iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** X **Month of publication:** October 2022

DOI: https://doi.org/10.22214/ijraset.2022.47021

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue X Oct 2022- Available at www.ijraset.com

Malware Analysis Using Memory Forensics

Pragya Paramita Das
Master of Technology in Information Technology (Cyber Security), School of Engineering, Pune MIT-ADT University,

Abstract: ***Malware is still the most dangerous issue facing internet users in today's online environment. The newly created malware is separate from the traditional kind, has a more dynamic design, and typically combines traits from two or more different malware types. comparing the various memory acquisition tools that are available, each of which has a varying performance dependent on the setups, installed hardware, and operating system version. If the ending character is not present. To address the growing malware issue, new methodologies like machine learning must be employed. Investigate how cybersecurity is used in this study for malware detection and machine learning. In this study will look at the PE (portable executable) headers of malware and non-malware samples in order to build a malware classifier that can identify if malware is there or not. The development of behavior-based malware detection and classification methods using various machine learning approaches is addressed in this study along with behavior-based detection methods itself.***
Keywords: Cybersecurity, detection, malware, machine learning,PE headers, classifier, preparation, memory forensics

I. INTRODUCTION

The usage of the internet, computers, and smart gadgets is widespread nowadays, and many people use them on a daily basis. In the same way that there are good people and bad people everywhere we travel, the online world certainly has its share of nefarious characters that wish to use loyal users for their own gain [19]. Malware attacks have become increasingly complicated in recent years. Malware is the most potent menace to the cyber world despite advances in detection and classification of the threat into its correct family class and ongoing evolution. Computers and the internet have been widely used by everyone from individuals to the most secure and confidential corporations in the world due to the wide range of services given, including communication, e-learning, entertainment, e-banking, shopping, and similar. [1]. Private information is frequently stored on these systems, which are vulnerable to assault. These assaults might harm these systems and result in large fines [2]. Malicious software, also referred to as malware, is one of the most dangerous risks. Malware is any program that has the ability to disrupt the system by removing, adding, or changing data or system software [3]. The usage of the internet, computers, and smart gadgets is widespread nowadays, and many people use them on a daily basis. In the same way that there are people with good and ill intentions everywhere we travel, this is also true in the online world, where such services are being used by an increasing number of people [5]. As in the physical world, there are persons on the web with malicious motives who prey on trusting customers whenever money is involved [9]. Examples of commercial services on the internet include online banking and advertising. These individuals are able to carry out their objectives thanks to malicious software, such as malware. Malware assaults have evolved in complexity over the past few years. Malware continues to evolve, making it the most significant threat to the online world, despite advancements in detection and classification of malware into their correct family classes throughout time. Malware comes in a variety of forms depending on the reason for its use, such as ransomware used for financial gain, spying spyware, etc. [1]. We need a fundamental understanding of the types of malwares and the methods they employed in order to analyze malware using machine learning. Their disagreement over the school justified their behaviors. as described below [9]

1) *Virus:* Virus is simply like other programs. The main difference is that the program runs within the system without prior permission of the user & also replicates itself to infect other programs within the machine.
2) *Worm:* Worms are nothing but the improved version of the virus. The main difference is that everyone the machines connected over the network are in danger and there's the likelihood that they get infected with the worm.
3) *Trojan:* The main aim of the Trojan design is that it should seem like legitimate software, so users were tricked into considering malicious as a legitimate one.
4) *Adware:* The main aim of this sort of malware is to display advertisements on the target machine.
5) *Spyware:* The name suggests, the malware that spies on the target and delivers sensitive, vulnerable information to the attacker.
6) *Backdoor:* The backdoor may be a style of malware that's used for creating a back door for entry into the target machine. It doesn't affect the system badly.
7) *Ransomware:* Ransomware is that the most well-liked form of malware nowadays. It encrypts all user data within the machine and asks for a ransom to urge them back machine to normal.

A. *Forensics of Memory*

The posterior is the general approach in memory forensics, regardless of whether you use it for malware investigation or incident response.[7]

Memory Development This entails obtaining (or jilting) a target machine's memory in order to fragment it. The target machine will either be a system (on your network) that you simply suspect to be infected or it may be an analysis machine in your lab environment where you executed the malware sample, depending on whether you're probing an infected system or using memory forensics as part of your malware analysis [9]. Memory evaluation This stage entails analyzing the memory after it has been abandoned to fragment in order to look for and value forensic remnants.

II. EVOLUTION OF MALWARE

The variety, complexity, and vacuity of malicious software provide huge obstacles in protecting networks and computer systems from attacks. Security researchers and scientists must develop stronger cyber defenses to stay up with malware's constant evolution [3]. Malware has become more prevalent as a result of the employment of polymorphic and metamorphic techniques to hide their genuine intentions and evade detection. Polymorphic malware makes use of a polymorphic machine in order to modify the law while preserving all of the original functionality [11]. Packaging and encryption are the two most used methods of law evasion [3]. Packers conceal a program's true law by adding one or more levels of contraction. Additionally, the discharging procedures reinstate the original law and carry it out in runtime memory. The virus or a portion of it is ciphered and altered by crypters to make it more difficult for experimenters to analyses the software. An end of a crypter is employed for nefarious law encryption and decryption [2]. When it spreads, metamorphic malware changes the law to its original form. The developers of malware have access to a variety of metamorphosis techniques, such as register renaming, law expansion, law revulsion, and the insertion of scrap law. The combination of the methods listed below led to a progressively smaller addition of malware, making forensic analyses of malware instances time-consuming, costly, and increasingly difficult. Traditional antiviral results that use hand-ground and heuristic/behavioral calculation methods have some shortcomings [8]. An executable can be distinguished from another by a hand, which is a special point or group of traits. The identification of unknown varieties of malware is still beyond the capabilities of hand- crafted techniques. Security researchers proposed geste- based discovery to address these issues. This method may require some time to search and estimate [11] but analyses the attributes and geste of the train to determine whether it is malware [14]. As machine literacy is well adapted for recycling vast volumes of data, experimenters have started enforcing it to condense their results in order to overcome the prior drawbacks of standard antivirus machines and stay up with new threats and variants [4].

III. MALWARE DETECTION

In this technique, cybercriminals spread malware in an effort to convince victims to install it. Drug users also don't understand the programmed because it appears to be legal [19]. Typically, we install it with the assumption that it is secure, but unfortunately, it is a big problem. The malware enters your system in this way. When displayed on a screen, it scatters and blends into numerous lines, making it extremely difficult to spot. It may establish a direct connection to the operating system and begin cracking it in order to pierce and record specific or relevant information [5]. The search for malware lines and directories is referred to as malware discovery. Malware detection is reliable and successful because to the variety of tools and techniques that are accessible. The following are some common methods for finding malware: hand-grounding and heuristic analysis

• Sandbox • Anti-malware software

There have been implemented a number of classifiers, including direct classifiers (logistic retrogression, naive Bayes classifier), support for vector ministries, neural networks, arbitrary timbers, etc. Malware can be linked by both static and dynamic analysis, without breaking the law, through behavioral analysis.

A. *Need For Machine Learning In Malware Detection*

The available antivirus programs primarily use signatures to detect malware. These signatures for detecting the infection are taken from malware samples that have already been gathered. If the virus has previously been identified, these signature-based solutions perform quite well, but they are unable to detect new copies of malware. Therefore, signature-based solutions aren't always enough. False positive and false negative rates are similarly high [4]. New detection techniques are required to combat the threat posed by upgraded malware. Possible solutions to this issue include [20] integrating signature-based analysis with machine learning approaches, which can produce higher accuracy than using a single signature-based strategy for detection alone. When using a signature-less technique, it is necessary to categorize a programme as harmful by taking into account a few preset rules.

One set of suspicious features might include things like "connection established to odd destination," "file or programme permission altered implicitly," "unwanted registry key modifications made," etc. Then, if any software that activates the aforementioned features exceeds the defined threshold, it can be deemed hazardous to the machine. Although this approach has some success, this is not always the case because each feature's influence varies depending on the situation. For instance, "Connection formed" typically has more of an effect on the system's danger or damage than "registry key changed." It should be highlighted that combining these qualities efficiently may be less likely than considering them separately [13]. These associations can be taken into account by machine learning algorithms to have a more accurate identification of harmful programs.

Machine Literacy has created a drastic change in numerous diligences, including cybersecurity, over the last decade. Among cybersecurity experts, there's a general belief that AI- poweredanti-malware tools can help descry ultramodern malware attacks and boost surveying machines. evidence of this belief is the number of studies on malware discovery strategies that exploit machine literacy reported in the last many times. The number of exploration papers released in 2019 is 7720, a 95 percent rise over 2018 and a 476 percent increase over 2020, according to Google Scholar,1. This rise in the number of studies is the product of several factors, including but not limited to the increase in intimately labelled malware feeds, the increase in computing capacity at the same time as its price drop, and the elaboration of the field of machine literacy, which has achieved ground- breaking success in a wide range of tasks similar as computer vision and speech recognition(6). Depending on the type of analysis, conventional machine literacy styles can be distributed into two main orders, static and dynamic approaches. The primary difference between them is that static styles prize features from the static malware analysis, while dynamic styles excerpt features from the dynamic analysis. A third order may be considered, known as cold-blooded approaches. mongrel styles incorporate rudiments of both static and dynamic analysis. In addition, learning features from raw inputs in different fields have outdistanced neural networks. The performance of neural networks in the malware sphere is imaged by recent developments in machine literacy for cybersecurity. (6)

IV. RELATED WORKS

The malicious binaries are actually executed in a sandboxed environment and behavioral report is generated [3].

Researcher uses conventional and recurrent neural networks to classify malware, but executes the malware in protected environment. The labels needed for the supervised learning has been obtained from Virus Total API.[4]

In this paper used deep learning to classify benign and malicious mobile applications (Android). 202 features are extracted which includes permissions, sensitive ap calls and dynamic behavior [8]. The deep learning model used here is quite interesting [9]. A semi supervised model which uses Restricted Boltzmann Machines (RBM) in pre-training phase(unsupervised) followed by regular supervised backpropagation phase [7]. Paper proposes a new malware classification technique based on the maximal common subgraph detection problem [5]. Extracts various features like byte entropy, PE metadata, strings and import features [11]. This paper uses neural networks to classify whether a file is benign or malware. The architecture of DNN has one input layer, two hidden layer and one output layer [10]. The activation function used in the hidden layers is Parametric ReLU. The activation function used in output layer is sigmoid activation function since it is a binary classification problem [5].

N-gram based signature generation for malware detection [8]. N-grams of every file is extracted which is used to generate signatures for detection of malware [11]. Using windows API calls from Import address table which is similar to our approach to detect zero-day malwares [9]. This paper has tried 8 different type of classifiers KNN, NB, Neural Networks – Backpropagation, SVM Normalized Poly Kernel, SVM Poly Kernel, SVM Puk, SVM Radial Basis Function (RBM) [15].

V. TRADITIONAL MACHINE LEARNING APPROACH

The research and application of machine literacy outcomes to the tasks of malware finding and bracketing have increased during the past ten years. Without the fusion of three recent advancements, machine literacy techniques would not have been as successful and connected.

1) The first change is the rise of labelled malware feeds, which means that for the first time, labelled malware is available to both the security and exploration communities. These feeds range in size from tiny, high-quality samples, like the bones Microsoft provided for the Gathering Anti-Malware Prediction Challenge (Ronen et al., 2018), to enormous quantities of malware, like the Zoo (Yuval Nativ, 2015) or Virus Share (2011).

2) The alternative scenario is that processing power has rapidly expanded while also becoming cheaper and closer to the means of the most ambitious experimenters. It thus enabled experimenters to accelerate the iterative training process and fit larger and more complicated models to the continuously growing data.

3) Third, the field of machine literacy has advanced more quickly over the past few decades, reaching ground-breaking achievements in terms of finesse and scalability on a variety of tasks, including computer vision, speech recognition, and natural language processing. A workflow in machine literacy is an iterative process that includes acquiring available data, structuring and preparing the data, building models, verifying the models, and integrating the models into the final product. The data treatment procedure of classic machine learning approaches includes preprocessing the executable to prize a set of features that give an abstract view of the software rather than dealing with raw malware. Later, the features are trained into a model to help it divide the task at hand. Due to the diversity of malware characteristics, it's crucial to distinguish between various types of malwares in order to better comprehend their capabilities. This goes beyond simply describing malicious software. The outcome returned is the primary distinction between machine learning outcomes for malware discovery or classification.

VI. MACHINE LEARNING APPROACH

There are three sections in the paper workflow.

The dataset is imported, and the various columns are discussed in the next section.

1) *Data Drawing:* After analyzing the dataset, the necessary steps are conducted to remove all null values and columns with low significance so that they won't be a problem during the training phase.
2) *Testing and Data Training:* When the data was transparent and prepared for training, we exposed it as a training dataset and a testing dataset at an 80:20 rate, revealing the information at an 80:20 rate overall.

In order to get the best accuracy, I will test four different algorithms in this research to see which will provide us with greater precision.

A. Machine Learning Algorithms

1) *Decision Tree:* Category and reversion models are largely created using decision trees. The malware is categorized in the design using a decision tree classifier. It creates a tree-like structure for the categories. Using a decision tree requires supervision. performed so beautifully, scoring 98 on delicacy. It also distinguishes between malware that falls into two categories: malicious and benign.
2) *Random Forest:* Using supervised categories, Random Forest creates results. Because Random Forest is more like collaborative decision trees that can create a timber, it closely relates to the Decision Tree algorithm. The number of trees in a piece of wood has a significant impact on the delicacy of the wood; fewer trees produce wood with lower delicacy, but higher performance; more trees provide wood with higher delicacy, but slower execution.
3) *Gradient Boosting:* Using supervised categories, Random Forest creates results. Because Random Forest is more like collaborative decision trees that can create a timber, it closely relates to the Decision Tree algorithm. The number of trees in a piece of wood has a significant impact on the delicacy of the wood; fewer trees produce wood with lower delicacy, but higher performance; more trees provide wood with higher delicacy, but slower execution.
4) *Linear Regression:* With the help of this approach, we may see a direct connection between a dependent(y) variable and one or more independent(y) variables. Since direct retrogression demonstrates the direct relationship, it identifies how the dependent variable's value changes in response to the independent variable's value. The link between the variables is depicted by a graded straight line in the direct retrogression model. Finding the fashionable fit line that indicates the difference between prognosticated values and actually values should be decreased is our main goal when working with direct retrogression. The line with the best fit will make the fewest mistakes.

VII. IMPLEMENTATION

The machine learning methods described here are used for the analysis of malware in this section. Objectives, Features, and Machine Learning algorithms are the three essential steps in malware analysis, according to the author [17]. In malware detection, the emphasis is on determining if a particular sample is harmful. In malware similarity analysis, we can determine if a file is harmful by looking for attributes that can categorize it clearly and aid in malware detection. The second phase is feature extraction depending on the requirement after the analysis's goal has been decided. Applying a machine learning method to reach the goal comes after that final phase [17]. The primary goal of malware analysis is detection. Malware is discovered using a special signature that was created based on earlier samples that were appropriately identified as harmful or benign [17]. Identifying whether a particular sample is malicious or not is always the first and foremost objective. The majority of review work focuses on finding malware [2], [3], [14].

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue X Oct 2022- Available at www.ijraset.com

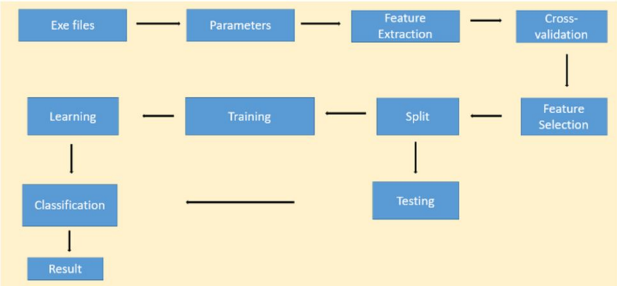


Fig: Working Architecture[3][6]

I developed a command-line tool for malware detection from memory that uses python code to extract the active processes from the suspected system's live memory. I used Python to construct a machine learning-based antivirus solution. The machine learns from training data sets and predicts results based on what it has learned.I import and read the dataset at first, and then I clean it up by removing extra features and null values. The dataset was then divided into training and testing. I import the required software to create six distinct classifiers. Once finished, I fit the data appropriately and provide predictions.
I am able to obtain a highly accurate prediction using a combination of these algorithms after training and testing the algorithms.

A. Dataset

Two datasets were employed: a training dataset and a test dataset. The clean files in this dataset (and in the scale-up dataset) are substantially bigger than the number of malware files because, as was already mentioned, my major objective is to achieve malware identification with as few (if any) false positives as feasible. 57 columns serve as features in the dataset, which contains roughly 138000 entries of legitimate and malicious PE headers. 96724 malicious files and 41323 legitimate ones make up the dataset. These data sets are used to test my machine and train it. Similar value-generating files for the selected feature set were only counted once. It should be noted that there are far less clean combinations—that is, combinations of feature values for clean files—in the datasets than there are malware-specific combinations. We require actual malware samples to prepare the dataset. Numerous malware samples have been gathered, largely through Virus Share, from open-source GitHub sources [2]. The majority of the malware that will be utilized for supervised learning is already classified in these sources.

1) *Data pre-processing:* One of the crucial steps prior to feature selection for the model is data pre-processing. If null values are present, delete them from the data collection during data pre-processing. Remove the columns that contain the categorical information. The first columns, Name and md5, which contain string values, are removed. Additionally, the last column legitimate contains a value that indicates if the binary is malicious or not.

B. Feature Extraction

There are two methods for removing the functionality from malware binary, which is primarily static or dynamic analysis utilized alone. The malware file is statically analyzed and examined without being put into action, every feature extracted is mostly from the PE header or by breaking down and examining the executable file assembled code. The executable file is used in dynamic analysis. In order to prepare dataset, we need to extract all the Import functions used by the malware. A small Python program is written which will extract all the imports from all the PE32 files present in the directory. MD5 hashing is used to prevent data duplication. Initially the program will create a file to store the imports used by all the executable of the same category. Creation to extract the static information of the program or software with the help of PE file python library and generate excel for storing information for every program or software. According to [14] the most form of malware falls under Portable Executable (PE) file format so I am proposing a system which classifies the malware which comes under portable executable file format. Some of the features of the datasets and the visualization of Imports used by various malware are listed below:

1) *SizeOfOptionalHeader* - Gets the size of the optional header, which is required for executable files but not for object files. This value should be zero for an object file.
2) *MajorLinkerVersion* - Gets the linker major version number.
3) *SizeOfCode* - Gets the size of the code (text) section, or the sum of all code sections if there are multiple sections.
4) *SizeOfInitializedData* - Gets the size of the initialized data section, or the sum of all such sections if there are multiple data sections.

5) *SizeOfUninitializedData* - Gets the size of the uninitialized data section (BSS), or the sum of all such sections if there are multiple BSS sections.
6) *AddressOfEntryPoint* - Gets the address of the entry point relative to the image base when the PE file is loaded into memory.
7) *BaseOfCode* - Gets the address of the beginning-of-code section relative to the image base when the image is loaded into memory.
8) *BaseOfData* - Gets the address of the beginning-of-data section relative to the image base when the image is loaded into memory.
9) *ImageBase* - Gets the preferred address of the first byte of the image when it is loaded into memory.
10) *SectionAlignment* - Gets the alignment (in bytes) of sections when they are loaded into memory.
11) *FileAlignment* - Gets the alignment factor (in bytes) that is used to align the raw data of sections in the image file.
12) *MajorOperatingSystemVersion* - Gets the major version number of the required operating system.
13) *MinorOperatingSystemVersion* - Gets the minor version number of the required operating system
14) *MajorImageVersion* - Gets the major version number of the image.
15) *MinorImageVersion* - Gets the minor version number of the image.
16) *MajorSubsystemVersion* - Gets the major version number of the subsystem.
17) *MinorSubsystemVersion* - Gets the minor version number of the subsystem.
18) *SizeOfImage* - Gets the size (in bytes) of the image, including all headers, as the image is loaded in memory.
19) *SizeOfHeaders* - Gets the combined size of an MS DOS stub, PE header, and section headers rounded up to a multiple of
20) *FileAlignment* - Gets the alignment factor (in bytes) that is used to align the raw data of sections in the image file.
21) *CheckSum* - Gets the image file checksum.
22) *Subsystem* - Gets the name of the subsystem that is required to run this image.
23) *DllCharacteristics* - Describes the characteristics of a dynamic link library. This enumeration supports a bitwise combination of its member values.
24) *SizeOfStackReserve* - Gets the size of the stack to reserve.
25) *SizeOfStackCommit* - Gets the size of the stack to commit.
26) *SizeOfHeapReserve* - Gets the size of the local heap space to reserve.
27) *SizeOfHeapCommit* - Gets the size of the local heap space to commit.
28) *NumberOfRvaAndSizes* - Gets the number of data-directory entries in the remainder of the PEHeader. Each describes a location and size.

In this part data is prepared for learning process. First columns containing string values are dropped and also last column containing value which tells about whether binary is malicious or legitimate the remaining data and data-frame is store in variable X. Similarly in variable y values from column legitimate are stored.

C. Feature Selection

Select the features which are closely related to the output or important for the prediction or classification of the given input. step Extra Tree Classifier from the sklearn library features is selected. Out of 57 features, 13 features are selected by the Extra tree classifier. To perform feature selection, the Extra-Trees classifier follows the conventional top-down approach and generates a collection of unpruned decision trees. In essence, it entails splitting a node in a tree while severely randomizing both attribute and cut-point selection. In the worst-case scenario, it generates entirely randomized trees with structures independent of the training sample's output values. It notably varies from conventional tree-based ensemble approaches on two counts: first, it separates nodes by selecting cut-points completely at random; second, it grows the trees using the entire training sample rather than a bootstrap replica. By using a majority vote, the forecasts of all the trees are pooled to determine the final prediction. Here, information gain will be the deciding factor. To generate independent Decision Trees for each and every feature, first determine the entropy of the data. Calculate the overall knowledge obtained for each feature, then choose the ones with the highest scores.

D. Training

After choosing the proper features, divide the data set into two halves for training and testing, as necessary. Ready to train the model once the dataset has been generated. There are 13 features in my model. The model is trained using four different algorithms: Decision Tree, Random Forest, Gradient Boost, and Linear Regression. I can easily see that Random Forest has the best accuracy with the fewest false positive and negative rates.

Algorithms	Accuracy
Random Forest	99.40%
Decision Tree	99.06%
Gradient Boost	98.81%
Linear Regression	52.92%

Table: Result Comparison

Now the Random Forest model and all the selected feature will be used for later prediction.

E. Classification

This is the last step where actual classification is predicted based on training done in the previous step. The algorithm selected for classification predicts whether the given input is a malicious or benign sample. In order to test the model on unseen file, it's required to extract the characteristics of the given file. Python's pefile.PE library is used to construct a build the above-mentioned feature vector and a ML model is used to predict the class for the given file based on already trained model.

VIII. CONCLUSION

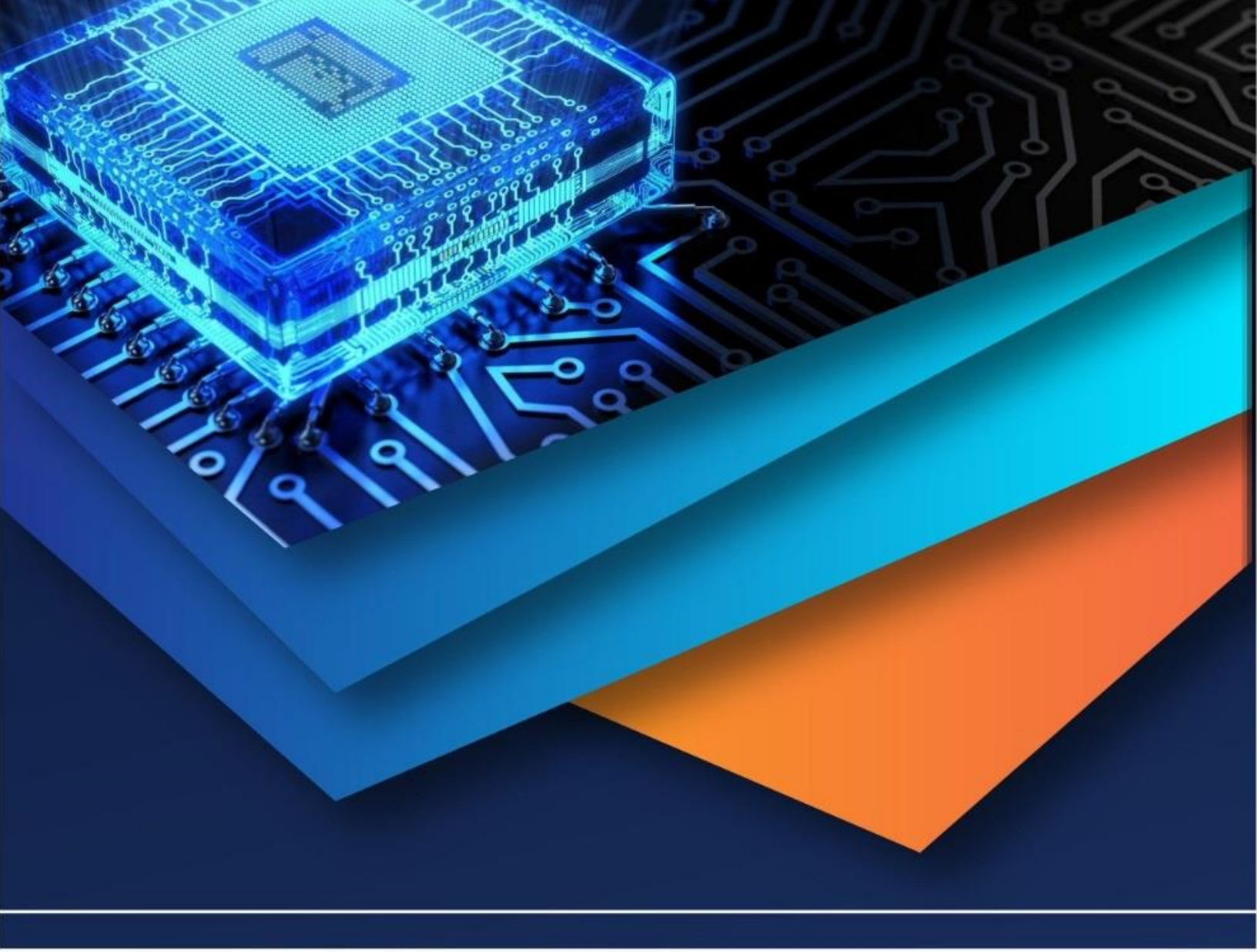
The machine learning approach considered were evaluated successfully.

The static features that were retrieved from both good and bad executable files were used by the machine learning method. This method makes it possible to determine if a particular file is harmful or not very quickly. It will be effective to utilize this method before signature-based ones in scenarios with high load to lessen the burden on dynamic analysis of executable. Although it is recommended that false negatives in these investigations be nil or almost zero. In the industrial context, the model is useless if the false-negative rate is larger. This program only runs on Windows. Additionally, it is unable to determine the malware's family. My major goal was to develop a machine learning system that can normally identify as many malware samples as feasible while adhering to the rigorous requirement of having a zero false positive rate. Despite coming very near to our goal, I still have a non-zero false positive rate. For this system to be included in a highly competitive commercial product, a variety of deterministic exemption mechanisms must be incorporated. In my opinion, machine learning-based malware detection will be an addition to, not a replacement for, the current malware detection techniques currently employed by anti-virus providers. Any commercial anti-virus program has memory and speed restrictions, so the most precise methods should be employed. In order to draw a conclusion, I'd like to say that this study needs more research in a few specific areas. As the number of internet users increases and new malware is created daily, new strategies are required to combat it.

REFERENCES

[1] Maryam Al-Janabi, Ahmad Mousa Altamimi, "A Comparative Analysis of Machine Learning Techniques for Classification and Detection of Malware", 2021 21st International Arab Conference on Information and Technology (AICT)
[2] Sana Qadir, Basirah Noor, "Applications of Machine Learning in Digital Forensics", 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)
[3] Sanket Agarkar, Soma Ghosh, "Malware Detection & Classification using Machine Learning", 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC) | 978-1-7281-8880-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/iSSSC50941.2020.935883
[4] Adam Duby, Teryl Taylor, Yanyan Zhuang, "Malware Family Classification via Residual Prefetch Artifacts", 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)
[5] Rajvardhan Patil, Wei Deng, "Malware Analysis using Machine Learning and Deep Learning techniques", SoutheastCon 2020 | 978-1-7281-6861-6/20/$31.00 ©2020 IEEE | DOI:10.1109/SoutheastCon44009.2020.936826
[6] Ashub Bin Asad, Raiyan Mansur, Safir Zawad, Nahian Evan, Muhammad Iqbal Hossain "Analysis of Malware Prediction Based on Infection Rate Using Machine Learning Techniques", 2020 IEEE Region 10 Symposium (TENSYMP), 5-7 June 2020, Dhaka, Bangladesh
[7] AV-Test, "The AV-test security report," 2020. [Online]. Available: https://www.av-test.org/.
[8] R. Sihwail, K. Omar and K. A. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," International Journal on Advanced Science, Engineering and Information Technology, vol. 8, no. 4–2, pp. 1662–1671, 2018.
[9] A. Khanan, S. Abdullah, A.H.M.E.D. Mohamed, A. Mehmood and K. A. Z. Ariffin, "Big data security and privacy concerns: A review," in Smart Technologies and Innovation for a Sustainable Future, Advances in Science, Technology & Innovation, Cham: Springer, pp. 55–61, 2019.
[10] Y. Ye, T. Li, D. Adjeroh and S. S. Iyengar, "A survey on malware detection using data mining techniques," ACM Computing Survey, vol. 50, no. 3, pp. 1–40, 2017.

[11] Mouhammd Al-kasassbeh Mohammad A. Abbadi, Ahmed M. Al- Bustanji. Robust intelligent malware detection using lightgbm algo- rithm. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9, 04 2020.
[12] Edward Raff and Charles Nicholas. An alternative to ncd for large sequences, lempel-ziv jaccard distance. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1007–1015, 2017.
[13] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Au- tomatic analysis of malware behavior using machine learning. Journal of Computer Security, 19(4):639–668, 2011.
[14] Igor Santos, Jaime Devesa, Felix Brezo, Javier Nieves, and Pablo Garcia Bringas. Opem: A static-dynamic approach for machine-learning-based malware detection. In International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions, pages 271–280. Springer, 2013.
[15] Joshua Saxe and Konstantin Berlin. Deep neural network-based malware detection using two-dimensional binary program features. In 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), pages 11–20. IEEE, 2015.
[16] Michael Sikorski and Andrew Honig. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press, USA, 1st edition, 2012.
[17] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. Computers & Security, 81:123–147, 2019.
[18] Dolly Uppal, Rakhi Sinha, Vishakha Mehra, and Vinesh Jain. Malware detection and classification based on extraction of api sequences. In 2014 International conference on advances in computing, communications and informatics (ICACCI), pages 2337–2342. IEEE, 2014.
[19] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. ACM Computing Surveys (CSUR), 50(3):1–40, 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY