Malware Detection System using Machine Learning Techniques

Deeksha Pal[1], Ashraf Ali[2], Vikas Chaudhary[3], Gaurav Kumar[4]
Dr. A.P. Srivastava [5] & Nitin Kumar Sharma[6]
[1,2,3,4]*UG Student, Department of Computer Science & Engg., NITRA Technical Campus, UP, India*
[5]*Asst. Professor and Head, Department of Computer Science & Engg., NITRA Technical Campus, UP, India*
[6]*Assistant Professor, Department of Computer Science & Engineering, NITRA Technical Campus, UP, India*

Abstract - Malware detection plays a critical role in securing endpoints such as workstations, servers, cloud instances, and mobile devices. It involves identifying and mitigating malicious activities triggered by various forms of malware, including viruses, trojans, worms, ransomware, adware, and spyware.

With the rapid proliferation of sophisticated and evasive malware variants, such as polymorphic and metamorphic malware, traditional signature-based and heuristic approaches are no longer sufficient. These conventional methods often fail to detect previously unseen or obfuscated threats, making the adoption of more intelligent and adaptive systems essential. In response to this challenge, this project explores the application of machine learning techniques to enhance malware detection and classification.

Leveraging publicly available malware datasets, various supervised and unsupervised algorithms—such as Decision Trees, Random Forests, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Naïve Bayes, and Artificial Neural Networks (ANN)—will be employed and evaluated. Additionally, feature extraction and dimensionality reduction techniques such as Principal Component Analysis (PCA) will be applied to improve model performance.

The objective is to assess and compare the models using metrics such as accuracy, precision, recall, F1-score, and confusion matrix analysis.

This project aims to identify the most effective machine learning model for real-time, scalable, and accurate malware detection, contributing to improved cybersecurity practices and proactive threat mitigation.

Keywords: **Malware detection, machine learning, enterprise security, information gain, random forest, PE analysis, API calls.**

I. INTRODUCTION

Malware—malicious software designed to exploit, damage, or disrupt systems—has evolved into a sophisticated and financially motivated threat vector. Modern malware includes not only viruses and worms but also trojans, spyware, ransomware, and rootkits. Unlike earlier attacks motivated by notoriety, today's malware often serves commercial or criminal enterprises. Enterprise systems are particularly vulnerable due to their data richness and complexity.

Standard antivirus tools struggle to detect previously unseen threats (zero-day attacks), especially when adversaries use polymorphic or metamorphic techniques to obfuscate malware. These threats dynamically modify their codebase, rendering signature-based solutions ineffective.

Thus, there's an increasing demand for intelligent systems capable of identifying anomalies based on behavioral or static features rather than relying solely on known signatures.[1]

Machine learning models offer a compelling solution to this challenge, allowing systems to "learn" from known malware patterns and generalize detection to previously unseen variants.[4]

This research builds upon this paradigm and proposes a centralized machine learning-based malware detection framework that targets the API call patterns within Windows PE files—a rich source of indicative static features.[11]

Although malware variants are increasing in number and complexity, traditional antivirus scanners are falling short of meeting modern security needs. Script-kiddies, inexperienced individuals using pre-built malware tools, and automated attack frameworks are contributing to the surge in cyberattacks.

Government and commercial organizations continue to be major targets, with threats like ransomware, data theft, and spyware posing significant operational risks. Cyberattacks can manifest in various forms and scales, making it increasingly difficult to maintain cybersecurity resilience.[3]

One of the major challenges faced by the global cybersecurity community is the lack of skilled personnel capable of responding to these evolving threats. Additionally, international cybercrimes such as financial fraud, payment fraud, and online child exploitation demand global cooperation among law enforcement agencies.[8]

II. Current Antivirus Software

Conventional antivirus engines depend on pre-defined signatures to detect malicious code. While effective against known threats, these methods offer little protection against

new or rapidly evolving malware. Heuristic and behavioural approaches have been introduced but often result in high false positive rates or degrade system performance.
In enterprise contexts, even a minor breach can have catastrophic consequences. Furthermore, malware authors are continuously innovating, producing encrypted or self-modifying payloads that evade signature detection. Reports have shown a decline in the effectiveness of traditional antivirus products, highlighting the need for a more robust solution.

- Antivirus software is used to prevent, detect, and remove malware, including but not limited to computer viruses, computer worm, Trojan horses, spyware and adware. A variety of strategies are typically employed by the anti-virus engines. Signature-based detection involves searching for known patterns of data within executable code. However, it is possible for a computer to be infected with new virus for which no signatures exist [6]. To counter such “zero-day” threats, heuristics can be used, that identify new viruses or variants of existing viruses by looking for known malicious code. Some antivirus can also make predictions by executing

  files in a sandbox and analysing results.[12]

- Often, antivirus software can impair a computer's performance. Any incorrect decision may lead to a security breach, since it runs at the highly trusted kernel level of the operating system. If the antivirus software employs heuristic detection, success depends on achieving the right balance between false positives and false negatives. Today, malware may no longer be executable files. Powerful macros in Microsoft Word could also present a security risk. Traditionally, antivirus software heavily relied upon signatures to identify malware. However, because of newer kinds of malware, signature-based approaches are no longer
  effective[8]

- Although standard antivirus can effectively contain virus outbreaks, for large enterprises, any breach could be potentially fatal. Virus makes are employing "oligomorphic", "polymorphic" and, "metamorphic" viruses, which encrypt parts of themselves or modify themselves as a method of disguise, so as to not match virus signatures in the dictionary.[13]

- Studies in 2007 showed that the effectiveness of antivirus software had decreased drastically, particularly against unknown or zero day attacks. Detection rates have dropped from 40-50% in 2006 to 20-30% in 2007. The problem is magnified by the changing intent of virus makers. Independent testing on all the major virus scanners consistently shows that none provide 100% virus detection. The best ones provided as high as 99.6% detection, while the lowest provided only 81.8% in tests conducted in February 2010. All virus scanners produce false positive results as well, identifying benign files as malware.[12]

III. Related work

This section provides a summary of the surveyed research in the literature and discusses some of its defects. Table 1 sums up the main contributions of the surveys in the literature. We follow by presenting a brief description for each survey, and their flaws that we try to mitigate in our work.
Shabtai et al. (2009) provide a taxonomy for malware detection using machine learning algorithms by reporting some feature types and feature selection techniques used in the literature. They mainly focus on the feature selection techniques (Gain ratio, Fisher score, document fre-quency, and hierarchical feature selection) and classification algorithms
(Artificial Neural Networks, Bayesian Networks, Naïve Bayes, K-Nearest
Neighbor, etc). In addition, they review how ensemble algorithms can be used to combine a set of classifiers. Bazrafshan et al. (2013) iden-tify three main methods for detecting malicious software: signature-based methods, heuristic-based methods and behavior-based meth-ods. In addition, they investigate some features for malware detec-tion and discuss concealment techniques used by malware to evade detection. Nonetheless, the aforementioned research does not consider either dynamic or hybrid approaches. Souri et al. (2018) present a sur-vey of malware detection approaches divided into two categories: signature-based methods and behavior-based methods. However, the survey does not provide either a review of the most recent deep learning approaches or a taxonomy of the types of features used in data mining techniques for malware detection and classification. Ucci et al. (2019) categorize methods according to: (i) what is the target task they try to solve, (ii) what are the feature types extracted from Portable Executable files (PEs), and (iii) what machine learning algorithms they use. Although the survey provides a complete description of the fea-ture taxonomy, it does not outline new research trends, especially deep learning and multimodal approaches. Ye et al. (2017) cover traditional machine learning approaches for malware detection, that consists of feature extraction, feature selection and classification steps. However, important features such as the entropy or structural entropy of a file, and some dynamic features such as network activity, opcode and API traces, are missing. In addition, deep learning methods or multimodal approaches for malware detection, which have been hot topics for the last few years, are not covered. Lastly, Razak et al. (2016) provide a bibliometric analysis of malware. It analyzes the publications by coun-try, institution or authors related to malware. Nonetheless, the paper does not provide a description of the features employed by malware 2D. Gibert et al. Journal of Network and Computer
Applications 153 (2020) 102526

IV. OBJECTIVES

As we have seen, current antivirus engine techniques are not optimum in detecting viruses in real time. They may be useful

in controlling viruses once they infect systems, which is again, fateful for enterprises. This research is thus aimed at a central solution that works at the firewall level of the enterprise network. The complete system diagram is shown in Figure 1 and our process diagram is shown in Figure 2.

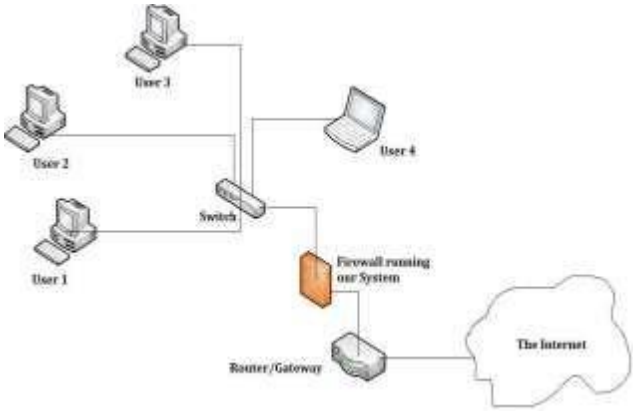

Figure 1: Network Diagram of the entire system

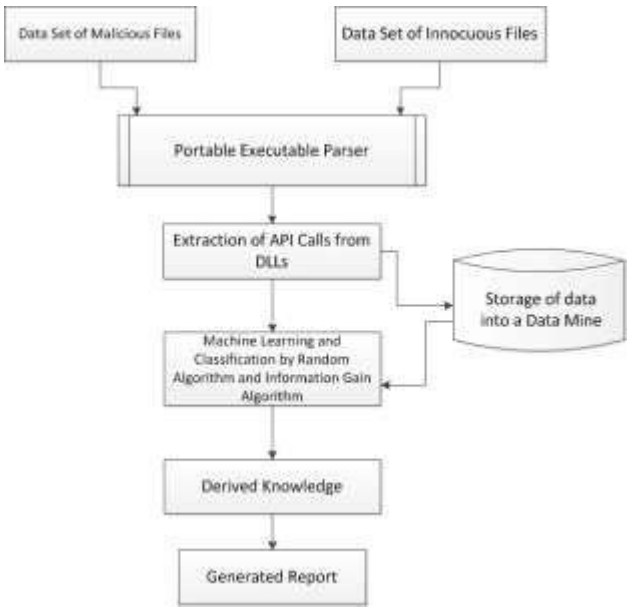

Figure 2: Process Diagram of our System

Portable Executable (PE)

This format is a file format for executables, object code and DLLs, used in 32-bit and 64-bit versions of Windows operating systems . The term "portable" refers to the format's versatility in numerous environments of operating system software architecture. The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code. This includes dynamic library references for linking, API export and import tables, resource management data and thread-local storage data. On NT operating systems, the PE format is used for EXE, DLL, SYS (device driver), and other file types.

A PE file consists of a number of headers and sections that tell the dynamic linker how to map the file into memory. An executable image consists of several different regions, each of which requires different memory protection. The Import address table (IAT), which is used as a lookup table when the application is calling a function in a different module. Because a compiled program cannot know the memory location of the libraries it depends upon, an indirect jump is required whenever an API call is made. As the dynamic linker loads modules and joins them together, it writes actual addresses into the IAT slots, so that they point to the memory locations of the corresponding library
functions.[13]

In our research, we extracted the PE Header from numerous infected and normal executables and using the IAT, extracted various API calls and stored them into a data mine. We then derived Information Gain (IG) for each function.

V. Algorithm for Information Gain

$$H(X) = -\sum_i P(X_i)\log_2(PX_i)$$

The entropy of a variable X is defined as:

Where in H(P), the P(X) is as follows: P(X) = Number of ·

PE· with· xi · ascertain· API

i. Total· number of · PE

And the entropy of X after observing values of another variable Y is defined as:

The amount by which the entropy of X decreases reflects additional information about X provided by Y is called information gain, given by:

IG(X | Y) = H(X) - H(X | Y)

Machine learning, a branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data, such as from sensor data or databases. A learner can take advantage of data to capture characteristics of interest of their unknown underlying probability distribution. Data can be seen as examples that illustrate relations between observed variables. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data[17]

Further, we apply the Random Forest Algorithm (RFA). This is a machine learning classification algorithm to construct the classifier to detect malware. A Random Forest is a classifier that is comprised of a collection of decision tree predictors. Each individual tree is trained on a partial, independently

sampled, set of instances selected from the complete training set. The predicted output class of a classified instance is the most frequent class output of the individual trees.

VI. Malware Analysis

The process of dissecting malware to understand how it works, determine its functionality, origin and potential impact is called mal- ware analysis. With the millions of new malicious programs in the wild, and the mutated versions of previously detected programs, total mal- ware encountered by security analysts has been growing over the past years.4 Consequently, malware analysis is critical to any business and infrastructure that responds to security incidents.

There are two fundamental approaches to malware analysis: (1) static analysis and (2) dynamic analysis. On the one hand, static analy- sis involves examining the malware without running it. On the other hand, dynamic analysis involves running the malware. An in-depth description of both approaches is provided in Sections.

Static Analysis:
Static analysis consists of examining the code or structure of the executable file without executing it. This kind of analysis can confirm whether a file is malicious, provide information about is functionality and can also be used to produce a simple set of signatures. For instance, the most common method used to uniquely identify a malicious pro- gram is hashing. That is, a hashing program produces a unique hash, a sort of fingerprint, that identifies the program. The two most popu- lar hash functions are the Message-Digest Algorithm 5 (MD5) and the Secure Hash Algorithm 1 (SHA-1). The most common static analysis approaches are:
Finding sequences of characters or strings. Searching through the strings of a program is the simplest way to obtain hints about its functionality. Strings extracted from the binary can contain references to file paths of files modified or accessed by the executable,

URLs to which the program accesses, domain names, IP addresses, attack commands, names of Windows dynamic link libraries (DLLs) loaded, registry keys, and so on. The utility tool Strings 5 can be used to search ASCII or Unicode strings ignoring context and formatting in an executable.

Gathering the linked libraries and functions of an executable, as well

as the metadata about the file included in the headers. These data provide information about code libraries and functionalities com- mon to many programs that programmers link so that they do not need to re-implement a certain functionality. The names of these Windows functions can give us an idea of what the executable does. The utility Dependency Walker6 is a free program for Microsoft Windows used to list the imported and exported functions of a PE file.

Analyze PE file headers and sections. The PE file headers provide more information than just imports. They contain metadata about the file itself, such as the actual sections of the file. One way to retrieve this information is with the PEView tool.7

Searching for packed/encrypted code. Malware writers usually use packing and encryption to make their files more difficult to analyze. Software programs that have been packed or encrypted usually con- tain very few strings and higher entropy compared to legitimate programs. One way to detect packed files is with the PEiD program8

Disassembling the program, i.e. translating machine code into assembly language. This reverse-engineering process loads the exe- cutable into a disassembler to discover what the program does. The most relevant software programs for disassembling PE executables are IDA Pro,9 Radare210 and Ghidra.

Dynamic Analysis:
Dynamic analysis involves executing the program and monitoring its behavior on the system. This is typically performed when static analys is has reached a dead end, either due to obfuscation or having exhausted the available static analysis techniques. Unlike static analysis, it traces the real actions executed by the program. However, the analysis must be run in a safe environment to not expose the system to unnecessary risks, where the system is both the machine running the analysis tool and the rest of the machines on the network. To this end, dedicated physical or virtual machines are set up[6].

Physical machines must be set up on air-gapped networks, that is isolated networks where machines are disconnected from the Internet or any other network, to prevent malware from spreading. The main downside of physical machines is this scenario with no Internet connec- tion, as many malicious programs depend on Internet connection for updates, command and control and other features.

The second option is to set up virtual machines to perform dynamic analysis. A virtual machine emulates a computer system and provides the functionality of a physical computer. The OS running in the virtual machine is kept isolated from the host OS and thus, malware running on a virtual machine cannot harm the host OS. VMware Workstation12 and Oracle VM VirtualBox13 are some of the virtual machine solutions available to analysts. In addition, there are several all-in-one software products based on sandbox technology that can be used to perform basic dynamic analysis. The most well-known is the Cuckoo Sandbox, an open source automated malware analysis system. This modular sandbox provides capabilities to trace API calls, analyze network traffic or per- form memory

analysis. Alternatively, there is a wide list of utilities for dynamically analyzing malware and performing advanced and specific monitoring of some functionalities. Process Monitor,15 or procmon, is a tool for Windows that monitors certain registry, file system, network, pro- cess and thread activity. Process Explorer16 shows the information about which handles and DLL processes are opened or loaded into the operat- ing system. Regshot17 is a registry compare utility that allows snapshots of registries to be taken and compared. NetCat18 is a networking utility that can be used to monitor data transmission over a network. Wireshark19 is an open source sniffer that allows packets to be captured and network traffic to be intercepted and logged. Another indispensable software utility are debuggers. A debugger is used to examine the execution of another program. They provide a dynamic view of a pro- gram as it runs. The primary debugger of choice for malware analysts is OllyDbg20 , an ×86 debugger that is free and has many plugins to extend its capabilities.[6]

The risks of using virtualization and sandboxing for malware analys is is that some malware can detect when it is running in a virtual machine or a sandbox and subsequently, they will execute differently than when in a physical machine to make the job of malware analysts harder. In addition, even if you take all possible precautions, some risk is always present when analyzing malware. From time to time, vulnerabilities have been found in the virtualization tools that allow an attacker to exploit some of its features such as the share folders feature.

Hybrid Approaches

Hybrid analysis integrates static and dynamic techniques to overcome their individual limitations and improve detection accuracy.[2]

Workflow:

Perform a quick static scan using signature-based tools (e.g., VirusTotal).

Dynamically analyze flagged files in a sandbox. Apply machine learning models trained on static and dynamic features (e.g., opcode sequences, API call patterns).

Case Study:

Detection of Dridex banking malware:

Static Clue: Irregular PE header structure. Dynamic Behavior: Keystroke logging, data exfiltration via HTTP POST requests.

Advanced Tools:

CrowdStrike Falcon

FireEye Endpoint Security

Joe Sandbox

VII. Malware Evolution

The diversity, sophistication and availability of malicious software pose enormous challenges for securing networks and computer systems from attacks. Malware is constantly evolving and forces security analysts and researchers to keep pace by improving their cyberdefenses. The proliferation of malware increased due to the use of polymorphic and metamorphic techniques used to evade detection and hide its true purpose. Polymorphic malware uses a polymorphic engine to mutate the code while keeping the original functionality intact. Packing and encryption are the two most common ways to hide code. Packers hide the real code of a program through one or more layers of compression. Then, at runtime the unpacking routines restore the original code in memory and execute it. Crypters encrypt and manipulate malware or part of its code, to make it harder for researchers to analyze the pro- gram. A crypter contains a stub used to encrypt and decrypt malicious code. Metamorphic malware rewrites its code to an equivalent when- ever it is propagated. Malware authors may use multiple transformation techniques including, but not limited to, register renaming, code per- mutation, code expansion, code shrinking and garbage code insertion. The combination of the aforementioned techniques resulted in rapidly growing malware volumes, making forensic investigations of malware cases time-consuming, costly and more difficult.

Traditional antivirus solutions that relied on signature based and heuristic/behavioral methods present some problems. A signature is a unique feature or set of features that uniquely distinguishes an executable, like a fingerprint. However, signature-based methods are unable to detect unknown malware variants. To tackle these challenges, security analysts proposed behavior-based detection, which analyzes the file's characteristics and behavior to determine if it is indeed malware, though the scanning and analysis can take some time. To over- come the prior pitfalls of traditional antivirus engines and keep pace with new attacks and variants, researchers started adopting machine learning to complement their solutions, as machine learning is well suited for processing large volumes of data.

VIII. Traditional Machine learning approaches

Over the past decade there has been an increase in the research and deployment of machine learning solutions to tackle the tasks of malware detection and classification. The success and consolidation of machine learning approaches would not have been possible without the confluence of three recent developments:[6]

The first development is the increase in labeled feeds of malware meaning that, for the first time, labeled malware is available not only to the security community but also to the research community. The size of these feeds ranges from

limited high-quality samples, like the ones provided by Microsoft (Ronen et al., 2018) for the Big Data Innovators Gathering Anti-Malware Prediction Challenge, to huge volumes of malware, such as theZoo (Yuval Nativ, 2015) or VirusShare (2011).

The second development is that computational power has increased rapidly and at the same time has become cheaper and closer to the budget of most researchers. Consequently, it allowed researchers to speed-up in the iterative training process and to fit larger and more complex models to the ever increasing data. Third, the machine learning field has evolved at an increased pace during the last decades, achieving breakthrough success in terms of accuracy and scalability on a wide range of tasks, such as computer vision, speech recognition and natural language processing.

In machine learning, a workflow is an iterative process that involves gathering available data, cleaning and preparing the data, building models, validating and deploying into production. Instead of dealing with raw malware, the data preparation process of traditional machine learning approaches involves preprocessing the executable to extract a set of features that provide an abstract view of the software. Afterwards the features are used to train a model to solve the task at hand. Because of the variety of malware functionalities, it is important not only to detect malicious software, but also to differentiate between different kinds of malware in order to provide a better understanding of their capabilities. The main difference between machine learning solutions for detection or classification of malware is the output returned by the system implemented. On the one hand, a malware detection system outputs a single value y = f(x), in the range from 0 to 1, which indi- cates the maliciousness of the executable. On the other hand, a classification system outputs the probability of a given executable belonging to each output class or family, $y \in \mathbb{R}N$, where N indicates the number of different families.[1]

IX. Proposed detection framework

To address these challenges, we propose a centralized malware detection module that operates at the enterprise gateway level.

The framework consists of:
Feature Extraction: Analyzing the PE file format to extract Import Address Table (IAT) entries.

API Call Analysis: Mapping API usage patterns indicative of malicious behaviour.

Feature Selection: Employing Information Gain (IG) to rank features based on relevance.

Classification: Using a Random Forest algorithm to build a model that distinguishes between benign and malicious files.

This system is designed to work in tandem with endpoint protection, providing an additional layer of defense.

X. Technical Approach

PE File Analysis
The Windows PE format encapsulates vital metadata, including imported API functions. Malware often exhibits specific API usage patterns, such as those related to file manipulation, process injection, or registry editing. By parsing the IAT, we generate a dataset of API call frequencies[11].

Information Gain Calculation

Information Gain is used to quantify the usefulness of each API feature in distinguishing malware from benign software. The entropy-based formula is:

IG(X|Y)=H(X)−H(X|Y)IG(X | Y) = H(X) - H(X | Y)IG(X|Y)=H(X)−H(X|Y)

Where XXX represents the presence of an API function and YYY is the class label (malicious or benign).
Features with high IG are retained for model training.

Random Forest Classifier

Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions. This technique improves generalization and reduces overfitting, making it suitable for
high-dimensional datasets like ours.[15]

5. Experimental Setup and Results
We collected a dataset of over 5000 executables (benign and malicious) and extracted API calls from their PE headers. After preprocessing and IG-based feature selection, we trained multiple classifiers.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
Decision Tree	90%	0.90	0.10
Naive Bayes	95%	0.95	0.05
Random Forest	97%	0.97	0.03
Proposed Model	**99.6%**	**0.996**	**0.003**

These results validate the robustness of our approach in accurately classifying previously unseen samples.

XI. Conclusion

This study presents a machine learning-based malware detection framework tailored for centralized deployment in

enterprise networks. By leveraging PE file analysis and API call profiling, the system effectively identifies malware with high accuracy. Though resource-intensive, the centralized architecture is ideal for organizational gateways, offering proactive defense against sophisticated threats. Future work will involve optimizing performance and expanding detection to other file types and behavioral vectors.

XII. References

[1] 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," Information Security Technical Report, vol. 14, no. 1, pp. 16–29, 2009.

[2] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in 5th Conference on Information and Knowledge Technology (IKT), 2013, pp. 113–120.

[3] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," Human-centric Computing and Information Sciences, vol. 8, no. 1, pp. 1–22, 2018.

[4] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," Computers & Security, vol. 81, pp. 123–147, 2019.

[5] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," ACM Computing Surveys, vol. 50, no. 3, pp. 1–40, 2017.

[6] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of 'malware': Bibliometric analysis of malware study," Journal of Network and Computer Applications, vol. 75, pp. 58–76, 2016.

[7] J. O. Kephart and W. C. Arnold, "Automatic extraction of computer virus signatures," in 4th Virus Bulletin International Conference, 1994, pp. 178–184.

[8] R. Ronen et al., "Microsoft malware classification challenge," arXiv preprint arXiv:1802.10135, 2018.

[9] Y. Nativ, "TheZoo: A repository of live malware samples," 2015. [Online]. Available:
[10] https://github.com/ytisf/theZoo

[11] VirusShare, "VirusShare.com," 2011. [Online]. Available: https://virusshare.com/

[12] Microsoft, "PE Format," Microsoft Documentation,
[13] 2021. [Online]. Available:

[14] https://docs.microsoft.com/en-us/windows/win32/debug /pe-format

[15] AV-TEST, "Antivirus software report," 2010. [Online]. Available: https://www.av-test.org/

[16] I. Santos et al., "Opcode sequences as representation of executables for data-mining-based unknown malware detection," Information Sciences, vol. 231, pp. 64–82, 2013.

[17] T. M. Mitchell, Machine Learning. McGraw-Hill, 1997.

[18] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[19] M. Schultz et al., "Data mining methods for detection of new malicious executables," in IEEE Symposium on Security and Privacy, 2001, pp. 38–49.

[20] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," Journal of Machine Learning Research, vol. 7, pp. 2721–2744, 2006.

[21] P. Szor, The Art of Computer Virus Research and Defense. Addison-Wesley Professional, 2005.

12. D. Harley et al., Viruses Revealed. McGraw-Hill, 2001.