DESIGN AND DEVELOPMENT OF A NOVEL FRAMEWORK FOR MALWARE IDENTIFICATION AND CLASSIFICATION

Thesis submitted for the award of the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science and Engineering

By

Vikas Verma

Registration Number: 42100296

Supervised By

Dr. Arun Malik

Computer Science and Engineering

Professor

LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

2024

DECLARATION

I, hereby declared that the presented work in the thesis entitled **"Design and Development of a Novel Framework for Malware Identification and Classification"** in fulfilment of degree of **Doctor of Philosophy (Ph. D.)** is outcome of research work carried out by me under the supervision of Dr. Arun Malik, working as Professor, in the School of Computer Science and Engineering of Lovely Professional University, Punjab, India. In keeping with general practice of reporting scientific observations, due acknowledgements have been made whenever work described here has been based on findings of other investigator. This work has not been submitted in part or full to any other University or Institute for the award of any degree.

(Signature of Scholar)

Name of the scholar: Vikas Verma

Registration No.: 42100296

Department/school: School of Computer Science and Engineering

Lovely Professional University,

Punjab, India

CERTIFICATE

This is to certify that the work reported in the Ph. D. thesis entitled "Design and Development of a Novel Framework for Malware Identification and Classification" submitted in fulfillment of the requirement for the award of degree of **Doctor of Philosophy (Ph.D.)** in the School of Computer Science and Engineering, is a research work carried out by Vikas Verma, 42100296, is bonafide record of his/her original work carried out under my supervision and that no part of thesis has been submitted for any other degree, diploma or equivalent course.

(Signature of Supervisor)

Name of supervisor: Dr. Arun Malik

Designation: Professor

Department/school: Computer Science and Engineering

University: Lovely Professional University

ABSTRACT

In the present era of technology, the internet plays a central role in most activities. Therefore, it is crucial to prioritize the protection of our applications, data, and information from potential attackers who constantly create malicious programs and make attempts in order to compromise our resources. Therefore, in the present era, the examination of malware has become a significant focus due to the fact that attackers continue to create a diverse array of malware that is constantly evolving in its characteristics. In recent times the use of new technologies such as virtualization technology and distribution tracking avoidance has made it difficult to track the latest cybercrimes. Furthermore, in the realm of cyber research, the tracking of harmful code distributors is more essential than analyzing malicious programs. Nowadays malware is quite different in terms of properties as compared to the time of its starting era and thus require different methods to trace. So, the detection of latest malware or the new malware is quite crucial before it start impacting the system. An important area of study that people want to learn more about is the malware used in Zero Day attacks that will happen in the future. Hence this proposal is primarily focused on the development of a novel technique for malware identification and classification.

Malware has to be discovered before it impacts a lot of systems to safeguard computer systems and the internet from them. Several types of research on malware detection techniques have recently been conducted. But it is still difficult to identify malware. There exist two main processes for recognizing malware. One strategy is based on signature detection, while the other technique is based on analyzing behavior. The signature-based technique is fast as well as effective in recognizing known malware. However, the behavior-based approach, which utilizes machine intelligence and other methods, is capable of detecting unknown and intricate malware to some degree. Nevertheless, this approach is more intricate in nature. However, none of the techniques are capable of identifying every type of malicious software, particularly zero-day attacks, especially when the number of malware instances continues to rise on a daily basis. Previous iterations of malware were easily detectable due to their ability to conceal their characteristics. However, modern malware employs many tactics, such as obfuscation, to prolong their anonymity. Additionally, they can circumvent network

and system security measures, including firewalls and other security checks. In addition, a variety of malwares are employed to initiate the attack, thereby amplifying the impact and causing more severe damage.

It is also important to find the malware family like identification as if malware is detected then what kind of malware category that is detected based on its properties needs to finalize for understanding its functionality and to execute some countermeasures for future to mitigate it. Malware classification can be done using many methods, such as image based where binary values are converted into images or by applying feature-based algorithms. To enhance the classification mechanism, the inclusion of additional information will be advantageous. In order to get better results when putting malware into groups, we need to make high-quality algorithms that use modern machine intelligence techniques.

One another important thing that is useful for malware analysis is the development of a framework or tool that helps to validate and test the files and other data items for malware identification and classification. To defend and anticipate future attacks, there is a need to utilize technologies for analyzing malware. The main choice is to use generally open-source tools. But with the increase in complexity of malware variations, it has become more difficult to understand and compare them. Thus, there is always a need for new tools and integration of more features in tools for extensive malware analysis. This motivates me for further research in the area of malware analysis and to devise few novel approaches for malware analysis and classification especially the new and unidentified malware through better framework development.

The challenges that we have perceived can be taken care of to some extent using the new machine intelligence techniques that can speed up the process of identification and classification to a larger extent. The fundamental idea of a machine learning approach is provide training to the model to perform a certain task based on an algorithm, such as classification, clustering, regression, etc. Training is carried out based on the input data set and the model built in is then used to predict. The main steps in the execution of the same are Intake of data, Transformation of data, Training of model, testing of the model followed by deployment of model. One important thing which is applicable for

all malware detection techniques is that the features that are used for the detection are different for different techniques. In signature-based byte sequence analysis, Dynamic Link Libraries (DLL) are utilized. Behavior-based analysis involves the usage of APIs and system calls, while heuristic analysis focuses on operation codes and context-free grammars.

Currently, Windows is one of the most extensively utilized operating systems. The proliferation of malware presents a substantial menace to the integrity and security of Windows operating systems. The aim of this research work is to devise a proficient methodology for recognizing and categorizing various forms of malicious software on the Windows operating system, with the purpose of tackling the prevalent problem of malware. A proposed method for effectively detecting and categorizing malicious software on Windows involves combining Support Vector Machine, Decision tree, and Logistic Regression techniques in an ensemble approach. The proposed approach utilizes methods of feature selection to identify the patterns and characteristics of many malware clubs. The authentic dataset for malware will be utilized to evaluate and appraise both proposed ensemble and the existing fundamental machine learning methods. Ultimately, this will aid both inexperienced and skilled individuals in the field of cyber security to understand and get ready for the always evolving dangers presented by the latest type of malicious software on Windows personal computers.

ACKNOWLEDGEMENT

Completing this Ph.D. thesis has been a challenging and rewarding journey, and it would not have been possible without the invaluable support, guidance, and encouragement of many individuals.

To my beloved mother **Ms. Shashi Bala**, who is no longer with me: your love, strength, and belief in me have been my constant source of inspiration. Though you are not here to witness this milestone, I carry your lessons and values in every step of this journey. This achievement is a tribute to you. I am profoundly grateful to my family, especially my father **Mr. Sushil Kumar Verma**, my wife **Ms. Poonam** and son **Tanmay Verma**, for their unwavering support, understanding, and encouragement during the most challenging phases of this journey.

I would like to express my heartfelt gratitude to my research supervisor **Dr. Arun Malik**, Professor, Lovely Professional University, for his continuous immortal support, motivation, and guidance. Without his support and kindness, I could not complete the thesis successfully. I wholeheartedly thank him for his inspiration and encouragement in providing valuable suggestions, from selecting the topic, clarifying doubts, publishing, writing, and completing my thesis.

I wish to record my deep sense of gratitude and profound thanks to **Dr. Rajeev Sobti**, Professor, Lovely Professional University, for inspiring guidance and constant encouragement with my work during all stages, to bring this thesis into fruition. I thank Prof. **Dr. Isha Batra**, Professor, Lovely Professional University for the invaluable guidance to address various issues and challenges encountered, immeasurable encouragement and kind cooperation during my research work.

My sincere thanks go to **Shamneesh Sharma**, **Dr. Rahul Sharma, Awadhesh Kumar Shukla**, **Aman Kumar** and **Aditya** who provided me with continuous support, stimulating discussions, and constructive feedback.

Finally, I wish to extend my appreciation to the countless individuals whose names might not appear here but who played a part in this journey, including friends, mentors,

and well-wishers. Your encouragement and contributions have left a lasting impact on my life.

Vikas Verma

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Networks
API	Application Programming Interface
APT	Advanced Persistent Threats
AUC	Area Under Curve
BP	Back Propagation
C&C	Command and Control
CNN	Convolutional Neural Networks
DBN	Deep Belief Network
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DT	Decision Tree
FDI	False Data Injection
FL	Federated Learning
GBM	Gradient Boosting Machines
GNN	Graph Neural Networks
GPSC	Genetic Programming Symbolic Classifier
IDS	Intrusion Detection Systems
IoT	Internet of Things
IPS	Intrusion Prevention Systems
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
NFF	Network Flow Feature

PCA	Principal Component Analysis
PE	Portable Executable
PSO	Particle Swarm Optimization
RAT	Remote Access Trojan
RF	Random Forest
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SAE	Stacked Auto Encoder
SE	Symbolic Expressions
SVM	Support Vector Machine
URL	Uniform Resource Locator

Chapter 1

Introduction

1.1 Overview

The advancement in the field of cybersecurity faces a rising challenge in contending the spread of malware, which continues to evolve in terms of complexity and erudition [1]. Malware, or Malicious software, poses a substantial threat to the integrity, security, privacy, and availability of digital systems, encompassing a lot of malicious software designed to penetrate, interrupt, or exploit vulnerable systems and networks [2].

Malware, an abbreviation for malicious software, is purposely developed program that disrupts, damages, or gains unauthorized entry to computer systems, networks, or devices. Malware encompasses a wide range of malicious software, each possessing distinct characteristics and capabilities. This includes viruses, worms, trojans, ransomware, spyware, adware, and rootkits.

There are many bad things that malware can do, based on how it works and what the attacker wants to do. These are some common things that malware can do [3]:

- Data Theft: Malware can steal important information such as personal credentials, confidential documents. This information can be used for identity the fraud related to finance or theft.
- System Interruption: Malware may interrupt the regular operation of digital systems by causing crashes, freezes, or slowdowns. This disruption can affect uptime, productivity, and damage to hardware or software components.
- File Modification or Deletion: Malware can modify, delete or encrypt the contents of the files and data stored on infected systems. This action can lead to data loss or irreparable damage to critical files and applications.
- Remote Control: There are certain types of malware, such as remote access trojans (RATs) and backdoors that let criminals get into affected systems without permission from afar.
- The user might know be knowing about the compromised systems and the attackers would have the control on the compromised systems enabling the

possibility of executing commands, and exfiltrate data without the user's knowledge.

- Botnet Formation: Malware can add computers that have been affected to a botnet that is a group of infected devices managed by a bad person or group. Users often connect botnets to work together to start organized attacks, send spam emails, carry out Distributed Denial-of-Service (DDoS) attacks, or mine cryptocurrency.
- Ransomware Attacks: Ransomware is a variant of malware that locks down the system and encrypts the files, so only authorized users can get to them. Attackers demand a ransom to get decryption keys or to open the system. If the ransom is not paid, they say they will delete or leak the protected data forever.
- Espionage and Surveillance: Certain malware variants, such as spyware and keyloggers, are designed to monitor and record user activity on systems. Such software generally works behind the scenes usually as background processes. This information can be used for espionage, surveillance, or corporate espionage, allowing attackers to steal sensitive information or monitor user behavior.
- Cryptocurrency Mining: Some malware strains, known as cryptojacking malware, hijack the computing resources of infected systems to mine cryptocurrencies such as Bitcoin or Monero. This activity consumes CPU and GPU resources, slowing down system performance and increasing energy consumption.
- Propagation: Malware often includes mechanisms for self-replication and propagation to taint other devices and systems. This can be achieved through email attachments, malicious links, network exploits, removable media, or drive-by downloads.
- System Vulnerability Exploitation: Malware can exploit known vulnerabilities in operating systems, software, or network protocols to acquire unauthorized access to systems or execute arbitrary code. Exploiting vulnerabilities allows malware to bypass security controls and escape detection mechanisms.

In summary, malware can have a wide range of detrimental effects on networks, computer systems, and users, ranging from data theft and system disruption to espionage and financial fraud. Detecting and mitigating malware threats are critical components of cybersecurity strategies aimed at protecting against malicious activities and safeguarding digital assets.

1.2 Types of Malware

Malware, which stands for "malicious software," is a broad term for a lot of different bad programs that are made to get into computer systems and networks and take over or damage them. Here are some common types of malware:

- **Viruses:** A virus is a program that replicates itself by infecting other files or programs on your computer. It can do harm by messing up data, taking data, or getting in the way of physical action. It is usually attached to an executable or file and is launched when the file is opened or executed. For example, the ILOVEYOU virus, which emerged in 2000, spread through e-mail links and caused serious damage to systems and user data [4]. Viruses can reduce system performance, cause invalid data, and cause frequent crashes. Some viruses, such as Melissa, which infected Word files in 1999, use macros to perform malicious operations. Preventing this virus includes using the latest anti-virus software, avoiding suspicious downloads, and updating important files [5].
  Examples: File-infecting viruses, macro viruses, boot sector viruses.

- **Worms:** Worms are autonomous programs that propagate via networks by taking advantage of weaknesses in operating systems or network services. These malicious programs have the ability to quickly infect a significant number of machines and spread without requiring any action from the user. Unlike viruses, worms can propagate without the need for user contact. An exemplary instance is the SQL Slammer worm, which rapidly propagated in 2003 by exploiting a flaw in Microsoft SQL Server, resulting in extensive network disruptions [6]. Worms have the ability to use a substantial amount of bandwidth and system resources, affecting the network performance. The Conficker worm, identified in 2008, took use of Windows vulnerabilities and established a network of compromised devices for a range of malevolent purposes [3].

Worms such as Stuxnet, which focused on industrial control systems in 2010, can also possess a high level of complexity and target specific entities. Preventive methods encompass the frequent upgrading of software, the utilization of firewalls, and the implementation of network security protocols. The Morris Worm, which emerged in 1988, caused significant disruption to numerous computer systems, therefore highlighting the enduring menace posed by worms. Efficient network surveillance and user consciousness are crucial in identifying and reducing the impact of worm infections.

Examples: Internet worms, email worms, network worms.

- **Trojans:** Trojans are insidious software applications that masquerade as genuine programs, yet harbor malevolent code. They frequently disguise themselves as beneficial software or files and can be employed to pilfer sensitive information, obtain unauthorized entry into systems, or aid other forms of malware contaminations. An example of this is the extensive utilization of the Zeus Trojan to illicitly acquire banking information through the process of recording keystrokes and obtaining confidential data. Trojans, unlike viruses and worms, do not have the ability to reproduce on their own, instead they depend on users to propagate them [7]. Attackers have the ability to establish backdoors in computer systems, which grants them remote access to computers that have been compromised. The Emotet Trojan, originally designed as a banking Trojan, has developed the capability to distribute many forms of malware, including ransomware. Trojans have the ability to deactivate security software, so granting attackers greater authority over the compromised system. To prevent Trojan infestations, it is necessary to utilize powerful antivirus software, exercise caution when dealing with email attachments, and exclusively download software from reliable sources [7]. The FakeAV Trojan masquerades as antivirus software, deceiving users into making payments for counterfeit virus elimination services [8]. It is essential to have a well-informed user and a high level of alertness in order to effectively recognize and prevent Trojan threats.

  Examples: Remote access Trojans (RATs), banking Trojans, spyware Trojans.

- **Ransomware:** Ransomware encrypts files or locks down whole systems so that only authorized users can't get to them. Attackers demand ransom payments in exchange for decryption keys, threatening to delete or leak the encrypted data if the ransom is not paid. One notorious instance is the **WannaCry** ransomware attack of 2017, which exploited a Windows vulnerability to infect thousands of computers over the world. Ransomware often spreads through phishing emails, malicious attachments, or by exploiting software vulnerabilities. Once infected, victims typically see a ransom note demanding payment, usually in cryptocurrency, to decrypt their files. The **Petya** ransomware, which emerged in 2016, not only encrypted files but also prevented computers from booting by encrypting the master boot record. Preventing ransomware involves regular data backups, using robust antivirus solutions, and keeping software updated [9]. The **Locky** ransomware, which spread through email attachments disguised as invoices, highlighted the importance of email security awareness [10]. Organizations and individuals should implement network segmentation and educate users about phishing scams to mitigate ransomware risks. It is not recommended to pay the fee because it does not guarantee file return and may lead to more crime.

  Examples: CryptoLocker, WannaCry, Ryuk.

- **Spyware:** Spyware is software that is meant to watch and record what a person does on their computer without them knowing. It can track keystrokes, capture screenshots, record browsing habits, and steal personal or financial data for malicious purposes. An example of spyware is **Keyloggers**, which record keystrokes to capture confidential information. Spyware can enter a system through infected downloads, malicious websites, or bundled with legitimate software. The **CoolWebSearch** spyware hijacked web browsers, redirecting users to unwanted websites and collecting search queries. Spyware often degrades system performance and compromises user privacy [11]. The **FinFisher** spyware, used for surveillance, highlighted the risks of spyware in targeted attacks. Preventing spyware involves using anti-spyware tools, keeping software updated, and avoiding suspicious downloads. The **DarkHotel** spyware

targeted business executives through hotel Wi-Fi networks, showing the sophistication of some spyware attacks [12]. Educating users about the dangers of downloading untrusted software and practicing safe browsing habits are crucial in combating spyware.

Examples: Keyloggers, screen capture spyware, webcam spyware.

- **Adware:** Without the user's permission, adware shows them ads they don't want to see or sends them to harmful websites. It often comes bundled with legitimate software and can degrade system performance or compromise user privacy. This type of software, often disguised within seemingly innocuous downloads, embeds itself into systems to inundate users with advertisements. A notorious case is the Superfish adware, pre-installed on Lenovo laptops, which not only flooded screens with ads but also compromised user security by intercepting encrypted connections [13]. Adware's intrusive nature extends beyond mere pop-ups, altering browser settings, redirecting searches, and clandestinely tracking online behavior to deliver targeted ads. Despite its non-malicious intent, adware can significantly degrade system performance, disrupt workflows, and compromise user privacy by harvesting sensitive data without consent. Uninstalling adware proves challenging, often requiring specialized tools or expert intervention. As users navigate the digital landscape, vigilance during software installations and regular security checks become paramount defenses against the encroachment of adware and its disruptive consequences. One example of adware is the notorious "Superfish" adware that made headlines due to its intrusive nature and security implications. Superfish was pre-installed on some Lenovo laptops between 2014 and 2015. It worked by injecting third-party advertisements into web pages the user visited, often without their consent.

  Examples: Browser hijackers, pop-up adware, click fraud adware.

- **Keyloggers:** Keyloggers record keystrokes typed by a user, allowing attackers to capture sensitive information such as passwords, credit card numbers, and other confidential data. They operate covertly in the background, often evading exposure by antivirus and security measures. One example of a keylogger is the

Zeus Trojan, which targeted financial institutions and harvested login credentials to perpetrate banking fraud. Keyloggers can be spread in many ways, such as through harmful email files, websites that are infected with malware, or software downloads that have been tampered with. Once they are installed, they quietly record keystrokes and send the information they collect to sites that are controlled by hackers. Keyloggers can be used for more than just stealing money. They can also be used for identity theft, business spying, and privacy invasion. Detection and removal of keyloggers require specialized tools and expertise, making them a persistent threat to both individuals and organizations. Vigilance in practicing good cybersecurity hygiene, such as regularly updating software and employing robust security measures, is essential to mitigate the risk posed by these stealthy adversaries.

Examples: Hardware keyloggers, software keyloggers, memory injection keyloggers [14].

- **Rootkits:** Rootkits are stealthy malware programs that conceal their presence and activity on a compromised system. They often manipulate operating system functions or system calls to bypass detection by antivirus software and security mechanisms. One prominent example of a rootkit is the Sony BMG rootkit, which was included in some Sony audio CDs in the mid-2000s. It installed itself when users played the CDs on their computers, opening a backdoor for potential exploitation by cybercriminals [3]. People often use rootkits for malicious things, like spying, data theft, and making other types of computer operations easier. Detecting and removing rootkits can be challenging due to their ability to cloak themselves from antivirus programs and system scans. Advanced security measures such as integrity checking and behavior analysis are necessary to detect and combat these stealthy threats effectively.

  Examples: Kernel-mode rootkits, user-mode rootkits, firmware rootkits.

- **Botnets:** A botnet is a group of computers that have been hacked and are directed by a central command-and-control server. Spam emails, DDoS attacks, coordinated strikes, and mining for cryptocurrencies are all things that can be done with them. One notorious example of a botnet is the Mirai botnet, which

gained notoriety for launching massive DDoS attacks in 2016, disrupting services of internet worldwide. Usually, computers are infected with malware, which lets a person called the "botmaster" control them from afar and use them for bad things like spamming, stealing private information, or coordinating hacks. Infected devices often include computers, smartphones, and IoT devices, making botnets a pervasive threat across multiple platforms. Detecting and dismantling botnets require association between law enforcement agencies, cybersecurity experts, and internet service providers due to their distributed and resilient nature. Implementing strong security measures, such as using antivirus software, regularly updating software, and securing network devices, can help mitigate the risk of botnet infections and prevent machines from becoming unwitting participants in these malicious networks [15].

Examples: Mirai, Zeus, Emotet.

- **Fileless Malware:** Fileless malware operates in system memory without leaving traces on disk, making it hard to identify using traditional antivirus. It leverages legitimate system tools and processes to execute malicious activities, such as exploiting vulnerabilities or stealing data. A notable example of fileless malware is Poweliks, which infected computers by exploiting vulnerabilities in Windows and using PowerShell scripts to execute its commands directly in memory [16]. Because fileless malware doesn't rely on traditional file-based techniques, it can evade detection by many antivirus programs and security mechanisms, making it particularly challenging to detect and mitigate. Its stealthy nature makes it a preferred tool for cybercriminals seeking to infiltrate networks, steal data, or carry out espionage with minimal risk of detection. Protecting against fileless malware requires a multi-layered security approach that includes behavioral analysis, Endpoint Detection and Response (EDR) solutions, and user education to find and avoid phishing attempts and suspicious activities.

  Examples: PowerShell-based malware, memory-resident malware, in-memory exploit payloads [17].

New variants and techniques of cyber threats keep coming out, which shows how important it is to have strong cybersecurity means to protect yourself from malware infections and cyberattacks.

1.3 Evolution of Malware

The historical development of malware spans several decades, evolving from rudimentary viruses, spyware and worms to complex ransomware and advanced persistent threats (APTs). This progression reflects the increasing sophistication and diversity of malware variants over time, underscoring the critical importance of robust detection and classification mechanisms in cybersecurity.

Initial Worms and Viruses [11]:

- The earliest forms of malware emerged in the 1970s and 1980s, primarily as experiments or proofs of concept. Examples include the Creeper virus, one of the first documented instances of malware, which spread through early computer networks like ARPANET.
- In the 1980s, viruses such as the Brain virus and the Morris worm gained notoriety for their ability to infect and propagate across computers, causing disruptions and drawing attention to the emerging threat of malware.

Proliferation of Computer Viruses:

- The 1990s witnessed a proliferation of computer viruses, driven by the widespread adoption of personal computers and the internet. Viruses like Michelangelo and Melissa demonstrated the potential for widespread damage and data loss, exploiting vulnerabilities in operating systems and applications [5].
- Virus authors began employing more sophisticated techniques such as polymorphic code and stealth mechanisms to evade detection and propagation, challenging the efficacy of traditional antivirus software.

Emergence of Worms and Trojans:

- The early 2000s saw a shift towards self-propagating malware, exemplified by worms like Code Red and SQL Slammer, which exploited vulnerabilities in network services to spread rapidly and cause widespread disruption.
- Trojans, disguised as legitimate software, became increasingly prevalent, enabling invaders to get illegitimate access to systems, launch targeted attacks, or steal sensitive information against individuals and organizations.

Rise of Ransomware and APTs:

- The 2010s marked the rise of ransomware as a prominent form of malware, leveraging encryption techniques to lock down systems and demand ransom payments for decryption. CryptoLocker, WannaCry, and NotPetya are all well-known examples of ransomware that caused broad problems and financial losses in many areas.
- Advanced persistent threats (APTs) [Advanced Persistent Threats (APT): evolution, anatomy, attribution] emerged as a sophisticated and stealthy form of malware, often attributed to nation-state actors or organized cybercriminal groups.

Diversity and Sophistication of Modern Malware:

- Today, malware variants exhibit a high degree of diversity and sophistication, ranging from traditional viruses and worms to fileless malware and supply chain attacks. Malware authors continuously innovate and adapt their tactics to evade detection and exploit vulnerabilities in software and systems.
- The increasing use of encryption, obfuscation, and anti-analysis techniques poses significant challenges for traditional detection methods, highlighting the need for advanced detection and classification mechanisms capable of identifying emerging threats in real-time.

In conclusion, the history of malware shows how cyber risks change over time and how important it is to have strong detection and classification systems in cybersecurity. Malware keeps getting smarter and more varied, so people who work in cybersecurity need to stay alert and take the initiative to create defenses that can adapt to new threats.

Certainly, here's the extended table 1.1 with more years included [3], Twenty-five years of cyber threats in the news:

Table 1.1: Malware types and their characteristics

Year	Malware Type(s)	Characteristics and Impact
1971	Creeper	It affected DEC PDP-10 computers that were using the TENEX OS and is thought to be the first computer virus. The message "I'm the creeper, catch me if you can!" was shown.
1981	Elk Cloner	It was one of the first viruses to infect a microcomputer. It spread through floppy files to Apple II computers and showed a funny poem.
1986	Brain	The first IBM PC-compatible virus, it spread via infected floppy disks and displayed the message "Welcome to the Dungeon".
1988	Morris Worm	One of the first internet worms, it infected UNIX-based systems and caused widespread congestion and disruption.
1991	Michelangelo	Triggered on March 6th, it infected DOS-based systems and overwrote the first 100 sectors of the hard disk, potentially causing data loss and system instability.
1999	Melissa	Spread via infected email attachments, causing email servers to overload and leading to widespread disruption. It could compromise personal and sensitive information.
2000	ILOVEYOU	It spread through email and attacked millions of computers around the world, damaging them badly and costing a lot of money. Other things it did were delete files, send copies of itself, and steal passwords.

2001	Code Red, Nimda	Code Red exploited vulnerabilities in Microsoft IIS web servers, while Nimda spread via multiple vectors, including email, websites, and network shares. It caused massive downtime and financial losses.
2003	Blaster (MSBlast)	It has taken advantage of a flaw in Microsoft Windows to spread quickly and hit important infrastructure with DDoS attacks. It messed up systems all over the world and caused crashes.
2004	Sasser	Exploited a vulnerability in Windows LSASS service to spread and caused widespread disruption, particularly in healthcare and transportation sectors. It crashed systems and networks.
2007	Storm Worm	Distributed spam emails and formed one of the largest botnets, demonstrating the power of botnets in cybercrime operations. It stole personal and financial information.
2010	Stuxnet	Targeted Iranian nuclear facilities, highlighting the potential for cyber-physical attacks and espionage. It sabotaged industrial systems and disrupted uranium enrichment processes.
2011	Duqu, Flame	Sophisticated espionage malware, linked to Stuxnet, targeting government and corporate networks. It stole sensitive data and provided remote access to compromised systems.
2013	CryptoLocker	Introduced ransomware as encrypting files, prominent threat, and demanding payment for decryption keys. It extorted money from individuals and organizations worldwide.
2014	Gameover Zeus, Heartbleed	Gameover Zeus operated as a banking Trojan, while Heartbleed exploited a critical vulnerability in

		OpenSSL, affecting millions of websites and systems. It stole financial and personal data.
2015	Angler Exploit Kit, Dyre	Angler Exploit Kit exploited vulnerabilities in web browsers, while Dyre targeted online banking systems and stole credentials. They facilitated widespread cybercrime and financial fraud.
2016	Mirai	Targeted IoT devices and formed botnets for DDoS attacks, leading to massive disruptions in internet services. It exploited weak or default credentials in IoT devices.
2017	WannaCry, NotPetya, Bad Rabbit	WannaCry exploited a Windows SMB vulnerability to spread globally; NotPetya caused widespread disruption, particularly in Ukraine; Bad Rabbit targeted Eastern European countries. They caused billions in damages and disrupted critical infrastructure.
2018	VPNFilter, Olympic Destroyer	VPNFilter infected routers and NAS devices, while Olympic Destroyer targeted Olympic organizations with destructive malware. They compromised network infrastructure and disrupted operations.
2019	Emotet, Ryuk	Emotet operated as a modular banking Trojan, while Ryuk targeted organizations with ransomware attacks. They caused financial losses and disrupted operations in various industries.
2020	COVID-19-themed Malware, SolarWinds Supply Chain Attack	Malicious actors exploited the COVID-19 pandemic with phishing campaigns and malware distribution; the SolarWinds supply chain attack targeted government and corporate networks. They compromised government agencies, corporations, and critical infrastructure, leading to data breaches and espionage.

This extended table provides a comprehensive overview of malware evolutions over a wider range of years, highlighting significant events and their impact on cybersecurity and society.

1.4 Anatomy of Malware

The anatomy of malware refers to its internal structure and components, which are designed to perform various malicious activities while evading detection and analysis. While specific malware variants may vary in complexity and functionality, they generally consist of several common components. Here's an overview of the typical anatomy of malware [18]:

a. **Propagation Mechanism:** Malware often includes mechanisms to propagate and spread to other systems or devices. This could mean taking advantage of flaws in software, networks, or protocols, or it could mean using social engineering methods like phishing emails, harmful links, or drive-by downloads.

b. **Payload:** The payload of malware contains the core malicious functionality that performs the intended attack or unauthorized activity. This may include code to steal sensitive information, encrypt files for ransom, launch denial-of-service attacks, or provide remote access to an attacker.

c. **Loader:** Many malware variants include a loader component responsible for initiating the execution of the malicious payload. The loader may be designed to evade detection by employing techniques such as code obfuscation, anti-debugging, or anti-analysis measures.

d. **Persistence Mechanism:** Malware often incorporates persistence mechanisms to ensure its longevity on an infected system. This may involve modifying system settings, creating registry entries, or installing startup scripts to ensure that the malware is executed automatically on every moment device boots up.

e. **Command and Control (C2) Communication:** Malware typically communicates to a remotely located command-and-control server functioned by the invader to receive instructions, exfiltrate stolen data, or download additional payloads. This communication may be encrypted or obfuscated to evade detection.

f. **Evasion Techniques:** Malware employs various evasion techniques to avoid detection by antivirus software, intrusion detection systems, and other security mechanisms. This may include polymorphism (changing its code signature), metamorphism (restructuring its code), or using rootkit techniques to hide its presence from security tools.

g. **Anti-Analysis Measures:** To thwart analysis by security researchers and malware analysts, malware may include anti-analysis measures such as code obfuscation, sandbox evasion, or the use of packers and crypters to encrypt or compress the malicious payload.

h. **Stealth Techniques:** Malware often uses sneaky methods to avoid being found by users and system admins. This may include hiding its presence in the file system, masquerading as legitimate processes or files, or disabling security features such as antivirus software and firewalls.

i. **Exfiltration Mechanisms:** If the goal of the malware is to steal sensitive information, it may include mechanisms to exfiltrate data from the infected system. This may involve uploading stolen data to remote servers, sending it via email, or transmitting it over covert channels.

j. **Self-Destruct Mechanism:** Some malware variants include a self-destruct mechanism to erase their presence from the infected system or render themselves inoperable to avoid detection or analysis by security researchers.

Understanding how malware works is important for cybersecurity experts who want to make effective defenses and response plans to stop and deal with attacks. By dissecting malware samples and analyzing their components, researchers may get visions into the tactics, methods, and procedures employed by attackers and develop better defences against evolving cyber threats.

1.5 Approaches to Malware Detection

Traditional approaches to malware detection encompass a lot of mechanisms aimed at identifying and mitigating malicious software threats. While these methods have been effective to some extent, they also have limitations, particularly in detecting novel and sophisticated malware variants. Here are some traditional approaches to malware detection:

1.5.1 Signature-Based Detection:

- Signature-based detection [19] makes use of predefined patterns or signatures of known malware to recognize malicious files or activities. Antivirus software, for example, maintains a database of malware signatures against which files are scanned during routine scans or when accessed.

- While signature-based detection is effective against known malware variants, it is unable to detect zero-day threats or malware with polymorphic or metamorphic characteristics that alter their signatures to evade detection.

1.5.2 Heuristic Analysis:

- Heuristic analysis [19] involves identifying potentially malicious behavior or attributes based on predetermined rules or heuristics. Instead of relying solely on known signatures, heuristic analysis looks for suspicious activities such as code injection, self-modification, or unusual network traffic patterns.

- While heuristic analysis can detect previously unknown malware variants, it may also generate false positives or miss sophisticated threats that employ evasion techniques to mimic legitimate behavior.

1.5.3 Behavioral Monitoring:

- Behavioral monitoring is the act of observing the actions of running programs or processes in order to identify abnormal or malicious activity [20]. This technique emphasizes activities such as alterations to the file system, revisions to the registry, establishment of network connections, and utilization of system resources.

- Behavioral monitoring can detect previously unknown malware based on its behavior, making it more effective against polymorphic and zero-day threats. However, it may also generate false positives and require significant computational resources to analyze system behavior in real-time.

1.5.4 Sandboxing:

- Sandboxing involves running potentially malicious files or programs in a well-ordered environment, known as a sandbox, in order to witness their conduct and analyze

their impact on the device. Sandboxes isolate the execution of suspicious code from the underlying operating system to prevent damage or compromise [21].

- Sandboxing can identify malware type by observing its behavior in safe surroundings, allowing security analysts to analyze its characteristics and determine its intent.

1.5.5 Static Analysis:

- Static analysis entails scrutinizing the code or structure of a file without actually running it in order to detect possible signs of malicious activity. This approach includes techniques such as file signature analysis, file hashing, and examination of metadata [21].

- Static analysis can quickly identify known malware based on static characteristics, making it useful for triaging and prioritizing suspicious files. However, it is less effective against polymorphic and obfuscated malware variants that alter their code to evade detection.

1.5.6 Network-Based Detection:

- Network-based detection entails the surveillance of network traffic to identify indications of malicious activity, such as recognized malware signatures, dubious patterns, or abnormal behavior. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are frequently employed for the examination of network flow of traffic and the prevention of probable security breaches [22], [23].

- Network-based detection can identify malware attempting to communicate with command-and-control servers, exploit vulnerabilities, or propagate across the network. However, it may be less effective against encrypted or obfuscated traffic and requires continuous monitoring and analysis to detect emerging threats.

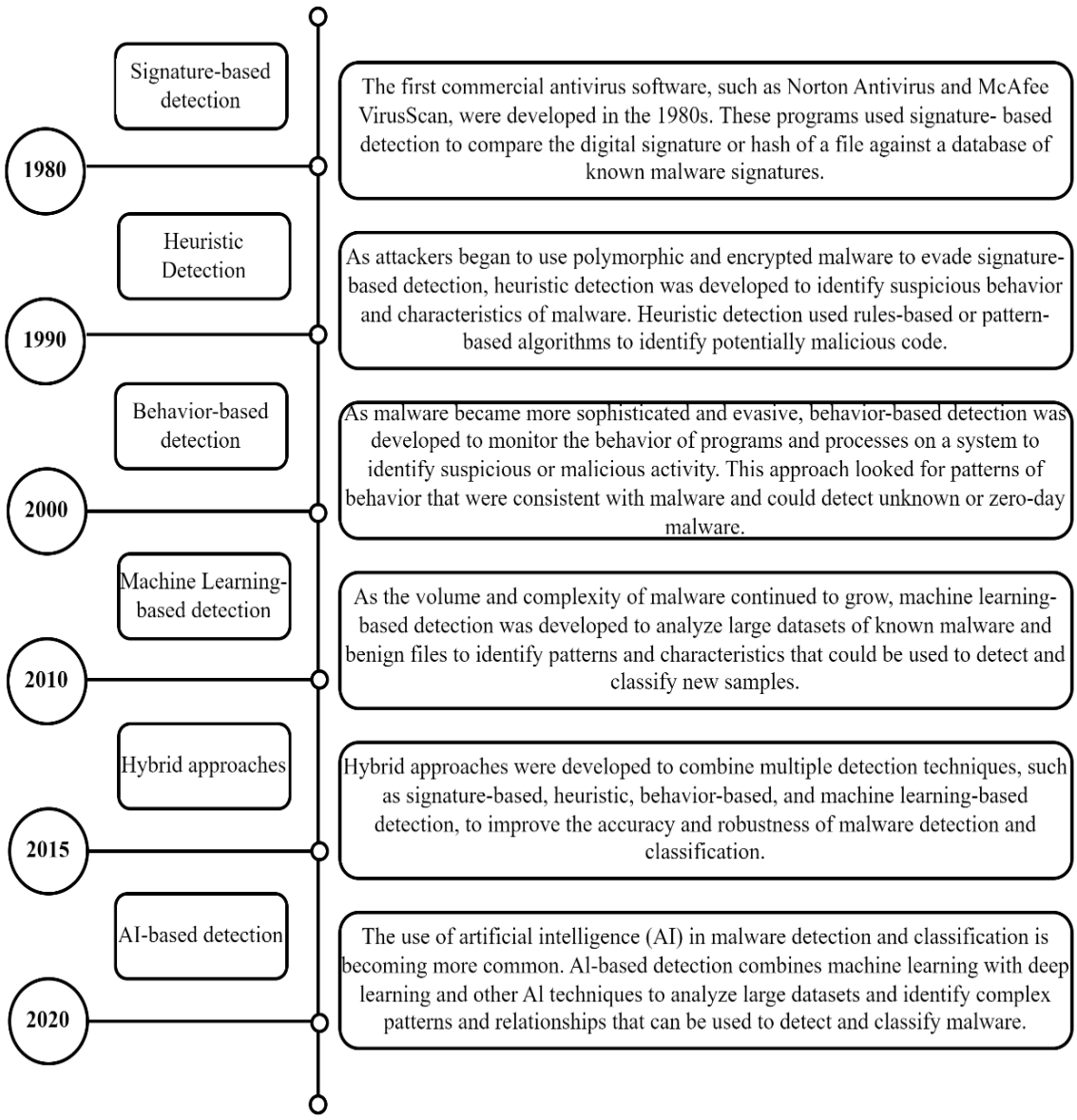

Figure 1.1: Historical Timelines for the Evolution of Malware Detection Techniques

While traditional approaches to malware detection have been instrumental in identifying and mitigating known threats, they have limitations in detecting novel and sophisticated malware variants that employ evasion techniques to evade detection. As cyber threats continue to evolve, there is a growing need for more advanced and adaptive detection mechanisms, such as machine learning-based approaches, to effectively combat the ever-changing landscape of malware. The timelines for evolution of malware detection techniques are shown in Figure 1.1.

1.6 Use of Machine Learning in Malware identification

Machine learning (ML) plays a crucial part in the identification and classification of malware. Conventional techniques that heavily depend on detecting specific patterns are not able to keep up with the fast-changing environment of malicious software. Machine learning offers dynamic and adaptable techniques that can identify new and unknown threats more effectively. Here's a summary of use of machine learning in malware identification and classification:

1.6.1 Feature Extraction

Feature extraction involves identifying characteristics of files that can be used to distinguish between benign and malicious software. These features can be static (derived from the code without executing it) or dynamic (observed during execution).

- **Static Features:** Static features are derived from the malware's code and binary structure without executing the file [23].

    A. **File Metadata:** These features include file metadata like file size, hashes, and timestamps.

        a. **Basic File Attributes**

            i. **File Name**: The name of the file, which might contain hints about its purpose or origin.

            ii. **File Path**: The directory path where the file is located, which can indicate how and where the malware was installed or executed.

            iii. **File Size**: The size of the file in bytes, which can be compared against known benign or malicious file sizes.

            iv. **File Type**: The type of the file (e.g., executable, document, image) determined by its extension or magic number.

        b. **File System Timestamps**

            i. **Creation Time**: The date and time when the file was created. This can be useful for identifying when the malware was first introduced to the system.

ii. **Modification Time**: The date and time when the file was last modified. Frequent modifications might indicate active malicious behavior.
iii. **Access Time**: The date and time when the file was last accessed. Unusual access patterns can be a sign of malware.
iv. **Executable File Metadata (Specific to Executables)**

c. **Headers**: Metadata in the file headers such as the PE (Portable Executable) header in Windows files, which includes:
    i. **Entry Point**: The address where the execution starts, which can be analyzed for unusual patterns.
    ii. **Sections**: Details about different sections of the executable (e.g., .text, .data), which can indicate the structure and nature of the executable.
    iii. **Import/Export Tables**: Lists of functions and libraries the executable imports or exports, revealing its dependencies and potential functionality.

d. **Digital Signatures**
    i. **Publisher Information**: Metadata about the publisher of the file, obtained from digital signatures. Verified signatures from trusted publishers indicate benign software, while unverified or suspicious publishers suggest potential malware.
    ii. **Certificate Information**: Details about the digital certificate used to sign the file, including issuer and validity dates.

e. **File Hashes**
    i. **Checksums**: The file is uniquely identified by cryptographic hash values such as MD5, SHA-1, and SHA-256. These are used to quickly compare files against known malware databases.

f. **Permissions and Ownership**
    i. **File Permissions**: Metadata indicating who can read, write, or execute the file. Unusual permissions can suggest malicious intent.

ii. **File Owner**: The user or process that owns the file, which can provide context about how the file was created or modified.

g. **Embedded Metadata**

i. **Embedded Information**: Metadata embedded within files, such as author names, software version, and other details, commonly found in document or media files.

The comparison of file metadata against known good and bad profiles helps identify anomalies indicative of malware. The patterns in metadata (e.g., unusual creation times, suspicious file sizes) thus recognized aids in detecting malicious files.

The metadata provides clues about the file's intended use and behavior, such as unusual execution paths or unexpected modifications. Also, the metadata such as digital signatures and publisher information can help attribute the file to a known entity, aiding in threat intelligence. However, there are certain challenges because of which we can't simply rely on finding benign and malicious files. Such challenges are mentioned below.

Challenges

- **Evasion**: Malware authors can manipulate metadata to evade detection, such as modifying timestamps or using fake digital signatures.
- **Inconsistencies**: Variations in metadata due to legitimate reasons (e.g., software updates) can complicate analysis, requiring sophisticated techniques to distinguish between benign and malicious changes.

B. **Opcode Sequences:** An opcode, also known as an operation code, is the segment of a machine language instruction that precisely indicates the operation that needs to be executed [24]. In a binary executable, each instruction to the CPU is represented by an opcode, which is part of the machine code. An opcode sequence is a series of opcodes extracted from an executable file. This sequence reflects the flow of operations that the program performs, providing a "signature" of the program's behavior.

Extraction Process

a. **Disassembly**:
    i. The first step in extracting opcode sequences is disassembling the binary executable.
    ii. Tools like IDA Pro, Radare2, and Ghidra convert the binary code into assembly code, from which the opcodes can be extracted.

b. **Sequence Generation**:
    i. Once the binary is disassembled, the opcodes are extracted and arranged in the order they appear in the code.
    ii. This results in a sequence of opcodes that represents the execution flow of the program.

Representation of Opcode Sequences: Typically, opcodes are expressed as N-grams. A successive sequence of n elements extracted from a provided text or audio sample is called as an N-gram. An n-gram in the context of opcodes denotes to an arrangement of n consecutive opcodes.

Example: For a sequence of opcodes [MOV, ADD, SUB, JMP], the 2-grams would be [MOV ADD], [ADD SUB], [SUB JMP].

During malware identification process, the sequences or patterns of opcodes are examined and analysed for malicious activity.

Opcode sequences can reveal patterns in the execution flow that are characteristic of malicious behavior. For instance, certain sequences may indicate operations like privilege escalation, keylogging, or network communication.

a. **Code Obfuscation**:

- Malware often uses obfuscation techniques to avoid detection. Opcode sequences can sometimes reveal hidden patterns even in obfuscated code.

b. **Signature Generation**:

- Opcode sequences can be utilized to create signatures for recognized malware, which can subsequently be employed to identify like risks in subsequent instances.

Opcode Sequences [25]**:** Opcode sequences are used to create feature that act like input for ML models. These vectors can be formed by considering the presence, frequency, and patterns of opcodes. Machine learning algorithms (Supervised learning and Unsupervised learning) can be used to identify unusual opcode patterns of malware. Supervised Learning: Models like Support Vector Machines (SVM), Decision Trees, Random Forests, and Neural Networks can be trained using labeled datasets of opcode sequences from benign and malicious binaries. Unsupervised Learning: Techniques like clustering and anomaly detection can be used to identify unusual opcode patterns indicative of new or unknown malware. In this process, the data is collected in the form of binaries of both malware and benign software. Then, a disassembler is used to convert binaries into opcode sequences followed by feature extraction in the form of n-grams, frequency vector or sequence patterns. This is followed by training the machine learning models and testing the model using standard metrics.

Challenges

1. **Code Obfuscation**:
   - Malware can use sophisticated obfuscation techniques to alter opcode sequences and evade detection.
   - Models must be robust to variations introduced by obfuscation.
2. **High Dimensionality**:
   - Opcode sequences can be very long, leading to high-dimensional feature spaces.
3. **Execution Context**:
   - The same opcode sequence can have different implications based on the execution context.

- Incorporating additional contextual features can improve detection accuracy.

C. **API Calls:** API calls, short for Application Programming Interface calls, are requests made by a program to the operating system or other software libraries to perform specific tasks. API calls are essential as they reveal the interactions between the malware and the system, providing insights into the malware's behavior. Here's a detailed explanation of API calls and their significance in feature extraction for malware identification:

- **System APIs**: These interact with the operating system, such as Windows API, POSIX for Unix-based systems, etc.
- **Library APIs**: These interact with third-party libraries and frameworks.

Role of API Calls in Malware Analysis

API calls are critical in understanding what actions a piece of software performs. By monitoring and analyzing these calls, one can infer the intentions and behavior of the software, distinguishing between benign and malicious activities.

Types of API Calls Analyzed

1. **File System Operations**: Malware might use the calls **CreateFile**, **ReadFile**, **WriteFile** to create, read, and write files respectively.
2. **Network Operations**: Malware might use the calls **InternetOpen**, **InternetConnect**, **HttpSendRequest** to initiate and manage network connections.
3. **Registry Operations**: Malware might use the calls **RegCreateKey**, **RegSetValue**, **RegDeleteKey** to manipulate the system registry and to achieve persistence or modify system configurations.
4. **Process and Memory Operations**: Malware might use the calls **CreateProcess**, **OpenProcess**, **VirtualAlloc** to create and manage

processes and memory allocation. This further may lead to launch additional malicious processes or inject code into other processes.

5. **System Information**: Malware might use the calls **GetSystemInfo**, **GetUserName** to retrieve system and user information and about the environment it is running in.

Challenges

1. **API Call Evasion**:
    - Advanced malware might use techniques to hide or obfuscate API calls, making it harder to detect.
    - Examples include direct system call invocation, dynamic API resolution, or encryption of strings.
2. **High Dimensionality**:
    - API call sequences can be very long and complex, leading to high-dimensional feature spaces.
    - Efficient feature selection and dimensionality reduction techniques are crucial.
3. **Behavior Variability**:
    - The same API call pattern might have different implications depending on the context.
    - Combining API calls with other features (e.g., system state, file metadata) can improve detection accuracy.

D. **Strings:** Another feature that is looked upon is searching for suspicious strings within the binary (e.g., URLs, IP addresses). Searching for suspicious strings within a binary file is a common technique in malware detection and analysis. Here's how this process generally works:

i. **String Extraction**: First, you extract all ASCII and Unicode strings from the binary file. These strings could include function names, URLs, registry keys, API calls, or any other identifiable text data embedded within the binary.

ii. **Filtering**: Filter out known benign strings or noise that are unlikely to be indicators of malicious activity. This step helps reduce false positives and focus on potentially suspicious strings.

iii. **Pattern Matching**: Use regular expressions or specific string patterns to search for known malicious indicators. These patterns could include - Command and Control (C&C) server URLs or IP addresses, Registry keys associated with persistence mechanisms API calls known to be used by malware (e.g., for file manipulation, network communication) Encryption or obfuscation routines (e.g., base64 strings) Strings related to system exploits or vulnerabilities.

iv. **Contextual Analysis**: Consider the context in which strings appear within the binary. For example, strings that are obfuscated or encrypted might indicate attempts to hide malicious behavior. Also, the combination of multiple suspicious strings or their proximity within the binary can strengthen the suspicion of malicious intent.

v. **Behavioral Analysis**: While string analysis is valuable, it's essential to combine it with behavioral analysis to understand the actual impact and behavior of the binary when executed. Dynamic analysis in a controlled environment (sandbox) can reveal runtime behaviors that static analysis might miss.

vi. **Manual Review**: In complex cases, manual review by security analysts is crucial. Analysts can identify subtle indicators or behaviors that automated tools might overlook.

vii. **Threat Intelligence**: Compare identified strings against known databases of malicious indicators and threat intelligence feeds to see if any matches exist, indicating a known malware family or variant.

viii. **Reporting and Action**: Finally, report findings and take appropriate action, which could include quarantining the binary, investigating further, or applying mitigation strategies.

- **Dynamic Features:** Dynamic feature monitoring in malware detection entails the observation and analysis of the runtime behavior of a program or binary in order to identify possibly dangerous actions. In contrast to static analysis, which involves the

examination of the code or binary without executing it, dynamic analysis involves the execution of the program in a controlled environment (sandbox) and the observation of its behaviors as it executes. Dynamic feature monitoring operates within the framework of malware detection to discover and analyze potential threats:

1. **Execution Environment**: The binary or software that is suspected of being malware is run in a precise environment like a sandbox or virtual machine. This environment enables analysts to observe the behavior of the system without incurring any potential harm to the actual system.
2. **Behavioral Monitoring**: During execution, various system-level and application-level activities are monitored in real-time. These activities may include monitoring file system operations like file creation, modification, deletion, changes being done in registry at run time, monitoring network connections, data transfers and communication protocols, monitoring process or thread creations, system and application programming interface calls made by the program or threads.
3. **Feature Extraction**: Specific features of interest are extracted from the monitored behavior. These features can include **API Call Sequences**, **Network Traffic**, **File System Activity**, **Process Behavior**, **Registry Changes** [23].
4. **Anomaly Detection**: The extracted features are analyzed for deviations from expected or normal behavior. This involves comparing observed behavior against known good behavior (baseline) or established patterns of malicious behavior (signatures or heuristics).
5. **Machine Learning and Pattern Recognition**: ML algorithms may be utilized to examine the data retrieved and identify intricate patterns that are characteristic of malware. This includes supervised learning (using labeled datasets) or unsupervised learning (clustering or anomaly detection).
6. **Behavioral Signatures**: Based on observed behaviors, behavioral signatures or profiles can be created. These signatures help in identifying similar malicious behaviors in future instances or variants of malware.

7. **Alerting and Reporting**: If suspicious or malicious behavior is detected, alerts are generated for further investigation by security analysts. Reports detailing the observed behavior and potential threat are prepared for remediation steps.

Dynamic feature monitoring is essential in modern malware detection as it allows for the detection of polymorphic and evasive malware that can bypass traditional signature-based detection methods. By focusing on behavior rather than static attributes, dynamic analysis provides a more robust approach to identifying and mitigating advanced threats.

1.6.2 Data Preprocessing

After extracting the characteristics, the data undergoes pre-processing to make it compatible with Machine learning algorithms. This involves normalizing the data, converting the data using some standardization, handling missing values, refining the data doing dimensionality reduction.

1.6.3 Model Training

Combining machine learning with traditional methods can enhance detection rates. For instance, using ML for initial screening and traditional methods for deeper analysis can provide a balanced approach.

Better malware detection and further classification can be performed using machine learning tools can be developed that can adapt to new threats, handle large amounts of data, and find minor trends that point to bad behavior. The continuous evolution of ML techniques ensures that cybersecurity measures stay one step ahead in the ongoing battle against malware. These methods can be used individually or in combination to detect and classify malware effectively, subject to the specific necessities and characteristics of the malware threat landscape. By leveraging a variety of detection techniques, users can improve their ability to defend against evolving malware threats and protect their systems and data from malicious attacks.

Machine learning algorithms, both supervised and unsupervised, play crucial roles in malware detection and their classification by enhancing the accuracy and efficiency of identifying malicious software.

A. Supervised Machine Learning: Supervised ML algorithms are trained using labeled datasets, where each data point is tagged with the correct output. In the context of malware detection and classification, these labels indicate whether a file is benign or malicious and, in the case of classification, the specific type of malware [26].

Key Algorithms:

1. **Decision Trees and Random Forests**: These algorithms create models that predict the class of a file based on its features. Decision trees are simple and interpretable, while random forests, an ensemble of decision trees that will reduce overfitting and improve accuracy
2. **Support Vector Machines (SVMs)**: SVMs are active in high-dimensional spaces and can classify files by finding the hyperplane that best separates benign from malicious files.
3. **Neural Networks and Deep Learning**: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) possess significant computational capabilities. CNNs are capable of analyzing byte sequences or file structures, whereas RNNs excel in processing sequential data, such as API call sequences [27].
4. **Logistic Regression**: This is a numerical model used for binary ordering, providing a probability score for a file being malicious.
5. **K-Nearest Neighbors (KNN)**: This algorithm employs a classification technique that categorizes files by determining their proximity to the nearest training samples in the feature space. This approach is straightforward and highly efficient for certain jobs [28].

B. Unsupervised Machine Learning: Unsupervised learning algorithms do not require labeled data. Instead, they find patterns and structures within the data to detect anomalies or cluster similar data points [29].

Key Algorithms:

1. **Clustering (e.g., K-Means, DBSCAN)**: These algorithms group files into clusters based on feature similarity. Malicious files often cluster differently from benign ones.
2. **Anomaly Detection (e.g., Isolation Forests, One-Class SVMs)**: These methods identify files that deviate significantly from the norm, which can indicate new or unknown malware.
3. **Autoencoders**: These neural networks learn to compress data and then reconstruct it. Files that reconstruct poorly compared to the training data (typically benign files) are flagged as anomalies [30].

Figure 1.2 shows the various techniques of ML for detecting and further classifying malware.

1.6.4 Evaluating the model

Assessing the effectiveness of the ML models is of utmost importance. Precision, recall, accuracy, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are commonly employed metrics in data analysis and machine learning. Utilizing cross-validation and testing on data that has not been previously seen aids in guaranteeing the model's resilience and capacity to apply to new data. After undergoing training and validation, models are implemented in real-world settings where they consistently assess incoming files and network traffic. Regular and ongoing monitoring and upgrading of models are essential in order to adjust to emerging malware strategies and approaches.

1.7 Problem Formulation

Malware identification and classification is a challenging task when limited features are used for the same. For better identification and classification multimodal approaches

are required that are going to considered more factors using some machine learning concepts.

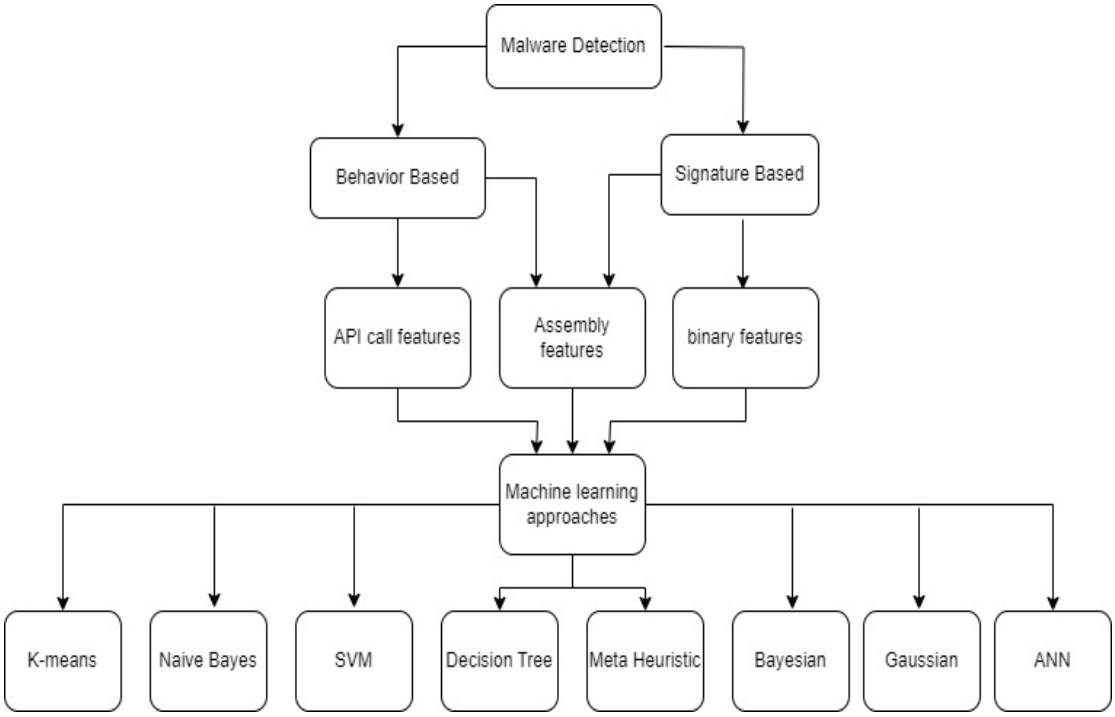

Figure 1.2: Malware identification and classification approaches [31]

Attackers are generating polymorphic malware which is generated by combining the features of metamorphic malware that usually change their features and these are not even detectable by anti-virus software. So there is a need for efficient malware detection schemes to take care of this malware which is part of the next generation family

Conventional methods of detecting malware based on signatures are ineffective in identifying zero-day malware or malware that has not been previously identified. The other method, behavior-based is efficient but lacks the ability to accurately define the specific behaviors that a system should exhibit. Hence, the need of approach that will consider the unknown malware is required.

Currently, there is no one method that is capable of identifying all categories of malicious software. Furthermore, the task of real-time monitoring poses significant challenges, since it primarily relies on data sets for analysis, which proves to be less effective in obtaining genuine insights. Therefore, it is currently necessary to have an

effective method for detecting a wide range of malware varieties and their methods of infiltration. The core purpose of this research work is to design a framework based on some novel malware identification and classification technique especially for the new malware that is arousing day by day. It requires proper background knowledge and an in-depth analysis of various existing techniques.

Despite substantial advancements in cybersecurity study and technology, malware remains an enduring and dynamic menace, requiring continuous research endeavors to comprehend its fundamental principles, mechanisms, and behaviors. A comprehensive understanding of malware is vital for designing and implementing effective defense policies, enhancing incident response capabilities, and alleviating the impact of cyber-attacks on individuals, organizations, and society as a whole.

1.8 Objectives

1. To study and analyze the existing malware identification and classification techniques.
2. To propose a novel approach for malware identification and classification.
3. To design a framework for the proposed approach for malware analysis.
4. To validate and compare the proposed approach.

Chapter 2

Review of Literature

The predominant methods rely on signatures but they prove to be inadequate when confronted with novel and unidentified types of malware. Machine learning approaches necessitate a substantial quantity of data, together with expert-level expertise, to construct precise models. Table 2.1 presents a comprehensive overview of research articles from 2019 to 2023 that have utilized malware detection approaches to address cyber security challenges in the field of information technology.

Table 2.1: Research Evaluation Matrix for Malware Detection Techniques

Research Reference	Signature-Based Detection	Heuristic Detection	Behavior Based Detection	Hybrid Approaches	ML/AI-Based Detection
[32]	✓	×	✓	✓	×
[33]	✓	×	×	×	×
[34]	×	✓	×	×	×
[35]	×	×	✓	×	✓
[36]	×	×	×	×	✓
[37]	×	×	×	×	✓
[38]	×	×	×	×	✓
[39]	×	×	×	✓	×
[40]	✓	×	✓	✓	×

The following Table 2.2 shows the work done in the same area of identification and classification of malware.

Table 2.2: Similar Work Done

Reference	Year
Nataraj et al. [41]	2011
Cui et al. [41], [42]	2018
Vinayakumar et al. [43]	2019
Abualhaj et al. [44]	2024
Wang et al. [45]	2023
Louk et al. [46]	2022

The study [26] explored ensemble approaches to find malware, focusing particularly on recognizing novel and unfamiliar malware that traditional methods struggle to detect accurately. The procedure involved two stages: feature extraction and classification using machine learning techniques. The study employed a stacked ensemble of fully-connected and one-dimensional convolutional neural networks (CNNs) to carry out the first round of classification. Subsequently, a machine learning technique was utilized for the ultimate phase of categorization. The meta-learner analysis entailed a direct comparison of 15 machine learning classifiers. The most effective outcome was obtained by combining seven neural networks and using the ExtraTrees classifier as the final classifier. The experiments employed the Windows Portable Executable (PE) malware dataset, and the results highlighted the effectiveness of the ensemble learning method in detecting malware. This strategy significantly enhanced the detection of novel and unfamiliar malware. The utilization of neural networks and machine learning classifiers has shown promising results in improving the ability to detect malware. The study's shortcomings included the need for further evaluation on diverse malware datasets to ascertain the suitability of the proposed technique. Moreover, there is the possibility of further research and enhancement in terms of increasing the scalability and real-time usability of the ensemble learning-based technique.

The publication [44] discussed the MW-KNN model, which leveraged the KNN method for effective malware detection, addressing the urgent need for accurate identification and classification of dangerous software. The study emphasized the importance of transformation and normalization of data for classification algorithms, focusing on execution-level identification due to the rise of polymorphic and

metamorphic malware. It introduced a two-stage process using random projections with the KNN method to enhance malware detection, demonstrating that increasing the projected vectors' dimensions improved outcomes by reducing unpredictable samples and false positives. Sahin et al. highlighted the increasing malware incidence on Android devices and proposed a permission weight strategy, showing superior outcomes compared to previous strategies. The study evaluated the MW-KNN model using the CIC-MalMem-2022 dataset, showcasing improvements in accuracy and F-score metrics with the KNN and NB methods, with the KNN method excelling in accuracy and Gaussian NB in the F-score metric. The publication contributed to the topic of malware detection by introducing innovative strategies like permission weight mechanisms and random projections with the KNN method, enhancing detection accuracy and performance on datasets like CIC-MalMem-2022. However, the research paper did not explicitly discuss the limitations of the MW-KNN model or the proposed strategies, leaving a gap in understanding the potential constraints or challenges faced in real-world implementations.

The literature review [45] presented a comprehensive examination of NLP and API sequence-based techniques for detecting and classifying malware, as well as Transformer-based approaches and training procedures. The research article presented the TTDAT (Two-step Training Dual Attention Transformer) as a solution for classifying malware. The focus of the paper was on analyzing API call sequences to tackle problems such as information loss and computational overhead. The proposed model achieved an average F1 score of 0.90 and an accuracy of 0.96. The research paper contributed by introducing a novel approach, TTDAT, for malware classification that leveraged Transformer architecture and dual attention mechanisms to streamline API sequence encoding and improve detection efficiency. One potential gap in research could have been the detailed discussion on the specific datasets used for training and testing TTDAT, which could have provided insights into the generalizability of the model. The limitations of the publication might have included the lack of in-depth analysis on the scalability of TTDAT to larger and more diverse malware datasets, as well as the computational resources required for training and inference.

The publication [46] compared various tree-based ensemble learning methods used in the analysis of PE malware and found that all tree-based ensembles performed well, with no statistically significant performance differences between algorithms when hyperparameters were properly configured. The publication contributed to the topic of PE malware analysis by highlighting the effectiveness of tree-based ensemble learning methods and their superior performance compared to other detectors. It emphasized the importance of hyperparameter tuning for optimal results and generalizability across different datasets. The limitations of the study included the focus on tree-based ensemble methods, potentially excluding other machine learning approaches, and the reliance on specific public datasets for evaluation, which might not cover the entire spectrum of PE malware characteristics.

The publication [47] examined intrusion risks and Distributed Denial of Service (DDoS) assaults in the Internet of Things (IoT). It proposed employing a sparse convolute network as a defence against these threats and attacks. The authors also explored the process of optimizing the network by employing evolutionary algorithms to find and recognize regular, error, and intrusion efforts in various scenarios. The main points of the publication included the utilization of machine learning techniques like Passban intelligent intrusion detection system, genetic optimized deep belief network, and other methods to detect and prevent intrusion activities in the IoT. It also emphasized the importance of network intrusion detection systems in providing network security. A gap in research could have been the need for further exploration of real-time threat detection mechanisms and the development of more advanced intrusion detection systems to address evolving cybersecurity challenges in the IoT environment. The publication contributed significantly to the topic of IoT security by proposing various intrusion detection models and techniques with high detection rates and accuracy. The model was able to achieve detection rate of 98.98% and accuracy of 99.29%, demonstrating the effectiveness of the proposed system. The dataset used in the research included UNSW-NB15 and NSL-KDD datasets for training and evaluating the intrusion detection models. The results showed a high detection rate, accuracy, and minimal processing complexity, with a performance ratio of 90.26%. The limitations of the publication might have included the focus on specific types of attacks and

datasets, which might not cover the full spectrum of potential threats in the IoT environment. Additionally, the proposed models might have required further validation and testing in real-world scenarios to assess their practical applicability and scalability.

The publication [48] contributed to the domain of malware detection by emphasizing lightweight models like ANNs, SVMs, and GBMs. The study's assessment of the ANN design demonstrated a classification accuracy of more than 94% in distinguishing between malware and genuine programs. This was achieved by reducing the number of parameters by 40 compared to other ANNs, while still ensuring correct generalization. The SVM and GBM architectures proposed in the study, although less effective than the ANN architecture, offered valuable insights into machine learning behavior for malware classification. The study's limitations included the focus on memory-optimized machine learning solutions and the specific evaluation of ANNs, SVMs, and GBMs, potentially leaving out other machine learning models or techniques that could contribute to malware detection.

The publication [49] focused on the harmful impacts of Ransomware, which encrypts user files to prevent access to infected systems. It emphasized the need to understand the vulnerabilities prevalent in OT systems that enable such attacks, highlighting the importance of Availability, Integrity, and Confidentiality in these systems. Patches had to undergo extensive testing and approval before being implemented in the ICSOT network. The main points of the publication revolved around the impact of Ransomware, the distinctions between IT and OT setups, and the critical aspects of safety, system integrity, and network diagram confidentiality in protecting against cyber threats. A gap in research could have been the need for further exploration into specific strategies or technologies that could effectively mitigate the risks posed by Ransomware attacks on OT systems. The publication contributed by shedding light on the vulnerabilities in OT systems, emphasizing the importance of security measures and thorough testing of patches to safeguard critical infrastructure. The dataset used in the publication was not explicitly mentioned in the provided contexts. The results on the dataset and accuracy of the findings were not specified in the given information. One limitation of the publication could have been the lack of detailed insights into specific case studies or real-world examples to illustrate the concepts discussed.

The publication [50] proposed a novel approach using the BiTCN and SFCWGAN methods. It involved the extraction of features from malware Opcode and API sequences, using Word2Vec for representation, and combining Spearman correlation coefficient and WOA-XGBoost algorithm for feature selection and simplification. The proposed method involved generating malware samples using CWGAN to supplement the imbalanced malware family dataset, enhancing the training process on BiTCN. The model demonstrated accuracies of 99.56% and 96.93% on the Kaggle and DataCon datasets, respectively, surpassing other approaches by 0.18% and 2.98%. However, the study acknowledged limitations such as the need for further improvement in accuracy and bias reduction, especially on the DataCon dataset, suggesting room for enhancement in minimizing bias and improving detection accuracy.

The publication [51]proposed a rapid binary visualization technique employing Fuzzy Set theory and the H-indexing space filling curve to overcome constraints in intrusion detection systems (IDS). The main points of the publication included the development of a signature-free IDS, testing the methodology on 5000 malicious and benign files. The result showed that the model has accuracy rate of 91.94%, precision of 90.63%, recall of 92.7%, and an F-score of 91.61% on average. Gaps in research included the need for further optimization and tuning of the proposed methodology, as well as the potential for exploring larger datasets and real-world applications to validate the system's performance in diverse environments. The publication contributed to the topic of malware detection by introducing a novel method that leveraged binary visualisation and fuzzy sets, achieving high accuracy rates, and demonstrating potential for a signature-free IDS. The limitations of the publication included the lack of optimal tuning in the methodology, the need for further improvements in computation time, and the necessity for validation in real-world scenarios to assess the system's robustness and scalability.

The literature [52] discussed the use of different classifiers such as ELM (Extreme Learning Machine), SVM (Support Vector Machine), K-nearest neighbour and others in the classification scenarios. The paper highlighted the shift towards deep learning approaches like Convolutional Neural Networks (CNN) for malware classification, showcasing the application of unique CNN models and architectures. Various

researchers had proposed innovative MC methods combining different deep CNN models like AlexNet, VGG16, ResNet50, and others, along with machine learning classifiers like Softmax, Multiclass SVMs, and more. The research paper also mentioned the use of ensemble classifiers, sequential multilayered Random Forest ensemble techniques, and machine learning approaches like Random forests, Xgboost, Extra trees classifier, and Logistic regression in malware classification. The proposed malware classification method in the paper addressed the challenges of low classification rates in existing techniques by achieving high accuracy rates, outperforming other methods with an accuracy rate of 95.42% and 96.84%. The research paper contributed significantly to the topic of malware classification by presenting a robust method that involved dataset preparation, feature extraction, and classification steps, leading to highly accurate results on the Malimg dataset. However, a potential gap in research could be the need for further exploration of the scalability and generalizability of the proposed method across different malware families and datasets, which could be a limitation of the study.

The publication titled [53] addressed the topic of detecting malware intrusions by utilizing machine learning techniques and analyzing system interactions. The paper introduced a risk-based system-call sequence aggregation approach that assigned riskiness values based on the risk value of the function. This method outperformed previous findings by achieving improved classification accuracy when utilizing SVM and DT methods. The research emphasized the importance of increasing malware detection accuracy while utilizing lightweight machine learning methods for practical applications. The paper proposed a risk-based system-call grouping strategy that effectively utilized lightweight machine learning techniques for detecting malware attacks, achieving accuracy levels comparable to deep learning methods. The dataset used in the study was not explicitly mentioned in the provided contexts. The suggested risk-based system-call grouping strategy yielded a 23.4% increase in classification accuracy with the SVM method and a 7.6% increase with the DT method, compared to earlier findings.

The publication [54] specifically addressed the identification of FDI (False Data Injection) attacks in smart grids through application of machine learning

methodologies. It discussed the application of supervised learning and hybrid methods to improve the performance of classification algorithms in detecting FDI assaults. The study utilized a dataset to evaluate the effectiveness of various technologies in identifying threats accurately. The gaps in research could include further exploration of advanced machine learning models or feature selection techniques to enhance detection accuracy and efficiency. The publication's contribution lay in improving the performance of classification algorithms for FDI attack detection using supervised learning and hybrid methods, which could be crucial for enhancing smart grid security. The accuracy of the detection strategies was evaluated based on how effectively they could identify different types of threats in the smart grid environment. Limitations of the study included the need for real-world implementation and validation of the proposed detection strategies to assess their practical applicability in smart grid security scenarios.

The publication [55] focused on enhancing malware detection accuracy using machine learning algorithms like K-Nearest Neighbours, Decision Tree, Logistic Regression, and Random Forest. The study aimed to extract the best feature selection to improve the detection of polymorphic malware. The main points of the publication included the use of machine learning algorithms to address the rising number of malware threats, the importance of feature selection in improving detection accuracy, and the comparison of different classifiers in detecting polymorphic malware. Additionally, there was a focus on reducing false positive and false negative rates to improve overall detection accuracy. The publication contributed significantly to the topic of malware detection by showcasing the effectiveness of the Random Forest algorithm with a detection accuracy of 99% on a relatively small dataset. It emphasized the importance of feature selection in enhancing the performance of polymorphic malware detection. The dataset used in the study included unknown malware (0), known malware (1), and polymorphic malware (2). The results on the dataset showed that the Decision Tree has accuracy of 93%, K-Nearest Neighbours has 94%, and Logistic Regression has an accuracy of 88%. The limitations of the publication included the focus on a relatively small dataset, which might not fully represent the complexity of real-world malware scenarios. Additionally,

the study acknowledged the importance of further research to address evolving malware threats and improve detection accuracy on larger datasets.

The research paper [56] aimed to develop a lightweight malware detection method that was multiclass and capable of identifying recent malware. This method was designed to be executed in IoT devices, with a particular emphasis on smart city applications. The processing model that is well-suited for implementation in IoT devices is a concise and expedient solution that combines the feature-learning capabilities of convolutional neural networks with the ability of bidirectional long short-term memory to describe temporal information. It introduced a robust and resource-efficient detection algorithm that outperformed other machine learning-based models in detecting obfuscated malware and identifying specific attack types. The research paper used the CIC-Malmem-2022 OMM dataset for extensive experiments. The results indicated that the method excelled in detecting OMM and identifying specific attack types, showcasing its effectiveness in defending against obfuscated malware. The proposed method achieved 84.56% detection accuracy with the RobustCBL model and 84.22% with the CompactCBL model, outperforming existing works in the field. Despite being smaller in size, the CompactCBL model performed remarkably well showcasing that the proposed approach is quite efficient. One limitation of the study was that it focused on a specific dataset (CIC-Malmem-2022 OMM dataset), which might limit the generalizability of the proposed method to other datasets or scenarios.

The publication [57] presented a novel approach for obfuscated malware detection in IoT Android applications using Markov images and CNN models. The main points of the publication included the rise of Android malware, the need for improved detection methods due to obfuscation techniques, and the effectiveness of CNN models trained on Markov images for malware detection and classification. A gap in research could have been the need for further exploration of the scalability and real-world application of the proposed system beyond the experimental setup with 12,000 Android applications. The publication contributed to the topic by demonstrating high accuracy rates in malware detection and classification: 99.41% for distinguishing malware from benign apps, 99.65% for identifying obfuscated malware, 99.81% for distinguishing obfuscated from non-obfuscated malware, and 99.67% for classifying obfuscated

malware into 14 categories. The limitations of the publication may have included the need for further validation in real-world IoT environments, potential challenges in scaling the system, and the generalizability of the results to diverse IoT systems beyond the Android platform.

The research paper [58] proposed the model's ability to automatically learn representations of network flow graphs that achieved high accuracy in detection. One limitation of the publication could have been the lack of detailed information on the dataset used, which could impact the reproducibility and generalizability of the results. Additionally, the paper did not delve into the explainability of model decisions, which could have been a valuable aspect to explore in future research.

The research paper [59] focused on image-based malware classification approaches. The authors concentrated on developing a streamlined ensemble architecture. This was accomplished by integrating a customized MLP-mixer with an Autoencoder to amplify the characteristics r utilizing the encoder-decoder structure of the autoencoder. Gaps in research could have included the need for further exploration of the optimal preprocessing techniques for image-based malware tasks. The publication contributed to the topic by introducing a novel lightweight ensemble architecture that outperformed other cutting-edge techniques in malware classification, utilizing fewer parameters compared to traditional models while achieving high performance through various experiments. The datasets used in the study were the Malimg dataset containing 9939 samples from 25 malware families and the Malheur dataset with 3133 variant samples from 24 malware families. The results indicated that ensemble method outperformed CNN-free models and outperformed a variety of conventional pure CNN models, thereby illustrating the efficacy of the approach in malware classification tasks. The accuracy of the proposed ensemble architecture was highlighted through experimental results, showcasing its superiority over other models and its ability to achieve high performance in malware classification tasks. However, the scalability and generalizability of the architecture require additional validation on a broader range of datasets.

The publication [60] focused on the development of a distinctive approach to malware classification that employed convolutional neural networks and dual attention. The

research paper highlighted gaps in existing malware detection techniques and emphasized the requirement for more accurate malware detection methods to combat modern automated malware creation methods. The publication's contribution to the topic lay in introducing a different approach to malware classification using ML frameworks, showcasing exceptional performance in malware detection and classification. The dataset used for evaluation included the benchmark dataset where the proposed model achieved accuracy rates of 98.14% and 98.95%, respectively. The limitations of the study included the need for further validation on diverse datasets, potential challenges in real-world implementation, and the necessity for continuous updates to adapt to evolving malware creation techniques.

The research paper [61] focused on addressing cybersecurity concerns related to Android platform. The authors discussed various existing techniques for Android malware detection, such as optimizing and effective ensemble learning-based methods, hybrid systems, DL models, and ML technique and hyperparameter tuning in enhancing classification performance. The research paper contributed to the topic by introducing the RHSODL-AMD model, which utilized Rock Hyrax Swarm Optimization and deep learning for Android malware detection, achieving a maximum accuracy of 99.05% on the Andro-AutoPsy dataset. The dataset used for experimental validation was the Andro-AutoPsy dataset, and the results showed promising performance with a maximum accuracy of 99.05% for the RHSODL-AMD technique. One limitation of the publication was that it focused on a specific model and did not extensively cover a wide range of existing techniques in Android malware detection, potentially leaving out other relevant approaches in the field.

The publication [62] explored the secure data aggregation methods and countermeasures against attacks in wireless sensor networks. The limitations of previous studies were discussed that focused on modeling malware propagation without considering the characteristics of WSNs, highlighting the need for more tailored models for malware propagation in WSNs. The research presented a novel fractional order model to accurately depict the dynamics of malware spread in WSNs. Gaps in research identified in the publication included the limited discussion on malware propagation based on fractional order models and the need for more studies focusing on the behavior

of different types of malware in WSNs. The dataset used in the publication was not explicitly mentioned in the provided context. However, the paper discussed the results of numerical simulations to evaluate the proposed model's performance and compare it with classical models, indicating a focus on assessing the accuracy and effectiveness of the new fractional order model for malware propagation in wireless sensor networks. The limitations of the publication may have included the need for further empirical validation of the proposed fractional order model, potential challenges in implementing the adaptive model for determining optimal control strategies, and the generalizability of the findings to diverse WSN environments. Additionally, the paper may not have extensively discussed the practical implications of the proposed model in real-world WSN scenarios.

The publication [63] highlighted the importance of memory analysis in detecting malware. The publication's contribution lay in demonstrating that memory data could be utilized for malware detection, achieving high accuracy levels with algorithms like Logistic Regression (99.97%) and Gradient Boosted Tree (99.94%). The dataset used was the balanced CIC-MalMem2022, and the results showed that memory analysis was very useful in detecting malware, with various algorithms achieving successful results. The limitations of the study included the specific characteristics of the dataset used, potential biases in the algorithms, and the need for further research to address more advanced malware detection challenges.

The publication [64] focused on malware detection using deep learning algorithms, specifically Long-Short-Term Memory Network (LSTM), Convolutional Neural Network (CNN), and Multitasking Deep Neural Network (DNN). Gaps in research could have included the need for further exploration of other deep learning algorithms or the investigation of different types of malware for detection. The publication contributed to the topic by showcasing the effectiveness of deep learning algorithms in malware detection, with an average accuracy of 96%, precision average of 97%, and recall average of 97%. The limitations of the publication could involve the need for further validation on larger datasets, exploration of real-time detection capabilities, or the consideration of different types of malware for detection.

The research paper [65] focused on the development of a highly accurate and efficient malware detection system based on one dimensional convolutional neural networks. The main points of the publication included the comparison of the proposed CNN detector with state-of-the-art techniques, such as a TF-IDF based benchmark detector and an existing embedding-based CNN detector, showcasing improved accuracy and training times. Gaps in research could have been related to the need for further exploration of the impact of different types of malware on the detection system, as well as the scalability of the model to larger datasets and more complex malware variants. The publication contributed significantly to the topic of malware detection by introducing a novel approach that outperformed existing techniques in terms efficiency and accuracy. The dataset consisted of 11,130 binaries, and the results on this dataset demonstrated the superior performance of the proposed CNN detector compared to benchmark detectors. One limitation could have been the lack of exploration into the generalizability of the model across different types of malware and the need for further validation on diverse datasets to assess its robustness and reliability.

The publication [66] was a comprehensive review on malware detection. The main points of the publication included the use of a dataset from Kaggle Data Set and VirusShare, consisting of 17,845 data captures based on network traffic containing both malware and non-malware, the training and testing process using TensorFlow Tools, and the comparison of various algorithms for malware classification. One gap in research could have been the exploration of more advanced Machine Learning algorithms or hybrid models for malware detection to enhance accuracy and efficiency. The publication contributed significantly to the topic of malware detection by achieving high accuracy levels, with the RF algorithm showing the accuracy of 99.95%, precision of 0.998, and recall of 0.999, enabling quick detection and mitigation of malware threats. The results showed that the Random Forest algorithm had the highest accuracy of 99.95%. The limitations of the study could include the need for further validation on different datasets to ensure the generalizability of the results and the exploration of more complex malware scenarios to test the robustness of the classification algorithms.

The publication [67] concentrated on utilizing machine learning (ML) and deep learning (DL) techniques to detect and classify malware. The aim was to assist cyber

forensic investigators in countering the proliferation of harmful software that specifically targets Android handsets, which store valuable and confidential information. The main points of the publication included addressing cybersecurity issues like intrusion detection and malware classification. The ECNN model presented in the research had shown high accuracy rates of 96.92%, 96.14%, and 95.8% for different Android Malware Datasets, with precision rates of 96%, 94%, and 94% on the three datasets. The research contributed to the field by presenting the ECNN model, which was faster and more accurate for smartphone malware analysis. The gaps in research could have included further exploration of the scalability of the proposed models across different types of malware and datasets, as well as the potential challenges in real-world implementation and adaptability to evolving malware threats. The publication's contribution to the topic lay in its innovative use of DL methods for cybersecurity applications, particularly in malware identification and categorization. The dataset used included Android Malware Dataset-1, 2, and 3, showcasing the effectiveness of the ECNN model in achieving high accuracy and precision rates. The limitations of the publication may have involved the need for extensive testing across a wider range of malware samples and datasets to validate the robustness and generalizability of the proposed models.

The research paper [68] focused on the detection and classification of malware in IoT networks using Artificial Neural Networks (ANN). The publication's main points included the challenges of protecting IoT networks from malware attacks, the need for efficient techniques and the comparing ANN methodology with traditional ML algorithms like k-NN and Naive Bayes. Gaps in research highlighted in the paper included the limited work on malware identification in IoT networks, emphasizing the substantial security threat posed by malware in IoT environments. The publication contributed to the topic by proposing a novel ANN methodology for detecting and classifying malware in IoT networks with high accuracy rates of 94.17% for detection and 97.08% for classification. The dataset used in the study comprised a total of 461,043 records, with 300,000 benign and 161,043 malicious instances, demonstrating the effectiveness of the proposed methodology. The limitations of the research included the focus on network traffic analysis for malware detection and classification, which

may not cover all possible attack vectors in IoT networks, and the need for further exploration of diverse malware types and behaviors in IoT environments.

The publication [69] specifically addressed the growing security risks faced by IoT infrastructure, apps, and devices as a result of the integration of IoT technology with 5G and artificial intelligence technologies. The main points of the publication included the challenges in detecting new and variant IoT malware quickly, the attractiveness of IoT devices to cybercriminals due to weak authentication and outdated firmware, and the development of a malware classification and detection system for IoT devices using machine learning techniques. The dataset used for malware classification and detection was not explicitly mentioned in the provided context. The publication emphasized the importance of accuracy in detecting and identifying various types of malware using static analysis with machine learning algorithms. One limitation of the publication could have been the lack of detailed information on the specific dataset used, which was crucial for evaluating the system's performance.

The publication [70] focused on proposing an automated way of classifying malware based on behavior analysis, utilizing Back Propagation Neural Network model as a classification technique. The authors emphasized the importance of automated tools in dealing with the diversity and volume of malware variants on network systems. Gaps in research could have included further exploration of advanced machine learning techniques beyond Back Propagation Neural Network for malware classification, enhancing feature extraction methods to capture more nuanced behavior patterns. The publication contributed significantly to the topic of malware classification by introducing an automated approach based on behavior analysis and neural network models. It highlighted the effectiveness of the proposed technique in classifying malware variants accurately, showcasing the potential of machine learning in cybersecurity applications. The dataset used in the research was not explicitly mentioned in the provided contexts. However, the results on the dataset demonstrated that the classification technique was effective in classifying malware variants and accurately detecting malware, as indicated by the experimental results. The limitations of the publication may have included the need for further validation on diverse datasets to ensure the generalizability of the proposed approach, potential challenges in feature

extraction from behavior analysis reports, and the necessity for continuous updates to adapt to evolving malware behaviors.

The publication [71] is focused towards examination and execution of binary malware classifier that is neural network based. It categorized Portable Executable (PE) files of windows according to the introduced function calls present in library. The main points of the publication included the limitations of traditional malware detection methods like hash-based, signature-based, and heuristic-based techniques, leading to the exploration of machine learning for malware detection. The research addressed the gap in achieving high efficacy in malware detection by proposing a neural network classifier that achieved an average accuracy of 97.8%, 97.6% precision, and 96.6% recall when classifying files as malicious or benign based on imported library function calls. The dataset used in the research was Windows Portable Executable (PE) files, and the results showed an average accuracy of 97.8%, recall of 96.6% and precision of 97.6% for classifying files as malicious or benign based on imported library function calls. The limitations of the publication included the focus on a Portable Executable files and the need for further exploration of new methods or approaches to enhance malware detection beyond the achieved results.

The publication [72] focused on the development of a hybrid model based on API call sequences. The main points of the publication included the significance of behavioral malware analysis, the integration of ML and deep learning algorithms, and the use of logistic regression to initialize neural network weights based on API call sequences. Gaps in research identified in the publication included the need for further exploration of weight initialization techniques in neural networks. The publication's contribution lay in offering techniques in neural networks for malware detection, achieving 83% accuracy and a 0.44 loss on a balanced dataset, and 98% accuracy with a 0.10 loss on an imbalanced dataset, outperforming state-of-the-art models. The research utilized a secondary dataset that contained API call sequences. The limitations of the publication may have included the need for further validation on larger and more diverse datasets, potential challenges in generalizing the model to different malware types, and the necessity for continuous updates to adapt to evolving malware threats.

The publication [73] provided a comprehensive examination of methodologies for scrutinizing and categorizing malware, with a particular focus on the difficulties presented by metamorphic and polymorphic malware, that possess the aptitude to modify the code in course of movement. The research highlighted the limitations of traditional signature-based methods in detecting new malware samples, emphasizing the importance of static and dynamic malware analysis techniques to understand risks associated with malicious code and to group unknown malware into existing families. The publication contributed to the topic of malware analysis by providing a comprehensive survey of techniques, emphasizing the role of machine learning in addressing the challenges posed by evolving malware variants, and highlighting the significance of behavioral patterns in classifying malware into known families. The dataset used in the research was not explicitly mentioned in the provided contexts, and specific results on a dataset or accuracy metrics were not detailed. However, the focus was on the methodologies and techniques used for malware analysis and classification, rather than specific datasets or results. One limitation of the publication could have been the lack of detailed discussion on specific case studies or real-world applications of the analyzed techniques, which could have provided more practical insights into the effectiveness of the proposed methodologies.

The publication [74] leveraged the CSE-CICIDS2018 dataset to employ techniques for intrusion detection and network security. The research article investigated the application of deep learning frameworks to identify attack categories and detect network intrusion traffic. It employed datasets such as NSL-KDD, KDD Cup 1999, CICIDS2017, and CICIDS2018. The publication provided insights into the effectiveness of various algorithms in detecting attacks.

The research paper [75] focused on the rise of ransomware and the challenges faced by the anti-malware industry due to the increasing malware threats. It discussed the evaluation framework centered on machine learning, comprising various modules like dataset compilation, file disassembly, data processing, decision making, and malware identification. The study highlighted the limitations of orthodox signature-based antivirus programs in identifying unfamiliar malware and tracking new forms of malware, leading to the need for advanced techniques like machine learning for

malware detection and classification. The publication emphasized the use of artificial learning and fundamental modeling techniques by academics and antivirus organizations for researching and identifying malware, showcasing the shift towards more sophisticated methods in the anti-malware industry. The research paper evaluated the effectiveness of different classifiers in the detection and classification of malware based on the accuracy of the complete process, demonstrating the importance of machine learning in enhancing malware identification. The gaps in research could have included further exploration of the specific machine learning algorithms used, the scalability of the proposed malware evaluation framework, and the adaptability of the system to evolving malware threats. The publication contributed significantly to the topic of malware detection and classification by showcasing the advantages of machine learning over traditional signature-based methods, providing insights into the challenges faced by the anti-malware industry, and presenting a comprehensive malware evaluation framework. The dataset used in the study consisted of two separate classes: malicious and benign software, with modules like grey images, Opcode n-gram, and decision-making mechanisms employed for malware identification. The results of the research paper were based on the accuracy of the complete process, which validated the effectiveness of the malware evaluation framework focused on machine learning in detecting and classifying malware threats. The limitations of the publication may have included the need for further real-world testing of the proposed framework, the generalizability of the results to different types of malware, and the potential challenges in implementing the system in diverse organizational settings.

The publication [76] examined the application of an adaptive genetic algorithm (AGA) and a hybrid analysis-based particle swarm optimization (PSO) to identify Android malware in autonomous vehicles. The study utilized the "CCCSCIC-AndMal-2020" dataset, which included of 13 distinct malware categories and 9504 hybrid characteristics, feature selection using PSO, and optimization of RF and XGBoost classifiers using AGA. A gap in research could have been the need for further exploration of the scalability and real-world applicability of the proposed approach. The research work achieved a 99.82% accuracy and F-score with the XGBoost classifier and 98.72% accuracy and F-score with the random forest classifier.

The research paper [77] suggested a two-level classification system, Macro and Micro, for identifying and categorizing various files and API calls as benign or harmful. It described data mining-based classification technique for malware discovery based on the characteristics and behaviors of viruses, utilizing dynamic analysis techniques. A virtual environment was used to run Cuckoo Sandbox to generate static and dynamic analysis reports. This results in high rates of detection and classification using various machine learning algorithms. Later on, the authors demonstrated performance effectiveness by utilizing WEKA. Various classification techniques and algorithms like K-nearest neighbor, Random Forest, and Light GBM were employed for malware detection out of which regression classification approach performed the best. The study discussed the categorization of malware samples based on characteristics and behaviors, with a focus on API calls and a mix of features to achieve high categorization rates. The dataset used in the research included malware samples from April 2020 to June 2021, with different malware categories and benign samples from various sources. The limitations of the research paper may have included the need for further exploration of malware sample tagging methods, potential biases in the dataset, and the generalizability of the proposed system to different malware types and behaviors.

The publication [78] focused on addressing the vulnerabilities and cyber-attacks faced by IoT networks through the use of Artificial Intelligence. The study focused on the creation of a method that utilized a Deep Neural Architecture called Pointer Networks to automatically choose the most effective mitigation steps for countering assaults on IoT networks. The proposed method aimed to optimize security-related Key Performance Indicators (KPIs) and had shown optimal solutions. The publication contributed to the topic by introducing an innovative approach that leveraged Artificial Intelligence to enhance cybersecurity in IoT networks. It presented a mechanism that optimized KPIs to select mitigation actions efficiently, showcasing promising results in terms of performance and scalability. The dataset used in the research was not explicitly mentioned in the provided contexts. However, the publication emphasized the optimization of security-related KPIs to enhance cybersecurity in IoT networks. The accuracy of the proposed method in selecting appropriate mitigation actions was demonstrated through the optimization of security-related KPIs, leading to efficient

countermeasures against attacks faced by IoT networks. One limitation of the publication was the lack of detailed information on the dataset used and the specific experimental setup, which could provide more insights into the performance and generalizability of the proposed method.

The publication [79] focused on addressing the challenges in malware detection by proposing a vigorous ML based anti-malware resolution that employed a imagining approach to epitomize malware as 2D images. The main points of the publication included the inefficiency of conventional malware detection systems. The proposed model made use of a layered ensemble approach that outperformed other deep learning techniques. The results obtained have the accuracy of 0.98, 0.97, and 0.97 for Malimg, BIG 2015, and MaleVis malware datasets, respectively. A gap in research highlighted was the need to identify unknown samples of untrained families by using a threshold, which could be a focus of future work. The stated work well performed with high uncovering rates on different malware datasets. The limitations of the publication included the need for further research on identifying unknown samples of untrained families and potential enhancements to the model's performance.

The publication [80] sought to improve the robustness of ensemble classifiers against adversarial assaults by utilizing varied feature selection and a stochastic aggregation technique. The main points of the publication included conducting experiments using Linear and Kernel SVMs on genuine datasets for spam filtering, malware detection, and handwritten digit recognition. A gap in research could have been the need for further exploration of the impact of the proposed ensemble approach on different types of datasets and classifiers to assess its generalizability. The publication contributed to the topic by demonstrating that the proposed ensemble approach significantly improved classifier robustness against evasion attacks without compromising classification accuracy, as evidenced by experiments on genuine datasets for spam filtering, malware detection, and handwritten digit recognition. The results on the dataset indicated a significant improvement in classifier robustness against evasion attacks, particularly when using Linear and Kernel SVMs. The accuracy of the classifiers was enhanced through the proposed ensemble approach, showcasing improved performance in terms of robustness against evasion attacks. One limitation of the publication could have been

the focus on specific types of classifiers and datasets, warranting further research to explore the generalizability of the proposed ensemble approach across a broader range of classifiers and datasets.

The research paper [81] focused on malware detection using hybrid features and artificial intelligence to enhance the process of identifying complex, polymorphic malware that exhibited varied behaviors. The GPSC (Genetic Programming Symbolic Classifier) algorithm was used to produce SE (Symbolic Expressions) in order to achieve optimal classification performance on a publically available dataset. The dataset employed in the investigation was divided into two categories: malicious and benign software. The dataset's imbalance was the primary concern, and balanced dataset was generated by employing oversampling techniques. The GPSC algorithm underwent training using a five-fold cross-validation technique to attain great accuracy in predicting SEs for each variation of the dataset. The study utilized multiple assessment criteria, such as precision, recall, F1-score, confusion matrix, and area under receiver operating characteristic curve (AUC), to evaluate the effectiveness of the GPSC algorithm. The results showed that the proposed method achieved accuracy of 0.9962 in detecting malware software which is obviously a high classification. The research contributed by demonstrating the effectiveness of combining hybrid features with AI for malware detection, providing a detailed methodology for applying the GPSC algorithm to achieve high classification accuracy. It addressed the issue of imbalanced datasets and presented a solution through oversampling techniques, leading to robust SEs with high accuracy. The limitations of the study included the need for further validation on larger datasets to ensure the generalizability of the results. Additionally, the research could have benefited from exploring the scalability of the proposed method to handle larger and more diverse datasets in real-world scenarios.

The publication [82] introduced a technique for categorizing malware using photos and deep learning. This involved representing malware binary files as color images and employing data augmentation approaches to improve performance. The research highlighted the need for further exploration in extracting more features for sample classification, combining dynamic analysis with static analysis, and conducting experiments on real-world malware datasets. The study evaluated the proposed method

against state-of-the-art classification models in the literature, showcasing significant advantages in accuracy over existing techniques. The dataset used in the study included the Microsoft malware dataset and the Google Code Jam dataset, achieving high accuracy rates of up to 99.99% and 99.38%, respectively. The publication contributed to the topic of malware classification by introducing a novel approach that outperformed existing methods in accuracy, demonstrating the effectiveness of deep learning and image-based techniques in this domain. The study has limitations such as the utilization of a significant number of model parameters, the analysis of only one feature of the sample, and the requirement for further investigation into additional aspects to improve categorization.

The paper [83] discussed various approaches proposed by different researchers who worked on the classification of Android malware apps using ML techniques. The authors had previously proposed different approaches for Android malware classification, including a hybrid approach integrating fuzzy C-means clustering with LightGBM, an evolving hybrid neuro-fuzzy classifier, an approach using adaptive neuro fuzzy inference systems. The research study introduced a new approach for classifying malware using a fuzzy integral-based multi-classifier ensemble. The experimental results showed that this approach achieved the maximum accuracy of 95.08%. The dataset used in the research consisted of 9476 Android goodware apps which comes in the category of benign files and 5560 Android apps which comes in the category of malicious software. The limitations of the research paper were not explicitly mentioned in the provided contexts.

The research paper [84] focused on the use of Federated Learning (FL) and Federated Transfer Learning (FTL) for NIDS (Network Intrusion Detection Systems) employing deep learning for classification of images. The main points of the publication included proposing novel methods for pre-processing Network Flow Feature (NTF) records, transforming them into images, and utilizing FL to maintain data privacy while achieving acceptable accuracy in DDoS attack identification. Gaps in research included the need for further exploration of the scalability and applicability of FL and FTL in diverse network environments and the potential challenges in implementing these methods in real-world network security systems. The publication contributed

significantly by addressing the issue of data privacy in NIDS training, achieving accuracy rates of 92.99% for FTL and 88.42% for FL in DDoS attack identification compared to Traditional Transfer Learning. The BOUN DDoS dataset was used for testing. The limitations of the study might have included the need for further validation on larger and more diverse datasets, as well as the practical implementation challenges of FL and FTL in complex network environments.

The publication [23] discussed the rapid development in Information Technology and the increasing number of new malware, highlighting the ongoing battle between attackers and defenders in the cybersecurity realm. The main points of the publication included the need for abundant malware samples for training machine learning models, the exponential growth of malware, the challenges faced by cybersecurity professionals in maintaining security, and the utilization of sandboxes to detect malicious activity. The paper made a substantial contribution to the field of cybersecurity by highlighting the significance of machine learning in detecting malware, the use of different characteristics such as call graphs and API calls for categorization, and the function of sandboxes in evaluating the behavior of malware. It shed light on the ongoing challenges in cybersecurity and the need for continuous innovation to combat cyber threats effectively. The dataset used in the publication included abundant malware samples for training machine learning models, with features such as call graphs, API calls, and strings being utilized for detection and classification. The dataset findings demonstrated the efficacy of machine learning in properly detecting malware and categorizing various types of malware. The dataset's conclusions highlighted the efficacy of machine learning in detecting and classifying malware, highlighting its potential to enhance cybersecurity measures. However, limitations may have included the need for more diverse datasets, the continuous evolution of malware techniques, and the requirement for ongoing updates to machine learning algorithms to stay ahead of cyber threats.

The publication [85] focussed on utilizing machine learning approaches to detect malware in Android applications, with a specific emphasis on pre-processing of data and reduction in dimensionality. The impact of various data pre-processing methodologies was investigated using four distinct datasets, including real-world

Android applications, to enhance test scenarios beyond existing datasets. The study identified support vector machines (SVM) and random forests (RF) as classifiers that achieved better performance in detecting malware in Android apps, with feature selection techniques playing a crucial role in reducing data dimensionality and enhancing explainability. The research contributed by addressing the importance of data pre-processing, identifying decisive features for malware detection, and evaluating the performance of SVM and RF classifiers in this domain. However, there were gaps in research related to the need for supplementary and weighty features beyond permissions and intents for effective malware detection in Android apps. The study utilized both publicly available datasets and custom-built applications to evaluate the performance of classifiers in accurately detecting malware. The results demonstrated the high accuracy of both classifiers in identifying malicious software. The limitations of the publication might have included the need for further research on additional features beyond permissions and intents for improved malware detection, as highlighted by previous studies.

The research paper [86] emphasized the need to evaluate various malware family classification methods fairly within a controlled environment. The research highlighted the importance of assessing various techniques for classifying malware families in a fair manner, with a specific focus on flow-level traffic classification approaches that classify each encrypted flow separately. The paper highlighted the significance of evaluating the sequential information of each TLS session for malware family classification. A gap in research identified was the lack of proper evaluation of the sequential information of TLS sessions for malware family classification. The dataset used in the research was not explicitly mentioned in the provided contexts. The test findings consistently showed that using a graph-based representation for sequential information resulted in superior performance across several classification algorithms. This provides valuable insights for researchers to develop enhanced machine learning classifiers. The accuracy metrics or specific numerical results on the dataset were not detailed in the provided contexts. One limitation of the publication could have been the lack of detailed information on the dataset used, specific numerical results, and

accuracy metrics, which could have enhanced the comprehensiveness of the findings and evaluation presented in the research paper.

The paper [87] examined the malware detection skills of different pretrained Convolutional Neural Network (CNN) models. The main components involved in this study were the application of feature extraction models including SqueezeNet , ResNet-50, GoogLeNet, DenseNet-201, and AlexNet. Additionally, feature selection was performed using PCA, classification was carried out using KNN, LR, SVM, GDA, RF, and ensemble learning techniques. Gaps in research could have included the need for further exploration of other CNN models, additional feature selection methods, and the impact of different datasets on malware detection performance. The publication made a substantial contribution by introducing DenseNet201-KNN algorithms that surpassed the current leading approaches, achieving an accuracy rate of 96% and a minimal error rate of 3.07%. The dataset used was the unbalanced Malimg datasets, and the results showed that KNN was the best classifier and DenseNet201 was the best pretrained model for malware detection. The limitations might have involved the need for more diverse datasets, exploration of real-time detection scenarios, and further investigation into the scalability of the proposed methods.

The research paper [88] focused on proposing an ensemble classification-based methodology for detecting malware using neural networks and machine learning models. The study emphasized the utilization of a stacked ensemble consisting of dense and Convolutional Neural Networks for first categorization, followed by a meta-learner utilizing 14 classifiers for the final categorization stage. Gaps in research included the need for more general and robust methods for malware detection to improve generalization potential and accuracy of classification. The studies were conducted using the ClaMP dataset. The finest performance was obtained by utilizing an ensemble of five dense and CNN neural networks, in addition to the ExtraTrees classifier. The results obtained from the dataset demonstrated that the suggested methodology surpassed alternative machine learning methods, attaining superior performance in the identification of malware.

The research paper [89] focused on the development of a novel six-step framework for identifying and categorizing IoT malware, addressing the challenges of detecting new

and modified malware strains in the context of increasing IoT adoption and vulnerabilities. The study employed a combination of a Ghost-Net ensemble and the Gated Recurrent Unit Ensembler (GNGRUE), which were trained on eight datasets of malware attacks. The models were then fine-tuned using the Jaya Algorithm (JA) to obtain exceptional performance in detecting malware. The literature review section of the paper provided an overview of various techniques used in classifying malicious software, covering dynamic, static, and AI-driven approaches. It discussed the limitations of signature-based malware detection and the challenges posed by zero-day malware, emphasizing the need for hybrid approaches that combine static and dynamic evaluation methods with computational intelligence techniques. The research paper contributed to the topic of IoT malware detection by introducing a novel framework that outperformed existing models by around 15% across metrics like AUC, accuracy, recall, and hamming loss, with a 10% reduction in time complexity. The gaps in research identified in the literature review included the limitations of signature-based techniques, the challenges of behavior analysis and anomaly detection methods, and the need for hybrid approaches that amalgamate static and dynamic evaluation methods with computational intelligence techniques to improve malware detection capabilities. The evaluation of the publication's contribution to the topic highlighted the development of a comprehensive framework that significantly enhanced smart IoT malware detection, providing more performance compared to existing models. The limitations of the research included the computational overhead associated with some dynamic assessment methodologies, the challenges of dealing with changing malware behavior within virtual environments, and the need for further research to address evolving malware threats and enhance the efficiency of malware detection techniques.

The publication [90] proposed a machine learning approach that utilizes a neural network to improve the accuracy of malware detection in response to insider threats. The procedure involved feature extraction, anomaly detection, and classification using the CERT4.2 dataset. The data was pre-processed by the authors through the encoding of text strings and the distinction between threat and non-threat entries. The publication focused on the creation of a machine learning model that included thick layers, ReLU activation functions, and dropout layers for regularization. The purpose of this model

was to accurately identify and categorize internal threats. Gaps in research could have included a more extensive exploration of various machine learning algorithms and their effectiveness in detecting insider threats, as well as a deeper analysis of the limitations of the proposed model in real-world scenarios. The publication contributed to the topic of insider threat detection by introducing a novel machine learning model that enhanced malware detection accuracy. The results on the dataset showed that the proposed machine learning model could detect malware more effectively with 100% accuracy. The limitations of the publication may have included the need for further validation of the model in diverse real-world scenarios, potential biases in the dataset used, and the scalability of the proposed method to larger and more complex datasets.

The research paper [91] aimed to improve the identification of Android malware and classify them into families by analyzing conversation-level network traffic parameters. The main points of the publication included the extraction of conversation-level network traffic features for Android malware detection, categorization, and family classification using an ensemble learning technique. Gaps in research could have included the need for further exploration of the effectiveness of different machine learning classifiers and feature selection algorithms in Android malware detection and classification. The results showed Extra-trees classifier on the higher side with accuracy percentage surpassing other classifiers. The accuracy of 87.75% was found for malware detection, 79.97% for malware categorization, and 66.71% for malware family classification. The dataset used in the study was the CICAndMal2017 dataset.

The research paper [92] focused on the integration of plant protection and information systems, modernizing pest level monitoring, and enhancing control capabilities in plant protection networks. The main points of the publication included the proposal of malware analysis scheme based on bicubic interpolation to address image size imbalance issues in malware images. The scheme utilized the Cycle-GAN model for data augmentation to balance samples among malware families and built an efficient malware classification model based on CNNs, resulting in significantly improved malware classification efficiency. Experimental results showed high accuracy rates of 99.76% for RGB images and 99.62% for gray images using the Microsoft Malware Classification Challenge Dataset (BIG2015). There may have been a lack of research

in investigating the scalability and flexibility of the proposed scheme for identifying and classifying malware in real-world plant protection information systems.

The publication [93] explored B398n networks using a susceptible-unexposed-infected-isolation-removed pandemic model developed by Ying Zhou, Yan Wang, and other researchers. The model used a non-linear dynamic equation to describe the spread of malware, and the basic reproduction number was derived using the next-generation approach. The publication focused on three key aspects: devising optimal ways to manage the proliferation of malware, conducting numerical simulations to examine the spread of malware in wireless sensor networks (WSNs), and examining the communication range of nodes to regulate the transmission of malware. One area that was not adequately studied was the requirement for additional investigation into how well the proposed model works in real-life situations and whether the control mechanisms can be scaled up in larger networks. The publication contributed to the topic by providing insights into malware propagation dynamics in wireless sensor networks, offering a model for controlling malware spread, and designing optimal node ranges to limit malware propagation.

The publication [94] examined malware by extracting common item sets in API call sequences. The publication focused on three primary aspects: categorizing 266 API requests into 23 unique categories for malware analysis, dividing the data into training and testing sets, and assessing the performance of machine learning models such as Naive Bayes, XGBoost, and K-Nearest Neighbor for detecting malware. The research gaps may have encompassed the necessity for further investigation into various machine learning techniques or the integration of supplementary attributes to enhance the precision of virus identification. The publication enhanced the subject matter by offering valuable perspectives on the utilization of different API categories by malware, presenting a systematic approach for malware analysis employing machine learning models. The specific dataset used in the study was not clearly stated in the given information. However, it is likely that the article employed a dataset having API calls for both malware and benign in order to train and evaluate the machine learning models. The dataset analysis involved categorizing API requests into several classes, training and testing machine learning models, and assessing model performance using metrics

such as accuracy, precision, recall, and F1 score. The paper may have been limited by the absence of a commentary on the generalizability of the findings to various types of malware, potential bias in the dataset utilized, or the necessity for further rigorous empirical validation of the suggested technique.

The research paper [95] provided an overview of previous studies related to Android malware detection and classification. The authors discussed the inadequacy of manual techniques in dealing with the intricate nature of contemporary malware and the constraints of static and dynamic evaluations. The evaluation emphasized the necessity of employing a hybrid analysis strategy that integrates both static and dynamic malware analysis in order to improve the detection and categorization of Android malware. The article examined a suggested architecture consisting of three stages: pre-processing which includes normalization and feature extraction, feature selection, and the implementation of a detection model utilizing a neural network. The literature study highlighted the enhanced precision attained by the hybrid technique in contrast to the separate consideration of static and dynamic information. This underscores the research's significant contribution to the domain of Android malware detection and classification.

The publication [96] focused on employing machine learning techniques for recommender systems-based IoT to forecast assaults in Android malware devices. The publication highlighted several key findings, including the application of static analysis to anticipate malware in Android apps, the development of a system to forecast and suggest blocking malicious devices from transmitting data to the cloud server, and the successful achievement of a 93% prediction rate using the K-Nearest Neighbor (KNN) machine-learning model. The research article examined the deficiencies in existing research by emphasizing the necessity for more sophisticated techniques to identify Android malware in the Internet of Things. It underscored the significance of feature selection and the utilization of machine-learning algorithms to achieve precise predictions. The model also demonstrated high accuracy, precision, recall, and F1 measures, with values of 93%, 95%, 90%, and 92% correspondingly. The research utilized a dataset consisting of over 10,000 Android applications. The purpose was to identify and ban malicious nodes from the cloud server. The KNN model demonstrated

exceptional accuracy and performance metrics. The release may have been limited by the requirement for additional validation on larger datasets, potential difficulties in implementing it in real-time, and the need for ongoing updates to address the ever-changing malware threats on Android devices.

The paper [97] discussed the limitations of traditional signature and heuristic-based methodologies for detecting malicious software, highlighting the need for advanced techniques like machine learning to achieve higher accuracy rates for unknown malware detection. Various deep learning algorithms and transfer learning techniques were explored to enhance malware detection resilience and accuracy, with different models like ResNet, GoogleNet, VGG16, and LSTM hybrid networks being employed. The researchers in the study employed a dataset including 8970 malicious and 1000 benign executable files. These files were pre-processed and transformed into pictures for the purpose of analysis. The research paper proposed a architecture that utilized two VGG-19 models but with certain modifications. First model was designed to determine the maliciousness of a file, achieving a testing set accuracy of 99%. The second model focused on identifying the specific type of malware, achieving an accuracy of 98.2%. The research study made a significant contribution to the field of malware detection by demonstrating the efficacy of deep learning approaches, particularly the VGG-19 models, in accurately detecting and classifying malware with a high level of accuracy. An identified weakness in the literature assessment was the possibility of adversaries avoiding detection if they knew the specific features utilized for feature extraction and classification. This highlights the necessity for additional research to tackle this issue.

The publication [98] focused on implementing machine learning techniques for identifying malware. It evaluated the performance of different algorithms, such as Naïve Bayes, Support Vector Machine, K-Nearest Neighbor, Decision Tree, Random Forest, and Logistic Regression, using a dataset that included both benign files and malware. The study emphasized the capacity of machine learning methods to accurately identify malware. The literature review portion analyzed multiple studies conducted by previous researchers on machine learning classification methods, highlighting the importance of choosing the right technique to accurately detect malware. The text also emphasized the significance of feature selection and dimensionality reduction

approaches in improving the efficacy of classifiers. The study lacks in the areas of advanced feature selection techniques to better depict malware features, as well as the exploration of ensemble methods and deep learning approaches to increase malware detection capabilities. The paper demonstrated that machine learning techniques may effectively detect malware, with Decision Tree and Random Forest algorithms exhibiting greater performance compared to other methods. The study yielded useful insights into the efficacy of several algorithms in detecting malware, hence emphasizing the potential of machine learning in this domain. The study utilized a dataset obtained from Kaggle, and data pretreatment techniques were applied to ensure the data's high quality. The dataset analysis revealed that machine learning techniques achieved high levels of precision and reduced false positive rates. Specifically, the Decision Tree and Random Forest algorithms demonstrated a remarkable accuracy of 100.00%. The study highlighted the significance of choosing pertinent and efficient characteristics for the identification of malware. The publication's limitations encompassed the requirement for additional study in augmenting feature engineering techniques, investigating advanced ensemble methods, and employing deep learning methodologies to enhance malware detection capabilities. Furthermore, the study recognized the significance of choosing the suitable machine learning method for efficient identification of malware.

The research paper [99] focused on malware identification and analysis, exploring various methods and techniques used in the field. It provided a detailed discussion of the approaches used for malware analysis, including signature-based identification, behavior-based detection, supervised machine learning methods, and the utilization of deep learning. The paper discussed the challenges faced in malware analysis, such as the need for robust datasets and the limitations of current techniques. The study assessed the impact of machine learning methods, specifically deep learning and convolutional neural networks, on the detection of malware. The research utilized a dataset comprising 1200 PDF samples, where 800 samples were allocated for training and 400 for testing. The objective was to maintain a balanced ratio of benign to dangerous files at 1:1. The dataset was analyzed using multiple machine learning classification techniques, such as stochastic gradient boosting, random forest, decision

tree, support vector classifier, and logistic regression. The effectiveness of the proposed work can be observed in the confusion matrix parameters. One of the study's shortcomings was the difficulty in categorizing PDF files using JavaScript code.

The publication [100] explored the utilization of deep learning models in the identification of malware in cyberspace, with a specific emphasis on their significance and contributions to bolstering cybersecurity endeavors. The study assessed the performance of various machine learning models, including Recurrent Neural Networks, LSTM, Deep Autoencoders, and Deep Neural Networks, in the detection of malware in cyberspace. It emphasized the specific advantages and practical uses of these models in real-world scenarios. The study highlighted the efficacy of deep learning models in autonomously categorizing malware samples into separate families or categories by acquiring attributes from huge datasets. The publication did not specifically address the research gaps in the given circumstances. The study highlighted the substantial enhancement in malware detection models, demonstrating their high precision and minimal occurrence of false positives in real-world situations. The utilization of diverse deep learning models in the identification of malware has emphasized their capacity to augment cybersecurity endeavors. The dataset used in the publication was not specified in the provided contexts. The results on the dataset and the accuracy achieved by the deep learning models were not explicitly mentioned in the provided contexts. Limitations of the publication were not explicitly discussed in the provided contexts.

The publication [101] focused on on creating an innovative malware detection model with an autoencoder network that merged a grey-scale picture of malware image to differentiate malware from harmless software. The study focused on using an autoencoder network to assess the effectiveness of grey-scale picture representations of malware by analyzing reconstruction errors. The study underscored the constraints of existing malware detection systems that rely on deep learning models, underscoring the necessity for more streamlined techniques for encoding malware feature images and conducting data pre-processing in order to investigate novel ways to malware detection. The paper made a significant contribution to the field by creating a detection model that performed better than traditional machine learning techniques. This demonstrated the

efficiency of using an autoencoder-based design for malware detection. The study utilized a dataset obtained from the Android platform. The findings of the study showcased the higher performance of the proposed detection model compared to existing machine learning approaches and specific deep learning malware detection models that rely on malware images. The suggested model of detection achieved an accuracy about 96% and a stable 96% F-score approximately, demonstrating the model's stability and efficiency in separating malware from benign software. An issue emphasized in the report was the need for human configuration in existing malware detection methods, which presents difficulties in efficiently detecting new forms of malware.

The publication [102] included an overview of malware detection nature (static, dynamic, and hybrid approaches), an investigation of recent advanced works on malware detection using deep learning frameworks, and the utilization of AI-based frameworks like machine learning, deep learning, and hybrid frameworks to provide solutions. The research paper discussed gaps in research by highlighting the importance of developing robust malware-free devices due to critical security issues in the digital world. The publication contributed to the topic of malware detection by providing a comprehensive literature review on the subject, exploring various detection approaches, and emphasizing the significance of security in the digital landscape. The dataset used in the publication was not explicitly mentioned in the provided contexts. The results on the dataset and accuracy metrics were not specified in the given contexts. The limitations of the publication were not explicitly outlined in the provided contexts.

The publication [103] worked on enhancing the accuracy of machine learning classifier for static PE malware detection through hyper-parameter optimization using covering arrays (CAs). The main points of the publication included the introduction of cAgen, a tool for generating covering arrays to tune ML approaches, the significance of covering arrays in optimizing hyperparameters for machine learning algorithms, and the promising results obtained in improving classification accuracy for malware detection. A gap in research highlighted in the publication was the lack of convincing rationale for specific parameter selections in machine learning algorithms, which could impact their performance. The publication contributed to the topic by introducing a novel

approach using covering arrays for hyper-parameter optimization, addressing the curse of dimensionality resulting from traditional systematic approaches like Grid Search. The dataset used in the research was not explicitly mentioned in the provided contexts. However, the publication emphasized the importance of parameter optimization for machine learning algorithms to enhance their performance. The research findings demonstrated that cAgen was a highly effective method for attaining optimal parameter selections for machine learning approaches, resulting in enhanced accuracy in classifying static PE malware for detection purposes. One limitation of the publication was the lack of detailed information on the specific dataset used for experimentation and validation of the proposed approach.

The publication [104] focused on the comparison of malware classification techniques utilizing Convolutional Neural Network based on API call streams. The paper focused on many key aspects, including the utilization of a database consisting of 7107 instances of API call streams and 08 distinct types of malware for the purpose of classification. Additionally, it involved the creation of a 1-Dimensional Convolutional Neural Network (CNN) for classifying different types of malware. Furthermore, the publication also entailed a comparative analysis of the obtained findings with other classification techniques. Potential research gaps may involve the necessity for additional investigation into various forms of malware and the creation of more sophisticated categorization methodologies to efficiently identify novel and unidentified malware. The publication demonstrated the efficacy of the suggested 1-D CNN in classifying malware by reaching an overall accuracy of 91% for both categorical and TF-IDF vectors. The dataset utilized included of API call streams and various forms of malware, with the findings showcasing the supremacy of the CNN methodology. The limitations of the publication may involve the need for more diverse datasets, further validation of the results on larger datasets, and potential challenges in real-time implementation of the proposed CNN model.

The publication [105] focused on risk management providing security business methods that are cloud-based. The gaps in research identified in the publication included the limited work done on cloud security risk management despite the significant discourse on risk management in business processes. The study highlighted

the importance of conducting further research in systems that were identified as facing difficulties, such as incorporating security risk management techniques into research areas that lack support and effectively managing security risks during the modeling and monitoring stages. The paper emphasized the significance of incorporating security risk management, developing methodologies, and standards into corporate operations to alleviate security concerns. Furthermore, it emphasized the necessity of validating methods in real-time to evaluate their practicality and efficiency. The dataset utilized for the literature study comprised esteemed conferences and journals that were meticulously searched to identify papers pertaining to the management of risks of security in cloud-based business methods. The search spanned from 2010 to October 2020. The review findings emphasized deficiencies in research and the significance of incorporating security risk management into corporate operations. Nevertheless, the context did not include precise information regarding the dataset utilized, the outcomes obtained, and the level of accuracy achieved. The limitations of the publication may involve the requirement for additional validation of methods and the investigation of security risk management in real-time situations to improve practicality and efficiency.

The publication [106] proposed a framework that use an evolutionary algorithm to generate adversarial samples and defend against them in deep learning-based Internet of Things (IoT) malware detection models. The key aspects covered were the utilization of evolutionary algorithms for sample rewriting to produce alterations, the procedure of generating modified samples using the evolutionary algorithm, and the advantages of this technique in enhancing the algorithm's speed and facilitating parallel computing. Potential research gaps encompass the need for additional investigation into the influence of various malware kinds on the efficacy of the suggested framework, the ability of the technique to handle larger datasets, and the flexibility of the method to address emerging malware threats. The study introduced an automated framework for producing adversarial samples without human interaction. This approach can improve the robustness and effectiveness of deep learning-based IoT malware detection models. The specific dataset utilized for the study was not specifically stated within the given context. The outcomes of the dataset were not explicitly stated within the provided context. The context did not give information regarding the accuracy of the proposed

framework on the dataset. The paper may have limitations such as the requirement for additional validation on distinct datasets, potential difficulties in implementing the framework in real-world scenarios, and the applicability of the framework to various types of malware.

The publication [107] included the necessity for effective malware detection mechanisms, the limitations of signature-based detection systems and the emergence of machine learning for rapid malware detection. The research enhanced knowledge by integrating machine learning techniques to enhance security, doing a bibliometric analysis, and delivering a full evaluation of anomaly detection approaches. The study conducted a thorough analysis of machine learning-based classification techniques used in anomaly detection over a period of almost twenty years. It focused on highlighting the strengths and weaknesses of different machine learning methods for detecting malware. The study highlighted the deficiencies in research by underscoring the constraints of signature-based detection systems, the difficulties in identifying zero-day assaults and polymorphic malware, and the necessity for more sophisticated detection methods to successfully counteract malware attacks. The assessment of the publication's impact on the topic encompassed its thorough examination of machine learning techniques for identifying malware, the introduction of methods for extracting and selecting relevant features, the comparative analysis of various machine learning approaches, and the identification of potential areas for future research in utilizing artificial intelligence for automated detection of malware at the system level. The specific dataset used, the conclusions obtained from it, and the accuracy measures were not specifically specified in the given situations.

The publication [108] addressed the difficulties encountered in detecting Android malware, highlighting the structural and characteristic disparities between computer malware and Android malware. Conventional detection techniques designed for PC malware may not be efficacious in detecting Android malware. The paper focused on the creation of a robust method for categorizing malware, specifically targeting Android malware, with high accuracy in detection. The methodology employed consisted of creating distinctive profiles based on signatures and behaviors for each application in the dataset, which were subsequently utilized for categorization purposes. A gap in

research highlighted in the publication was the limited generalizability of suggested detection approaches for Android malware, especially in detecting zero-day malware. This limitation was attributed to factors like the availability of datasets with specific examples. The publication's contribution to the topic was the introduction of a malware classification approach that aimed to improve detection accuracy for Android malware. The approach was evaluated using artificially generated examples to assess its reliability and effectiveness. The dataset used in the publication was not explicitly mentioned in the provided context. However, the publication evaluated the approach using artificially generated examples to demonstrate its detection accuracy. The results on the dataset were not explicitly mentioned in the provided context. Still, the publication presented a malware classification approach with reliable detection accuracy, indicating positive results in improving Android malware detection. The accuracy of the detection approach proposed in the publication was highlighted as reliable, aiming to address the challenges in detecting Android malware effectively. One limitation of the publication was the potential lack of generalizability of the suggested detection approaches for Android malware, particularly in detecting zero-day malware due to constraints like dataset availability.

The research paper [109] focused to improve malware detection in order to mitigate the detrimental effects of malware on performance, reliability, energy consumption, and other quality aspects. The authors sought to enhance and refine the fundamental tools for malware detection by employing a comprehensive dynamic ontology model that incorporated a substantial volume of data, resulting in improved accuracy. The publication contributed by summarizing existing approaches and presenting the APKOWL method, which utilized SPARQL queries based on malware behavior to detect malware at the design stage, showing promising results for SMS malware detection. The results of the APKOWL method on the dataset CICMalDroid 2020 showed higher accuracy compared to other state-of-the-art methods, indicating the effectiveness of using a full dynamic ontology model for Android malware detection. The limitations of the publication included the focus on SMS malware detection and the need for further evaluation on detecting other types of Android malware to assess the method's broader applicability.

The publication [110] conducted an analysis on the efficacy of machine learning models in detecting online malware. Specifically, it examined the explainability and interpretability of ML models like SVM Linear, SVMRBF, Random Forest, Feed-Forward Neural Net, and Convolutional Neural Network models. These models were trained on an online malware dataset using the Shapley Additive exPlanations (SHAP) technique. The publication addressed a number of critical topics, such as the challenges that arise from the opaque nature of neural networks in the detection of malware, the importance of transparency and explainability in the decision-making processes of machine learning models, and the use of SHAP techniques to interpret the results of various models that were trained on an online malware dataset. The research provided valuable insights into the contributions of various machine learning models in detecting online malware. It emphasized the significance of explainability in improving the sharing of cyber threat intelligence. The study also demonstrated the performance of different models, including SVM Linear, SVM-RBF, RF, FFNN, and CNN, on the dataset. Notably, CNN achieved the highest accuracy rate of 97.01%. The study utilized an online malware dataset and demonstrated that the CNN model had superior performance compared to other models, with an accuracy rate of 97.01%. The study also assessed the feature contributions of several models utilizing SHAP approaches such as KernelSHAP, TreeSHAP, and DeepSHAP. The publication's limitations encompassed the intricate nature of neural networks, the difficulty in reading black-box models, and the necessity for more study to investigate supplementary techniques of explainability and enhance the comprehensive comprehension of machine learning models in online malware detection.

The publication [111] conducted an analysis and comparison of the efficacy of malware detection utilizing contemporary machine learning techniques such as K-Nearest Neighbors, Extra Tree, Random Forest, Logistic Regression, Decision Tree, and neural network Multilayer Perceptron. The study utilized the UNSWNB15 dataset and implemented feature encoding and selection techniques for classification purposes. The study underscored the need of robust internet security in safeguarding users against detrimental conduct and emphasized the escalating menace of malware in the realm of cybersecurity. The paper demonstrated that Random Forest achieved the highest

accuracy of 97.68% among the machine learning models examined, offering valuable insights into effective strategies for detecting malware. The study revealed constraints in existing deep learning models for detecting and categorizing malware, highlighting the necessity for more research to improve detection capabilities.

The research paper [112] discussed various malware detection methods, including signature, behavior-based, and heuristic techniques using Deep Learning (DL) approaches for malware detection. It highlighted the limitations of signature techniques, behavior-based techniques, and heuristic approaches in detecting complex malware variants. The review also mentioned the development of DL-based malware detection frameworks, such as Stacked Auto-Encoder (SAE), Deep Belief Network (DBN), and Transfer Learning models, to improve malware detection rates. Ensemble learning techniques have been suggested in the literature for detecting malware. These techniques involve mixing different machine learning algorithms such as Naive Bayesian, Decision Tree, Random Forest, and Support Vector Machines. The review highlighted the significance of ensemble learning in enhancing the efficacy of individual malware detection algorithms. The research study presented a novel approach to malware detection using an ensemble-based parallel deep learning classifier. The classifier utilized five deep learning base models and a neural network as a meta model, resulting in exceptional accuracy rates across five different malware datasets. The proposed ensemble method enhanced deep learning models using a hybrid optimization method that combines Back-Propagation (BP) and Particle Swarm Optimization (PSO) algorithms. This approach showed effectiveness, efficiency, and scalability in detecting malware. The parallel implementation of the ensemble method greatly improved computational speed by a factor of 6.75, demonstrating its effectiveness in handling enormous amounts of data.

The publication [113] focused on proposing a malware detection approach using a modified DenseNet model for feature extraction and classification of binary images. The model was trained by inputting images directly into the initial convolution layer, leveraging CNNs to extract distinctive features and learn task-specific features. The main points of the publication included illustrating the design of the malware detection approach, showcasing the flow of the modified DenseNet model, emphasizing the use

of DenseNet for feature extraction from malware datasets, and training the model on top of the extracted features to classify binary images. A gap in research identified in the publication was the need for further exploration into the effectiveness of DenseNet models for malware detection across different types of malware datasets. The publication contributed to the topic of malware detection by introducing a novel approach that utilized DenseNet for feature extraction and classification, showcasing the potential of CNNs in extracting features from binary images for accurate classification. The dataset used in the study was not explicitly mentioned in the provided context. Nevertheless, the proposed DenseNet model's ability to accurately classify binary images was demonstrated by the results of the dataset. This was supported by the training and test accuracy and loss figures that were presented in the publication. The limitations of the publication may have included the lack of detailed information on the specific malware datasets used, potential biases in the training data, and the need for further validation on diverse datasets to assess the generalizability of the proposed malware detection approach.

The publication [114] examined the comparison of attribute extraction approaches and machine learning algorithms for the purpose of static malware categorization and detection. The study primarily focused on evaluating the efficacy of combining PCA attribute extraction with SVM classifier for malware detection. This approach yielded the highest accuracy rate while using the least number of attributes. The study also explored advanced approaches for detecting sophisticated malware and tactics for defending computer systems. The study noted a research gap in the absence of an evaluation of current deep learning techniques and a comprehensive explanation of the characteristics employed in data mining methods for identifying and tracking malware. The publication made a significant contribution to the field of malware detection by presenting a strategy that improved the accuracy of malware detection to 96% through the use of PCA attribute extraction and SVM classifier. The specific dataset employed in the study was not clearly stated in the given context. However, the findings obtained from this dataset demonstrated superior performance in comparison to previous methodologies, thereby emphasizing the effectiveness of the proposed technique. The publication's drawbacks encompassed the absence of particular information regarding

the dataset employed, the lack of an evaluation of deep learning techniques, and a comprehensive description of the attributes employed in data mining methods for identifying and tracking malware.

The publication [115] focused on the Android platform and its impact on node feature disparities in a function call graph (FCG) through the utilization of Graph Neural Networks (GNNs). The publication highlighted several key aspects, including the implementation of an API-based node feature, the extraction of FCG and function features from decompiled APK files, the calculation of the API coefficient, the extraction of a subgraph called S-FCSG that contains sensitive function calls, and the utilization of a GCN model for feature extraction and classification. The research emphasized the significance of augmenting node feature disparities in FCGs (Function Call Graphs) for detecting Android malware. It demonstrated that the suggested approach surpassed models utilizing alternative features, suggesting a promising avenue for future investigations in malware detection that rely on graph structures and GNNs (Graph Neural Networks). The research utilized a dataset consisting of 978 Application Programming Interfaces (APIs) that have robust security measures and are often called upon. These APIs were used to construct a sensitive collection of APIs for the purpose of detecting malware. The findings obtained from the dataset demonstrated that the suggested methodology effectively increased the disparities in node characteristics inside FCGs, resulting in improved accuracy in detecting compared to models utilizing alternative features. The paper enhanced the field of Android malware detection by bringing a unique methodology that utilized graph structures and GNNs, offering valuable insights into the possibilities of graph-based methods for improving detection accuracy. One gap in the research could have been the need for further exploration of dynamic subgraph extraction methods and the evaluation of node importance within the graph structure. The limitations of the publication may have included the focus on node features and GCN model characteristics, potentially overlooking other aspects of graph structures that could impact malware detection effectiveness.

The publication [116] addressed the topic of detecting Android malware through the utilization of machine learning-based methods. The publication primarily focused on

three key topics: the proliferation of Android malware, the efficacy of machine learning and genetic algorithms in detecting Android malware, and the utilization of developing Genetic Algorithms to enhance feature subsets for training machine learning algorithms. The publication may have addressed research gaps such as the necessity for additional investigation into the use of various machine learning algorithms in combination with Genetic Algorithms, the influence of larger datasets on outcomes, and the assessment of alternative machine learning methods for detecting Android malware. The publication enhanced the field of Android malware detection by conducting a comprehensive analysis of machine-learning-based methods, emphasizing the efficacy of machine learning and genetic algorithms, and discussing the architecture and security considerations of Android that are pertinent to malware detection. The dataset utilized in the publication was not specifically specified in the given context. The results pertaining to the dataset were not specifically specified within the given context. The publication reported that Support Vector Machine and Neural Network classifiers achieved a classification accuracy of over 90-91 percent when working with lower dimension feature sets. This approach also reduced the complexity of the training process. The publication may have been improved by addressing several limitations, such as the requirement for additional study of the effects of various machine learning techniques, the utilization of larger datasets, and the applicability of the findings to a wider range of Android malware scenarios.

Chapter 3

Malware Identification and Classification

The exponential growth of internet-connected devices, particularly accelerated by the Covid-19 pandemic, has brought forth a critical global challenge: safeguarding the security of transmitted information. The integrity and functionality of these devices face significant threats from various forms of malware, leading to behavioral distortions. Consequently, a vital aspect of cybersecurity entails accurately identifying and classifying such malware, enabling the implementation of appropriate countermeasures. Existing literature has explored diverse approaches for malware identification, encompassing static and dynamic analysis techniques like signature-based, behavior-based, and heuristic-based methods. However, these approaches face a key issue of inadequately identifying unknown malware variants, often resulting in misclassifications of new strains as benign. To tackle this challenge, this study introduces a novel ensemble-based approach for identifying and classifying malware on Windows platforms, with a specific focus on detecting new and previously unknown variants. The proposed methodogy leverages multiple machine learning schemes to identify elusive unknown malware that proves challenging for existing methods.

The process of determining whether a suspicious entity is malware or benign involves an array of stages, such as malware data acquisition, pre-processing, feature extraction, transformation, selection, and classification. An overview of the methodology employed in this work is presented in Figure 3.1.

- Data Collection: Samples are collected from Windows-based platforms in the form of binary files. For this study, the malware classification dataset provided by Quick Heal is utilized.
- Data Pre-processing: Unwanted data, such as digitally signed documents, is removed from the collected dataset, focusing on images and files.
- Feature Extraction and Reduction: Execution traces are logged by analyzing the malware samples. Data mining techniques are employed to extract malware characteristics from these logs. Data mining involves the discovery of patterns and previously unknown values in large databases. During the extraction of

malware features, various elements such as byte sequences, strings, opcodes, assembly guidelines, system calls, API calls, and a variety of DLLs may be utilized. The feature extraction process employs a classifier, and PCA (Principal Component Analysis) is employed for feature reduction. This step identifies and eliminates irrelevant features from the data.

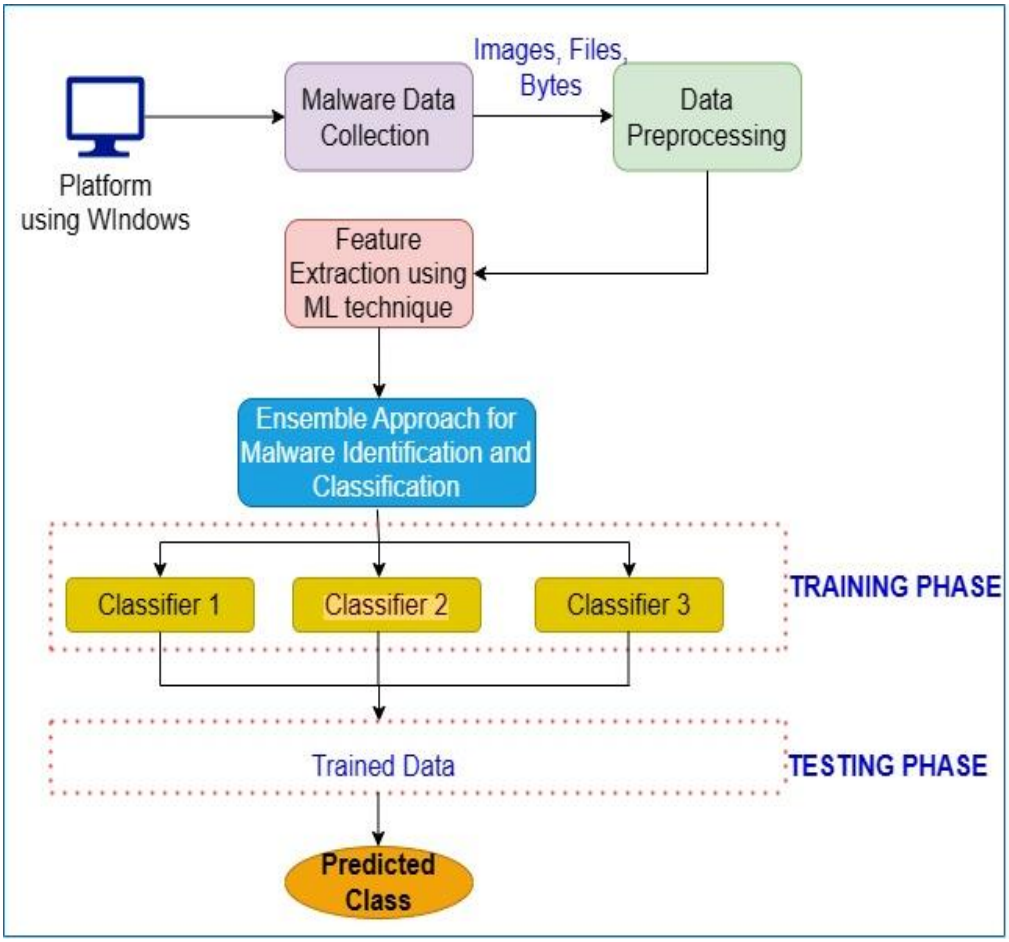

Figure 3.1: Methodology depicting flow of proposed ensemble approach

Selection and Classification: The proposed ensemble approach is used to extract malware features and perform accurate malware classification. PCA captured the data's variability while reducing dimensionality. The primary components are determined by combining the initial features. The top components (95%) that account for the majority of the variance are used for further analysis. The reduced feature set is used to train the proposed hybrid malware detection model. To confirm PCA's efficacy, performance indicators are compared (such as accuracy and precision) before and after applying

feature selection. It eliminates unwanted features and enhances the accuracy of the classification process.

3.1 Ensemble Approach for Malware Identification and Classification

Detecting and classifying malware poses significant challenges due to the objectives of malware developers, which include information theft, extortion, and network attacks. Traditional methods have been effective in identifying known malware, but they struggle with newly emerged malware, known as zero-day malware. However, the advancement of ML platforms has greatly enhanced the capabilities of malware detection models in identifying threats. ML techniques enable malware detection to be performed in two crucial steps: feature extraction and selection, followed by data classification or clustering. This proposed approach focuses on ML techniques, which can effectively identify both harmful and benign files and accurately predict the nature of previously unseen files.

The proposed approach introduces an ensemble classifier strategy for malware detection and classification. This strategy involves incorporating a base classifier into each modified training dataset, resulting in a collection of base classifiers that form an ensemble. This ensemble formation is the core principle of the approach. To achieve this, the training datasets are reorganized using various resampling or weighting methods, creating multiple variations.

3.1.1 Ensemble Classifier Design

It comprises several steps, including the clustering process and the implementation of an ensemble-based classifier for malware identification and classification. The clustering step is conducted prior to applying the ensemble classifier and utilizes the K-means clustering approach to group similar information together. The clustering is based on word frequency, where words with similar frequency indices are clustered into the same group. The number of clusters represented by the centroids is determined based on the desired quantity.

The K-means algorithm begins by selecting initial centers for the clusters from the data patterns at k points. Subsequently, the distance between the center of each cluster and the sample is determined, and the sample is assigned to the cluster that is closest. The

average value of the data objects within each newly formed cluster is computed to determine the new center for that cluster. These steps are iteratively repeated until the clustering centers of consecutive iterations do not significantly change, indicating convergence and maximum achievement of the primary clustering function. The ensemble approach consists of three phases:

Phase 1: Preparation of the ensemble involves selecting N base classifiers and choosing a meta-learning algorithm.

Phase 2: The ensemble is trained by training each of the M base learners using the training dataset. The predictions are recorded after each base learner undergoes K-fold cross-validation.

Phase 3: Testing of the ensemble is conducted using new and unknown data. The decisions made by the base learners are recorded, and the meta-learner ensemble decisions are derived from these base-level decisions.

Selection of N Base Classifiers for Ensemble

The available literature provides a wide range of classifiers, each with its own predictive capabilities. To leverage the strengths of these classifiers and create an innovative ensemble classifier, we adopt the stacked ensemble technique. This approach combines the predictions of diverse base models to achieve improved classification accuracy and reduce the risk of misclassification. In the proposed approach, we incorporate three specific base classifiers:

- Support Vector Machine (SVM): SVMs are a distinctive learning method rooted in statistical learning theory. They are constructed based on a limited number of samples from the training data, aiming to achieve optimal classification results. Initially designed for binary classification tasks, SVMs have been extended to handle large-scale data management and classification in the context of advancements in computer, network, and database technologies.
- Decision Tree (DT): DT is a generally used classification technique with applications in various real-world scenarios. This symbolic learning method constructs a hierarchical structure by analyzing the training dataset. The

structure consists of nodes and branches representing different decisions based on the attributes of the dataset.

- Logistic Regression (LR): LR is a fundamental statistical and data mining technique widely utilized by statisticians and researchers for analyzing and classifying binary and proportional response datasets. One of its key characteristics is the ability to generate probabilities automatically, making it applicable to both binary and multi-class classification problems.

Various ensemble techniques, including stacking, boosting, blending, and bagging, are available for constructing ensemble models. In this study, we employ the stacking method to create ensemble. At Level 0, SVM and DT models are built, while at Level 1, an LR model is constructed. The overall process is illustrated in Figure 3.2. Once the data has undergone pre-processing, we utilize the term frequency-inverse document frequency (TF-IDF) technique to calculate the frequency of a specific type of malware. The RF model then works on the malware frequency, taking it into account. To generate uncorrelated variables, the data is subjected to PCA, which involves dividing a set of correlated variables into linearly independent subsets. The PCA algorithm processes the malware data with the highest frequency as input and eliminates those with the lowest frequency. This reduces the number of extracted features using the PCA approach. By transforming the data into a lower-dimensional representation, PCA evaluates the effective level of variation present in the data. The PCA technique primarily intends to discover a linear transformation vector that capitalize on the data variance in the projected space, as represented in Eq. (1).

$$t_{k(i)} = w_{l(i)} T_{x_i} \qquad (1)$$

where t is a sequence or vector of values, the subscript k_((i)) denotes the ith element of a sequence, where k is another sequence or index that specifies the order or position of the elements in t. w is a matrix, where w_I(i) represents the i-th row of the matrix. The subscript l(i) refers to the i-th element of the sequence or index l. T_(x_i) denotes the transpose of the vector x_i. x_i represents the i-th input vector.

To maximize the variance, the original weight vector w_i must satisfy the following condition, as shown in Eq. (2).

$$w_i = (\sum(x_i \cdot w)^2) \qquad (2)$$

where, w_i represents the ith element of the vector w and x_i represents the i^{th} element of the vector x.

To group similar information together, an additional clustering step is applied. Malware samples with similar characteristics are clustered together, based on their frequency indices. The number of centroids is equal to the number of clusters, as determined during the calculation. The K-means algorithm starts by selecting k points as the initial cluster centers from the data patterns. Then, the distance between each sample and the center of its corresponding cluster is calculated. The sample is assigned to the closest cluster based on this distance. Afterwards, the average value of each newly formed cluster's data objects is used to calculate the new center for that cluster. These steps are iteratively repeated until the clustering centers of two consecutive iterations do not significantly change. At this point, the clustering process has converged, and the primary clustering objective has been maximized. The algorithm utilizes the Euclidean distance to compute the distance between data samples. The clustering performance is assessed using the sum of squared errors criterion. The K-means technique divides the sample set D=(x_1,x_2,…,x_m) into C=(x_1,x_2,…,x_k) clusters to minimize the squared error, as shown in Eq. (3):

$$E = \sum_{i=1}^{k} \sum x \in \|x - \mu_i\|^2 \qquad (3)$$

where E represents the total sum or cumulative value of the expression on the right-hand side of the equation. It is considered as the result or output of the equation. x represents an individual value or observation in a dataset. In the equation, x is used as a summation variable, indicating that the subsequent expression is evaluated for each value of x. k represents the number of groups or clusters in the dataset. It defines the range or limits of the summation in the equation, specifying that the expression is evaluated for values of i ranging from 1 to k. Here, i represent the index of each group or cluster in the dataset and is used as a summation variable, indicating that the subsequent expression is evaluated for each group or cluster. μ_i˙ represents the mean or centroid of the ith group or cluster. It indicates the average or central value of the observations within that particular group.

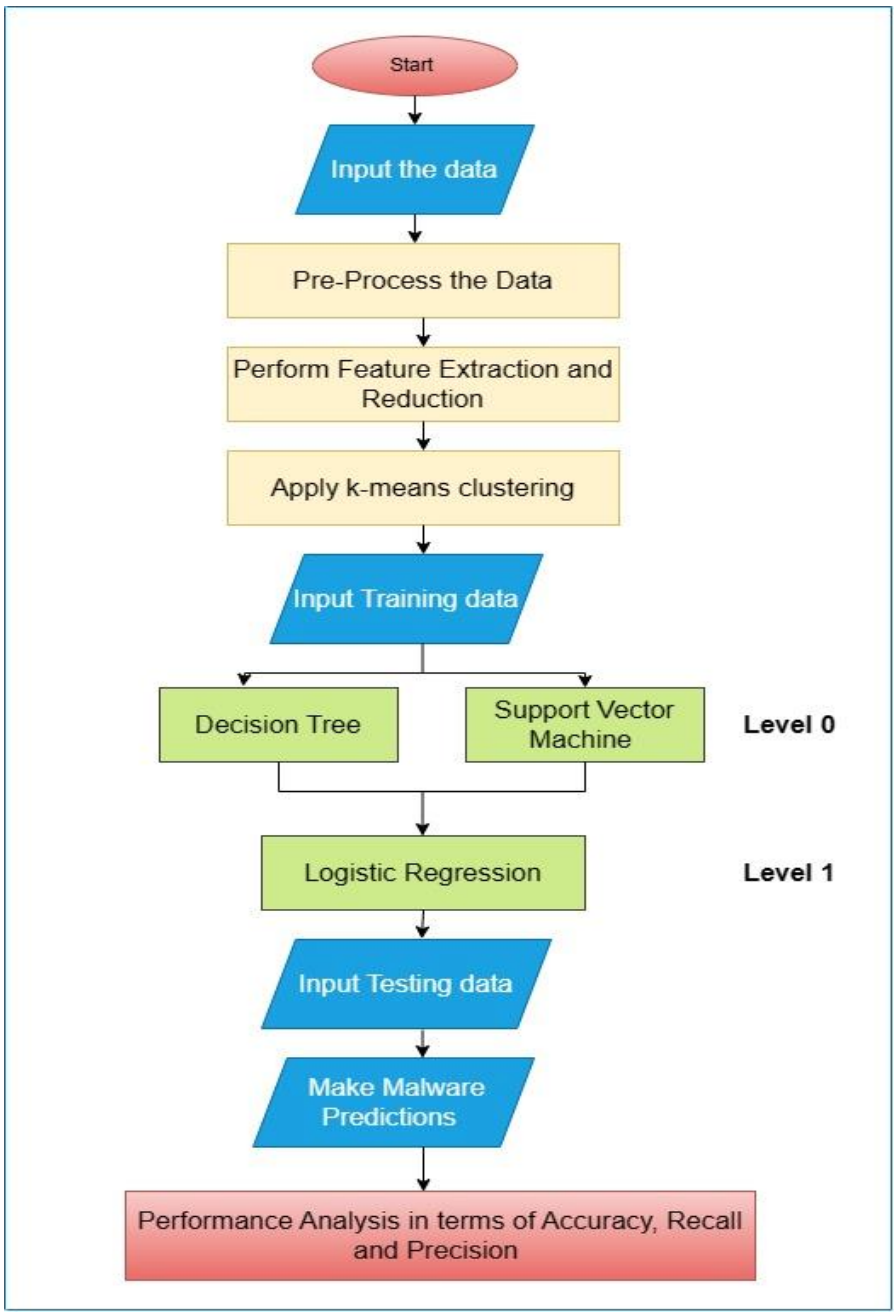

Figure 3.2: Clustering and classifiers used in Hybrid ensemble approach for malware detection.

Incorporating all the steps, the ensemble approach is developed by combining the DT, SVM, and LR classifiers. DT is used in ensemble as it supports interpretability. When interpretability and transparency are crucial, DTs are a common option since they are easy to comprehend and visualize. SVM is deployed as it has capability to handle high-dimensional data effectively. Moreover, SVMs are very effective for issues when the

numbers of features are large as compared to the number of samples. LR generates probability scores between 0 and 1, which represent the possibility of falling into a specific class, rather than binary predictions (0 or 1). Depending on the needs of the application, this probability score may be useful for making judgments, evaluating forecasts, and establishing various decision thresholds.

Chapter 4

Results and Discussion

The malware detection pipeline is a systematic procedure that starts with data preparation and encompasses duties such as data cleansing, processing of missing values, normalization, and feature extraction. It then integrates characteristics from both malicious and harmless samples into a comprehensive set of attributes. Feature hashing is employed to decrease the dimensionality of the feature space, whilst K-means clustering combines comparable data points together to create clusters. The dataset is divided into two sets: a training set and a testing set. The training set has 80% of the data, while the testing set contains 20%. The training process involves utilizing Decision Tree and Support Vector Machine models to train the models, and subsequently obtaining predictions from these trained models. The predictions from various models are aggregated and used as inputs to a higher-level model. Logistic Regression is trained by utilizing the stacked predictions to generate a conclusive decision. Evaluation metrics are stored for the purpose of referencing and comparing, while visualizations are generated to depict performance data. The models are retained for future utilization without the need for retraining. The pipeline utilizes the taught models to classify a fresh binary file and make predictions. Features are derived from the binary file, and a K-means transformation is utilized to ensure consistency of the features. Predictions are derived from the training models, and the stack predictions are combined for the new binary file. The Logistic Regression model is used to obtain the final prediction, and a final prediction label is constructed based on the stacked predictions. The forecast result and level of certainty are stored for future use. The primary data set for the proposed model is provided by Quick Heal and consists of malware and benign files. The data set is divided into two major groups- one of benign files and the other of malware files. The malware files are further divided into categories of malware family: virus, worm, and ransomware for the purpose of training and testing the proposed model.

Model Training and Evaluation

The sequence diagram presents a thorough procedure for training and evaluating models in the field of malware detection. The process encompasses the steps involved in preparing the data, saving the trained models, and evaluating the outcomes. The process commences with data preparation, encompassing the collection and preprocessing of unprocessed data, addressing any missing values, standardizing data, and readying it for feature extraction. Feature extraction encompasses the process of extracting significant features from the pre-processed data. This entails utilizing feature hashing to decrease computational complexity, and subsequently performing clustering using K-means clustering. Subsequently, the data is divided into separate training and testing sets, facilitating enhanced generalization. The process of model training includes training a Decision Tree model, a Support Vector Machine, generating stacked predictions, and training a Logistic Regression model. The ultimate model is trained by utilizing stacked predictions. Model evaluation entails assessing the performance of the trained models using several measures, including accuracy, precision, and recall. Metrics are computed for the purpose of comparing and visualizations are produced. The collected data is saved for subsequent reference and reporting purposes. Models are stored in files for the purpose of documentation and comparison, while the computed metrics are also saved for documentation and comparison. This framework provides a flexible and strong approach for machine learning applications, making it appropriate for many kinds of data and models.

Prediction Process for New Binary

The flowchart shown in Figure 4.1 offers a methodical methodology for forecasting the malignancy of a novel binary file. The procedure starts by initializing the prediction phase, when pre-trained machine learning models such as Decision Tree, SVM, and Logistic Regression are loaded from storage. The technique entails extracting pertinent characteristics from the binary file, employing feature hashing to decrease the dimensionality of the collected characteristics, and utilizing the pre-trained K-means clustering model on the hashed characteristics. The Decision Tree model is utilized to derive predictions by using its acquired decision rules. The SVM model is utilized to

derive predictions by utilizing the acquired decision boundary. The aggregated forecasts are merged to generate a resilient collection of characteristics for the ultimate model. The Logistic Regression model is employed to provide the ultimate forecast by utilizing stacked predictions. Subsequently, the ultimate forecast is assessed to ascertain whether the binary file is harmful or benign. If an item is categorized as malicious, it is stored as malicious, and if it is categorized as benign, it is stored as benign. The result is documented as "Malicious" for subsequent action or evaluation. When the binary file is determined to be harmless, it is labelled as "Benign" to prevent it from being identified as a security risk.

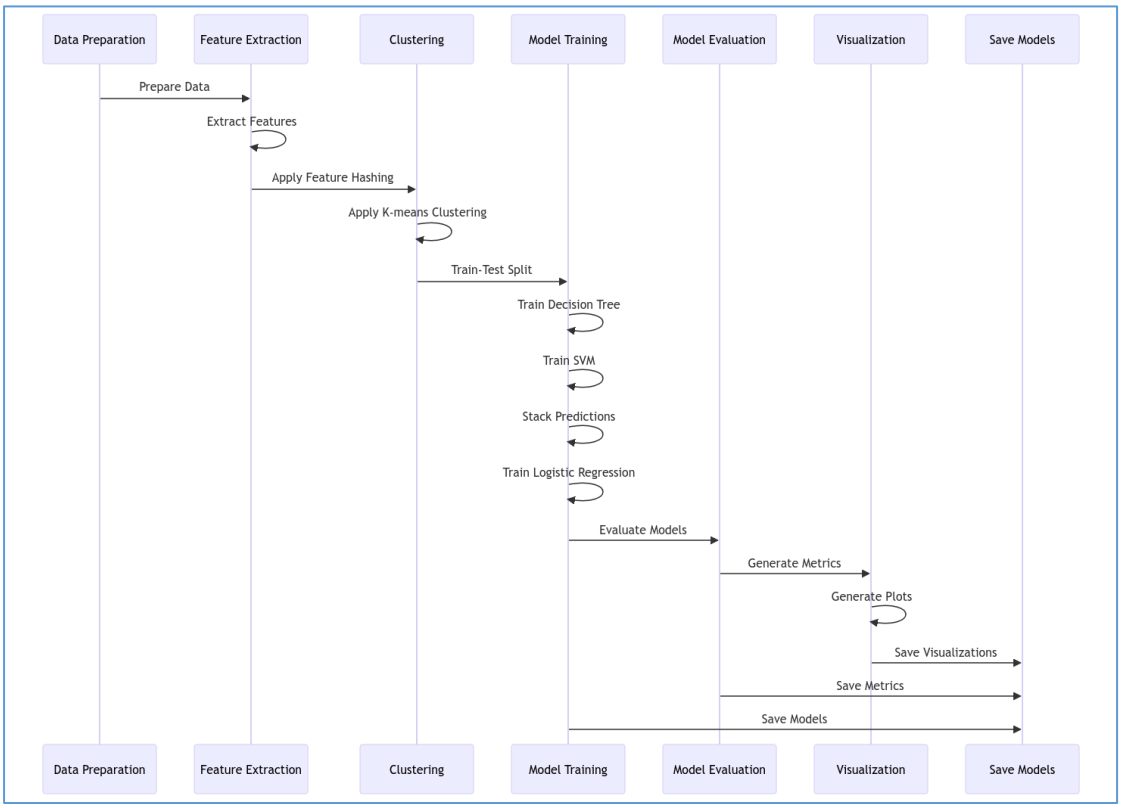

Figure 4.1: Stages involved in model training and evaluation

Through the utilization of the K-means clustering technique, it is possible to differentiate between malicious software samples and benign software samples by classifying them into distinct groups. Cluster 0 (purple) represents one group of samples, and Cluster 1 (yellow) represents another group. The data is graphically grouped according to its allocated clusters, with Cluster 0 (purple) representing one

specific set of samples. Two separate clusters can be formed from the data points that have been collected. In the K-means clustering depicted in Figure 4.2, the arrangement of data points shows a narrow dispersion with all of them clustered together. The color bar transitions from red at a value of 0.0 to 1.0, but at the value of 0.0, it displays a uniform yellow tone, indicating that all points in the plot are in one cluster. This observed homogeneity indicates the absence of more refined clusters within the dataset.

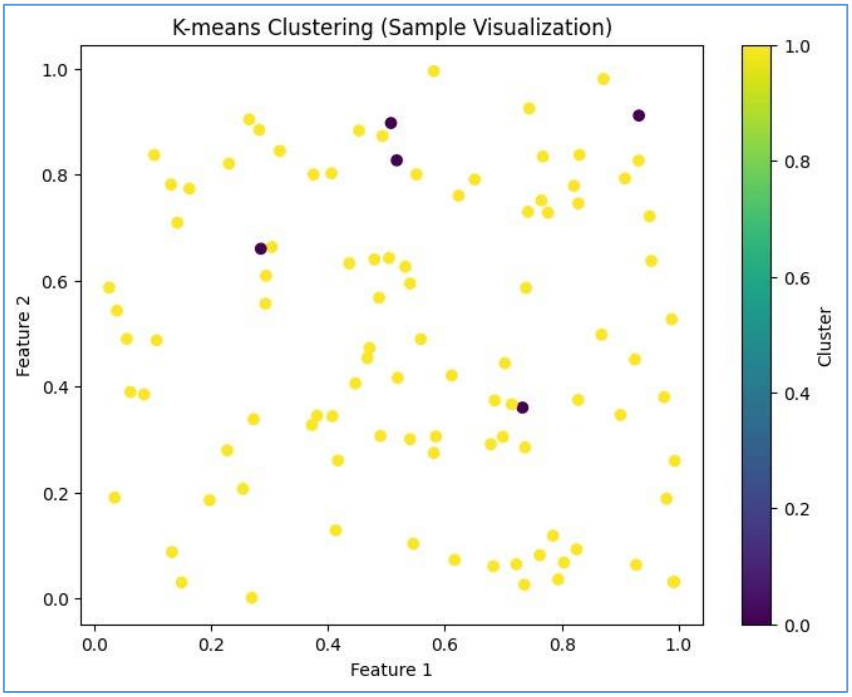

Figure 4.2: Clustering of feature vectors obtained from malware and benign samples using K-means clustering

The ability of the algorithm to differentiate between the two groups is demonstrated by the fact that Cluster 0 and Cluster 1 will be distinguished from one another. The pattern of clustering demonstrates that the selected features successfully separate the data into significant groups, which is essential for the establishment of a classification system that can be relied upon. To discriminating between the two clusters, it is vital to have features that successfully capture large variance in the dataset. The smooth distribution of feature values across the scatter plot is an indication that these features properly capture this variance.

A comparison matrix is offered to enable a direct comparison of the Support Vector Machine (SVM) and Decision Tree models predictions on a dataset. Results of the two models' classification are shown in the matrix. With 198 samples in the cell at the top-left corner (0,0), the models agree significantly. There were no situations in which both models agreed to classify a sample as malware, as seen by the absence of samples in the 1,1 cells. Four samples that show a misclassification make up the 1,0 cell. These samples were classed as benign by the SVM but malware by the Decision Tree. This difference implies that the SVM shows a different sensitivity to the features used for classification, or that the Decision Tree may be more prone to generate false positive results. A scatter plot of extended K-means clustering with additional noise is shown in Figure 4.2. Various colors for the data points indicate different clusters. The data points appear to be diagonally aligned and exhibit a linear relationship. To test the resilience of the method, noise may have been inserted, particularly among the teal and yellow points. The code demonstrates the process of training and evaluating a Decision Tree classifier using TF-IDF vectorized features extracted from text data. The train_decision_tree function loads malware and benign data paths, extracts features using TF-IDF vectorization, and trains a Decision Tree classifier. The evaluate_decision_tree function evaluates the classifier's performance by calculating accuracy, precision, recall, and F1-score, and visualizes these metrics using a bar plot.

The latest deployment of sophisticated detection algorithm has shown remarkable outcomes in detecting the existence of the specific malware category. Thorough testing on datasets demonstrated that the system regularly identifies harmful activity with considerable precision. The investigation demonstrates that the algorithm effectively identifies unique patterns and behaviors that are specific to the malware family, resulting in a notable enhancement in the speed and dependability of threat identification. The results shown here showcase the strength and effectiveness of method, highlighting its ability to improve cybersecurity measures by proactively detecting and reducing the impact of malware threats. The proposed model categorizes the identified malware into virus, worm and ransomware depending upon the model trained.

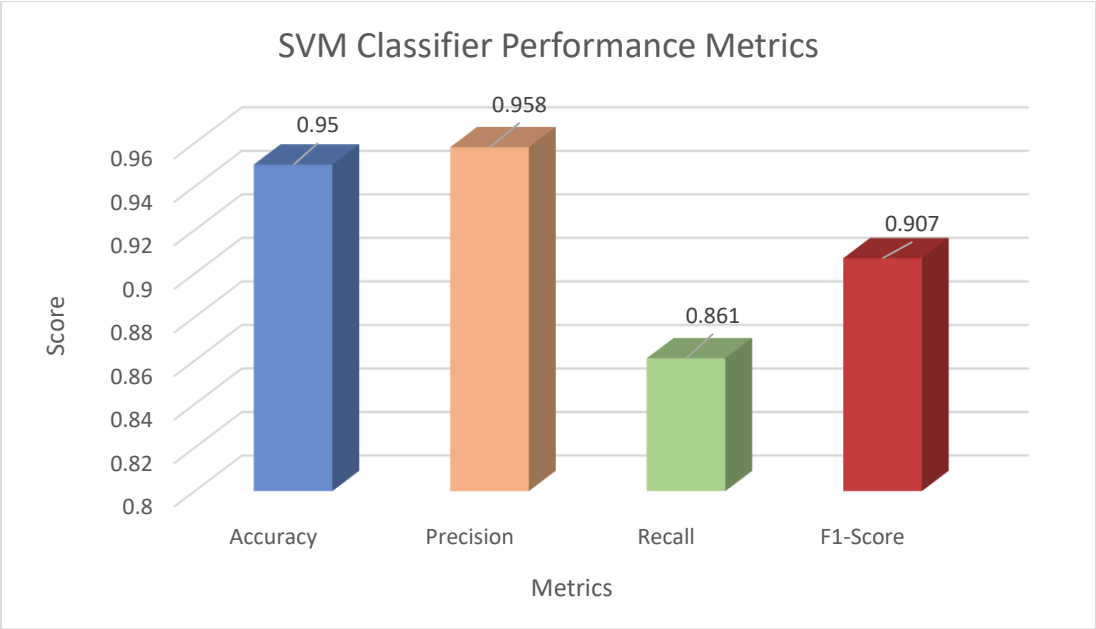

Figure 4.3: Performance matrix for SVM Classifier

Figure 4.3 is the bar chart for SVM Classifier Performance Metrics which provides an overview of the performance classifier by utilizing four essential assessment metrics. These metrics are Accuracy, Precision, Recall, and F1-score. The accuracy value is 0.95, which indicates that a significant number of predictions were realized correctly. Additionally, the precision score is 0.958, which indicates that the classifier is successful at recognizing true positives within the data. It is clear that the classifier is able to recognize the majority of true positive cases, as evidenced by the recall score, which is somewhat lower than the precision score but is still rather high. In order to ensure consistent performance in recognizing genuine positives while simultaneously decreasing false positives and false negatives, the F1-score, which is the harmonic mean of precision and recall, provides a comprehensive measure that strikes a balance between the two metrics for assessment. The graphic allows for the interpretation of the performance of the SVM classifier across all the metrics that were assessed, hence proving its efficiency and dependability in classification tasks. The fact that the SVM model has high values for metrics - accuracy, precision, recall, and F1-score jointly show that it is well-suited for differentiating between various classes with a low amount of error. Taking everything into consideration, the strong performance of the SVM classifier indicates that it is suitable for applications that require classification

capabilities that are both exact and balanced. The utilization of tools like scikit-learn for metric computation and Matplotlib for visualization highlights the methodological rigor and clarity that is present in the presentation of the model's performance.

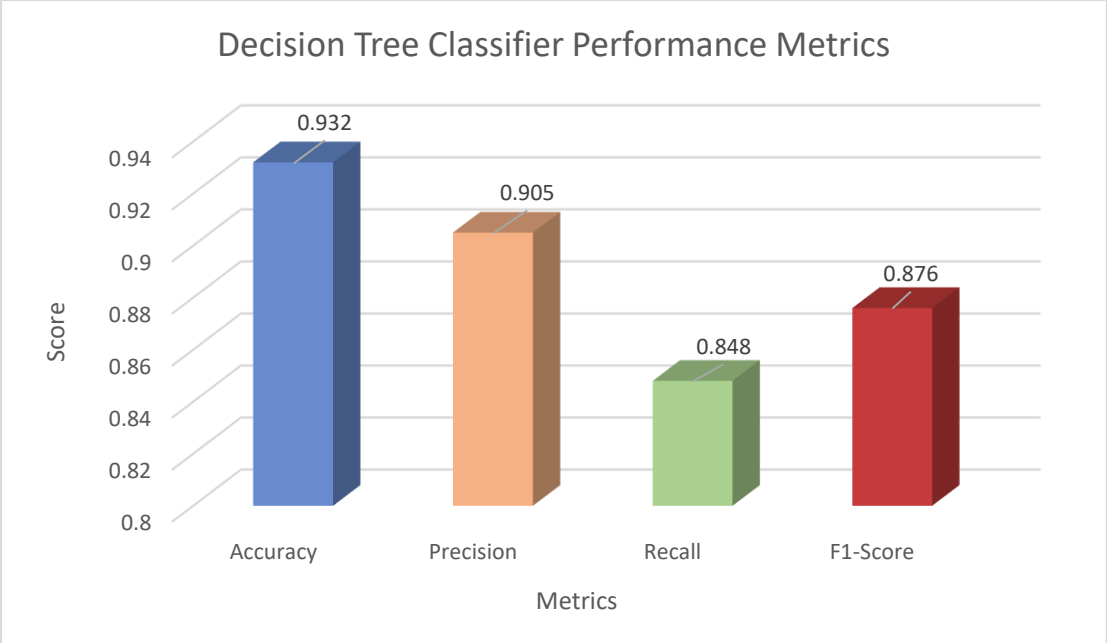

Figure 4.4: Performance matrix for Decision Tree Classifier

The performance of a DT classifier is evaluated using four crucial metrics: accuracy, precision, recall, and F1-score. Figure 4.4 is the bar chart for Decision Tree Classifier Performance Metrics which offers a comprehensive overview of the performance of a Decision Tree classifier. The classifier achieved a high level of accuracy, as evidenced by a score of 0.932, indicating that 93.2% of its predictions were correct. The precision measurement yielded a value of 0.905. To clarify, this suggests that around 90.5% of the instances identified as positive by the classifier were indeed positive. A recall value of 0.848 means that the model successfully identified 84.81% of the actual positive cases. Recall is a quantitative measure that assesses the classifier's capacity to accurately detect all instances that are positive. The F1-score was calculated to be 0.876. This value suggests that the performance of both measures was well-balanced. Overall, these performance measurements indicate that the Decision Tree classifier achieves both high accuracy and maintains a good trade-off between identifying real

positives and limiting false positives. This can be seen in the chart, where the bar heights represent these measures. The classifier strikes a commendable balance.

Figure 4.5 is the representation for Logistic Regression Classifier Performance Metrics and offers a comprehensive examination of the performance of a Logistic Regression classifier in terms of four crucial metrics: Accuracy, Precision, Recall, and F1-score. The classifier attained an Accuracy of 0.982, signifying that 98.2% of its predictions were accurate. The precision, defined as the ratio of true positive predictions to all positive predictions, was calculated to be 0.987, indicating that approximately 98.7% of the cases labeled as positive were correct.

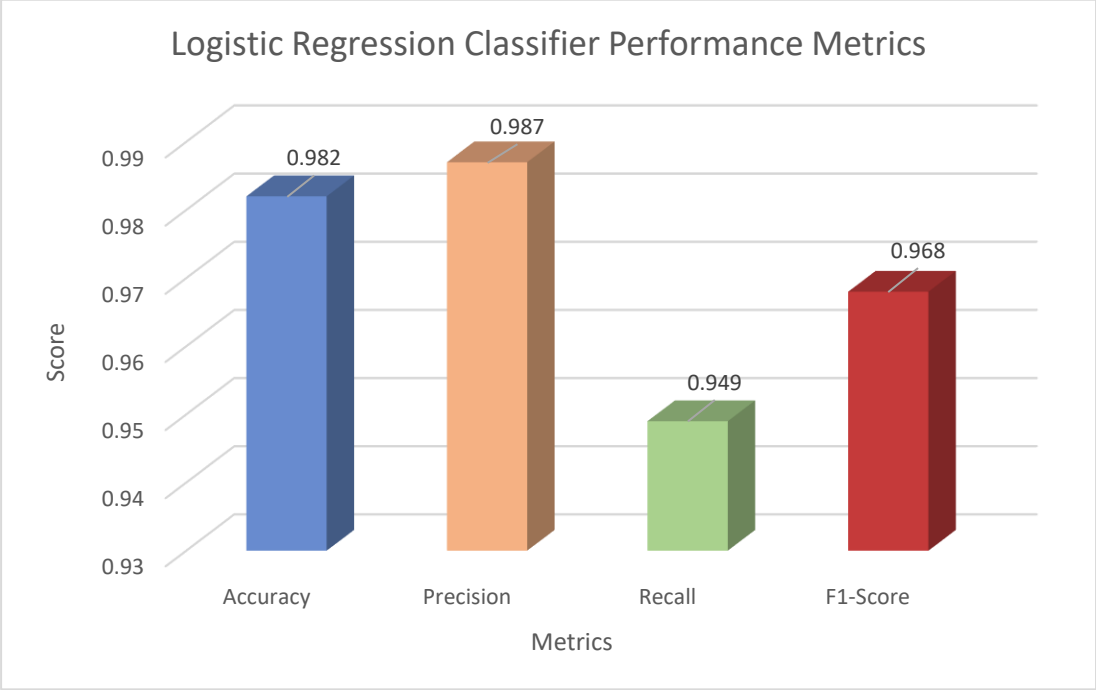

Figure 4.5: Performance matrix for Logistic Regression Classifier

The recall, which measures the classifier's ability to accurately identify all positive occurrences, was 0.949. This indicates that the model successfully captured 94.9% of the real positive examples. The F1-score was determined to be 0.968, indicating a very balanced performance between these two metrics. In summary, the Logistic Regression classifier showed outstanding performance based on these metrics. It achieved a high level of accuracy and effectively balanced the identification of true positives with the minimization of false positives, as shown by the bar heights in the chart.

At Level 0, the models used are Support Vector Machine and Decision Tree. At Level 1, the output from Level 0 is used as input for Logistic Regression. Each performance indicator, including Accuracy, Precision, Recall, and F1 Score, is displayed in individual subplots. The accuracy of Level 1 (Logistic Regression) is the highest, that is 0.982, suggesting that the combination of inputs from SVM and Decision Tree improves the prediction performance of the model. Logistic Regression demonstrates the best precision (0.987), indicating its superior capability to accurately identify positive examples in comparison to individual models at Level 0. The recall has the greatest F1 Score of 0.949, indicating its superior balance between precision and recall compared to Level 0 models. Logistic Regression at Level 1, which incorporates the results of SVM and Decision Tree from Level 0, exhibits exceptional performance in all measurable parameters. The utilization of numerous models in a layered strategy indicates that combining their individual strengths can result in improved predicted accuracy and reliability.

Table 4.1 provides a comparative analysis of the performance of three machine learning models based on Support Vector Machine (SVM), Decision Tree, and Logistic Regression. All three models demonstrated great accuracy, with Logistic Regression achieving the best accuracy of 0.982, followed by SVM with an accuracy of 0.95, and Decision Tree with an accuracy of 0.932.

Table 4.1: Comparative Analysis at both the levels

Level	Model	Accuracy	Precision	Recall	F1-Score
0	Support Vector Machine	0.95	0.958	0.861	0.907
0	Decision Tree	0.932	0.905	0.848	0.876
1	Logistic Regression	0.982	0.987	0.949	0.968

The Decision Tree accurately identified a substantial proportion of positive cases, achieving a recall rate of 0.848. Nevertheless, when evaluating accuracy, both Support Vector Machines (SVM) and Logistic Regression exhibited superior performance. Logistic Regression slightly surpassed SVM in performance, with precision rates of 0.987 and 0.958, respectively. The recall rate for Logistic Regression was 0.949, which

was the highest among the three models. SVM had a recall rate of 0.861, while Decision Tree had a rate of 0.848. The F1-score reached its peak value of 0.968 for Logistic Regression, indicating exceptional overall performance. When comparing the two, SVM and Decision Tree showed F1-scores of 0.876 and 0.907, respectively.

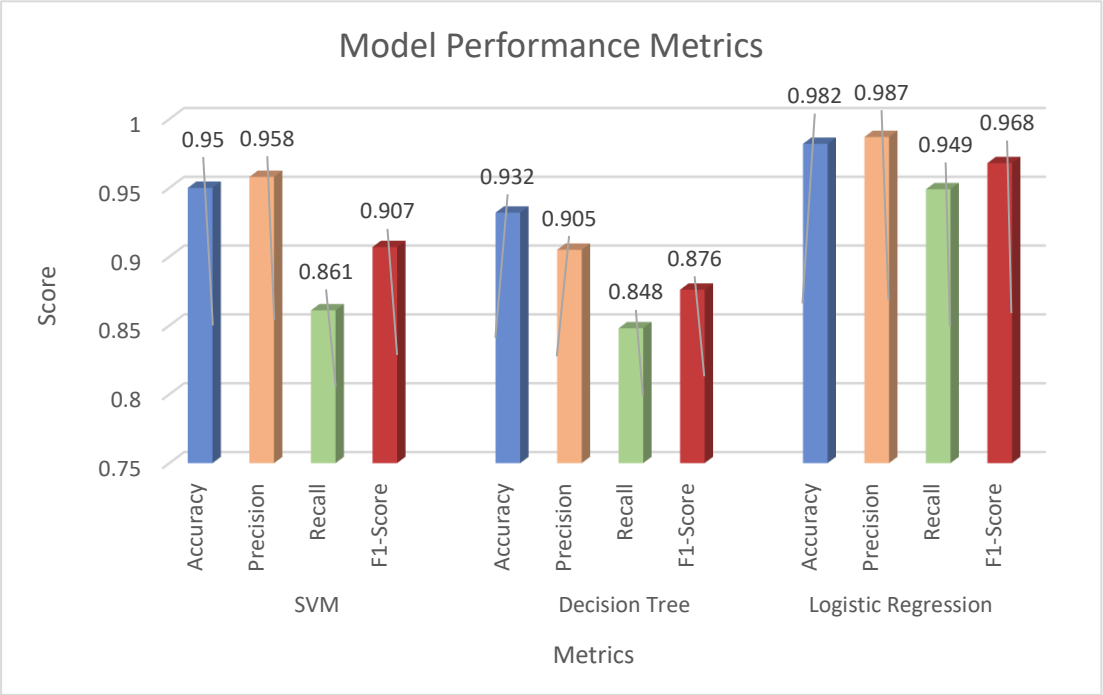

Figure 4.6: Performance matrix for SVM and Decision Tree at level 0 and for Logistic regression at Level 1.

Figure 4.6 provides a clear visual representation of the comparative performance of these models, underscoring Logistic Regression as the superior model among the three, based on the evaluated metrics. However, it is important to consider additional factors such as model interpretability, training time, and computational resources when making a final decision on model selection. The figure depicts three confusion matrices, which offer a comparative evaluation of the efficacy of three machine learning models: Support Vector Machine (SVM), Decision Tree, and Logistic Regression. Each matrix displays the number of true positives (located in the bottom-right), true negatives (located in the top-left), false positives (located in the top-right), and false negatives (located in the bottom-left) for the corresponding models. The Support Vector Machine (SVM) model exhibits strong and reliable performance, correctly identifying 199

instances as negatives and 68 instances as positives. It only misclassified 03 instances as positive when it was actually a negative, and 11 instances as negatives when they were actually positives. These measures demonstrate a low level of misclassification, highlighting the model's strong precision and recall.

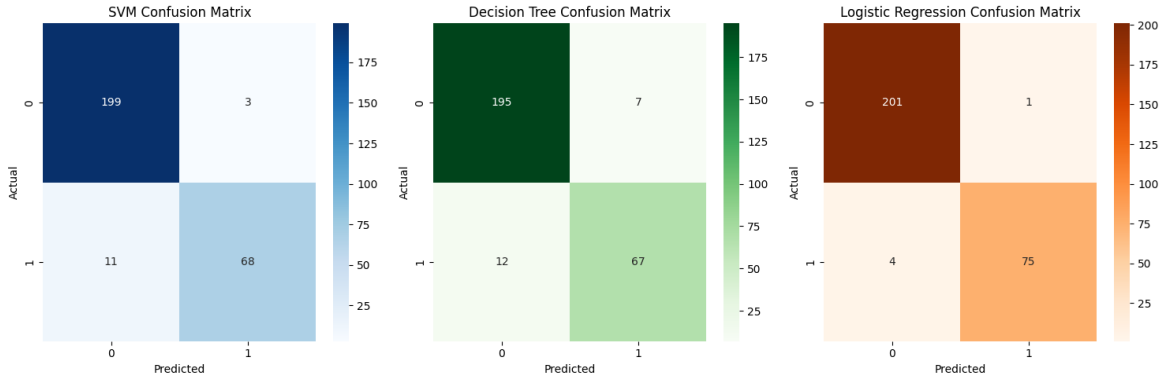

Figure 4.7: Confusion matrix for SVM and Decision Tree at level 0 and for Logistic regression at Level 1.

By contrast, the Decision Tree model exhibits 195 instances of true negatives, 67 instances of true positives, 07 instances of false positives, and 12 instances of false negatives. This model demonstrates a minor decrease in performance measures, characterized by a notable rise in both false positives and false negatives, indicating a modest decline in overall classification accuracy and dependability. The Logistic Regression model yields a total of 201 true negatives, 75 true positives, 01 false positive, and 04 false negatives. Although this model at level 1 has comparable performance to the SVM (applied at level 0) in terms of true negatives and false positives, it demonstrates a greater incidence of false negatives, suggesting a minor compromise in sensitivity. The confusion matrices offer a detailed perspective on the classification accuracy of each model, demonstrating how they handle both correct and incorrect classifications. The Support Vector Machine (SVM) and Logistic Regression models have higher performance in eliminating misclassification errors, specifically false positives, when compared to the Decision Tree model. This comprehensive comparative research highlights the significance of taking into account both accurate

and inaccurate categorization metrics when evaluating and choosing machine learning models for particular applications.

The graph depicts the accuracy of a model during four epochs, both in terms of training and validation. The accuracy measure is represented by the y-axis, while the number of epochs is represented by the x-axis. The training accuracy is shown by blue dots, whereas the validation accuracy is represented by a blue line. Both measures exhibit a steady and consistent increase over the epochs, indicating the model's enhanced performance. At the beginning, the training accuracy is roughly 0.75 and increases to around 0.90 by the fourth epoch. Similarly, the validation accuracy starts at around 0.70 and gradually rises to about 0.85. This sequence of events indicates that the model is successfully acquiring knowledge and applying it to new data, resulting in enhanced precision on the validation set as well.

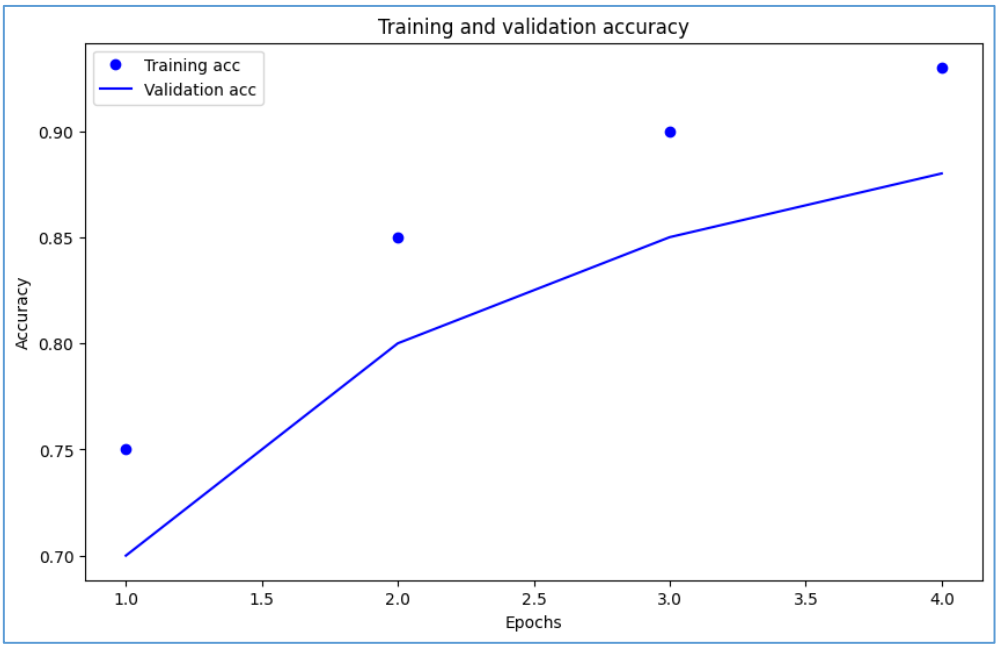

Figure 4.8: Training and validation accuracy.

The simultaneous enhancement of both the training and accuracy of validation suggests that model is not excessively conforming to the training data and is improving its performance on unknown data. This pattern indicates that the model is improving its ability to make precise predictions as the training continues, which is a positive sign of its ability to apply what it has learned to new situations and its overall effectiveness.

The bar chart displays the relative performance of three well-known machine learning models, namely Decision Tree, Support Vector Machine (SVM), and Logistic Regression, in terms of important evaluation. These metrics are essential for evaluating the effectiveness of models in classification tasks, where larger scores indicate better performance.

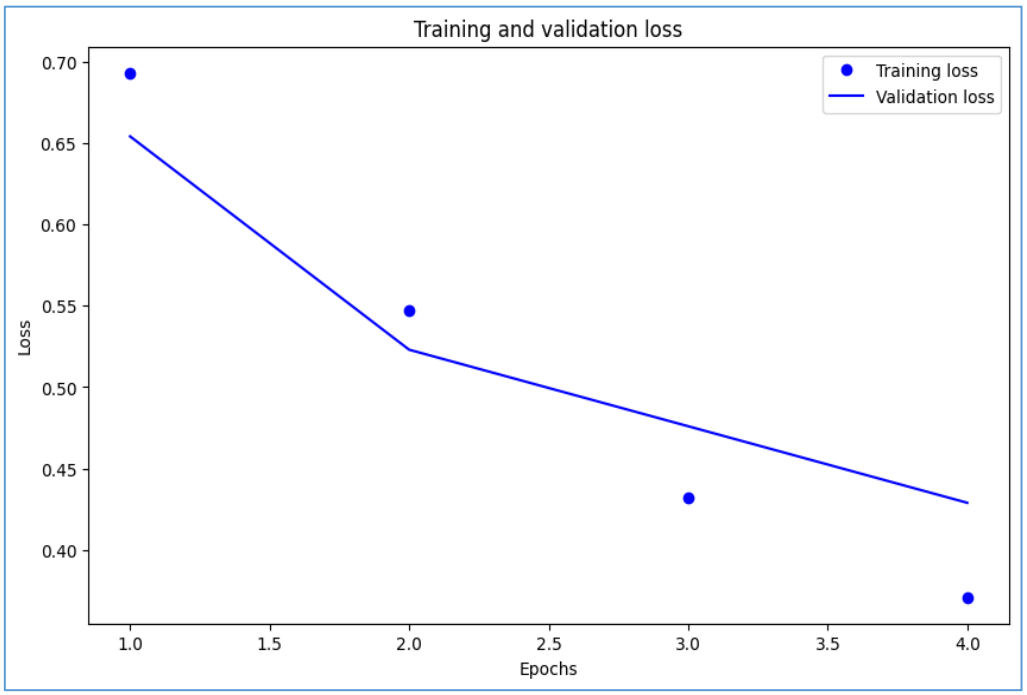

Figure 4.9: Training and validation loss.

Each model exhibited exceptional accuracy across all tested criteria, indicating its overall proficiency in classification. Nevertheless, Support Vector Machines (SVM) and Logistic Regression demonstrated a significant superiority in terms of Precision and F1-Score. This superiority implies that these models attain a more advantageous equilibrium between accurately detecting positive occurrences and reducing false positives in comparison to the Decision Tree model. Based on the performance metrics, SVM and Logistic Regression are identified as strong options for classification tasks, showcasing their effectiveness in providing precise and dependable predictions across several evaluation criteria.

Comparative Analysis with Existing Techniques

Table 4.2: Ensemble learning results with different meta-learners

Techniques	Accuracy	Precision	Recall	F1 Score
Wang et al. [45]	0.96	N/A	N/A	0.9606
Baker del Aguila et al. [48]	0.947	N/A	N/A	0.94
Saridou et al. [51]	0.918	0.893	0.943	0.918
Almaleh et al. [72]	0.98	0.99	1.00	0.99
Taha et al. [83]	0.951	0.924	0.946	0.935
Syeda et al. [94]	0.96	0.99	0.96	0.96
Cai et al. [117]	0.969	0.971	0.969	0.970
Aslan et al. [118]	0.978	N/A	N/A	0.958
Zhang et al. [119]	0.986	0.954	0.976	0.956
Proposed approach	0.983	0.987	0.95	0.969

The proposed approach exhibits greater performance in comparison to existing models as depicted in Table 4.2, attaining the high accuracy of 0.983 and the highest precision of 0.987. This demonstrates its remarkable capacity to accurately detect affirmative cases and achieve high accuracy. Nevertheless, its recall rate of 0.95, although competitive, is not the highest. The F1 score of the suggested technique is 0.969, which is comparable to that of Zhang et al. [119] (0.956), but lower than that of Cai et al. [117] (0.969). However, the suggested technique's combination of high precision and overall accuracy makes it a compelling choice for applications that require limiting false positives. Nevertheless, the potential decrease in recall should be taken into account, depending on the specific application's tolerance for false negatives.

Chapter 5

Conclusion and Future Work

In this research, we have delved into the evolving landscape of malware, identifying its diverse types and the sophisticated techniques employed by attackers to evade detection. The pervasive nature of malware poses significant challenges to the integrity, confidentiality, and availability of digital systems. The study emphasizes the critical need for robust malware analysis frameworks that can adapt to the rapidly changing threat environment.

We have explored various malware types, including viruses, worms. Each of these categories presents unique characteristics and threats, necessitating tailored detection and mitigation strategies. The research highlights the limitations of traditional static and dynamic analysis methods and underscores the need for more advanced techniques that leverage machine learning and artificial intelligence to detect and classify new and unknown malware. The development of a novel malware analysis framework provides a more comprehensive solution for identifying and mitigating threats. This framework's ability to incorporate real-time data analysis and advanced machine learning algorithms enhances its effectiveness in detecting sophisticated malware variants, including zero-day attacks. The experiments were conducted on a dataset that included malware and benign files from Windows Portable Executables (PE). The framework can be deployed to detect indications of malware within Windows based Industrial Control Systems including command injections and inappropriate communication patterns. Additionally, it may employ sandboxing techniques to conduct a secure analysis of questionable files. Another application would be in supply chain management software, logistics tracking systems. The data exchanges may be monitored for any vulnerabilities in the form of malware and thus, ensuring the integrity of sensitive data. Other industrial areas where the framework is of use are Systems used for financial transactions, Web Servers etc.

Future Scope

The discipline of malware analysis is characterized by its dynamic nature and constant evolution, which is propelled by the ever-sophisticated nature of cyber threats.

Subsequent investigations should prioritize the subsequent domains to further augment the proficiency in countering malware:

Enhanced Behavioral Analysis Techniques:

Future research should focus on enhancing behavioral analysis techniques to detect subtle and complex malware behaviors. This involves the development of sophisticated algorithms that can analyze system activities, network traffic, and user behaviors to identify anomalies indicative of malware presence. By understanding the behavior of malware in real-time, we can develop more proactive defenses. The future of malware analysis lies in the integration of advanced machine learning techniques.

Development of Comprehensive Threat Intelligence Platforms:

The development of comprehensive threat intelligence platforms that aggregate data from multiple sources, including honeypots, network sensors, and user reports, can provide a holistic view of the threat landscape. These platforms should leverage big data analytics and machine learning to identify emerging threats and predict future attack vectors. Sharing threat intelligence across organizations can also enhance collective cybersecurity resilience.

Focus on Mobile and IoT Security:

The distinctive security challenges posed by the IoT (Internet of Things) devices must be the focus area of future research due to their increasing prevalence. Developing malware detection and mitigation strategies specifically tailored for mobile and IoT environments will be critical in safeguarding these devices from cyber threats.

Implementation of Blockchain Technology:

Blockchain technology can provide a decentralized and secure framework for sharing threat intelligence and verifying the integrity of software and data. Future research should explore the use of blockchain for enhancing malware detection and response mechanisms, particularly in distributed environments where traditional security measures may be insufficient.

Emphasis on User Awareness and Training:

As human error continues to be a significant factor in the success of cyber-attacks, future research should focus on developing effective user awareness and training programs. Educating operators about malware risks and best practices for cybersecurity can help reduce the likelihood of malware infections and enhance the overall security posture of organizations.

Exploration of Quantum Computing for Malware Analysis:

Quantum computing offers fresh prospects for the investigation of malware. Further investigation is needed to explore the potential of quantum computing in improving the efficiency and precision of malware detection, as well as its ability to overcome existing encryption techniques employed by malware creators.

Regulatory and Policy Developments:

Policymakers and regulatory bodies must stay well-informed of the latest developments in malware threats and adapt regulations to ensure robust cybersecurity frameworks. Future research should focus on the impact of regulatory measures on malware analysis and how policy can support the expansion of advanced security technologies.

Information Sharing and Collaboration:

Encouraging information sharing and collaboration among industry, academia, and government organizations can lead to more effective malware analysis and mitigation strategies. Future research should explore the benefits of collaborative approaches to cybersecurity and develop frameworks for secure and efficient information sharing.

In conclusion, while noteworthy development has been added in the area of malware analysis, the continuous evolution of cyber threats demands ongoing research and innovation. By focusing on these future research areas, we can develop more robust and adaptive defenses against the ever-present threat of malware, thereby enhancing the security and resilience of the digital systems.

REFERENCES

[1] A. Khanan, Y. Abdelgadir Mohamed, A. H. H. M. Mohamed, and M. Bashir, "From Bytes to Insights: A Systematic Literature Review on Unraveling IDS Datasets for Enhanced Cybersecurity Understanding," *IEEE Access*, vol. 12, pp. 59289–59317, 2024, doi: 10.1109/ACCESS.2024.3392338.

[2] Oluwasanmi Richard Arogundade, "Network Security Concepts, Dangers, and Defense Best Practical," *Computer Engineering and Intelligent Systems*, Mar. 2023, doi: 10.7176/ceis/14-2-03.

[3] M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang, "Evolution of Malware Threats and Techniques: A Review," *International Journal of Communication Networks and Information Security*, vol. 12, no. 3, pp. 326–337, Dec. 2020, doi: 10.17762/ijcnis.v12i3.4723.

[4] M. Boholm, "Twenty-five years of cyber threats in the news: A study of Swedish newspaper coverage (1995-2019)," *J Cybersecur*, vol. 7, no. 1, 2021, doi: 10.1093/cybsec/tyab016.

[5] F. Sulianta, "Comparison of The Computer Viruses from Time to Time," *ASIA AND THE CAUCASUS English Edition*, vol. 23, p. 2022, doi: 10.37178/ca-c.23.1.139.

[6] T. M. Chen and J.-M. Robert, "Worm epidemics in high-speed networks," *Computer (Long Beach Calif)*, vol. 37, no. 6, pp. 48–53, Jun. 2004, doi: 10.1109/MC.2004.36.

[7] G. Ali, Maad M. Mijwil, Bosco Apparatus Buruga, and Mostafa Abotaleb, "A Comprehensive Review on Cybersecurity Issues and Their Mitigation Measures in FinTech," *Iraqi Journal for Computer Science and Mathematics*, vol. 5, no. 3, pp. 45–91, Jun. 2024, doi: 10.52866/ijcsm.2024.05.03.004.

[8] M. A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao, "The Nocebo * Effect on the Web: An Analysis of Fake Anti-Virus Distribution," in *USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More (Berkeley, CA, USA), LEET'10, USENIX Association*, 2010.

[9] A. L. Y. Ren, C. T. Liang, I. J. Hyug, S. N. Brohi, and N. Z. Jhanjhi, "A Three-Level Ransomware Detection and Prevention Mechanism," *EAI Endorsed Transactions on Energy Web*, vol. 7, no. 26, 2020, doi: 10.4108/eai.13-7-2018.162691.

[10] A. O. Almashhadani, M. Kaiiali, S. Sezer, and P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," *IEEE Access*, vol. 7, pp. 47053–47067, 2019, doi: 10.1109/ACCESS.2019.2907485.

[11] Val Saengphaibu, "A Brief History of The Evolution of Malware." FortiGuard Labs Threat Research, 2022.

[12] D. Yadav, G. Kumar, D. Lakshmi Kameshwari, V. K. Gunjan, and S. Kumar, "Malware Techniques and Its Effect: A Survey," in *Lecture Notes in Electrical Engineering*, vol. 828,

Springer Science and Business Media Deutschland GmbH, 2022, pp. 1215–1225. doi: 10.1007/978-981-16-7985-8_127.

[13] J. A. Reuben and N. Ware, “Approach to Handling Cyber Security Risks in Supply Chain of Defence Sector,” *Industrial Engineering Journal*, vol. 12, no. 7, 2019.

[14] M. Bayzid, M. Shoikot, J. Hossain, and A. Rahman, “Keylogger Detection using Memory Forensic and Network Monitoring,” *Int J Comput Appl*, vol. 177, no. 11, pp. 17–21, Oct. 2019, doi: 10.5120/ijca2019919483.

[15] X. Zhang, O. Upton, N. L. Beebe, and K. K. R. Choo, “IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers,” *Forensic Science International: Digital Investigation*, vol. 32, Apr. 2020, doi: 10.1016/j.fsidi.2020.300926.

[16] Sudhakar and S. Kumar, “An emerging threat Fileless malware: a survey and research challenges,” *Cybersecurity*, vol. 3, no. 1, Dec. 2020, doi: 10.1186/s42400-019-0043-x.

[17] O. Khalid *et al.*, “An Insight into the Machine-Learning-Based Fileless Malware Detection,” *Sensors*, vol. 23, no. 2, Jan. 2023, doi: 10.3390/s23020612.

[18] S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi, “A Survey on malware analysis and mitigation techniques,” May 01, 2019, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2019.01.002.

[19] G. Nguyen, B. M. Nguyen, D. Tran, and L. Hluchy, “A heuristics approach to mine behavioural data logs in mobile malware detection system,” *Data Knowl Eng*, vol. 115, pp. 129–151, May 2018, doi: 10.1016/j.datak.2018.03.002.

[20] M. S. Akhtar and T. Feng, “Evaluation of Machine Learning Algorithms for Malware Detection,” *Sensors*, vol. 23, no. 2, Jan. 2023, doi: 10.3390/s23020946.

[21] V. Kuriyal, D. Bordoloi, and V. Tripathi, “A Comprehensive Study on Malware Detection Techniques Using Machine Learning,” *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, no. 7, 2021, [Online]. Available: www.ijseas.com

[22] H. Y. Kwon, T. Kim, and M. K. Lee, “Advanced Intrusion Detection Combining Signature-Based and Behavior-Based Detection Methods,” *Electronics (Switzerland)*, vol. 11, no. 6, Mar. 2022, doi: 10.3390/electronics11060867.

[23] M. Goyal and R. Kumar, “A Survey on Malware Classification Using Machine Learning and Deep Learning,” *International Journal of Computer Networks and Applications*, vol. 8, no. 6, pp. 758–775, Nov. 2021, doi: 10.22247/ijcna/2021/210724.

[24] A. Bensaoud and J. Kalita, “CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls,” *Knowl Based Syst*, vol. 290, p. 111543, Apr. 2024, doi: 10.1016/J.KNOSYS.2024.111543.

[25] M. Sai, A. Tyagi, K. Panda, and S. Kumar, “Machine learning-based malware detection using stacking of opcodes and bytecode sequences,” in *PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing*, Institute of

Electrical and Electronics Engineers Inc., 2022, pp. 204–209. doi: 10.1109/PDGC56933.2022.10053307.

[26] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, “Windows PE Malware Detection Using Ensemble Learning,” *Informatics*, vol. 8, no. 1, Mar. 2021, doi: 10.3390/informatics8010010.

[27] H. Manthena, S. Shajarian, J. Kimmell, M. Abdelsalam, S. Khorsandroo, and M. Gupta, “Explainable Malware Analysis: Concepts, Approaches and Challenges,” Sep. 2024, [Online]. Available: http://arxiv.org/abs/2409.13723

[28] V. Gazeau, K. Gupta, and M. K. An, “Advancements of Machine Learning in Malware and Intrusion Detections,” in *Proceedings of the 2024 IEEE International Conference on Computer, Information, and Telecommunication Systems, CITS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/CITS61189.2024.10608018.

[29] T. Zoppi, A. Ceccarelli, T. Capecchi, and A. Bondavalli, “Unsupervised Anomaly Detectors to Detect Intrusions in the Current Threat Landscape,” *ACM/IMS Transactions on Data Science*, vol. 2, no. 2, pp. 1–26, May 2021, doi: 10.1145/3441140.

[30] A. H. Salem, S. M. Azzam, O. E. Emam, and A. A. Abohany, “Advancing cybersecurity: a comprehensive review of AI-driven detection techniques,” *J Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-024-00957-y.

[31] V. Singh *et al.*, “Rising Threats in Expert Applications and Solutions,” in *Advances in Intelligent Systems and Computing*, vol. 1187. [Online]. Available: http://www.springer.com/series/11156

[32] H. Y. Kwon, T. Kim, and M. K. Lee, “Advanced Intrusion Detection Combining Signature-Based and Behavior-Based Detection Methods,” *Electronics (Switzerland)*, vol. 11, no. 6, Mar. 2022, doi: 10.3390/electronics11060867.

[33] T. Sommestad, H. Holm, and D. Steinvall, “Variables influencing the effectiveness of signature-based network intrusion detection systems,” *Information Security Journal*, vol. 31, no. 6, pp. 711–728, 2022, doi: 10.1080/19393555.2021.1975853.

[34] R. M. Sharma and C. P. Agrawal, “MH-DLdroid: A Meta-Heuristic and Deep Learning-Based Hybrid Approach for Android Malware Detection,” *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 4, pp. 425–435, 2022, doi: 10.22266/ijies2022.0831.38.

[35] Kirubavathi G, “Behavioural Based Detection of Android Ransomware Using Machine Learning Techniques,” 2023, doi: 10.21203/rs.3.rs-2555218/v1.

[36] K. Lee, S. Y. Lee, and K. Yim, “Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems,” *IEEE Access*, vol. 7, pp. 110205–110215, 2019, doi: 10.1109/ACCESS.2019.2931136.

[37] I. Shhadat, B. Bataineh, A. Hayajneh, and Z. A. Al-Sharif, “The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware,” in

Procedia Computer Science, Elsevier B.V., 2020, pp. 917–922. doi: 10.1016/j.procs.2020.03.110.

[38] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[39] P. Akshara and B. Rudra, "Study of a Hybrid Approach Towards Malware Detection in Executable Files," *SN Comput Sci*, vol. 2, no. 4, Jul. 2021, doi: 10.1007/s42979-021-00672-y.

[40] S. Shukla, G. Kolhe, S. M. P. D, and S. Rafatirad, "MicroArchitectural Events and Image Processing-based Hybrid Approach for Robust Malware Detection," in *Proceedings of the International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion*, New York, NY, USA: ACM, Oct. 2019, pp. 1–2. doi: 10.1145/3349569.3351538.

[41] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *ACM International Conference Proceeding Series*, 2011. doi: 10.1145/2016904.2016908.

[42] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, and J. Chen, "Detection of Malicious Code Variants Based on Deep Learning," *IEEE Trans Industr Inform*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018, doi: 10.1109/TII.2018.2822680.

[43] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[44] M. M. Abualhaj, A. A. Abu-Shareha, Q. Y. Shambour, S. N. Al-Khatib, and M. O. Hiari, "Tuning the k value in k-nearest neighbors for malware detection," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 2, p. 2275, Jun. 2024, doi: 10.11591/ijai.v13.i2.pp2275-2282.

[45] P. Wang, T. Lin, D. Wu, J. Zhu, and J. Wang, "TTDAT: Two-Step Training Dual Attention Transformer for Malware Classification Based on API Call Sequences," *Applied Sciences*, vol. 14, no. 1, p. 92, Dec. 2023, doi: 10.3390/app14010092.

[46] M. H. L. Louk and B. A. Tama, "Tree-Based Classifier Ensembles for PE Malware Analysis: A Performance Revisit," *Algorithms*, vol. 15, no. 9, Sep. 2022, doi: 10.3390/a15090332.

[47] M. H. Ali *et al.*, "Threat Analysis and Distributed Denial of Service (DDoS) Attack Recognition in the Internet of Things (IoT)," *Electronics (Switzerland)*, vol. 11, no. 3, Feb. 2022, doi: 10.3390/electronics11030494.

[48] R. Baker del Aguila, C. D. Contreras Pérez, A. G. Silva-Trujillo, J. C. Cuevas-Tello, and J. Nunez-Varela, "Static Malware Analysis Using Low-Parameter Machine Learning Models," *Computers*, vol. 13, no. 3, Mar. 2024, doi: 10.3390/computers13030059.

[49] K. Singh Sangher, A. Singh, and H. M. Pandey, “Signature based Ransomware detection based on optimizations approaches using RandomClassier and CNN algorithms,” *International Journal of System Assurance Engineering and Management*, doi: 10.21203/rs.3.rs-2716621/v1.

[50] B. Xuan, J. Li, and Y. Song, “SFCWGAN-BiTCN with Sequential Features for Malware Detection,” *Applied Sciences (Switzerland)*, vol. 13, no. 4, Feb. 2023, doi: 10.3390/app13042079.

[51] B. Saridou, J. R. Rose, S. Shiaeles, and B. Papadopoulos, “SAGMAD—A Signature Agnostic Malware Detection System based on Binary Visualisation and Fuzzy Sets,” *Electronics (Switzerland)*, vol. 11, no. 7, Apr. 2022, doi: 10.3390/electronics11071044.

[52] B. T. Hammad, N. Jamil, I. T. Ahmed, Z. M. Zain, and S. Basheer, “Robust Malware Family Classification Using Effective Features and Classifiers,” *Applied Sciences (Switzerland)*, vol. 12, no. 15, Aug. 2022, doi: 10.3390/app12157877.

[53] T. Vyšniūnas, D. Čeponis, N. Goranin, and A. Čenys, “Risk-Based System-Call Sequence Grouping Method for Malware Intrusion Detection,” *Electronics (Switzerland)*, vol. 13, no. 1, Jan. 2024, doi: 10.3390/electronics13010206.

[54] S. Aziz, M. Irshad, S. A. Haider, J. Wu, D. N. Deng, and S. Ahmad, “Protection of a smart grid with the detection of cyber- malware attacks using efficient and novel machine learning models,” *Front Energy Res*, vol. 10, Aug. 2022, doi: 10.3389/fenrg.2022.964305.

[55] N. Syuhada Selamat and F. Hani Mohd Ali, “Polymorphic Malware Detection based on Supervised Machine Learning,” *Journal of Positive School Psychology*, vol. 6, no. 3, pp. 8538–8547, 2022, [Online]. Available: http://journalppw.com

[56] S. S. Shafin, G. Karmakar, and I. Mareels, “Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications,” *Sensors*, vol. 23, no. 11, Jun. 2023, doi: 10.3390/s23115348.

[57] D. K. A. *et al.*, “Obfuscated Malware Detection in IoT Android Applications Using Markov Images and CNN,” *IEEE Syst J*, vol. 17, no. 2, pp. 2756–2766, Jun. 2023, doi: 10.1109/JSYST.2023.3238678.

[58] J. Busch, A. Kocheturov, V. Tresp, and T. Seidl, “NF-GNN: Network Flow Graph Neural Networks for Malware Detection and Classification,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jul. 2021, pp. 121–132. doi: 10.1145/3468791.3468814.

[59] T. Van Dao, H. Sato, and M. Kubo, “MLP-Mixer-Autoencoder: A Lightweight Ensemble Architecture for Malware Classification,” *Information (Switzerland)*, vol. 14, no. 3, Mar. 2023, doi: 10.3390/info14030167.

[60] A. M. Alnajim, S. Habib, M. Islam, R. Albelaihi, and A. Alabdulatif, “Mitigating the Risks of Malware Attacks with Deep Learning Techniques,” *Electronics (Switzerland)*, vol. 12, no. 14, Jul. 2023, doi: 10.3390/electronics12143166.

[61] A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, “Metaheuristics with Deep Learning Model for Cybersecurity and Android Malware Detection and Classification,” *Applied Sciences (Switzerland)*, vol. 13, no. 4, Feb. 2023, doi: 10.3390/app13042172.

[62] Y. Zhou, B. T. Liu, K. Zhou, and S. F. Shen, “Malware propagation model of fractional order, optimal control strategy and simulations,” *Front Phys*, vol. 11, 2023, doi: 10.3389/fphy.2023.1201053.

[63] M. Dener, G. Ok, and A. Orman, “Malware Detection Using Memory Analysis Data in Big Data Environment,” *Applied Sciences (Switzerland)*, vol. 12, no. 17, Sep. 2022, doi: 10.3390/app12178604.

[64] M. Altaiy, İ. Yıldız, and B. Uçan, “MALWARE DETECTION USING DEEP LEARNING ALGORITHMS,” *AURUM Journal of Engineering Systems and Architecture*, vol. 7, no. 1, pp. 11–26, 2023, doi: 10.53600/ajesa.1321170.

[65] A. Sharma, P. Malacaria, and M. Khouzani, “Malware Detection Using 1-Dimensional Convolutional Neural Networks,” in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, Jun. 2019, pp. 247–256. doi: 10.1109/EuroSPW.2019.00034.

[66] C. Dike Chukunda, D. Matthias, E. Bennett, and C. Author, “Malware Detection and Classification System Using Random Forest,” *Quest Journals Journal of Software Engineering and Simulation*, vol. 8, no. 5, pp. 2321–3809, 2022.

[67] S. A. Hashmi, “Malware Detection and Classification on Different Dataset by Hybridization of CNN and Machine Learning,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 6s, pp. 650–667.

[68] A. Jamal, M. Faisal Hayat, and M. Nasir, “Malware Detection and Classification in IoT Network using ANN,” *Mehran University Research Journal of Engineering and Technology*, vol. 41, no. 1, pp. 80–91, Jan. 2022, doi: 10.22581/muet1982.2201.08.

[69] Sayali Khirid, Sakshi Veer, Tanushika Gupta, Vishwajeet Waychal, and Mrs. Asmita R. Kamble, “Malware Detection and Classification Framework for IOT Devices,” *International Journal of Advanced Research in Science, Communication and Technology*, pp. 1–8, May 2022, doi: 10.48175/ijarsct-3877.

[70] Z.-P. Pan, C. Feng, and C.-J. Tang, “Malware Classification Based on the Behavior Analysis and Back Propagation Neural Network,” *ITM Web of Conferences*, vol. 7, p. 02001, Nov. 2016, doi: 10.1051/itmconf/20160702001.

[71] B. Bashari Rad, M. Kazem Hassan Nejad, and M. Shahpasand, “Malware Classification and Detection using Artificial Neural Network,” *610 Journal of Engineering Science and Technology Special Issue on ICCSIT*, vol. 12, pp. 14–23, 2018.

[72] A. Almaleh, R. Almushabb, and R. Ogran, “Malware API Calls Detection Using Hybrid Logistic Regression and RNN Model,” *Applied Sciences (Switzerland)*, vol. 13, no. 9, May 2023, doi: 10.3390/app13095439.

[73] E. Gandotra, D. Bansal, and S. Sofat, “Malware Analysis and Classification: A Survey,” *Journal of Information Security*, vol. 05, no. 02, pp. 56–64, 2014, doi: 10.4236/jis.2014.52006.

[74] R. Shrestha, A. Omidkar, S. A. Roudi, R. Abbas, and S. Kim, “Machine-learning-enabled intrusion detection system for cellular connected uav networks,” *Electronics (Switzerland)*, vol. 10, no. 13, Jul. 2021, doi: 10.3390/electronics10131549.

[75] S. A. Habtor and A. H. H. Dahah, “Machine-Learning Classifiers for Malware Detection Using Data Features,” *Journal of ICT Research and Applications*, vol. 15, no. 3, pp. 265–290, Dec. 2021, doi: 10.5614/ITBJ.ICT.RES.APPL.2021.15.3.5.

[76] L. Hammood, İ. A. Doğru, and K. Kılıç, “Machine Learning-Based Adaptive Genetic Algorithm for Android Malware Detection in Auto-Driving Vehicles,” *Applied Sciences (Switzerland)*, vol. 13, no. 9, May 2023, doi: 10.3390/app13095403.

[77] D. A. Kumar and S. K. Das, “Machine Learning Approach for Malware Detection and Classification Using Malware Analysis Framework,” *Original Research Paper International Journal of Intelligent Systems and Applications in Engineering IJISAE*, vol. 2023, no. 1.

[78] A. Mpatziakas, A. Drosou, S. Papadopoulos, and D. Tzovaras, “IoT threat mitigation engine empowered by artificial intelligence multi-objective optimization,” *Journal of Network and Computer Applications*, vol. 203, p. 103398, Jul. 2022, doi: 10.1016/j.jnca.2022.103398.

[79] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, “Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm,” *IEEE Access*, vol. 8, pp. 206303–206324, 2020, doi: 10.1109/ACCESS.2020.3036491.

[80] F. Zhang, K. Li, and Z. Ren, “Improving Adversarial Robustness of Ensemble Classifiers by Diversified Feature Selection and Stochastic Aggregation,” *Mathematics*, vol. 12, no. 6, Mar. 2024, doi: 10.3390/math12060834.

[81] N. Anđelić, S. Baressi Šegota, and Z. Car, “Improvement of Malicious Software Detection Accuracy through Genetic Programming Symbolic Classifier with Application of Dataset Oversampling Techniques,” *Computers*, vol. 12, no. 12, Dec. 2023, doi: 10.3390/computers12120242.

[82] Z. Zhao, D. Zhao, S. Yang, and L. Xu, “Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model,” *Security and Communication Networks*, vol. 2023, 2023, doi: 10.1155/2023/6390023.

[83] A. Taha, O. Barukab, and S. Malebary, “Fuzzy integral-based multi-classifiers ensemble for android malware classification,” *Mathematics*, vol. 9, no. 22, Nov. 2021, doi: 10.3390/math9222880.

[84] J. Toldinas, A. Venčkauskas, A. Liutkevičius, and N. Morkevičius, “Framing Network Flow for Anomaly Detection Using Image Recognition and Federated Learning,” *Electronics (Switzerland)*, vol. 11, no. 19, Oct. 2022, doi: 10.3390/electronics11193138.

[85] C. Palma, A. Ferreira, and M. Figueiredo, “Explainable Machine Learning for Malware Detection on Android Applications †,” *Information (Switzerland)*, vol. 15, no. 1, Jan. 2024, doi: 10.3390/info15010025.

[86] J. Ha and H. Roh, “Experimental evaluation of malware family classification methods from sequential information of tls-encrypted traffic,” *Electronics (Switzerland)*, vol. 10, no. 24, Dec. 2021, doi: 10.3390/electronics10243180.

[87] F. A. Abdulazeez, I. T. Ahmed, and B. T. Hammad, “Examining the Performance of Various Pretrained Convolutional Neural Network Models in Malware Detection,” *Applied Sciences*, vol. 14, no. 6, p. 2614, Mar. 2024, doi: 10.3390/app14062614.

[88] R. Damaševičius, A. Venčkauskas, J. Toldinas, and Š. Grigaliūnas, “Ensemble-based classification using neural networks and machine learning models for windows pe malware detection,” *Electronics (Switzerland)*, vol. 10, no. 4, pp. 1–26, Feb. 2021, doi: 10.3390/electronics10040485.

[89] A. A. Almazroi and N. Ayub, “Enhancing Smart IoT Malware Detection: A GhostNet-based Hybrid Approach,” *Systems*, vol. 11, no. 11, Nov. 2023, doi: 10.3390/systems11110547.

[90] M. H. Kabir, A. Hasnat, A. J. Mahdi, M. N. Hasan, J. A. Chowdhury, and I. M. Fahim, “Enhancing Insider Malware Detection Accuracy with Machine Learning Algorithms †,” *Engineering Proceedings*, vol. 58, no. 1, 2023, doi: 10.3390/ecsa-10-16234.

[91] M. Abuthawabeh and K. Mahmoud, “Enhanced android malware detection and family classification, using conversation-level network traffic features,” *International Arab Journal of Information Technology*, vol. 17, no. 4 Special Issue, pp. 607–614, 2020, doi: 10.34028/iajit/17/4A/4.

[92] Z. Chen, S. Xing, and X. Ren, “Efficient Windows malware identification and classification scheme for plant protection information systems,” *Front Plant Sci*, vol. 14, 2023, doi: 10.3389/fpls.2023.1123696.

[93] Y. Zhou, Y. Wang, K. Zhou, S. F. Shen, and W. X. Ma, “Dynamical behaviors of an epidemic model for malware propagation in wireless sensor networks,” *Front Phys*, vol. 11, 2023, doi: 10.3389/fphy.2023.1198410.

[94] D. Z. Syeda and M. N. Asghar, “Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning,” *Applied Sciences*, vol. 14, no. 3, p. 1015, Jan. 2024, doi: 10.3390/app14031015.

[95] F. Taher, O. AlFandi, M. Al-kfairy, H. Al Hamadi, and S. Alrabaee, “DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection,” *Applied Sciences (Switzerland)*, vol. 13, no. 13, Jul. 2023, doi: 10.3390/app13137720.

[96] H. Babbar, S. Rani, D. K. Sah, S. A. AlQahtani, and A. Kashif Bashir, “Detection of Android Malware in the Internet of Things through the K-Nearest Neighbor Algorithm,” *Sensors*, vol. 23, no. 16, Aug. 2023, doi: 10.3390/s23167256.

[97] A. I. A. Alzahrani, M. Ayadi, M. M. Asiri, A. Al-Rasheed, and A. Ksibi, “Detecting the Presence of Malware and Identifying the Type of Cyber Attack Using Deep Learning and VGG-16 Techniques,” *Electronics (Switzerland)*, vol. 11, no. 22, Nov. 2022, doi: 10.3390/electronics11223665.

[98] M. Azahari *et al.*, “Detecting Malware with Classification Machine Learning Techniques,” *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, p. 2023.

[99] S. S. Alshamrani, “Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files,” *Security and Communication Networks*, vol. 2022, 2022, doi: 10.1155/2022/7611741.

[100] A. Redhu, P. Choudhary, K. Srinivasan, and T. K. Das, “Deep learning-powered malware detection in cyberspace: a contemporary review,” 2024, *Frontiers Media SA*. doi: 10.3389/fphy.2024.1349463.

[101] Deepa K R and V. Bhavyashree, “Deep Learning Malware Detection Using Auto Encoder,” *Arts, Science and Humanities*, vol. 11, no. 1, Jul. 2023, doi: 10.34293/sijash.v11iS1-July.6314.

[102] S. S. Hussain, M. F. A. Razak, and A. Firdaus, “Deep Learning Based Hybrid Analysis of Malware Detection and Classification: A Recent Review,” 2024, *River Publishers*. doi: 10.13052/jcsm2245-1439.1314.

[103] F. T. ALGorain and J. A. Clark, “Covering Arrays ML HPO for Static Malware Detection,” *Eng*, vol. 4, no. 1, pp. 543–554, Mar. 2023, doi: 10.3390/eng4010032.

[104] M. Schofield *et al.*, “Comparison of Malware Classification Methods using Convolutional Neural Network based on API Call Stream,” *International Journal of Network Security & Its Applications*, vol. 13, no. 2, pp. 1–19, Mar. 2021, doi: 10.5121/ijnsa.2021.13201.

[105] T. E. Abioye, O. T. Arogundade, S. Misra, K. Adesemowo, and R. Damaševičius, “Cloud-based business process security risk management: A systematic review, taxonomy, and future directions,” Dec. 01, 2021, *MDPI*. doi: 10.3390/computers10120160.

[106] F. Wang, Y. Lu, C. Wang, and Q. Li, “Binary Black-Box Adversarial Attacks with Evolutionary Learning against IoT Malware Detection,” *Wirel Commun Mob Comput*, vol. 2021, 2021, doi: 10.1155/2021/8736946.

[107] N. K. Gyamfi, N. Goranin, D. Ceponis, and H. A. Čenys, “Automated System-Level Malware Detection Using Machine Learning: A Comprehensive Review,” *Applied Sciences*, vol. 13, no. 21, p. 11908, Oct. 2023, doi: 10.3390/app132111908.

[108] R. Khurram. Shahzad, “Automated Malware Detection and Classification Using Supervised Learning,” Blekinge Institute of Technology, 2024.

[109] D. Aboshady, N. E. Ghannam, E. K. Elsayed, and L. S. Diab, “APKOWL: An Automatic Approach to Enhance the Malware Detection,” *Mobile Networks and Applications*, 2023, doi: 10.1007/s11036-023-02159-x.

[110] H. Manthena, J. C. Kimmel, M. Abdelsalam, and M. Gupta, “Analyzing and Explaining Black-Box Models for Online Malware Detection,” *IEEE Access*, vol. 11, pp. 25237–25252, 2023, doi: 10.1109/ACCESS.2023.3255176.

[111] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, “Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches,” *Heliyon*, vol. 10, no. 1, Jan. 2024, doi: 10.1016/j.heliyon.2023.e23574.

[112] M. N. Al-Andoli, K. S. Sim, S. C. Tan, P. Y. Goh, and C. P. Lim, “An Ensemble-Based Parallel Deep Learning Classifier With PSO-BP Optimization for Malware Detection,” *IEEE Access*, vol. 11, pp. 76330–76346, 2023, doi: 10.1109/ACCESS.2023.3296789.

[113] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, “An efficient densenet-based deep learning model for Malware detection,” *Entropy*, vol. 23, no. 3, Mar. 2021, doi: 10.3390/e23030344.

[114] N. Albishry, R. Alghamdi, A. Almalawi, A. I. Khan, P. R. Kshirsagar, and Barudebtera, “An Attribute Extraction for Automated Malware Attack Classification and Detection Using Soft Computing Techniques,” *Comput Intell Neurosci*, vol. 2022, 2022, doi: 10.1155/2022/5061059.

[115] H. Wu, N. Luktarhan, G. Tian, and Y. Song, “An Android Malware Detection Approach to Enhance Node Feature Differences in a Function Call Graph Based on GCNs,” *Sensors*, vol. 23, no. 10, May 2023, doi: 10.3390/s23104729.

[116] R. Srinivasan, S. Karpagam, M. Kavitha, and R. Kavitha, “An Analysis of Machine Learning-Based Android Malware Detection Approaches,” in *Journal of Physics: Conference Series*, Institute of Physics, 2022. doi: 10.1088/1742-6596/2325/1/012058.

[117] L. Cai, Y. Li, and Z. Xiong, “JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters,” *Comput Secur*, vol. 100, p. 102086, Jan. 2021, doi: 10.1016/j.cose.2020.102086.

[118] O. Aslan and A. A. Yilmaz, “A New Malware Classification Framework Based on Deep Learning Algorithms,” *IEEE Access*, vol. 9, pp. 87936–87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[119] Y. Zhang *et al.*, “A Robust CNN for Malware Classification against Executable Adversarial Attack,” *Electronics (Switzerland)*, vol. 13, no. 5, Mar. 2024, doi: 10.3390/electronics13050989.

LIST OF PUBLICATIONS

1. V. Verma, A. Malik and I. Batra, "Assessing the Impact and Performance of Malware Detection Systems," 2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gautam Buddha Nagar, India, 2023, pp. 106-109, doi: 10.1109/UPCON59197.2023.10434443.

2. Vikas Verma and Arun Malik, "Malware Detection and Classification: A Comprehensive Review," 7th International Joint Conference on Computing Sciences (ICCS-2023) "KILBY100".

3. Vikas Verma, Arun Malik, and Isha Batra, "Analyzing and Classifying Malware Types on Windows Platform using an Ensemble Machine Learning Approach," Int J Performability Eng, 2024, 20(5): 312-318.

4. Vikas Verma, Arun Malik, Isha Batra and A. S. M. Sanwar Hosen, "A Novel Ensemble-based Approach for Windows Malware Detection," International Journal of Artificial Intelligence (IJ-AI).