time.
N. Maleki et.al. 2019[69]	761 - Malware, 210 - Benign	DT classifier	98.26	The low detection error rate	PE header of malware can be forged by malware writers to evade detection
		NN	97.92		
		ID3	95.83		
		NB	95.14		
		SVM	95.14		
Wadkar et al., 2020[70]	26245	SVM	-	Detect evolutionary changes in malware families	Malware obfuscation has a great impact on static feature
Jagsir Singh et. al. 2020 [71]	8634 - Malware, 6434- Benign	kNN	98.4	High Accuracy	-
		Decision Tree	98.14		
		SVM	98.14		
		Multi Naive Bayes	85.4		
		Random Forest	99.1		
Ajay Kumar et.al. 2020 [72]	Brazilian malware dataset with 1,21,000 rows	Random Forest	99.7	High accuracy	-
		Decision Tree	99.7		
		Gradient Boost	98.48		
		SVM	96.9		

SURVEY ARTICLE

		Logistic Regression	96.8		
		XGBoost	96.7		
		AdaBoost	94.3		
JeyaprakashHemalatha et.al. 2021[73]	MalImg - 7268(training) BIG 2015-8338(training) MaleVis-9958(training) Malicia-testing	k-NN	76.75	-	1. MalImg and BIG 2015 are imbalanced datasets
		LR	56.33		
		SVM	80.33		
		Naïve Bayes	46.93		
		Decision Tree	77.77		
		Random Forest	82.10		
		Adaboost	75.31		
Omar N. Elayan et.al. 2021[74]	347 - Benign 365 - Malware	SVM	96.2		
		k-NN	97.2		
		Decision Tree	96.6		
		Random Forest	97.8		
		Naive Bayes	93.9		

Table 3 Summary of Algorithms and Their Advantages and Limitations Used in the Detection of Malware

5. NEURAL NETWORK

3.5. Artificial Neural Network (ANN)

It is inspired by the working of the human brain along with statistics and applied math. There are large numbers of brain cells in the human brain which are interlinked with each other. These are called neurons. These neurons are used for transferring information in the form of signals. Human sense organs receive a lot of information and the neuron web interprets it to the sensory part of the brain. Similarly, ANN works in the form of artificial neurons. Each neuron consists of a non-linear function.

The neurons in ANN are organized in the form of layers. ANN consists of 3 types of layers namely input layer, hidden layer, and output layer. Data transform from the input layer to the output layer via the hidden layer. ANN can be known as shallow learning. When there are multiple hidden layers it becomes deep learning [12].

3.6. Deep Learning

Deep Learning is a subset of machine learning. It can learn from unsupervised data which is not structured or even not labeled. It works by imitating the human brain, processes data and creating patterns by self-understanding and making the decision on basis of it. Deep Learning consists of interconnected neurons. The neural network's structure is composed of connected layers. Neural Network consists of many layers like an input layer, an output layer, and a hidden layer. A hidden layer is any layer that is in between the input layer and output layer. The network consists of more than 2 layers is represented as deep. The signal strength provided as input to the next layer depends upon bias, weight, and activation function [82]. The complexity of the network increases with the increase in the number of layers.

In the case of machine learning for classification purposes there need to extract features from images, while in deep learning it is capable to extract features by itself. Some practical applications of deep learning are in the field of natural language processing, self-driving cars, virtual assistants, etc. Deep learning used by various authors on malware datasets is shown in Table 4 and accuracies achieved by various authors using deep learning are shown in Figure 3.

SURVEY ARTICLE

Author and year	Paper cited	Feature	Dataset	Number of Malware samples	Number of Malware Classes	Method	Accuracy
Aziz Makandar et.al. 2015 [83]	58	Malware Binary	Mahenhur dataset	3131	24	ANN	96.35 %.
Kyoung Soo Han et.al. 2015 [84]	115	Visualized images and entropy graphs	-	1,000	50	-	97.9%
Espoir K. Kabangaet.al. 2017[85]	35	Visualizing Malware as Image	MalIMG	9458	25	CNN	98%
Mahmoud Kalash et.al. 2018 [86]	111	Visualizing Malware as Image	MalIMG	9339	25	CNN	98.52%
HiromuYakuraet.al . 2018 [87]	41	To extract important byte sequences to reduce human effort	-	147803	542	CNN	50.97%
Matilda Rhode et.al. 2018[88]	128	Snap sort of behavioral data	-	2345 benign 2286 malicious samples	2	RNN	96.01%
Author and year	Paper cited	Feature	Dataset	Number of Malware samples	Number of Malware Classes	Method	Accuracy
Jin-Young Kim et.al. 2018 [89]	90	Malware Images	-	10800	9	transferred deep-convolution generative adversarial network	95.74%
Daniel Gibertet.al. 2019 [12]	37	Visualizing Malware as Image	MalIMG	9458	25	CNN	98.48%
			Microsoft Malware Dataset	10868	9	CNN	97.49%
Danish Vasan et.al. 2020 [90]	50	Image-based malware	MalIMG	9435	25	CNN	98.82%
Jeyaprakash Hemalatha et.al. 2021[73]	15	Binary Images	MalImg	7268(training)	2	CNN	71.42
						VGG16	77.66
			BIG 2015	8338(training)		VGG19	82.92
						Inception-v3	83.7
			MaleVis	9958(training)		Resnet-50	83.52
						Densenet-121	83.02
			Malicia	Testing		Xception	83.02
						DenseNet based	89.48

SURVEY ARTICLE

						proposed	
Omar N. Elayan et.al. 2021[74]	-	API calls +Permissions	-	347-Benign 365 - Malware	2	GRU of RNN	99.2
Abdul basit Darem 2021[91]	-	Malware binary Images	BIG 2015	10868	9	XGBoost+Opcode	92.67
						XGBoost+Opcode+Segment	94.2
						XGBoost+Opcode+Segment+Secondary features	96.8
						CNN+Malware Images	98
						Ensemble	99.12

Table 4 List of Work Done by Various Authors for Malware Classification using Deep Learning

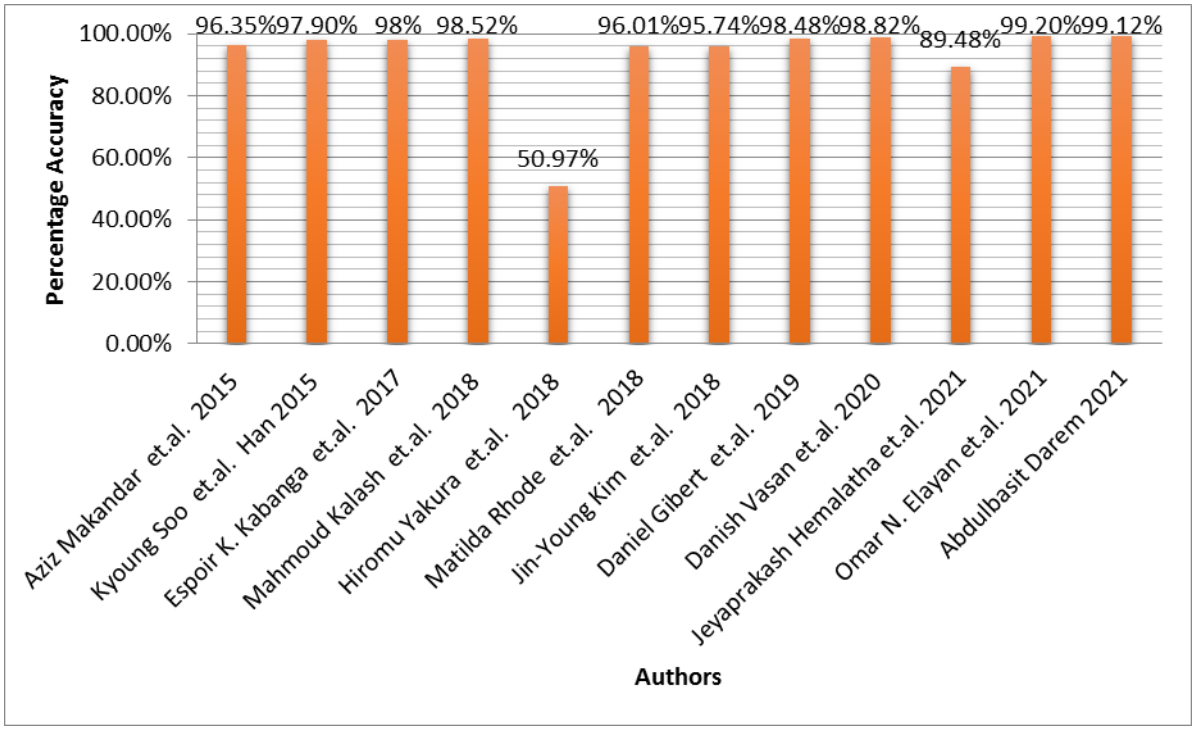


Figure 3 Malware Classification Accuracy in Surveyed Paper Using Deep Learning

5.2.1. Recurrent Neural Network (RNN)

RNN is a class of neural networks in which a directed graph along with a temporal sequence is used to make a connection between nodes. It can capture information about the sequence present in data [12]. In the case of RNN, it has some memory i.e. it remembers the past and its decisions are influenced by what is learned in past. So, we can say it is a feed backward network. Feed forward networks can only remember what to do but feed backward can also remember what was done in past.

5.2.2. Convolution Neural Network (CNN)

CNN is feeding forward neural networks. It consists of a convolution layer, pooling layer and fully connected layer. The convolution layer is the building block of CNN. This layer is made up of filters which are useful to detect a specific pattern in an image. Multiple filters are applied in parallel on the whole image symmetrically. Each filter detects a different type of pattern in an image. Strides are also used along with filters which define the movement of several pixel units of a

SURVEY ARTICLE

filter [92]. Activation functions are also used in convolution layers like relu, sigmoid, softmax, etc.

Pooling layers are used to reduce the dimensions of an image so that complexity of calculation can be reduced. The window size used for the pooling layer is less than the size of the filter. Generally, the pooling layer is used of 2 x 2 sizes to minimize the reduction of data. 3 types of pooling layers are used namely max-pooling, min-pooling and avg-pooling. In the case of min-pooling minimum value is chosen out of these partitions. Similarly in max-pooling and avg-pooling maximum and average values are chosen from the partition. A fully connected layer gets input from each neuron of the previous layer. These layers are generally followed by a dropout layer to avoid over-fitting the model.

6. SURVEY RESULTS

The mushroom growth of malware has raised challenges for its detection. A large number of new malware is reported daily and lots of automated toolkits for the development of malware are available like Zesus [93]. These toolkits use methods to evade malware at an early stage which has increased complexity for various researchers and antivirus companies. We have studied some papers in which researchers have worked to detect malware using various approaches. The numbers of malware samples used by various researchers are shown in Figure 4. In Figure 5 algorithms used by various researchers are listed on the x-axis and the percentage accuracy achieved by it is on the y-axis.

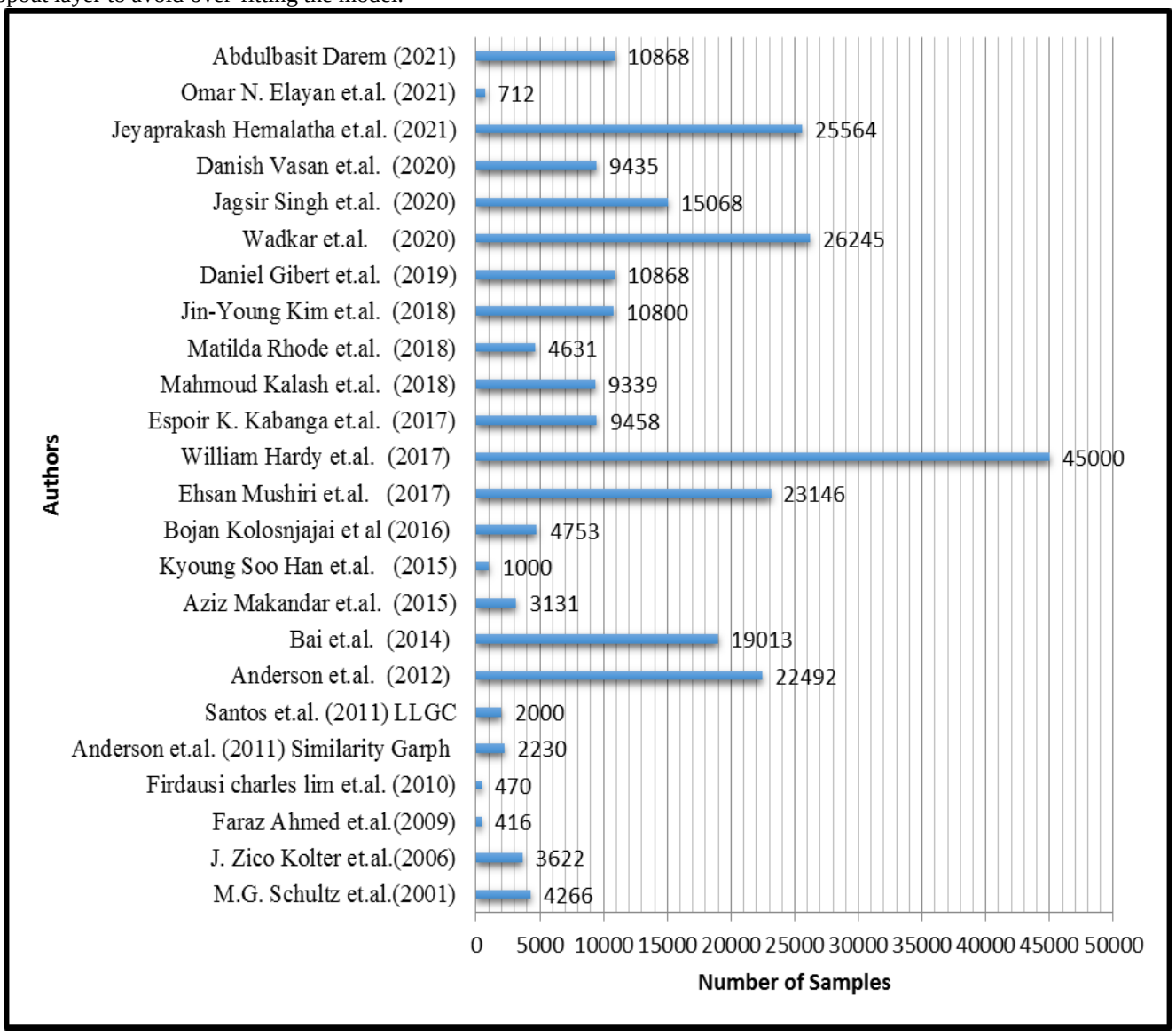


Figure 4 Number of Dataset Samples in Surveyed Papers

SURVEY ARTICLE

6.1.

Comparative Analysis of Machine Learning and Deep Learning

There is a significant contribution of machine and deep learning algorithms for malware detection. These algorithms provide great aid to increase the accuracy of malware detection. In 2001 M.G. Schultz [56] was the first to use machine learning in terms of malware detection. He changed the whole way of detection of malware. Various machine learning algorithms like Naïve Bayes, Multi Naïve Bayes, SVM, Decision Tree, Random Forest are used by authors. Some authors used machine learning as well as deep learning algorithms on their dataset. H.William [67] used both machine and deep learning algorithms and got maximum accuracy of 93.68% on deep learning while the highest accuracy on machine learning.

Algorithms are achieved by him is 88.24% using CG_SVM. Similarly, Jeyaprakash Hemalatha et.al. 2021[73] and Omar N. Elayan et.al. 2021[74] also achieved maximum accuracy using deep learning as 89.48% and 99.2% respectively. While they received the highest accuracy using machine learning are 82.10% and 97.8% using Random Forest respectively. Two observations are made while doing a comparative analysis of machine and deep learning algorithms, firstly while working on the same dataset deep learning is always providing better results than machine learning. Secondly, no single machine learning algorithm can provide the best result on all datasets.

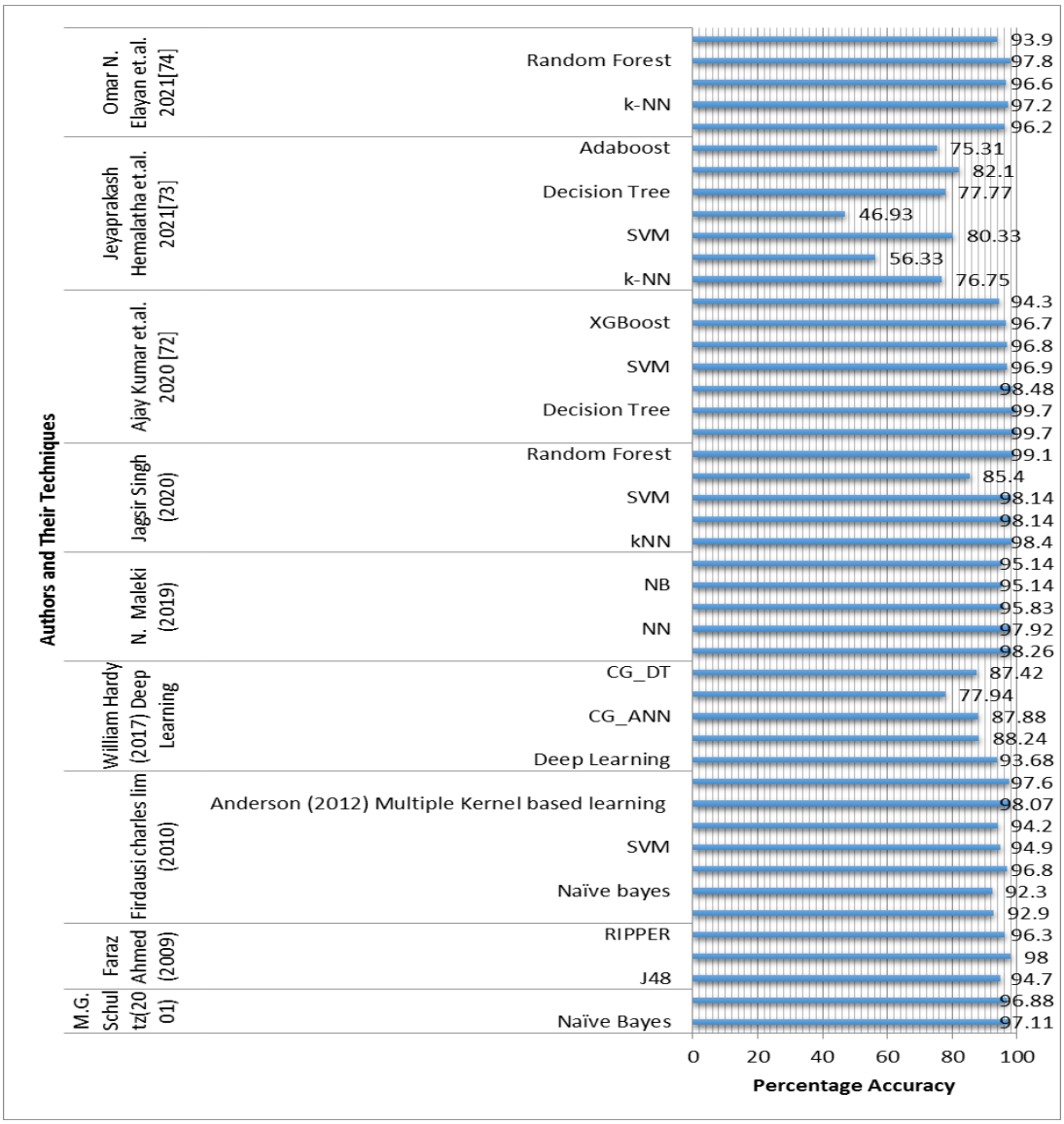


Figure 5 Malware Detection Accuracy in Surveyed Papers

SURVEY ARTICLE

7. CONCLUSION

There is an exponential growth of malware with each passing day. Malware writers are also writing code to evade the signatures used by various antivirus companies. This has become a great threat and providing security to the information has become challenging for cyber professionals.

Moreover, many tools are available in the market which can write malware code from scratch and can also obfuscate the malware code to prevent it from being detected at an early stage. Due to this mushroom growth of malware, efficient and intelligent algorithms for malware detection are required which can detect malware in less time to minimize the damage. The process of malware detection consists of two phases, the first phase is the extraction of features and the other phase is classification/clustering. Accuracy of classification majorly relies on the features so the feature extraction phase has become important. Malware features can be of two types: static well as dynamic. Static features are extracted from binary itself without its execution while dynamic features are extracted after the execution of the sample in an isolated environment.

In this paper, a survey is conferred on malware features and classification techniques using machine learning and deep learning. Firstly, the term malware and its types are explained. Then malware features like n-gram, opcodes, call graph, control flow graph, PE file characteristics, Memory Access, CPU registers, Raised Exceptions, Network, Sandbox Submissions, and API calls are explained. A summary is also provided for malware features used by various authors. Furthermore, machine learning techniques like Naive Bayes, kNN,

VM, Decision Tree, and Random Forest are explained for the classification of malware. Machine learning classifiers used by authors along with the advantages and disadvantages faced by them are also summarized.

Then deep learning is explained and a list of the work done by various authors for malware classification using deep learning is presented. Afterward survey results are explained which consists of the number of dataset samples used in surveyed papers and malware detection accuracy in surveyed papers. As per observations made in this paper, there is no single classifier that can always provide the best result in all kinds of features. Each type of malware detection, namely static and dynamic, has its advantages and disadvantages. Extraction of static features is fast and can detect malware at an early stage while these cannot detect zero-day-malware while dynamic features are more reliable to detect this zero-day-malware. However, to enhance the performance of malware classification abundant number of samples of malware as well as benign are required for training.

REFERENCES

[1] "Panda Lab: Pandalabs annual report 2015," 2015. http://www.pandasecurity.com/mediacenter/src/uploads/2014/07/Pandalabs-2015-anual-EN.pdf (accessed Mar. 19, 2020).
[2] "McAfee Labs Threats Reports – Threat Research | McAfee." https://www.mcafee.com/enterprise/en-in/threat-center/mcafee-labs/reports.html (accessed Jun. 01, 2021).
[3] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue Univ., vol. 48, pp. 2007–2, 2007.
[4] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," ACM Comput. Surv. CSUR, vol. 44, no. 2, pp. 1–42, 2008.
[5] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in The 5th Conference on Information and Knowledge Technology, 2013, pp. 113–120.
[6] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," J. Inf. Secur., vol. 2014, 2014.
[7] C. LeDoux and A. Lakhotia, "Malware and machine learning," in Intelligent Methods for Cyber Warfare, Springer, 2015, pp. 1–42.
[8] I. Basu, N. Sinha, D. Bhagat, and S. Goswami, "Malware detection based on source data using data mining: A survey," Am. J. Adv. Comput., vol. 3, no. 1, pp. 18–37, 2016.
[9] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," ACM Comput. Surv., vol. 50, no. 3, pp. 1–40, Jun. 2017, doi: 10.1145/3073559.
[10] D. Ucci, L. Aniello, and R. Baldoni, "Survey on the usage of machine learning techniques for malware analysis," ArXivPrepr. ArXiv171008189, 2017.
[11] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," Information, vol. 10, no. 4, p. 122, 2019.
[12] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," J. Comput. Virol. Hacking Tech., vol. 15, no. 1, pp. 15–28, 2019.
[13] "Worm definition by The Linux Information Project." http://www.linfo.org/worm.html (accessed Mar. 20, 2020).
[14] "Keyloggers: How they work and how to detect them (Part 1) | Securelist." https://securelist.com/keyloggers-how-they-work-and-how-to-detect-them-part-1/36138/ (accessed Mar. 20, 2020).
[15] J. Allain, The Ugly Truth About Adware and Spyware. Lulu Press, Inc, 2015.
[16] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," Inf. Secur. Tech. Rep., vol. 14, no. 1, pp. 16–29, 2009.
[17] C.-T. Lin, N.-J. Wang, H. Xiao, and C. Eckert, "Feature Selection and Extraction for Malware Classification.," J Inf Sci Eng, vol. 31, no. 3, pp. 965–992, 2015.
[18] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in Proceedings of the sixth ACM conference on data and application security and privacy, 2016, pp. 183–194.
[19] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification: bridging the static/dynamic gap," p. 12, 2012.
[20] J. Sexton, C. Storlie, and B. Anderson, "Subroutine based detection of APT malware," J. Comput. Virol. Hacking Tech., vol. 12, no. 4, pp. 225–233, 2016.
[21] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," Future Gener. Comput. Syst., vol. 90, pp. 211–221, 2019.
[22] P. Khodamoradi, M. Fazlali, F. Mardukhi, and M. Nosrati, "Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms," in 2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), 2015, pp. 1–6.

SURVEY ARTICLE

[23] M. Belaoued, A. Boukellal, M. A. Koalal, A. Derhab, S. Mazouzi, and F. A. Khan, "Combined dynamic multi-feature and rule-based behavior for accurate malware detection," Int. J. Distrib. Sens. Netw., vol. 15, no. 11, p. 1550147719889907, 2019.

[24] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K.-K. R. Choo, "An opcode-based technique for polymorphic Internet of Things malware detection," Concurr. Comput. Pract. Exp., vol. 32, no. 6, p. e5173, 2020.

[25] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, Oakland, CA, USA, 2001, pp. 38–49. Accessed: Apr. 06, 2020. [Online]. Available: http://ieeexplore.ieee.org/document/924286

[26] R. Ito and M. Mimura, "Detecting Unknown Malware from ASCII Strings with Natural Language Processing Techniques," in 2019 14th Asia Joint Conference on Information Security (AsiaJCIS), 2019, pp. 1–8.

[27] A. A. Elhadi, M. A. Maarof, and B. Barry, "Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph," Int. J. Secur. Its Appl., vol. 7, no. 5, pp. 29–42, Sep. 2013, doi: 10.14257/ijsia.2013.7.5.03.

[28] M. Graziano et al., "Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence," in 24th {USENIX} Security Symposium ({USENIX} Security 15), 2015, pp. 1057–1072.

[29] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13, Chicago, Illinois, USA, 2013, p. 1357. doi: 10.1145/2487575.2488219.

[30] J. Kwon and H. Lee, "BinGraph: Discovering mutant malware using hierarchical semantic signatures," in 2012 7th International Conference on Malicious and Unwanted Software, Fajardo, PR, USA, Oct. 2012, pp. 104–111. doi: 10.1109/MALWARE.2012.6461015.

[31] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna, "Polymorphic worm detection using structural information of executables," in International Workshop on Recent Advances in Intrusion Detection, 2005, pp. 207–226.

[32] I. Chionis, S. Nikolopoulos, and I. Polenakis, "A survey on algorithmic techniques for malware detection," 2013.

[33] T.-Y. Wang, C.-H. Wu, and C.-C. Hsieh, "Detecting Unknown Malicious Executables Using Portable Executable Headers," in 2009 Fifth International Joint Conference on INC, IMS and IDC, Seoul, South Korea, 2009, pp. 278–284. doi: 10.1109/NCM.2009.385.

[34] G. Laurenza, L. Aniello, R. Lazzeretti, and R. Baldoni, "Malware Triage Based on Static Features and Public APT Reports," in Cyber Security Cryptography and Machine Learning, vol. 10332, S. Dolev and S. Lodha, Eds. Cham: Springer International Publishing, 2017, pp. 288–305. doi: 10.1007/978-3-319-60080-2_21.

[35] M. Asquith, "Extremely scalable storage and clustering of malware metadata," J. Comput. Virol. Hacking Tech., vol. 12, no. 2, pp. 49–58, May 2016, doi: 10.1007/s11416-015-0241-3.

[36] J. Yonts, "Attributes of malicious files," Inst. InfoSec Read. Room, 2012.

[37] J. Bai, J. Wang, and G. Zou, "A Malware Detection Scheme Based on Mining Format Information," Sci. World J., vol. 2014, pp. 1–11, 2014, doi: 10.1155/2014/260905.

[38] M. Wadkar, F. Di Troia, and M. Stamp, "Detecting malware evolution using support vector machines," Expert Syst. Appl., vol. 143, p. 113022, 2020.

[39] M. Egele, M. Woo, P. Chapman, and D. Brumley, "Blanket execution: Dynamic similarity testing for program binaries and components," in 23rd {USENIX} Security Symposium ({USENIX} Security 14), 2014, pp. 303–317.

[40] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, "OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection," in International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions, vol. 189, Á. Herrero, V. Snášel, A. Abraham, I. Zelinka, B. Baruque, H. Quintián, J. L. Calvo, J. Sedano, and E. Corchado, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 271–280. doi: 10.1007/978-3-642-33018-6_28.

[41] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in International Workshop on Recent Advances in Intrusion Detection, 2007, pp. 178–197.

[42] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering.," in NDSS, 2009, vol. 9, pp. 8–11.

[43] T. Lee, "Behavioral classification," Proc. EICAR 2006 4, 2006.

[44] M. Lindorfer, C. Kolbitsch, and P. M. Comparetti, "Detecting environment-sensitive malware," in International Workshop on Recent Advances in Intrusion Detection, 2011, pp. 338–357.

[45] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis, "Measuring and Detecting Malware Downloads in Live Network Traffic," in Computer Security – ESORICS 2013, vol. 8134, J. Crampton, S. Jajodia, and K. Mayes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 556–573. doi: 10.1007/978-3-642-40203-6_31.

[46] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis, "Measuring and detecting malware downloads in live network traffic," in European Symposium on Research in Computer Security, 2013, pp. 556–573.

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

[48] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," ACM Comput. Surv., vol. 44, no. 2, pp. 1–42, Feb. 2012, doi: 10.1145/2089125.2089126.

[49] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in Proceedings of the 1st conference on India software engineering conference - ISEC '08, Hyderabad, India, 2008, p. 5. doi: 10.1145/1342211.1342215.

[50] A. Pfeffer et al., "Malware Analysis and attribution using Genetic Information," in 2012 7th International Conference on Malicious and Unwanted Software, Fajardo, PR, USA, Oct. 2012, pp. 39–45. doi: 10.1109/MALWARE.2012.6461006.

[51] "Rootkits - Computing and Software Wiki." http://wiki.cas.mcmaster.ca/index.php/Rootkits (accessed Mar. 21, 2020).

[52] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences," in AI 2016: Advances in Artificial Intelligence, vol. 9992, B. H. Kang and Q. Bai, Eds. Cham: Springer International Publishing, 2016, pp. 137–149. doi: 10.1007/978-3-319-50127-7_11.

[53] K. Huang, Y. Ye, and Q. Jiang, "ISMCS: An intelligent instruction sequence based malware categorization system," in 2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, Hong Kong, China, Aug. 2009, pp. 509–512. doi: 10.1109/ICASID.2009.5276989.

[54] D. Uppal, R. Sinha, V. Mehra, and V. Jain, "Malware detection and classification based on extraction of API sequences," in 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Delhi, India, Sep. 2014, pp. 2337–2342. doi: 10.1109/ICACCI.2014.6968547.

[55] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," Int. J. Comput. Appl., pp. 1–8, Feb. 2020.

[56] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, 2000, pp. 38–49.

SURVEY ARTICLE

[57] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables," in Machine Learning and Data Mining for Computer Security, Springer, 2006, pp. 47–63.
[58] F. Ahmed, H. Hameed, M. Z. Shafiq, and M. Farooq, "Using spatio-temporal information in API calls with machine learning algorithms for malware detection," in Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence, 2009, pp. 55–62.
[59] C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining for malware detection," 2010.
[60] I. Firdausi, A. Erwin, and A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," in 2010 second international conference on advances in computing, control, and telecommunication technologies, 2010, pp. 201–203.
[61] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," J. Comput. Virol., vol. 7, no. 4, pp. 247–258, 2011.
[62] I. Santos, J. Nieves, and P. G. Bringas, "Semi-supervised learning for unknown malware detection," in International Symposium on Distributed Computing and Artificial Intelligence, 2011, pp. 415–422.
[63] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification: bridging the static/dynamic gap," in Proceedings of the 5th ACM workshop on Security and artificial intelligence, 2012, pp. 3–14.
[64] J. Bai, J. Wang, and G. Zou, "A malware detection scheme based on mining format information," Sci. World J., vol. 2014, 2014.
[65] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in Australasian Joint Conference on Artificial Intelligence, 2016, pp. 137–149.
[66] E. Moshiri, A. B. Abdullah, R. Mahmood, and Z. Muda, "Malware Classification Framework for Dynamic Analysis using Information Theory," Indian J. Sci. Technol., vol. 10, no. 21, pp. 1–10, 2017.
[67] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, "Dl4md: A deep learning framework for intelligent malware detection," in Proceedings of the International Conference on Data Science (ICDATA), 2016, p. 61.
[68] M. A. Jerlin and K. Marimuthu, "A new malware detection system using machine learning techniques for API call sequences," J. Appl. Secur. Res., vol. 13, no. 1, pp. 45–62, 2018.
[69] N. Maleki, "A behavioral based detection approach for business email compromises," University of New Brunswick., 2019.
[70] M. Wadkar, F. Di Troia, and M. Stamp, "Detecting malware evolution using support vector machines," Expert Syst. Appl., vol. 143, p. 113022, 2020.
[71] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," Int. J. Comput. Appl., pp. 1–8, 2020.
[72] A. Kumar et al., "Malware Detection Using Machine Learning," in Iberoamerican Knowledge Graphs and Semantic Web Conference, 2020, pp. 61–71.
[73] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient DenseNet-based deep learning model for malware detection," Entropy, vol. 23, no. 3, p. 344, 2021.
[74] O. N. Elayan and A. M. Mustafa, "Android Malware Detection Using Deep Learning," Procedia Comput. Sci., vol. 184, pp. 847–852, 2021.
[75] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," IBM J. Res. Dev., vol. 3, no. 3, pp. 210–229, Jul. 1959, doi: 10.1147/rd.33.0210.
[76] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," ArXivPrepr. ArXiv13024964, 2013.
[77] E. Fix and J. L. Hodges, "Discriminatory analysis: Nonparametric discrimination: Consistency properties: (471672008-001)." American Psychological Association, 1951. doi: 10.1037/e471672008-001.
[78] T. Joachims, "Making large-scale support vector machine learning practical, Advances in Kernel Methods," Support Vector Learn., 1999.
[79] J. R. Quinlan, "Induction of decision trees," Mach. Learn., vol. 1, no. 1, pp. 81–106, Mar. 1986, doi: 10.1007/BF00116251.
[80] L. Breiman, "[No title found]," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[81] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acmsigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.
[82] P. M. Kavitha and B. Muruganantham, "A study on deep learning approaches over Malware detection," in 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE), 2020, pp. 1–5.
[83] A. Makandar and A. Patrot, "Malware analysis and classification using artificial neural network," in 2015 International conference on trends in automation, communications and computing technology (I-TACT-15), 2015, pp. 1–6.
[84] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," Int. J. Inf. Secur., vol. 14, no. 1, pp. 1–14, 2015.
[85] E. K. Kabanga and C. H. Kim, "Malware images classification using convolutional neural network," J. Comput. Commun., vol. 6, no. 1, pp. 153–158, 2017.
[86] M. Kalash, M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in 2018 9th IFIP international conference on new technologies, mobility and security (NTMS), 2018, pp. 1–5.
[87] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," in Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, 2018, pp. 127–134.
[88] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," Comput. Secur., vol. 77, pp. 578–594, 2018.
[89] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," Inf. Sci., vol. 460, pp. 83–102, 2018.
[90] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," Comput. Netw., vol. 171, p. 107138, 2020.
[91] A. Darem, J. Abawajy, A. Makkar, A. Alhashmi, and S. Alanazi, "Visualization and deep-learning-based malware variant detection using OpCode-level features," Future Gener. Comput. Syst., vol. 125, pp. 314–323, 2021.
[92] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6.
[93] D. Song et al., "BitBlaze: A new approach to computer security via binary analysis," in International Conference on Information Systems Security, 2008, pp. 1–25.

Authors

Manish Goyal is pursuing his Ph.D in Computer Science and Engineering at I. K. Gujral Punjab Technical University, Kapurthala. He did his Bachelor of Technology in Computer Science and Engineering from Yadavindra College of Engineering, Punjabi University GuruKashi Campus, Talwandi Sabo. He did his Master ofTechnology in Computer Science and Engineering fromYadavindra College of Engineering, Punjabi UniversityGuru Kashi Campus, Talwandi Sabo. His major area of research is Malware Detections and Information Security. (Email: manishgoyalpup@gmail.com, er.manishgoyal.ghudda@gmail.com).

Raman Kumar is working as an Assistant Professor in the Department of Computer Science and Engineering, I. K. Gujral Punjab Technical University, Kapurthala, formerly worked at DAV Institute of Engineering and Technology, Jalandhar as Assistant Professor in the Department of Computer Science and Engineering. Before joining

SURVEY ARTICLE

I K Gujral Punjab Technical University, Kapurthala, He did hisBachelor of Technology with honors in Computer Science and Engineering from Guru Nanak Dev University, Amritsar (A 5 Star NAAC University). He did his Master of Technology with honors in Computer Science and Engineering from Guru Nanak Dev University, Amritsar (A 5 Star NAAC University). He did his Ph. D with an A Grade in Computer Science and Engineering from the National Institute of Technology, Jalandhar (Deemed University). His major area of research in Cryptography, Security Engineering and Information security. He has published many papers in refereed journals, chapters, books and conference proceedings on his research areas. (Email: raman.kumarptu@gmail.com, er.ramankumar@aol.in).

How to cite this article:

Manish Goyal, Raman Kumar, "A Survey on Malware Classification Using Machine Learning and Deep Learning", International Journal of Computer Networks and Applications (IJCNA), 8(6), PP: 758-775, 2021, DOI: 10.22247/ijcna/2021/210724.