International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I1P132

Volume 73 Issue 1, 371-384, January 2025

Original Paper

Malware Analysis and Detection using ML tools: Current State and Challenges

Gulshan[1], Neetu Sharma[2]

[1,2] *School of Computer Science and Engineering, Galgotias University, Greater Noida, India.*

[1]*Corresponding Author : gulshanjat@gmail.com*

Received: 25 September 2024 Revised: 13 January 2025 Accepted: 16 January 2025 Published: 31 January 2025

Abstract - *In the era of digitalization, a major issue that must be addressed is cyber security. The use of technologies and advancements has endangered the user's information and data. Here, the main focus is on malware that should be detected in the early stages. Malware detection identifies and mitigates malicious software threats to computer systems and networks. With the increase in cyber-attacks, malware detection has become critical for individuals and organizations to safeguard their digital assets and sensitive information. In this paper, here discussion of the current state of malware detection, including challenges and advancements in the field. It also covers the most commonly used malware detection techniques, such as 'signature-based detection', 'behaviour-based detection', and 'machine learning-based detection'. At last, it quantifies the ml-based method for detection in various parameters.*

Keywords - *Malware Detection, Cyber Security, Machine Learning, Cyber-Attacks, Ransomware.*

1. Introduction

Malware, often referred to as "malicious software" is specifically generated to harm, damage, or disruption to computing systems or devices. Malware can be found in many different forms, including worms, trojans, ransomware, adware, and spyware. A prevalent method employed by attackers to infect computers with malware is through phishing scams. In these scams, fraudulent emails, websites, or social media messages are used to deceive individuals into downloading and installing malware onto their devices. Once installed, malware can steal sensitive data, monitor user activity, damage files, and even control the entire system.

The term "threat" encompasses all factors that contribute to the vulnerability of cybersecurity. It encompasses the dangers that arise when these vulnerabilities are exploited, allowing attackers to carry out a sequence of actions known as a penetration strategy. It encompasses various types, each with unique characteristics and effects. In Figure 1 depicted the classification of malware [5].

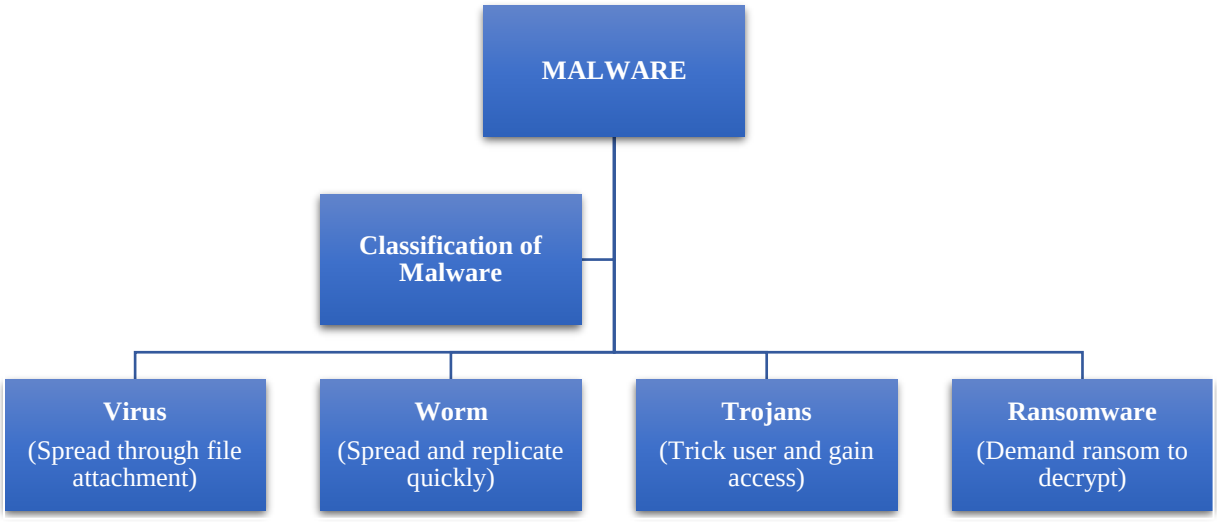


Fig. 1 Classification of malware

- *Virus* is a malware that infects a computer by connecting to a legitimate program and replicating itself. As a consequence, malicious programs can influence files, software, and hardware, presenting a very convenient pathway for further transmission to additional systems through networks and email attachments.
- *Ransomware* is a dangerous software program that prohibits access to files on a targeted machine and advises payment for a key that will unscramble the encrypted components. The possibility exists for important data loss and financial integrities for both individuals and establishments.
- *Worms* are famously known for using untruthful tactics, as they are programs that can replicate themselves and benefit from operating system flaws, enabling them to take advantage of any vulnerabilities in apps or systems. This malware causes an increase in network activity, a decline in operational efficiency, and security problems that violate confidentiality, resulting in data theft.
- *Trojans*, Malware categorized as Trojan horse - or 'Trojans' - scams users into viewing it as genuine software in an effort to unlawfully enter target computers. After it is installed, it has the capability for unauthorized access to the system or the theft of confidential information.
- *Adware* is a type of malicious program that shows a barrage of unwanted advertisements and pop-up windows either in a user's browser or on their actual computer. Among the problems with this intrusive software is the impact it has on system performance, added to its capability to violate personal privacy by accumulating personal information.
- *Spyware* conceitedly spies on a victim's computer activity, including keystrokes, internet browsing, and use of emails, software, and hardware can be compromised by malware, which can also spread to other computers via networks and email attachments.
- *Rootkit* a rootkit is malware that conceals itself from detection by security software and can grant an attacker complete control over the victim's computer system.

The ever-evolving cyber threat landscape is comprised of a multitude of attackers, all driven by their own nefarious objectives. These threat actors utilize a wide array of attack vectors and methodologies, with the sole purpose of causing harm. Through unauthorized access, data theft, service denial, fraud, alteration, extortion, and countless other means, their actions can have devastating consequences. In fact, the projected cost of cyber security attacks against businesses within the next five years is estimated to range from a staggering $5.2 trillion to $6 trillion in 2020 alone.

1.1. Organization of Paper

Here it begins with an Introduction that outlines the importance of cyber space and malware detection in cybersecurity, different types that exists, the evolution of malware threats, and the purpose of the study. Follow this with a Background section that explains key concepts and terms in malware detection, setting the stage for the detailed analysis. The core of the paper categorizes and discuss various malware detection techniques, such as signature-based, anomaly-based, and heuristic methods, highlighting their strengths, weaknesses, and recent advancements.

After the literature review, include a Comparison and Analysis section that contrasts these techniques, identifies trends, and discusses the effectiveness of different approaches in real-world scenarios. The Challenges explore current limitations in malware detection. Conclude with a Summary encapsulates the key findings of the paper and reinforces the significance of continued innovation in malware detection methods with future directions suggest areas for future research and improvement. Finally, include a References section, listing all the sources cited in this paper.

2. Literature Review

In this section, it summarizes all the literature in our study and highlight their main contributions with key findings and in other aspects as illustrated in Table 1. Malware analysis and detection are crucial components in understanding and combating malicious software threats. Numerous studies have explored various facets of malware analysis and detection to bolster cybersecurity defenses. (Nkongolo, 2023) examined cyclostationary malware detection through feature selection and classification, emphasizing the importance of recognizing periodically shifting malware behaviours using cyclostationarity, with internet protocol serving as a notable cyclostationary feature pattern employed by malware.

(Singh et. al., 2023) stressed the necessity for a unified platform to share and verify malware analysis findings, facilitating the replication and validation of research outcomes. The lack of such a platform compels malware researchers to develop context-specific datasets and detection mechanisms, which can be complex and time-intensive.

(Molina-Coronado et. al., 2023) explored the effects of specific obfuscation techniques on common features extracted via static analysis for Android malware detection. The purpose of the investigation was to find out whether these variations significantly affect the efficacy of ML detectors that explore static analysis features. (Pratomo et al., 2023) identified the need for dynamic malware analysis to estimate the functionality of malicious software and generate useful strategies for its detection and defence. The abilities required to understand the actions of malware at runtime may be acquired through dynamic analysis, which also helps to thwart the techniques malware uses to avoid static analysis.

Table 1. Contribution in malware detection in recent years

Sr. no	Authors	Problem area	Key findings	Dataset/Method used	References
1	Daniel et al., 2020	Malware detection using machine learning	• Provided complete review with description and features using machine learning. • Highlighted the limitations and challenges.	67 research papers	[1]
2	Valerian Rey et al., 2022	Malware Detection in IoT Devices	• Modelled a privacy preserving approach for data in IoT devices despite using federated learning.	N-BaIoT dataset	[10]
3	ABDELOUAHAB AMIRA et al. 2023	malware analysis using community detection algorithms	• Provided most recent survey on analysing malware using community detection. • Highlighted the possible change or improvement can be done	55 literatures	[2]
4	Akshat Gaurav et al., 2022	malware detection in IoT-Based systems	• Compared the recent literature of multiple detection techniques.	161 literatures	[3]
5	Singh et. al., 2023	malware detection by offering a customizable feature generation process and a centralized platform for malware analysis data.	• Achieved a high accuracy rate of 98.8% and an AUC of 0.97 in a real-world scenario using a decision tree algorithm for ransomware detection based on PE entropy. • Centralized repository for malware analysis data.	3000 ransomware and benign samples	[13]
6	Gaber et. al., 2023	detection of sophisticated and evasive malware using AI	• Identifies gaps in the literature, particularly the need for robust AI models that can generalize well across different datasets and environments • Highlights the challenges posed by sophisticated and evasive malware, which often uses anti-analysis techniques to threat detection tools	57 papers	[14]
7	Fehmi Jaafar et. al., 2016	Ransomware detection	• Proposes a privacy-preserving method using Federated RNN for ransomware detection	RNN dataset	[18]

(Yan et al., 2023) put forth a method utilizing GPT-4 for prompt engineering-assisted dynamic malware analysis, with the objective of resolving challenges associated with the API call concept drift encountered in malware analysis, seeking to boost the performance of dynamic analysis. [16]

(Gaber et al., 2023) executed a systematic analysis of literature concerning the application of AI for malware detection, investigating original research and difficulties in this arena. The focal point of this review was the current best practices for building accurate and strong AI-enabled malware detection systems covering diversity of aspects, such as malware sophistication, analysis methods, feature extraction, and comparison between machine learning and deep learning. (As illustrated by Liu et al., 2024), an analysis of machine learning tools was conducted for revealing concealed malicious patterns in Android applications, and it was indicated that a stronger emphasis on ML-driven solutions is essential. The research presented a thorough assessment of Android malware detection using ML with empirical and exacting information.

A cost-effective malware detection system through memory dump analysis was proposed by (Hasan et al., 2024) using various machine-learning algorithms, aiming to increase cybersecurity effectiveness by evaluating the performance of machine learning algorithms in obfuscated malware threat detection. (Ponte et. al., 2024) presented SLIFER, a novel Windows malware detection pipeline that integrates static and dynamic analysis methods. The study identified gaps in existing malware detection pipelines and developed a streamlined strategy that deploys both static and dynamic analysis effectively.

(Quertier et. al., 2024) designed a Transformer model that is lean for the purpose of dynamic malware analysis and

detection, concentrating on behavior-based techniques. Utilizing the architecture of Transformers, an Encoder-Only model was designed to examine malicious files for their API call sequences. It is clear, when taking a close look at these studies, that important progress has been made in the areas of techniques and resources for interpreting and classifying malware.

(Daniel et al., 2020) reviewed 67 research articles and provided a detailed description of three approaches: static, dynamic, and hybrid, with their operations based on machine learning. They systematically arranged the available literature and performed a comparative analysis of different malware detection approaches. In order to detect malware, they provide classifiers that rely on many feature types or data modalities and introduce new research paths. This comparative analysis allowed them to identify the strengths and limitations of each approach, and to recommend the most effective methods for detecting malware in various contexts [2].

(Rey et al., 2022) proposed a privacy preserving method for IoT malware detection in federated learning. They train and test both supervised and unsupervised model on IoT devices with compromising privacy. Also, the adversarial effect is managed using a robust method that prevent one malicious client to endanger the whole federation. At final stage the performance in term of accuracy in detection is calculated in reference of non-privacy preserving model and shown their efficiency [10].

(Amira et al., 2023) presents a survey on malware analysis using community detection algorithms. It reviews state-of-the-art solutions based on five facets: ‘Analysis task’, ‘Community detection approach’, ‘Target platform’, ‘Analysis type’, and ‘Source of features’. The findings suggest room for improvement in the field.

The advantages and limitations of the solutions are discussed, along with open issues and future research directions. The paper highlights the importance of leveraging graph theory techniques to achieve bulk detection of malware families and variants, reducing detection time significantly. (Gaurav et al., 2022) provides a comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information systems.

It discusses various attack mitigation strategies, focusing on the use of machine learning for detecting malware attacks in IoT-based systems due to its accuracy and adaptability. The survey covers static, dynamic, adversarial, and hybrid malware detection techniques. The paper highlights the importance of early malware detection in IoT devices to prevent potential damage. Various machine learning algorithms and techniques are reviewed for their effectiveness in detecting malware in IoT-based enterprise information systems. [2]

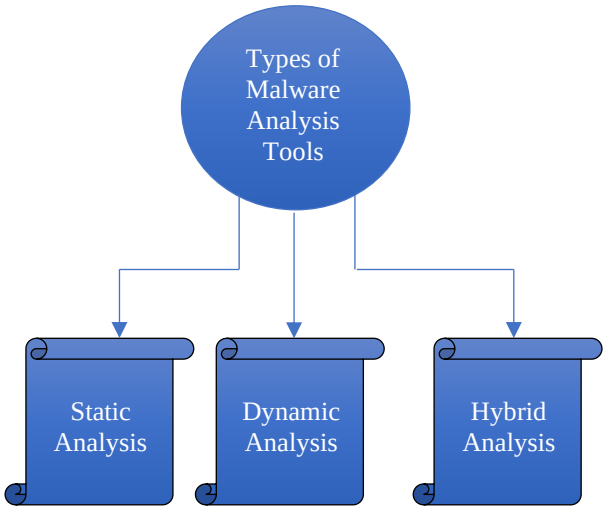


Fig. 2 Types of analysis tools

3. Background

3.1. Malware Analysis

The process of examining and reviewing the effect of malicious software is called malware analysis. This process is conducted in a controlled environment where the chance of harm is minimal or controlled. The information or data is extracted from malware using different monitoring tools and extraction techniques. The analyses can be performed in both run and rest modes. Thus, there are three methods of analysis as depicted in Figure 2.

3.1.1. Static Analysis

Here, the malware program is not in a running state or, in other words, not triggered. Conversely, static analysis examines malware code and structure without execution.

The primary steps and methods in static malware analysis include:

- Sample Acquisition: Secure a malware sample from a reliable source or controlled setting, preventing accidental system infection.
- File Format Determination: Ascertain the sample's file format, which may be an executable (EXE), dynamic link library (DLL), script (VBScript, PowerShell), or other file types.
- Disassembly/Decompilation: Transform the malware's binary code into a more comprehensible format. Changing machine code into assembly is referred to as disassembling, while the activity of processing the coding languages found in C and Java is called decompilation.
- Code Examination: When you analyse it closely, look into the code to learn what it does and consider any terrible jobs it might perform. Analysts are trying to find signals and set guidelines that would reveal harmful behaviour.
- String and Resource Extraction: Claiming authority over harmful code’s strings and resources can help us gather

information about its expected use, the communication mechanisms it adopts, and any encoded signatures significant for detection.

- Function and API Call Analysis: The operations and APIs of malware inform us that it can change files, communicate via networks, and take control of the registry.
- Packer and Obfuscation Identification: The goal of malware creators is usually kept hidden through obfuscation and packers, a method they frequently use. Studying these techniques allows one to find out the essential motive of the original code.
- Behavioural Rule Creation: Design a collection of behaviour patterns that can aid in detecting imminent malware that is associated, referring to the results.
- Code Reverse Engineering: The need exists to investigate reverse engineering in order to study complicated algorithms and security strategies commonly found in malware.
- Sandbox Analysis (Optional): Within some static analysis programs exists a form of limited sandboxing that replicates code snippet execution without having to run full malware, thereby introducing a dynamic aspect to static analysis.

Enhancing malware analysis performance commonly requires ongoing application of both static and dynamic analysis, along with other methods such as memory analysis and network traffic analysis, which is common in everyday cyber security operations.

3.1.2. Dynamic Analysis

The methodology is frequently practiced within cybersecurity to exhaustively investigate malicious software (malware) in an environment that is both monitored and safeguarded securely. This method diverges from the standard of static analysis, where malware's code and structure are judged without being put to use. A review of dynamic analysis leads to more detailed understanding of malware behavior, which fosters more robust methods for locating and moderating security threats.

The primary steps and techniques involved in dynamic malware analysis are as follows:

- Sample Isolation: In order to shield additional systems or networks from a malware intrusion, it is run in an isolated space called a sandbox or virtual machine.
- Activity Monitoring: There is continuous monitoring of malware activity during the time it is active. These procedures a variety of inquiries to the system, events in the file system, registry modifications, functions across the network, and changes to vital system components.
- Network Traffic Analysis: Through dynamic analysis, analysts can both log and study network traffic created by harmful software. Methods of exfiltration, probability of command-and-control servers, and the ways they communicate are subjects you can learn about with the proper assistance.
- Memory Analysis: Over its lifespan, malware has been subject to memory analysis to spot suspected damaging code invasions or process changes in systems.
- API Hooking: In specified circumstances, the use of API hooking techniques to directly monitor and follow up on the API calls of malware forms part of dynamic analysis. This understanding of malware behaviours is provided without actually modifying the malware's legitimate code.
- System Monitoring: What are given is an understanding that the analytical environment provides techniques that facilitate the detection of any system alterations, including the inclusion of novel threads, processes, or irregular shifts in attributes.
- Runtime Debugging/Dynamic Code Analysis: A special method permits specialists to investigate the orders of malware, look into its memory, and comprehend its implementation strategies.
- Triggering Payloads: Certain malware forms will only demonstrate specific actions when particular definitive triggers or conditions are present. To see a variety of situations, analysts may attempt to activate these payloads.

3.1.3. Hybrid Malware Analysis

Hybrid malware analysis is a methodology that combes static and dynamic analysis methods in order to develop a more complete understanding of harmful software. The purpose of this method is to optimize the advantages of these two strategies, intending to amend the inconsistencies inherent in each, while providing a more detailed analysis of the workings and strengths of malware.

The typical process of hybrid malware analysis involves several steps:

Static Analysis

This initial component includes a look at the criminal code and construction without actually putting it into effect. At this junction, signatures of known malware are graspable, texts and images are obtainable, and all obfuscation or packing executed by the harmful software are identifiable.

Behavioural Analysis (Dynamic Analysis)

In reproach to static analysis, is the launch of malware inside a controlled context, like a sandbox or virtual machine, fostering surveillance. Over this time, there undergoes a detailed study of the ways malware and the system engage, putting emphasis on API calls, file access, and network activities, which will be vital in the future for understanding the real-time actions of the malware and developing suitable defensive techniques.

Table 2. Benefit comparison of static and dynamic malware analysis

Sr. no	Static Analysis	Dynamic Analysis
1	Secure and precisely targeted investigation of malware free from the threat of infection.	It seems static at first, yet a subsequent, more structured environment is needed for analysis.
2	This can transpire without the reliance on the internet, so as to ensure a broader and more detailed analysis	Whereas static analysis might overlook the capabilities of the malware, the offline analysis of dynamic analysis may not be accessible.
3	Aids in the differential identification of already examined samples from malware that is known.	promotes the development of strategies for detection and mitigation through revealing the relationships between the malware and the system and network.
4	Main component of assessing the threat of malware and what its likely effects will be.	The ability to observe the real-life behaviour of the malware, including any approaches to hide from observation or to take evasive action.

Table 3. Limitation comparisons of static, dynamic, and hybrid malware analyses

Topic	Static Analysis	Dynamic analysis	Hybrid analysis
Runtime inability	Inability to observe runtime behaviours	Designed to observe runtime behaviours but without runtime no information can be fetched.	Have both abilities but also, time taking process.
Advanced malware handling	Advanced malware may use anti-analysis techniques to make static analysis more challenging	It may not be effective against certain types of advanced or sophisticated malware that can detect and resist dynamic analyses.	Less chances to evade in this analysis
Encrypted Codes handling	Encrypted or heavily obfuscated code can be difficult to understand statically	Quite good in obfuscated codes because of runtime activities.	Useful in encrypted codes
Risk of environment	Analysis environment is safer among all	Potential risk of infecting the analysis environment	Risk can be Avoided in early stages

Code Reversing (Optional)

The notion of dynamic analysis could potentially harmonize with a range of approaches designed to interpret serious threats or those with a considerable level of complexity. The exploration covers an analytic reverse engineering process intended to comprehend the one-of-a-kind algorithms, encryption programs, and protocols perpetuating the malware activities.

Memory Analysis (Optional)

Sometimes, decoding memory data serves to be part of dynamic analysis for understanding malware behavior within system memory. This method allows the recognition of several kinds of malware, including instances of code injections and rootkits in addition to more intricate types of malwares.

Network Traffic Analysis (Optional)

As it is carried out, the network activity generated by the malware can be a part of a hybrid analysis approach. This technique provides several advantages in the discovery of control and command servers, the understanding of data extraction processes, and the identification of different communication techniques.

Indicators of Compromise (IOCs) Generation

This evaluation concentrates on selected characteristics or activities of the malware, interpreting them as Indicators of Compromise (IOCs). The power to record these signatures or patterns can be a reliable vehicle for identifying dubious software that could emerge over time. Using both static and dynamic analysis results in a detailed understanding of malware characteristics that are frequently missed by standard analysis alone as illustrated in Table 2. Dynamic analysis facilitates real-time surveillance of malware targets for a more total comprehension of its influence on its ecosystem. Hybrid malware analysis is now a favoured strategy in present-day cybersecurity, allowing experts to stay at the forefront of changing malware dangers and create successful safeguards for systems and sensitive information as illustrated in Table 3.

4. Malware Detection Techniques

In the face of evolving cyber threats, malware detection techniques have become crucial for system protection. Signature-based detection remains a fundamental approach, utilizing recognized patterns to identify and counter malware. Nonetheless, as threats intensify, heuristic and behavioural-based methods have taken on a more important role, giving a pre-emptive way to recognize potential malware through observations of deviations from standard behavior. The field of machine learning has changed our approach to malware detection, empowering algorithms to identify undesirable attributes from extracted code or behavior. Sandboxing technologies are vital for defence, permitting threat researchers to try malware in isolated environments and therefore protect the host set up. In combination, these methods develop a detailed strategy to defend against cyber threats (Figure 3).

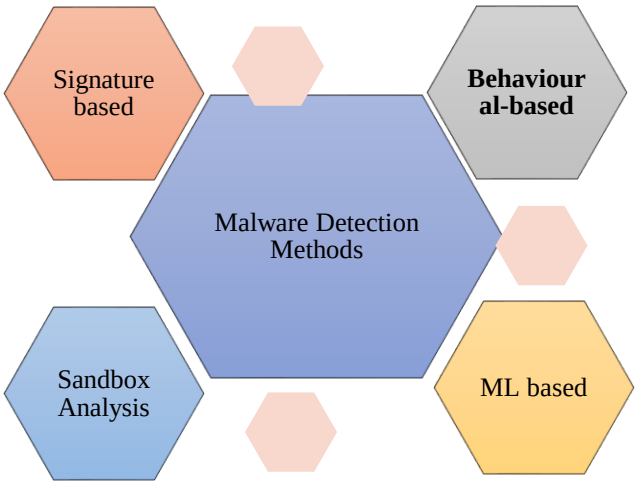


Fig. 3 Malware detection methods

4.1. Signature-Based Detection

Signature based detection systems deploy antivirus software and intrusion detection systems to discover known viruses, in order to cut down on and pinpoint malware risks.

The process involves:

4.1.1. Signature Generation

Security analysts who are experts analyse malware samples in order to detect attributes that could serve as signatures. The components under investigation may include byte values, particular properties in files, operational habits, or code segments connected to a single malware type. A new detection system for identifying polymorphic malware, known as MalHunter, has been introduced by Borojerdi and Abadi.[11] This technique entails the sequence clustering and correspondence of malware features to establish signatures reflecting the behavioural behaviours observed. This goal is achieved through addressing the challenge of spotting polymorphic malware, which can alter its program structure to circumvent recognition from conventional signature-based types of systems. The MalHunter detection system promises to significantly improve malware detection and defence tactics by improving efficiency and effectiveness in dealing with sophisticated malware threats.

4.1.2. Signature Storage

When discovered, these signatures are compiled and saved in a signature database as input for antivirus software, or a security tool.

4.1.3. Scanning Process

An antivirus software performs a check on files or network activity during a scan using the entries stored within its database. If a data set or file matches what is recognized as a known signature, it is deemed malicious. To detect malicious files, one looks at the qualities of a particular file alongside those signatures saved in the database. A file is considered malicious when its characteristics correspond to any of the signatures in place; this technique is key in cybersecurity and is crucial for fending off cyber-attacks.[1][12]

4.1.4. Quarantine or Blocking

Once it is detected, the antivirus software runs according to its settings. A methodology of this sort could include separating infected files, limiting actions linked to the malware on the network, or removing all of the infected material.

4.1.5. Regular Updates

For the identification and protection from original, imaginative, and novel dangers, updates to signature databases are needed. Generally, these advances are made available by antivirus companies through the updates of their definitions. Because of its ability to locate familiar malware varieties, signature-based detection is restricted in some ways. Signatures that haven't been developed cannot be used by it to detect malware that is completely new or unknown. Therefore, the analysis of behavior and heuristics is frequently added to signature-based recognition to offer all-inclusive security against a broad spectrum of threats.

4.2. Heuristic/Behavioural Focused Detection

Security software takes a proactive approach using heuristic and behavioural-based detection methods to notice likely infection based on unusual changes in regular behavior. These strategies, instead of relying only on some signatures or frameworks, assess the properties and usability of software to look for signals of risk. This is a summary of how various methods operate:

4.2.1. Heuristic Analysis

This strategic approach recognizes typical behavioural trends typically attributed to malware. Also included are reconfigurations during operations, access to private information, and the distribution across networks. Rule-based detection functions by examining application features that are determined by known rules or heuristics. For instance, an application may set off a heuristic alert if it attempts to alter important system files without authorization. [13] Another heuristic engine that runs dubious applications in a controlled setting is code emulation. It keeps an eye on how these apps behave and searches for indications of malicious activity without endangering the system itself.

4.2.2. Behavioural Analysis

The real-time behavior of programs is observed by security software through the use of dynamic monitoring and behavioural analysis, which tracks how they interact with the operating system, files, registry, network, and other system resources. To identify departures from the norm, a baseline of typical behavior is created and compared with the observed behavior. These anomalies, such as abrupt increases in network traffic or unapproved access to private documents, could point to malevolent behavior. Sophisticated methods of behavioural analysis use machine learning algorithms to find minute variations or abnormalities in program behavior that might point to the existence of malware. Over time, these algorithms can adjust to increase the accuracy of detection. The capacity of graph-based detection systems to identify malware programs has drawn a lot of interest in recent years. One such plan was put forth in [14], where system calls were converted into a behavior graph that showed data reliance by way of transitions between system calls represented by edges and nodes, respectively. Since the program graph to be marked was extracted and compared with the current graph to evaluate whether the given program was dangerous, the suggested model was successful in recognizing known malware. The model has trouble identifying unidentified malware, though. This emphasizes the want of additional study and advancement in this area to raise the precision and potency of graph-based detection systems.

4.2.3. Anomaly Detection

System logs, network traffic, and other data sources are analysed by anomaly detection algorithms to find odd or anomalous patterns that might point to a security risk. Systems for detecting anomalies establish cutoff points for different metrics and sound an alarm when actual values surpass these limits. For instance, suspicion can be aroused if a user starts to access an abnormally high number of files or establish many network connections. Zero-day or previously unknown attacks without specific signatures can be effectively identified using heuristic and behavioural-based detection techniques. They might, however, also result in false positives if malware-like behavior is displayed by legitimate software. Because of this, these strategies are frequently combined with other detection approaches to offer complete protection against a variety of threats.

4.3. Machine Learning-Based Detection

Using features taken from code or behavior, machine learning-based detection uses methods and algorithms from the field of machine learning to identify malware. Here's a discussion of how machine learning is applied in this context:

4.3.1. Feature Extraction

ML algo needs input features to reach the output. In the case of malware detection, features can be extracted from various sources, including static analysis, which involves features extracted directly from the code or file, such as opcode sequences, API calls, file metadata, and byte-level patterns, and dynamic analysis, which involves features derived from the behaviour of software during execution, such as system calls, network traffic, file operations, and registry modifications. Additionally, hybrid approaches combining features from both static and dynamic analysis can be used to capture a comprehensive view of the software's characteristics. Choosing the most suitable attributes, referred to as features, is crucial, typically organized into a matrix known as a feature vector. However, not all features are beneficial; some may be redundant or irrelevant. [15]

4.3.2. Training Data

Using labelled datasets that include samples of both benign and malicious software, machine learning models are developed. The extracted features of these samples serve as a representation. For the machine learning model to be effective, the training data's quality and diversity are essential. In order to achieve resilience, it has to incorporate multiple factions of the malware, optional benign software and even outliers that might be encountered.

4.3.3. Model Selection and Training

One way of categorizing the multiple ways of detecting a malicious program is through subdivision into different levels of machine learning. Among these are supervised learning techniques like deep neural networks (DNN), support vector machines (SVM), decision trees, and random forests. With these methods, a large amount of labelled data is employed to train the models to recognize the malware features and their classification. Besides, there exist certain shared lessons from the studies on the usage of unsupervised learning techniques like anomaly detection algorithms and k-means clustering, which can help to isolate anomalies or peculiarities which may be suggestive of malware. In all these cases, the interpretability of the model, complexity of the dataset, and availability of computational resources are all contributors to how the best option is arrived at. However, once the more generalized algorithm has been found for the simplified problem, the model is then trained with the labelled dataset and the model parameters are varied so as to increase the accuracy and reduce the prediction error rates.

4.3.4. Evaluation and Validation

Model performance is validated on additional test datasets out of which level of performance is defined as well as performance metrics with an example of f1 score. For example, model selection and model evaluation techniques can also employer the use of cross validation procedure to increase the degree to which the model performs well on several different datasets and reduce chances of excess fitting the model to the data.

4.3.5. Deployment and Monitoring

After passing through the validation process, a machine learning model is incorporated into working systems, where it assesses the incoming files, network activity, and behavior of the system. It is necessary to perform retraining on such datasets periodically to afford the model defence regarding novel malware and emerging threats. This calls for regular checks and revisions. Stereotypically, machine learning based detection comes with advantages of high levels of automation and scalability as well as the ability to detect some previously unknown malware embodiments. Adversarial attacks, unbalanced data, and model interpretability are some of the challenges it faces as well. Due to this, it is often combined with other detection techniques to provide adequate security coverage.

4.4. Sandbox Analysis

Employing these methods for sandboxing, it is possible to execute certain types of software, which are called malwares, in a protected environment and be able to observe their actions without harming the host computer. An overview of sandboxing methods may be presented here.

4.4.1. Virtual Machine Sandboxing

Virtual machine sandboxing involves executing viruses in a simulated environment which is designed to replicate that of actual computer hardware and software environment. Access to the host OS is prevented and protected, and malware is executed in the virtual machine guest OS.

4.4.2. Containerization

Within a containerized environment, every application, along with its dependencies, is packaged up using containerization, which is a form of virtualization that is not heavy-duty. Even malware will operate in a containerized environment, where it can run on the same kernel and use the same resources, yet it has been set apart from the host OS. In this regard, sandboxed malware evaluation environments are popularly provided and controlled through virtualization tools such as Docker or Kubernetes.

4.4.3. Emulation

Emulation involves the creation of a software replica of the operating system and hardware configuration where the malware will operate in. Since the malware functions in an emulation system, it is possible to monitor the action of the malware without endangering the actual computer. Though emulation may be more resource intensive than virtualization, it offers more flexibility and containment to malware analysis regardless of the operating system in use.

4.4.4. Hardware-Based Sandboxing

Certain protective systems and equipment apply hardware approaches to sandboxing in order to contain and study malware. Such remedies create isolated execution environments for suspicious applications by employing various dedicated hardware components such as memories and processors. A background of high performance and scalability of hardware-based sandboxing can enable effective real-time analysis of large quantities of mass malware.

4.4.5. Network Sandboxing

Techniques are employed to investigate the inner workings of malware which is contained within controlled network traffic by intercepting and executing suspicious files or data streams. This helps security analysts in easily identifying and controlling the movement of malware within an organization's network infrastructure by viewing how such malware acts and interacts with other resources and processes. [17]

4.4.6. Cloud Based Sandboxing

Cloud-based sandboxing services provide dynamic and scalable options for the purpose of malware assessment. Files or URLs that are confirmed to be malicious are uploaded to a cloud sandbox and executed while monitoring for any negative activity. Due to its default features of extensibility and flexibility, this mode of sandboxing is best suited for corporations of varying elevated levels of security demands.

The use of sandboxing methodologies is pivotal in the evaluation of malware and in the detection of threats because they provide the ability to study suspicious software within a controlled and safe environment. This enables researchers to learn about the functioning of malware and enables them to find out ways to prevent such threats from emerging in the future. The effectiveness of malware detection methods is still critical in the never-ending arms race between cyber attackers and defenders. The basis is laid by signature-based detection, which is based on recognizing recognized malware patterns. But given the ever-changing threat landscape, heuristic and behavioural-based detection must be adopted in order to identify possible malware proactively based on anomalous behavior. Machine learning enhances the capacity for detection by utilizing advanced algorithms to identify minute trends within large datasets. Furthermore, sandboxing methods offer a vital defensive layer and a secure setting for analysing and comprehending malware behavior. Cybersecurity professionals may strengthen their defences against the dynamic threat landscape and maintain the integrity and resilience of digital ecosystems by utilizing the combined effects of these strategies as illustrated in Table 4.

Table 4. Comparisons of detection methods of malware

Detection Method	Description	Advantages	Disadvantages	Use Cases
Signature-based	Detects malware by comparing files to a database of known malware signatures.	-Fast and efficient for known threats.	- Ineffective against new or modified malware (zero-day threats).	- Legacy systems, basic antivirus software.
Heuristic-based	Analysis code behaviour to detect potentially malicious activities, even if not previously known.	- Can detect new, unknown malware.	- May produce false positives.	- Advanced antivirus solutions, intrusion detection systems.
Behavior-based	Keeps an eye on program behavior in real time to spot questionable activity.	- Effective at detecting zero-day threats.	- High resource usage, potential for false positives.	- Real-time monitoring, advanced security suites.
Sandboxing	Runs doubtful files in a private setting to watch how they behave before allowing them to.	- Highly effective at detecting complex threats.	- Resource-intensive, can be bypassed by sophisticated malware.	- High-security environments, malware analysis labs.
Cloud-based Detection	Transfers analysis to the cloud, where sophisticated algorithms and massive databases are available for malware detection.	- Reduces local resource usage, can leverage big data for accuracy.	- Requires internet connection, potential privacy concerns.	- Lightweight antivirus solutions, mobile security.
File Integrity Monitoring	Keeps track of alterations made to important system files and folders in order to find illegal changes.	- Effective at detecting tampering and persistent threats.	- Limited to file changes, may not detect all types of malwares.	- High-security environments, compliance-driven sectors.
Network-based Detection	Searches for patterns in network traffic that point to the presence of malware.	- Can detect malware before it reaches endpoints.	- May miss threats that do not exhibit network behavior.	- Network security appliances, enterprise-level monitoring.
Machine Learning-Based	Uses learning algorithms to predict and identify malware based on data patterns	- Continuously improves detection accuracy	- Requires large datasets and training; potential for adversarial attacks	- Used in next-gen antivirus and endpoint protection systems
YARA Rules	Identifies malware using a set of rules or patterns based on binary or linguistic patterns found in files.	- Flexible, can be tailored to specific threats.	- Requires expertise to create effective rules.	- Targeted threat hunting, malware research labs.

5. Methodology

Here the setup of a ml-based malware detection model on python platform. The implication stages are below explained (Figure 4).

5.1. Data Pre-Processing

Data pre-processing is necessary to clean up and prepare data for machine learning models, which improves the model's efficacy and accuracy. Three stages have been involved in the pre-processing of the data: reading, verifying, and cleaning the data. The dataset comprises special characters like '?' and 'S'. In the pre-processing of the data, these special characters are given NaN values and the dropna() function is applied to delete any row in the data frame that contains a NaN value.

5.2. Data Classification

The data is split into training and test sets with the aid of train_test_split function within the sklearn.model_selection package. These sets are then used to train and evaluate a machine-learning model. To allow the model to be evaluated on data that hasn't been seen before and reduce the chance of overfitting, the data must be divided into training and test sets. The dataset is split up in the code so that the test set only includes the last column-which is the target variable-and the training set includes every characteristic except the last column, which is the target variable. This setup makes it easier to efficiently feed data into the model for testing and training.

The parameter << test_size = 0.2 >> establishes the ratio of data assigned for testing. Here, twenty percent of the data is used for testing and the other eighty percent is used for training. To make sure that samples from every class are evenly distributed between the training and test sets, the data is then randomly generated.

5.3. Training & Testing the Model

Machine learning algorithms that learn from such data and store the information for future prediction depend heavily on training data. Every artificial intelligence (AI) and machine learning (ML) project starts with training data because a system that learns from humans and makes predictions for humans requires training.

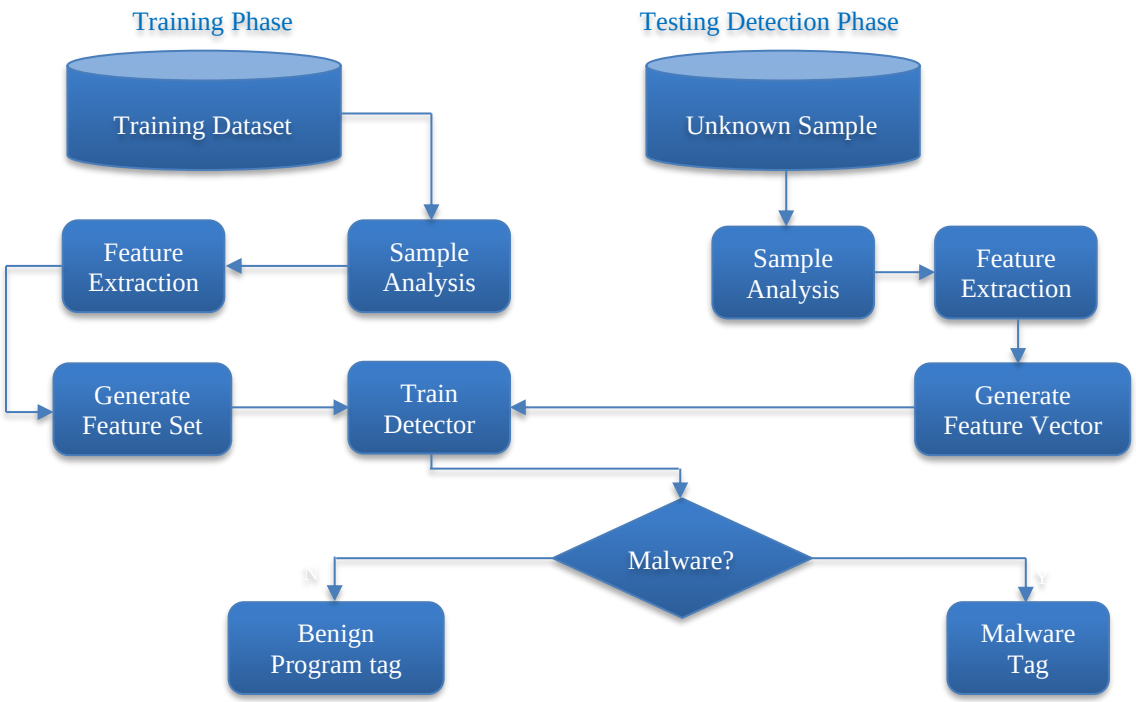


Fig. 4 ML based malware detection model

The algorithm is trained by providing it with labelled examples so that it may discover patterns and connections in the data. Predictions are then based on newly acquired knowledge and never-before-seen data. The model's performance and capacity for generalization are strongly influenced by the calibre and variety of the training data.

This code trains the model on the training set of data using the Sequential model's fit () method. The eight-training set's objective variable is called train_y, while the training set's input characteristics are called train_x.

The validation set is used to keep an eye on the model's performance throughout training and to guard against overfitting. An epoch is a set of training data iterations.

Figure 5 shows the plot of the training accuracy and validation accuracy over a variety of epochs in the first subplot. Plotting the training loss and validation loss across the same range of epochs is the second subplot. These subplots let us determine whether the model is overfitting or underfitting the data by comparing how the model performs on the training and validation sets during the training process.

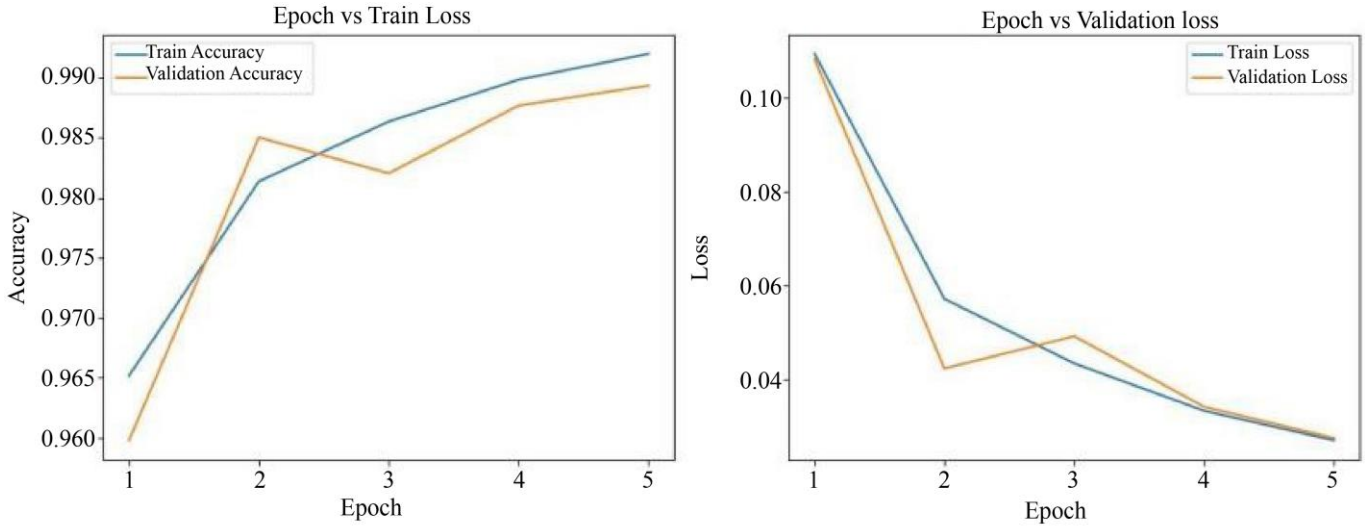


Fig. 5 Plots of the training accuracy and validation accuracy as a function of epochs

The model used to create predictions on the test set following training. In the next section, to evaluate their effectiveness using three key metrics: precision, recall and F1 score. Precision indicates the degree of correctness in positive predictions by computing the ratio of true positive predictions to the total positive predicted cases. This metric demonstrates the model's effectiveness in controlling the rate of false positive errors. In contrast, the recall captures the ability of the model to find all relevant cases by determining the proportion of relevant instances that have been retrieved over the total relevant instances. The performance of the model in all its aspects is presented using a single figure that is called F 1 score which is the geometric mean of precision and recall.

This measure is particularly beneficial in evaluating recall and precision as it strives to find the balance between the two which is often necessary with regards to imbalanced datasets.

6. Result and Discussion

An efficient model for detecting malware tends to reduce the occurrence of false described positives (i. e. benign samples being classified as malware) while maximizing true positive within detecting malware class (i.e. classifying malware samples as malware) (Figure 7). In this context, a high recall value explains that the model is capable of identifying all the malware samples correctly categorized as malware with few if any misclassifications or false negatives, in which case, the malware samples are misleadingly identified as clean. A model with a high F1 score balance both recall and precision, meaning that the model is performing very well indeed. These evaluation measures provide good insight into the performance of the model with respect to the test data set and can be used to evaluate difference performance models as well as fine tune model parameters. (Figure 6)

```
In [16]: y_pred = model.predict(test_x)
         for i in range(len(y_pred)):
             if y_pred[i] > (1-y_pred[i]):
                 y_pred[i]=1
             else:
                 y_pred[i]=0
         print("\n\n\nPrecision : ",precision_score(test_y,y_pred)*100)
         print("Recall : ",recall_score(test_y,y_pred)*100)
         print("F1 Score : ",f1_score(test_y,y_pred)*100)

         WARNING:tensorflow:Model was constructed with shape (None, None, 
         ape=(None, None, 215), dtype=tf.float32, name='dense_input'), nam
         nse_input'"), but it was called on an input with incompatible sha
         94/94 [==============================] - 0s 445us/step


         Precision :  98.97674418604652
         Recall :  98.79294336118849
         F1 Score :  98.88475836431226
```

Fig. 6 The Results of accuracy achieved

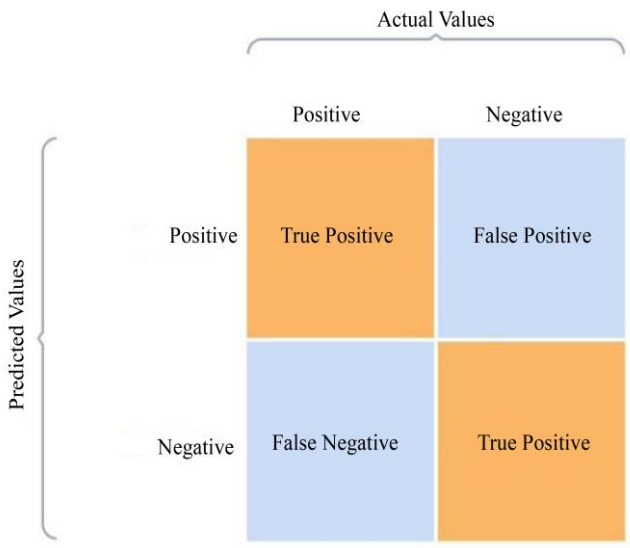
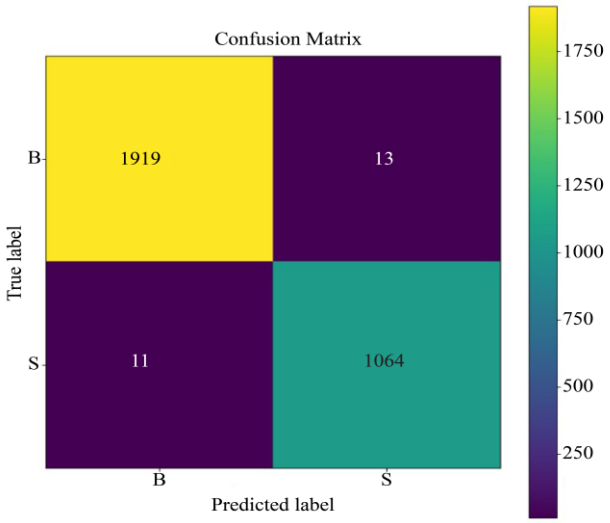


Fig. 7 Confusion Matrix (a) Sample space



(b) Model output

6.1. Challenges in Malware Analysis and Classification

Malware research and classification is challenged by the ever-evolving and intricate nature of malware. The reason is that due to the fact that creativity of a malware writer knows no bounds, it is always difficult to recognize and classify new and novel malware strains. In addition, another hindrance is the vast spectrum of current malware. Keeping up with current threats may prove impossible given that malware authors are never short of new releases. This issue requires researchers and security professionals to keep abreast of developments in malware and its deterrents. The other challenge encountered in the analysis and classification of malware is the likelihood of false positives and false negatives. This means a failure to detect a malware threat constitutes a false negative, whereas a legitimate benign file incorrectly reported as a malware threat constitutes a false positive. It goes without saying that detecting malware is equally important if systems and their information are to be protected from compromises.

6.1.1. Limitations of AI and Machine Learning in Cybersecurity

One of the key limitations of AI and machine learning systems is their heavy reliance on data to develop and improve their threat detection capabilities. If these systems are trained on small or corrupted data sources, then they will be severely hindered in their ability to accurately detect or predict threats. This creates an inherent "arms race problem" in which the effectiveness of such systems is entirely reliant on the availability and quality of data. Cybersecurity systems must possess the capability to handle sudden and unpredictable changes in cyber-attacks, but unfortunately, this aspect has rarely been thoroughly tested or proven in the practical implementation of AI and machine learning. It's important to recognize that AI is not naturally inclined towards generating adaptive responses that possess the same level of flexibility as humans exhibit when confronted with such dynamic changes. Consequently, after several decades of dedicated work in the field of cybersecurity and AI, most existing systems still struggle to generate effective and adaptive responses that can adequately cope with rapidly evolving cyber threats. It is crucial to acknowledge that most AI and machine learning algorithms can be easily deceived by malicious actors who tamper with their training data, among various other adversarial machine learning attacks.

7. Case Studies and Real-World Implementations

Concurrently, by inculcating the principles of the Sender Network Authority, a bespoke machine learning model amalgamates various features such as historical identity behavior and contextual information to effectively counter spear-phishing attempts via email and preempt any domain or account impersonations. As a result, ML initiatives have proven to be instrumental in mitigating the detrimental impact of spam and deceptive emails, even during the unprecedented challenges posed by the global COVID-19 pandemic. For instance, these initiatives have successfully countered spam messages offering false remedies for COVID-19, such as the infamous claim stating that "delivery will commence upon payment of shipping fees." It is important to note, however, that the dissemination of transparent information regarding the exact monetary amounts or percentages of spam thwarted varies across different organizations and has not been made readily accessible to the general public.

8. Ethical and Legal Implications

Professionals in machine learning and artificial intelligence, and the developers at large, have a profound ethical obligation to ensure that the AI systems they are producing will not be used for malicious purposes. The potential uses being discussed are truly staggering and are continuously growing at an exponential rate. While it may be desirable for a robot to assume the role of a defender, the question of why there are deploying robots to engage in lethal activities or similar actions is complex and warrants a deeper examination. When contemplating the implementation of AI for cybersecurity purposes, it is of paramount importance to actively strive towards preventing unauthorized individuals from gaining access, rather than inadvertently enabling their intrusion. In this particular case, the battlefield is not clearly demarcated. Black hat hackers may exploit the power of AI to conceive and execute sophisticated malware attacks that possess the unnerving ability to constantly morph and evolve until their objectives are achieved. At present, AI can be harnessed to execute the most efficient deployment of malignant software and consequently has the potential to learn from its failures.

9. Conclusion and Future scope

In conclusion, malware analysis and classification are critically important adversarial soft technologies for the detection of threats and mitigating the damage caused by attack applications. In this area, for example, hybrid, dynamic, or static analysis are usually used. Nevertheless, there are also a lot of barriers, including the innovation of malware and its increasing complexity, the wide spectrum of malware's genomes, and the issues of false positive and negative results.

Malware analysis and categorization are nevertheless pivotal in protection of computer systems and networks against the onslaught of malware attacks, in spite of the challenges. The range of malware detection solutions available sees to such needs as the need to protect systems from dynamic reach of threatening forces. These are signature-based detection, which is one of the traditional approaches, heuristic and behavioural-based detection, machine learning detection, and lastly sandbox detection of the malware. The need for an integrated approach to malware analysis and detection has been advocated in order to enhance cyber security through the use of different techniques developing synergy. The paper also examines malware analysis tool that includes static analysis, dynamic analysis,

reverses engineering, the and tool for visualizing highlighting their pivotal role in dissecting and understanding malware at a granular level. The accuracy and efficiency of malware analysis and classifications can be improved by researchers by combining cutting-edge machine learning techniques and natural tongue processing algorithmics. This would allow for more effective defence against malware-related threats. state of machine learning-based malware detection is generally encouraging, with deep learning-based methods exhibiting the most promise. is definitely space for development, particularly when it comes to integrating different machine-learning models and handling feature extraction and selection. Additionally, there is a need for more robust evaluation methods.

References

[1] Daniel Gibert, Carles Mateu, and Jordi Planes, "The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges," *Journal of Network and Computer Applications*, vol. 153, pp. 1-22, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Abdelouahab Amira et al., "A Survey of Malware Analysis Using Community Detection Algorithms," *ACM Computing Surveys*, vol. 56, no. 2, pp.1-29, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Akshat Gaurav, Brij B. Gupta, and Prabin Kumar Panigrahi, "A Comprehensive Survey on Machine Learning Approaches for Malware Detection in IoT-Based Enterprise Information System," *Enterprise Information Systems*, vol. 17, no. 3, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Tony Quertier et al., "A Lean Transformer Model for Dynamic Malware Analysis and Detection," *arXiv*, pp. 1-10, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] Mihai Christodorescu, Somesh Jha, and Christopher Kruegel, "Mining Specifications of Malicious Behavior," *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 5-14, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[6] Mike Nkongolo, "Assessing Cyclostationary Malware Detection via Feature Selection and Classification," *arXiv*, pp. 1-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Halit Bakır, and Rezan Bakır, "Droidencoder: Malware Detection Using Auto-Encoder Based Feature Extractor and Machine Learning Algorithms," *Computers and Electrical Engineering*, vol. 110, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Aastha Sharma, Divya Upadhyay, and Shanu Sharma, "Enhancing Blockchain Security: A Novel Approach to Integrated Malware Defence Mechanisms," *Engineering Research Express*, vol. 6, no. 2, pp. 1-14, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Baskoro Adi Pratomo et al.,, "Enhancing Enterprise Network Security: Comparing Machine-Level and Process-Level Analysis for Dynamic Malware Detection," *arXiv*, pp. 1-31, 2023.[CrossRef] [Google Scholar] [Publisher Link]

[10] Valerian Rey et al., "Federated Learning for Malware Detection in IoT Devices," *Computer Networks*, vol. 204, pp. 1-14, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Rajif Agung Yunmar et al., "Hybrid Android Malware Detection: A Review of Heuristic-Based Approach," *IEEE Access*, vol. 12, pp. 41255-41286, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Borja Molina-Coronado et al., "Light up that Droid! On the Effectiveness of Static Analysis Features against App Obfuscation for Android Malware Detection," *arXiv*, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Avinash Singh, Richard Adeyemi Ikuesan, and Hein Venter, "MalFe-Malware Feature Engineering Generation Platform," *Computers*, vol. 12, no. 10, pp. 1-20, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Matthew G. Gaber, Mohiuddin Ahmed, and Helge Janicke, "Malware Detection with Artificial Intelligence: A Systematic Literature Review," *ACM Compuing Surveys*, vol. 56, no. 6, pp. 1-33, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] S.M. Rakib Hasan, and Dhakal Aakar, "Obfuscated Malware Detection: Investigating Real-world Scenarios through Memory Analysis," *arXiv*, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Pie Yan et al., "Prompt Engineering-Assisted Malware Dynamic Analysis Using GPT-4," *arXiv*, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Andrea Ponte et al., "SLIFER: Investigating Performance and Robustness of Malware Detection Pipelines," *Computers and Security*, vol. 150, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Fehmi Jaafar, Gabriela Nicolescu, and Christian Richard, "A Systematic Approach for Privilege Escalation Prevention," *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Vienna, Austria, pp. 101-108, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[19] L. Nataraj et al., "Malware Images: Visualization and Automatic Classification," *VizSec '11: Proceedings of the 8th International Symposium on Visualization for Cyber Security*, pp. 1-4, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[20] Igar Santos et al., "Opcode Sequences as Representation of Executables for Data-Mining-Based Unknown Malware Detection," *Information Sciences*, vol. 231, pp. 64-82, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[21] Konrad Rieck et al., "Learning and Classification of Malware Behaviour," *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2008)*, pp. 108-125, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[22] Clemens Kolbitsch et al., "Effective and Efficient Malware Detection at the End Host," *Proceedings of the 18th USENIX Security Symposium*, pp. 351-398, 2009. [Google Scholar] [Publisher Link]

[23] Michael Bailey et al., "Automated Classification and Analysis of Internet Malware," *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection (RAID 2007)*, pp. 178-197, 2007. [CrossRef] [Google Scholar] [Publisher Link]