

Malware strategies and issues in examination

Mathe Ramakrishna *

Department of Computer Science and Engineering

Career Point University

Kota

Rajasthan

India

Aravapalli Rama Satish [†]

School of Computer Science and Engineering

VIT-AP University

Amaravati 522237

Andhra Pradesh

India

Gudela Ashok [§]

Department of Mathematics

Adikavi Nannaya University

Rajamahendravaram

Andhra Pradesh

India

Abstract

The field of information technology is experiencing rapid advancements, presenting significant challenges in security and privacy for cybersecurity professionals. Research indicates a concerning increase in the development of new malware strains. A perpetual cycle exists between the evolution of malicious software and the efforts to safeguard against them. Antivirus companies continually update their malware signatures, while attackers persistently seek ways to evade detection. Machine learning offers promising capabilities for detecting malware, with feature extraction as the initial step and classification as the subsequent phase. The effectiveness of classification algorithms is influenced by the quality

* E-mail: mathe.ramakrishna@gmail.com (Corresponding Author)

[†] E-mail: rama.satish@vitap.ac.in

[§] E-mail: ashokavf@gmail.com

of the extracted features. This articles aim is to assist aspiring researchers in selecting suitable algorithms for their investigations.

Subject Classification: Primary 68T15, 68T05, 68T3 Secondary 11T71.

Keywords: Malware, Machine learning, Security and privacy, Cybersecurity, Encryption.

1. Introduction

Malware, initially a tool for playful pranks, has evolved into a significant threat. Its proliferation has surged dramatically over the years. For Instance, Panda Antivirus detected 84 million new malware variants in 2014. Similarly, McAfee reported a 420% increase in new MacOS malware in just the third quarter of 2020. Once crafted for entertainment, computer viruses are now exploited for nefarious activities such as financial theft and unauthorized data access [1].

In an illustrative experiment, the San Diego Supercomputer Centre (SDSC) set up a machine with Red Hat Linux 5.2 and an internet connection in December 1999, intentionally omitting security updates. Alarminglly, the system faced its first attack within just 8 hours post-installation, with subsequent assaults occurring 20 times within 21 days, culminating in a full-scale hack within 40 days.

While antivirus companies primarily use signature-based detection for known malware, this method struggles with zero-day threats—newly emergent and uncatalogued malware. Malware developers use encryption and obfuscation to evade detection. With the rise in sophisticated attacks, proactive defence and vigilance are crucial. The article first explores distinguishing characteristics of malware, then delves into stages of machine learning and deep learning techniques, discussing their pros and cons. It aims to help aspiring researchers choose suitable algorithms.

2. Literature Review

In their 2012 study, Egele et al. examined techniques and tools for malware detection. They outlined malware characteristics, types, and infection vectors. Various analysis methods were discussed, including function parameter analysis, function call monitoring, information flow tracking, and instruction tracing. They profiled programs like Anubis, CWSandbox Norman Sandbox, Joe Box, and WiLD CAT for executing malware samples. Dynamic analysis tools assess system calls and APIs

beforehand to gauge data sensitivity, aiding in potential malware detection [2,3].

In 2013, Bazrafshan et al. delineated three primary techniques for malware detection: behaviour-based, signature-based, and heuristic-based methods [3-5]. The authors emphasized heuristic detection techniques utilizing elements like opcodes, API calls, and N-grams, while also discussing strategies employed to conceal malware infections.

Gandotra et al. conducted a comprehensive review of machine learning applications in malware analysis in 2014 [6,7]. Their classification categorized research into static, dynamic, and hybrid analysis methodologies.

LeDoux and Lakhota (2015) focused on the relationship between malware and artificial intelligence (AI). They discussed the difficulties in malware analysis, described a pipeline method to malware detection, and looked at supervised, unsupervised, semi-supervised, and ensemble learning as AI concepts. The researchers highlighted the problems with malware analysis and suggested AI-driven fixes to address these difficulties.

In 2016, Basu et al. [4] spearheaded research focusing on data mining techniques for identifying malware. The study offered insights into the methodologies adopted by various developers, their data sources for viral testing, and the specific techniques employed. Notably, the researchers differentiated works by different authors based on components such as API calls, byte sequences, and PE headers. They introduced a systematic approach for malware identification, utilizing the Xpstrus tool for API call recognition and employing Naive Bayes for training and validation.

In a related study in 2017, Ye et al. [5] explored data mining methodologies encompassing static and dynamic features, along with textual descriptions. The researchers segmented the malware identification process into two distinct stages [9-11]: feature extraction and subsequent clustering. Their findings emphasized the potential of data mining techniques to achieve malware detection with a minimal false positive rate.

In 2017, UCCI et al. [6] proposed innovative strategies for malware detection, categorizing malware features based on their specific types. The researchers identified limitations in existing datasets, highlighting the lack of diversity in the data samples. To address this issue, they advocated for the development of benchmark datasets that encompass a broader range of malware variants, categorizing them based on their functionality, complexity, and evolution over time. Additionally, UCCI et al. introduced

the concept of economic considerations in malware analysis, emphasizing the need for a balanced approach that considers both performance metrics and practical costs.

In 2019, Berman et al. [7] introduced two dynamic evaluation methods: the first method focuses on distinguishing defined behaviours, while the second method observes real-time behaviour. The researchers also conducted a comparative analysis of signature-based and behaviour-based malware detection techniques, highlighting their respective advantages and limitations.

Similarly, Gibert et al. [8] explored machine learning techniques in 2019, including both deep learning algorithms and shallow methods like Artificial Neural Networks (ANN). They presented a comprehensive framework encompassing various deep learning architectures such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), among others, and illustrated evaluation methodologies using multiple metrics.

While numerous studies have showcased the application of AI techniques for malware analysis, this article will concentrate on the specific methodologies employed by experts in the field of AI-driven systems, emphasizing the inherent advantages and challenges associated with these approaches.

3. NTYPES of Malware

Malware, often known as malicious software, is a class of software that enters a user's computer without authorization but has the potential to cause harm or steal important information.

3.1 Infection

Malicious entities infiltrate systems, propagate, and cause harm by integrating with executable files. Creeper, developed by Bob Thomas at BBN in 1971, was an early example. Though not damaging data, it was a significant milestone in malware history. Later, in 1982, Rich Skrenta created "Elk Cloner," spread via floppy disks and embedded in the Apple II operating system, further advancing malware tactics.

3.2 Worms

Worms are a form of digital infection capable of spreading autonomously across devices on the internet, without human interaction.

The first practical computer worm emerged in 1978. A notable example is the destructive Mydoom worm, utilized by attackers to inflict damage in January 2004.

3.3 Trojan

A Trojan is malicious software disguised as legitimate, aimed at compromising systems. Similar to a Trojan horse, it employs deception for security breaches. An instance occurred in August 2006, when Nordea bank clients received spam emails deploying the Haxdoor Trojan, resulting in a theft of over \$1 million by keylogging and transmitting sensitive data to the attackers' server [13-15].

I Optional Section

This strategy subtly bypasses security measures, often seen in software development where optional components for testing may introduce vulnerabilities. For example, using default passwords during initial setups can create exploitable weaknesses. A notable incident occurred in February 2005 when a Florida investor lost \$90,000 due to unauthorized access through such an optional component, compromising his computer system.

3.5 Rootkits

Rootkits grant attackers elevated privileges, surpassing standard user permissions to access sensitive data and system functions clandestinely. Their insidious nature conceals their presence within host PCs and software applications, evading detection while maintaining unauthorized access and persistent infiltration.

3.6 Keylogger

A Keylogger can be deployed in various perverse activities, often in conjunction with other malicious software to record user keystrokes and sometimes screenshots, to steal sensitive information like banking details. In February 2006, Brazilian police caught 55 people involved in a scheme that stole \$4.7 million from victims in six countries using such tactics. To combat these attacks, banks have started using virtual secure input methods [16].

3.7 *Spyware*

Spyware, considered potentially unwanted software, collects user data without consent, including passwords and account details. It gained attention around 2000 but was conceptualized in 1995. Adware, a subset of spyware, seeks user permission to display notifications, messages and pop-ups rather than causing direct harm to the system.

3.8 *Remote Access Trojan (RAT)*

It is malware that allows an attacker to control a victim's computer remotely and undetected. It can disguise itself as legitimate software, enabling the attacker to execute commands, manipulate files, and surveil activities without the victim's consent.

3.9 *Ransomware*

It is a kind of malware that encrypts victims' files, demanding a ransom for decryption. It makes data inaccessible, holding it hostage until payment, often in crypto currency, is made for the decryption key. These attacks can lead to substantial financial losses, disrupt business, and compromise sensitive information.

4. Feature of Malware

Malware attributes fall into static and dynamic categories. Static features are identifiable without execution, while dynamic ones manifest post-execution, requiring dynamic analysis. Advanced AI techniques aid in comprehensive malware analysis. This research outlines data acquisition methodologies and key attributes for detection, as detailed herein.

N-gram: A gram is a sequential set of n bytes obtained from a hexadecimal dump of a file. The concept of **Uni-gram** (1-gram) was introduced by Lin et al. [13], Ahmadi et al. [14], Anderson et al. [15], and Sexton et al. Exploration extended to **Bi-gram** with contributions from Kolter [16] and associates. Ahmadi, Kolter, and Khodamoradi led research into **4-gram** and beyond. **N-gram** usage adheres to a non-redundancy principle where a non-repeating byte is unique within an instance but may recur in subsequent grams.

Opcodes: Programmable executables (PE) perform specific machine-level operations using opcodes encapsulated within executable code. Sexton et al. [16] investigated branch rules, counts, and memory direction counts. Belaoued et al. explored mathematical operations, stack

manipulations, and memory directives. Ahmadi et al. analyzed directional aspects using data, while Darabian et al. utilized opcodes in malware research.

Strings: Schultz et al. defined strings as consecutive printed characters and noted that plain text within PE-file headers could segregate data. They also observed encoded strings in non-PE executables, which can be compared with their datasets. Ye et al. highlighted the usefulness of “legible” strings in different applications. For virus detection, Ito & Mimura combined ASCII texts with natural language processing techniques.

PE File: Microsoft introduced the PE file structure for Win32 platforms, containing crucial information. A static examination of the PE header reveals key data that includes details like size, security measures, and compiler information, often aiding in malware identification. Researchers have analysed various PE attributes, such as resource image checksums, section signatures, PE header checksums, section names, counts and sizes, and import table details. They’ve also explored trends in malware families using PE file attributes over time.

Memory Access: Understanding memory access nuances will help you better understand malware. Memory-based features were utilized by Egele et al. to log system calls, import library functions, heap and dynamic stack operations, and register return values. Other scholars, such as Kong et al., investigated I/O read-write jobs and memory read-make activities with an emphasis on various API and opcode components.

PC processor Register: Kong et al. investigated register modifications and operations such as examination and count creation. Egele et al. analyzed return values stored in the %rax register of instruction sets, while Ahmadi et al. focused on repeated utilization of specific registers and associated values. These studies demonstrate the broad applications and methodologies concerning computer registers in research contexts.

Raised Exceptions: malicious researchers frequently employ the tactic of raising an unusual instance in a program and throwing malicious code into an exception. Therefore, it makes more sense to investigate unusual occurrences in order to understand virus avoidance techniques.

Network: PE’s interactions with networks reveal crucial insights into malware behaviour. For instance, keyloggers may extract and transmit sensitive data, while botnets relay commands over the internet. Some malware initiates fake connections to increase network traffic. Studying these communications provides valuable data. Vadrevu et al. (2013) emphasized metrics like affiliation and unique IP counts, and HTTP

response types, while M. Belaoued et al. focused on monitoring network traffic, domain names, and IP addresses for remote access.

Sandbox Submissions: Sandboxes execute code in a restricted environment to detect malicious activity. They're widely available online. Malware developers exploit sandboxes to test their code and evade antivirus detection. Targeting sandbox vulnerabilities yields valuable malware insights.

Application Programming Interface (API) Calls: APIs act as intermediaries between software and operating systems, facilitating functions like file operations. Researchers analyse API calls such as sequences of API calls, aid in identifying tactics like library traversal or self-replication. Jinrong Bai et al. used inferred Data Definition Language (DDL) commands in their analysis, while Bojan Kolosnjaji et al. focused on Kernel API, employing neural networks. Some studies delve into system call patterns to understand software behaviours further.

5. Machine Learning

Malware, including keyloggers, can capture and transmit user data to malicious actors, who may control the malware. To combat evolving threats, cybersecurity relies on advanced techniques like Machine Learning (ML), coined by Arthur Samuel in 1959. ML enables computers to mimic human-like decision-making by analysing vast data during training. As predictive models refine with more data, accuracy improves. AI aids in identifying malware types like contaminants, worms, and rootkits, allowing for effective labelling and categorization.

Naive Bayes Classification: Naive Bayes classifiers rely on Bayes' theorem and assume feature independence within a set, making them 'naive'. This assumption means that the presence or absence of one feature is unrelated to others when considering a class. Despite this simplicity and potential feature interdependencies, they compute probabilities using formula.

$$P(C | X) = \frac{P(X|C)P(C)}{P(X)}$$

k-Nearest Neighbour (k-NN): The essence of this calculation lies in identifying traits shared by objects in the same class using a distance measure among their k nearest neighbours. It determines distances using methods like Euclidean, Manhattan, Minkowski, or Hamming before

assigning a class. While k-NN is easy to use, it becomes slow when there are incalculable features.

Support Vector Machine (SVM): SVM seeks the optimal hyperplane to effectively separate two classes. Linear SVM uses a straight hyperplane but may struggle with non-linear datasets. Polynomial SVM, with a polynomial kernel, captures non-linear class relationships for better separation in such scenarios.

Decision Tree (DT): Decision Trees are interpretable, popular for data analysis and modelling. They organize data hierarchically, with nodes representing features and leaves indicating class labels or values. Splitting the dataset by features maximizes homogeneity within subsets. At each decision point, the algorithm selects the feature that best separates data, continuing until meeting stopping criteria.

Random Forest (RF): The Random Forest, an advanced Decision Tree variant, has gained popularity for classification tasks. It comprises multiple trees that collaborate, with randomness introduced through techniques like bootstrap sampling and feature selection. Each tree contributes to predictions, and the final prediction is the class receiving the most “votes” across all trees. Utilizing diverse trees with random variations often leads to more accurate predictions than using a single tree.

Gradient Boosting: Gradient Boosting and Random Forest both improve the basic Decision Tree classifier’s performance. However, they differ in approach. Random Forest combines multiple trees for predictions, while Gradient Boosting sequentially trains new trees to correct errors of previous ones. It uses regression to refine predictions by minimizing residual errors at each step, leading to improved performance.

XG Boost: It is an optimized version of Gradient Boosting developed by Tianqi Chen. Each decision tree learns from residuals, refining the model for accuracy. Subsequent trees address earlier shortcomings, and XGBoost combines multiple learners’ predictive power. Aggregating contributions from all trees, it provides a robust model for accurate predictions.

6. Conclusion

Malware evolves to evade antivirus signatures, demanding skilled professionals to counter threats. Available tools complicate detection efforts by enabling code creation and obfuscation. Agile detection methods are crucial amid an expanding malware landscape. Malware analysis involves preparation/clustering and pivotal feature extraction. Dynamic

features manifest post-execution, while static features are discerned pre-execution. This study delves into malware characteristics, leveraging machine learning and deep learning. It explores n-grams, opcodes, API calls, and network behaviours. Naive Bayes, KNN, SVM, Decision Trees, and Random Forests are introduced for malware representation, highlighting their pros and cons.

References

- [1] 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, pp. 113–120 (2013).
- [2] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *J. Inf. Secur.*, vol. 2014 (2014).
- [3] C. LeDoux and A. Lakhota, "Malware and machine learning," in *Intelligent Methods for Cyber Warfare*, Springer, pp. 1–42 (2015).
- [4] 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).
- [5] 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.
- [6] D. Ucci, L. Aniello, and R. Baldoni, "Survey on the usage of machine learning techniques for malware analysis," (2017). ArXivPrepr. ArXiv171008189.
- [7] D. S. Berman, A. L. Buczak, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122 (2019).
- [8] 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).
- [9] "Worm definition by The Linux Information Project." <http://www.linfo.org/worm.html> (accessed Mar. 20, 2020).
- [10] Altameem, Ayman, et al. "P-ROCK: a sustainable clustering algorithm for large categorical datasets." *Intell. Autom. Soft Comput* 35.1 : 553-566 (2023).

- [11] J. Allain, *The Ugly Truth About Adware and Spyware*. Lulu Press, Inc (2015).
- [12] Gudela Ashok, S. Ashok Kumar, D. Chaya Kumari & Mathe Ramakrishna. A type of public cryptosystem using polynomials and pell sequences, *Journal of Discrete Mathematical Sciences and Cryptography*, 25:7, 1951-1963 (2022), DOI: 10.1080/09720529.2022.2133237.
- [13] 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).
- [14] 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*, pp. 183–194 (2022).
- [15] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification: bridging the static/dynamic gap," p. 12 (2012).
- [16] 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).
- [17] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangariah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Gener. Comput. Syst.*, vol. 90, pp. 211–221 (2019).
- [18] Singh, Abhay Pratap & Singh, Mahendra. Real time malware detection in encrypted network traffic using machine learning with time based features, *Journal of Discrete Mathematical Sciences and Cryptography*, 26:3, 841-850 (2023), DOI: 10.47974/JDMSC-1760.
- [19] Sharma, Vijay Prakash, Yadav, Narendra Singh, Adavi, Sundar Suhrith , Reddy, D. Sikha Datta & Gupta, Brij B. A two stage hybrid intrusion detection using genetic algorithm in IoT networks, *Journal of Discrete Mathematical Sciences and Cryptography*, 26:3, 667-676 (2023), DOI: 10.47974/JDMSC-1737.