Zero Day Malware Detection in the Windows Ecosystem: A Hybrid Contrastive and Behavioral Learning Approach

by

Wahid Hossain Nizami
21201156
Mamnun Ahmed Zihad
21201061
Abrar Murshed
23241095
Marzia Sultana
22101841

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
June 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Marzia Sultana

22101841

Mamnun Ahmed Zihad

21201061

Wahid Hossain Nizami

21201156

Abrar Murshed

23241095

Approval

The thesis titled "Zero Day Malware Detection in the Windows Ecosystem : A Hybrid Contrastive and Behavioral Learning Approach " submitted by

1. Wahid Hossain Nizami (21201156)
2. Mamnun Ahmed Zihad (21201061)
3. Abrar Murshed (23241095)
4. Marzia Sultana (22101841)

of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in June 19,2025.

Examining Committee:

Supervisor:
(Member)

Muhammad Iqbal Hossain, PhD

Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Partha Bhoumik

Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

The fast development of digitalization of the basic activities of the every-day life in this century has conditioned more dependence on the technological devices which made the observable enhancement of interdependence between the ecosystems, especially in Windows . With key areas of services, including banking, medical, and e-commerce services shifting to be online, the insecurities in these systems mean that more cyber jeopardy are faced by users, especially the zero-day malware attacks. Such attacks are also a huge challenge to cybersecurity on desktop and mobile platforms since they take advantage of vulnerabilities that cannot be easily identified and prevented. This thesis proposes a new hybrid method of detecting zero-day malware where ideas of contrastive and behavioral learning frameworks are combined. Through combination of these techniques, the proposed design will enhance the process of detecting and preventing unknown malware in windows platform. The hybrid detection system is able to sample the behavior of applications and compare against unique attributes, and provide a more flexible and effective barrier to defense. This method aims at giving input to the creation of scalable, real-time and Windows-centric platform security-solutions to address the shifting security requirements of the digital ecosystems.This paper proposes a staged framework for malware detection that increases its level of analysis gradually, from static filtering to behavioral modeling, self-supervised byte-level anomaly detection and network-wide monitoring, all within a single pipeline focusing on the Windows platform. Unlike other methods that rely on labeled malware datasets or a one-stage approach, this one focuses on a self-supervised approach and gradual risk assessment to effectively detect previously unseen malware, also known as zero-day malware which is the key novelty of this work.

Keywords: Zero-Day Malware, Windows-centric Platform Security, Hybrid method, Contrastive & Behavioral learning, Real-time detection, Digital ecosystems

Acknowledgement

Firstly, thanks to Almighty Allah for letting us complete our thesis successfully.

We extend our gratitude to our supervisor Muhammad Iqbal Hossain Sir , co- supervisor Partha Bhoumik sir for their guidance throughout this journey. Their detailed insights have helped us navigate this challenge.

Last but not the least, we express our gratitude to our parents and siblings, whose love and encouragement motivated us to reach our goal.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

A zero-day exploit is a cyberattack that uses an uncovered or unfixed security vulnerability in computer hardware, software, or firmware [5]. The term "zero-day" refers to a software or hardware fault that a vendor cannot repair before hostile actors exploit it to get access to affected systems [31]. What is risky about these vulnerabilities is that they can be used prior to any preventive mechanisms like these because new systems are implemented or patched, and systems are left vulnerable to attacks. These adventures may result in unauthorized access, data intrusion, and malware in the long run deployment[33]. Attackers employ numerous paths to factor in the zero-day failures,Network connectivity and user interaction. Phishing is one of the most popular ways that consists of false messages or emails that swindle users into clicking infected xml or download of files that use the zero-day vulnerability. Network-based attacks analyze systems for unpatched vulnerabilities or open services before sending malicious material to exploit them remotely. Infected portable media, such as USB drives, can also spread malware by exploiting a vulnerability when they connect to a computer. Real-world incidents like the Nimda worm, which rapidly spread through email, network shares, and web server vulnerabilities, and the SQL Slammer worm, which caused widespread denial-of-service by exploiting a SQL server vulnerability, demonstrate the devastating speed and reach of zero-day attacks when combined with multiple propagation vectors[3].

The diagram below presents the life cycle of zero-day vulnerability-

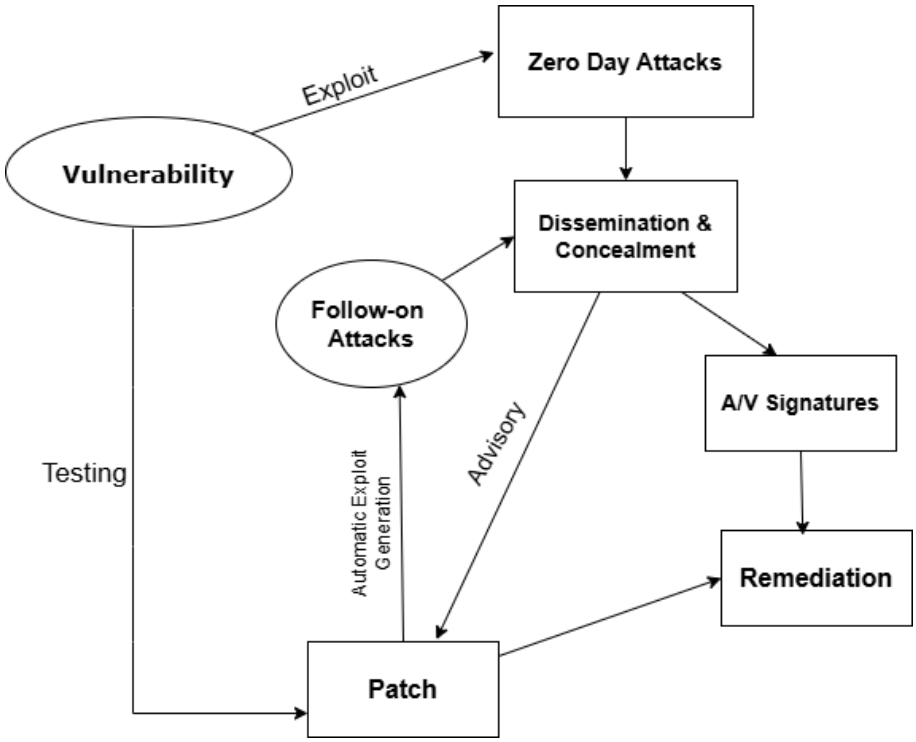

Figure 1.1: Zero-day Vulnerability Life Cycle [3]

As shown in figure 1.1, the life cycle of a zero-day vulnerability consists of seven stages. Zero-day vulnerabilities typically start out as software bugs. If the software maker doesn't find the problem, attackers could be able to exploit it. When software makers find a vulnerability, they usually make the security issue public [9]. The vendors then update anti-virus signatures to reflect the new understanding of zero-day assaults and release a patch for the zero-day vulnerability.

Even after a vendor has detected and addressed a vulnerability, fresh exploits may be developed and deployed against targets who have not yet received the patch. It may take several years for remediation to occur following this cycle of patching and exploitation, at which point the vulnerability no longer affects systems [16]. Despite growing awareness and improvements in cybersecurity solutions, the detection and prevention of zero-day attacks remain profoundly challenging. Mainly because of their unknown nature, polymorphism, obfuscation, runtime evasion, and encrypted payloads. The malware can use these strategies to hide its actual behavior, seem to be legitimate software, and evade tools of static and dynamic analysis. Besides, absence of labeled sets and high false positive rates makes it less effective. providing a machine learning-based detection, and device constraints inhibit real-time protection [22] This problem is not that easy to solve, since it is traditional machine learning techniques perform well in known threats but perform poorly in situations where it is unfamiliar unknown. The models largely depend on the attack patterns found in the past due to the reason that,they would commonly be trained on fixed, labelled data. Consequently they become hard to sleep.Adapting to unknown threats. Considering, in case a malware sample implements slightly alternative technique, the model cannot identify it. The deficiency in runtime behavioral context, dependence on static attributes (like permissions or API calls) and non-possibility.To test obfuscated or encrypted payloads all are less efficient. In addition, one-sided datasets,

benign samples are more in number than malicious samples, enhance the tendency towards false failure. That is why there is a necessity to address them in a hybrid way.limitations. We will include both behavioral and contrastive learning to achieve this achieve a higher level of prevention and detection of the zero-day attacks.

1.2 Rational of the Study or Motivation

The purpose of this research emerges from the growing threat that zero-day malware poses due to its ability to exploit systems and breach data before any defense mechanism is created. As mentioned before that traditional machine learning models often fail to detect these attacks, it has caused significant damage to data security. Moreover, due to its increasing complexity and frequency, these losses are not only limited to data breach but also include grave financial losses.

For example, The HAFNIUM attack was a significant cyber operation attributed to the state-sponsored group HAFNIUM, operating out of China, targeting Microsoft Exchange Servers. In early January 2021, suspicious activities were detected on numerous Microsoft Exchange servers, which upon investigation, were found to involve zero-day vulnerabilities exploited by HAFNIUM to gain unauthorized access. Over 30,000 organizations in the US alone were affected initially, with hundreds of thousands globally impacted, including government agencies, healthcare, law firms, and research institutions. The attack network was exploited to spread ransomware such as Dear Cry, demanding ransom payments up to $16,000,leading to financial and operational damage .

Another example is, In 2010, a highly advanced computer worm called Stuxnet made headlines when it targeted Iran's nuclear facilities. It exploited four unknown security flaws in Microsoft Windows [33], allowing it to infiltrate the systems controlling the facility. Once inside, Stuxnet secretly sent destructive commands to the centrifuges used for enriching uranium, making them spin uncontrollably fast until they broke down. In total, about 1,000 centrifuges were damaged in the attack Android systems are also vulnerable to zero-day attacks mainly due to their fragmented ecosystem. Many reasons like inconsistent software updates, multiple manufacturers with varying android versions and the open source nature of android allows attackers to exploit it easily. There's also the risk of installing compromised apps, which will later ask for unnecessary permissions. Therefore, there is a significant need for a real time and adaptive detection system that will tackle these problems going beyond traditional approaches. So our proposed solution is a hybrid approach that combines contrastive learning and behavioral analysis allowing for robust detection of zero-day threats, even in restricted environments.

1.3 Problem Statement

Despite making consistent progress on detecting vulnerabilities, zero-day attacks on Windows OS remain a challenge due to its unpredictable nature of attacks and changeability. These attacks are not yet discovered by vendors or developers as they

never ceased to exist in the first place. One of the most noteworthy instances of such an attack is the Eternal Blue exploit which was developed by the NSA and later hacked by a group of threat actors called Shadow Brokers in 2017. Eternal Blue exploit helped the hackers acquire unauthorized access and execute malicious commands in a remote manner along with initiating expeditious spreading of the ransomware across 200,000 computers in 150 countries, which caused approximately $8 billion impact on the global economic landscape [23].

This poses a significant threat for the scenery of the Windows ecosystem as it is the most popular and widely used operating system because of its user-friendly and intuitive GUI, software compatibility, and application ecosystem. Businesses and individuals, who operate on Windows on a regular basis, are at constant risk of cyberattacks because of these unknown vulnerabilities, which security vendors have not yet identified. The most concerning issue remains with the fact that conventional security measures lack the ability to timely detect and neutralize these new threats. These signature-based techniques prevail as ineffective in detecting zero-day vulnerabilities as these threats are not yet documented or analyzed [19]. Researchers, in recent times, used signature-based detection techniques which enormously depend on identification of malware using formerly known patterns. Such signature-based solutions prove to be insufficient in detecting zero-day patterns as datasets for Machine Learning models have no labeled examples of these attacks. The detection becomes more challenging and strenuous due to the constant altering nature of malware programs such as, obfuscation of codes and polymorphism [6]. As a result, these conventional approaches cannot offer effective real-time detection and prevention of malware; thus, systems remain vulnerable to exploits.

A close challenge is also seen in mobile environments such as Android, which is a frequent target due to its widespread adoption and open-source ecosystem [39].previous works have shown that permission can be effective for detecting malware in Android ecosystems [11]. Due to platform fragmentation and delayed security patch deployment, Android shows additional complexities that are beyond the scope of this work. Therefore, this research focuses exclusively on zero-day malware detection within the Windows operating system.

Although attempts are made by several researchers to reach a solution to discover and diminish real-time zero-day attacks facilitated by unknown vulnerabilities using Machine Learning models and heuristic analysis, the methods were not totally efficient as the procedures could not effectively dispense and deal with the newly sophisticated version of these attacks [2] [19].

Hence, this study's objective is to establish a hybrid approach by combining ***Contrastive*** and ***Behavioral*** learning techniques with adaptive learning feedback loop to systematically discover and mitigate zero-day attacks, designated specifically for Windows ecosystems. The purpose of this line of action is to refine zero-day vulnerability detection accuracy and efficiency which will minimize the chances of exposure to new and advanced malware. In this manner, it will help create a safer Windows ecosystem with vulnerabilities mitigated before they can be used extensively by attackers.

1.4 Research Questions

RQ1: How effective is a layered escalation-based framework in detecting zero-day malware within the Windows ecosystem compared to single-stage detection approaches?

RQ2: In self-supervised anomaly with byte level detection, is it possible to detect previously unknown?

RQ3: What is the effect of using a combination of static analysis, behavioral monitoring and network-level? introduce greater accuracy in detecting a zero-day and reduction of false positives with observation?

RQ4: How staged escalation affects system performance and detection. performance in Windows real time?

1.5 Objective

In the recent years, the sophistication of cyberattacks, especially zero-day, has also been increasing.Threats, has put into focus the limitations of the conventional signature based systems of detection. With attackers taking advantage of unidentified vulnerabilities, the need to increase is on the rise of dynamic security solutions. This study fills this gap by utilizing the power of artificial intelligence (AI) to identify malware by detecting the behavior.

Key Objectives

1. **Grant Behavior Baselines to Common File types:** Setting up baseline behavior of file type such as RAR files; this involves interactions between the files with system resources.
2. **Eschew Abnormal Behavior:** Indicate unusual behavior, including as a RAR file trying to start the command prompt, to open the registry or to start network connections, things that may suggest malicious activities behavior.
3. **Adopt AI and Machine Learning Models::** Build models which learn through both malign and benign pattern activity, making it easier to detect accuracy over time.
4. **Develop a Real-Time Response Mechanism:** Develop a system that has the capability to automatically hinder suspicious files prior to the occurrence of damaging actions.
5. **Minimize False Positives and Enhance Adaptability:** Target on minimizing Incorrect identify and adapting the system to the changing threats, ensuring continuous protection.

This study seeks to combine both behavioural analysi and the use of AI to clarify a decision to develop an effective, proactive security system, which can perform better than conventional signature-based approaches in the process of examining zero-day attacks.

1.6 Methodology in Brief

This research proposes a tiered approach to malware detection which is able to identify known malware as well as unknown malwares also known as zero-day malware that is aimed at Windows-based systems. The system is based on a tiered model in which files go through a series of detection levels with further analysis applied to those that are suspicious to begin with The intention is to improve the detection rates without increasing false positive or processing overhead. The second layer is the behavioral modeling. This layer analyses the file based on the behavior of the file if it is executed, It also involves the analysis by rules. This layer verifies whether the file behaves in a way that is consistent with the type of file it is. For example, a document file should not initiate actions on the system. This layer identifies suspicious behaviors by using rule-based analysis [1].

The second layer involves behavioral modeling. This layer analyzes the file based on the behavior of the file when executed. It also involves rule-based analysis. This layer checks if the file behaves in a manner that is consistent with the type of file it is. For example, a document file should not trigger system actions. This layer identifies suspicious behaviors using rule-based analysis[3].
The third layer involves in-depth analysis of the file contents using self-supervised learning. This layer analyzes the file's contents at the byte level. Unlike the previous layer, this layer does not look at the file's structure; instead, it learns what normal and benign files look like. This layer is very good at detecting zero-day malware [8]. The last layer involves combining the results of the previous layers. This layer combines the anomaly scores of the previous layers and the escalation rules. This layer generates a final decision on whether the file should be allowed, escalated for further analysis, or blocked. Although the system also has the capability to escalate the file for further analysis, the main aim of this layer is malware detection[36].

Experiments have shown that the system improves the effectiveness of malware detection,including the zero day malware detection[22].

1.7 Scopes and Challenges

Scope of Hybrid Approaches in Detecting Zero-Day Attacks

Traditional malware detection models mostly fail to detect zero-day attacks because those models mostly work for known malware. In the hybrid model, we are combining multiple detection approaches together which is increasing the efficiency of the model in detecting zero-day attack. This approach covers a diverse analysis approach for different types of files [31] [30].

1. Comprehensive Threat Coverage
Hybrid model is a combination of static analysis, behavior monitoring and also machine learning approaches. This increases the chance of detecting known and unknown malware. It can also detect hidden malware within any kind of file type. This model overcomes the limitation of other approaches because it adds multiple

layers which helps in increasing the efficiency. Some models work well for executable files but fail to detect hidden malware. This model overcomes those shortcomings [22].

2. Real-Time Threat Response and Containment
This hybrid model has software defined networking (SDN) in layer three which isolates the infected files or suspicious files. This real time working feature helps to minimize the damage if the system gets infected and also helps to prevent malware from working in the system [36].

3. Scalability for Enterprise Environments
This model can be deployed into multiple environments and systems. By federated learning, we can collaborate between different systems by sharing learned threats while not exposing any kind of user data. This approach will help to respond faster and also it will make the model more accurate and updated [31].

4. Adaptability Through Feedback Loops
This model learns and updates itself with the feedback in multiple layers for a file. Behavioral analysis can update the signature based attack database with new detected attack data. New unpacked malware patterns can update the static analysis database. This approach will help the model to be relevant with time and it will be also known to newer malware and how they approach in a system [31].

Challenges in Detecting Zero-Day Malware Using Hybrid Approaches

1. High Computational and Resource Overhead
Omnipack approach needs real time execution tracking because it monitors the activities of secondary memory which requires a lot of CPU and primary memory. CNN or VGG16, these deep learning models need a strong GPU to work faster. It is hard to use these models in low devices. These systems may get slower if we use them to store a lot of files [3].

2. False Alarms and Confusing Behaviors
Sometimes normal program behavior looks suspicious like changing their own code in the memory, this thing happens when an app gets updated. Some advanced apps behavior can be confusing like word files. They can be flagged without being a malware. It's actually hard to maintain a perfect behavioral analysis for each file so that safe programs do not get flagged [20].

3. Difficulty Keeping Rules Up to Date
YARA patterns or PE file structures need to be updated because malware keeps changing with time. Threat actors often use file-type spoofing to hide the malware in different file types. Rules need to be checked and updated with newer file types and how newer malware behavior is changing [6].

4. Machine Learning Can Be Tricked
Hackers can create files which are a little bit different from known malware. It can confuse the ML models into thinking the files are safe. Malware can be encrypted which makes them hard to recognize by the ML model. Without a proper approach with the ML models, it can be hard to detect zero-day attacks [22].

5. Malware That Hides or Avoids Detection

Zero-day malware always use newer approaches to make them undetectable. If they detect that they are executing in the sandbox, the malware can hold their running. Malware sometimes runs child processes so that they can hide them from getting flagged. Malware can also be injected into trusted sources so that they can gain user trust even if the models flagged them. To detect these kinds of malware tricks, it often needs the model's deep access in the system which can create privacy issues [20].

6. No Standard Way to Combine Techniques
There is no method which can mix static analysis, behavioral analysis, and machine learning together. So, these methods need different custom setup and also need to be connected with each other in order to work together as a different layer of a single system. They will not work together without any kind of communication link between them [31].

7. Not Enough Good Data for Training
There are many malware datasets, but those are obsolete or incomplete. Real newer malware samples are hard to find due to evasion and obfuscation techniques. If we train our machine learning models with old datasets, they would not be particularly effective at detecting zero-day threats [6].

1.8 Thesis Structure

Chapter 1 Introduces the research topic, problem statement, research questions, objectives and scope of the study .

Chapter 2 reviews the existing research and the gaps in it.

Chapter 3 talks about the social and economic impacts in real life, as well as ethical issues and risk management.

Chapter 4 outlines the design and structure of the multilayer system.

Chapter 5 focuses on the result and comparison with traditional methods.

Chapter 6 draws conclusion, talks about limitations and showcases some aspect of future work.

Chapter 2

Literature Review

2.1 Preliminaries

Zero-day malware is the type of malware that exploits the vulnerabilities no one knew about until after the damage starts. So spotting it with traditional antivirus is extremely difficult. As there is no defence ready when the attack takes place, these threats bypass the standard security protocol. With the increasing complexity and diversity of modern devices, zero day attacks are no longer limited to a single platform. Windows with its vast personal and professional use remains the prime target of these attacks. Also , android - the most popular mobile operating system is also exposed due to their open environment and vast app store. Most existing detection methods that scan codes statically or rely heavily on well known behavioral patterns- they fail to discover novel or complex polymorphic malware. This research proposes to address these limitations with an hybrid approach that incorporates behavioral analysis with contrastive learning for zero-day malware detection on Windows ecosystem. The aim is to build a robust, windows-focused detection model that can spot brand-new threats it has never met before. Behavioral analysis watches how the code acts while in execution and contrastive learning improves the system's ability to tell apart subtle similarities and glaring differences between harmless and harmful behavior. Combining these two approaches should boost both the accuracy and the real-world flexibility of zero-day malware detection targeting Windows systems.

2.2 Review of Existing Research

The weakness of current IDS solutions that are mainly concentrated on this limitation is on familiar attack signatures or obsolete data. Conventional IDS approaches usually presuppose define such attacks when they use predefined signature attacks of these attacks bureaucratic datasets that fail to consider changing attack patterns. The paper also argues that the constraints of training models on aging or biased data set. the necessity to train models again in order to fit in with new attack types. Though deep Learning models prove to be promising in novelty detection, aspects such as limited memory,computational requirements as well as deployment environments may affect the performance of models in real-world scenarios. Thus,

the review suggests researching on a range of methods and algorithms of zero-day attack detection and monitoring the development and obstacles to network anomaly detection. It also stresses the significance of coming up with strong IDS systems that are not biased and can identify both known and unknown attacks.

Huthifh et al.highlighted in this paper. [12] was able to determine 353 out of 361 malwares carried out of 97.78%design a novel zero-day attack prevention and detection system of Software-Defined Networks (SDNs) by configuring the Cuckoo sandbox. tool. Untested programs are tested on a sandbox environment in isolation protect contamination of actual surroundings. The modified Cuckoo sandbox improves the zeroday attacks by giving an isolated and automated environment on which unknown files or URL can be executed safely to investigate. In this process, it scans the action of potential malware in several windows VM settings,recreating memory dumps, registry modification, and files created. In this article, HAFNIUM Attack have a case study. The HAFNIUM attack attacks on specific Microsoft Exchange servers with different zero-day vulnerabilities.The federalists are considered to have been attacked by themselves, which is presumed to be a government-sponsored group operating out of China, and which is estimated to have breached a significant amount of companies in the world-over. The article focuses on the need to counter these threats by highlighting the necessity of preemptive defence means like fast patch release, modern intrusion detection and prevention frameworks, feature analytics and network separation Keeping a current systems library of software, automation of patch implementations and a culture of security sensitivity within the organization is an important measure in reducing vulnerabilities. They are however limited in some aspects. They often lack documented data on contemporary real-time identifications of exploits installed zero-days, rigorous processes of hunting threats proactively, and engineering processes objections of proposed solutions.

In this work [36], ZSDN-Guard, a Zero Trust protection framework, is offered. on SDN, deep learning based on anomaly detection in live and real-time access. regulation. Today, SDN security models are based on anomaly which is machine learning. identification yet typically employ obsolete data. The downside to this existing approach is that the vast majority of approaches center around the protection of the controller, but not the protection of the entire network. The perimeter-based defense is incapable of changing dynamically to threats in SDN. Thus, this paper suggests some solutions such as ZSDN-Guard or NetSeqDL.To identify network anomalies with 99.75 accuracy using (a deep learning model) only as the use of dynamic trust-based access control, rather than rules. It points out that work in the future will be aimed to enhance performance of the framework further especially within the more complicated network environments, and investigating its applicability through wide and extensive networks.

Ibraheem Tosho [27] proposes the traditional security methods are based on signaturebased detection that is powerless in case of the zero day attacks. Other recent studies involve machine learning-based models to monitor live abnormalities with random forests, etc. SVM, gradient boosting and decision trees were also coming out as promising techniques. But ML-based methods have issues such as gathering, labeling, and unification of data into security workflows. This paper is a proposal

to employ machine learning models (especially random forest) to identify zero-day attacks on the basis of the distinctions between data patterns. The Models were trained and tested on CSE-CIC-IDS2018, and were sufficiently detected of attack patterns. The findings show that random forest has a higher accuracy ,it is a powerful contender in upgrading cybersecurity.The paper presents an overview of the latest situation in the performance of zero-day attack detection and protection. It speaks of approaches like analysis of protocol behaviour, anomaly detecting, data. matching, honeypots, traffic signatures, vulnerability-based signatures and new methods such as C auto worm-detection. The primary challenge with existing methods is their inability to promptly identify and prevent zero-day attacks due to rapid threat evolution and the limitations of reactive signature updates, which can take hours or days, during which attackers can exploit unpatched vulnerabilities. Future directions suggested include developing proactive, AI-driven detection mechanisms and integrating multi-layered security architectures to provide higher performance and more robust zero-day defenses, thereby reducing the window of vulnerability.

The title of this paper is "Zero-day attack detection and prevention in software defined networks". This approach works using the SDN environment. In zero-day attack, the attacker exploits an unknown issue of a system. Mostly, traditional ways to detect malware. The authors of this paper proposed a dual layered defence system using cuckoo sandbox. All traffics which are passing through the switches is inspected by SDN controller, it extracts files and URLs and analysis them in a windows based sandbox. The files are assigned a number by the sandbox indicating how well the files behave with malware. A custom UNIX-based sandbox that monitors the critical components which is used for the controller protection. It monitors the service ports and OS features which detects any kind of unauthorized access and triggers the alarm. This paper's approach is done by using Mininet and Openflow. They used malware of size from 2 KB to 1400 KB and got detection rate of 97.78%. This shows the effectiveness of this approach. In future, we can try to make this approach better by increasing the malware size [13].

This paper shows a deep learning approach for malware classification where they used pre-trained VGG16 CNN model and it also shows us the challenges of traditional signature based methods. The researchers converted a malware into a grayscale image and each malware creates a different image. The pre-trained image recognition system is used to analyze those images. The system was 98% accurate in detecting malware types[2]. This system also works for most of the kind of malware. This approach is faster than trying a new system because they used a pre-trained system and made it better. But there are also some issues like some malware type was over presented in the test data. This system struggles a lot in detecting malware with mostly same visual variants. Advanced malware which hides its code can bypass this system because it only looks for the code. This system also should look for the behaviours of a file which can increase its accuracy to detect zero-day attack. For future research, this researchers suggested that we should add a way to find out unknown patterns of a malware and also we checkout how a program is behaving when we are running it [14].

A malware always changes its code to avoid detection. This paper tried to solve

this issue. The system they developed mainly checks how files call system functions as well as code pattern. They tried three different machine learning ways. The methods used are Decision Tree, Random Forest and LightGBM. Within all of them lightBGM worked best because it made less mistakes then others. But sometimes it also missed some actual malware which is very risky for a system. This research suggest us that we need bigger datasets to know about how newer malware behaves while running. This system normally detects normal known malware but it also can detect unknown malware by checking that if its pattern matches with the pattern of a malware file. This system mainly check out the byte sequence, opcodes, PE header and API calls from malware file, it helps to detect a malware without executing the file [14]. But to make this system more accurate they needed bigger datasets which is a limitation of them. False alert was the biggest issue in this system. To make this system better they could try a hybrid approach by combining multiple systems [14] [22].

Omnipack is a way to detect a malware when the malware is compressed or encrypted so that it can bypass normal signature based malware detection systems. Omnipack checkout that how a programme is getting executed. It uses hardware based assisted memory protection. This tracks that when a program is executing its code from the memory page its previously used, it's a sign of that other code is getting executed from that place after execution of the main code. Omnipack has high efficiency because it has hardware feature like non-executable page and using Coarse-grained monitoring. It also increases the safety by using malware detectors like clamAV only when potentially harmful system calls like read ,write are tried,it checks that if those calls are safe before they can harm the system. This approach is practical because it takes different amount of times for detecting packed and unpacked malware. This approach performs better than other ways of preventing malware [1]. It also has issues such as giving false alarms because it cannot monitor the child processes. Despite having some issue, this approach is a production ready solution for the malware detection system [1] [22].

JothiShri et al. [28] consider the issue of deep learning in network security, paying attention to the problem of anomaly detection and real time threat analysis. CNNs and RNNs are used in the study to boost cyber security defences. Namely, CNNs are used on network traffic data and their capacity is to extract spatial patterns in packets. By contrast, RNNs are used to perform time series processing over network-based log data identifying bad trends over time. CNNs are aimed at identifying the deviations that can be perceived as possible breaches, whereas the RNNs are geared at finding the anomalies like the unauthorized logins or the abnormal data transfer. Methodology mentions the hybrid combination of these models, but allows to cover the whole possible set of security incidents. Important indicators of this paper imply that the detection rate used in this research in anomaly detection is 95% with a known attack accuracy of 98% and a prior unknown attack accuracy of 85%. The system shows astonishing performance of 0.8 ms per packet, which indicates how effective the deep learning models are in real-time processes. The paper focuses more on the issue of model interpretability that is of high concern to security analysts who are required to trust AI-driven systems. Irrespective of these limitations, AI models can be extremely useful in the identification of zero-day vulnerabilities owing to

their flexibility and adaptability to new and changing information.

Another solution inspired by the work of Sarker et al. [18] introduces a hybrid AI-based model combining both supervised and unsupervised learning architecture that can be used to detect known threats and novel or unknown threats on real-time network traffic via decision trees and k-means clustering. The hybrid model is proposed to offer an assignable solution to the predefined cyber-threats as well as the emerging ones, relying on the two strengths of the particular machine learning techniques. In supervised learning, decision trees are used on labeled data, to detect common patterns of attacks, and this is unlike the unsupervised learning where the use of k-means clustering helps in detecting the undesired behaviors that are outside the current profiled threats. Such hybridization allows making the model ready to pursue a great variety of cybersecurity situations. The authors also use the Natural Language Processing (NLP) to process the unstructured data like the logs and the incident reports and this aspect makes the model able to identify the emerging threat, which is yet not deemed in the structured data. It is a particularly valuable part of NLP when it comes to responding to the incident in real-time, as it allows identifying the threat and its mitigation faster. The measure of performance of the model can be mentioned as 90% accuracy in detecting network anomalies and 92% precision in detecting suspicious behaviors; this indicates that the model is reliable. Nevertheless, its recall in the detection of new and unseen threats is 78%, which means that the model is less efficient when it comes to discovering completely new attack strategies. Nevertheless, the system has a great potential, and its speed is rapid enough to do real-time analysis.

Li et al. [29] presents the PAFE methodology, which is a new framework of the malware naming technique, based on visualization to represent the malware binary as a gray-scale image. This approach does not rely on the conventional possibility of previous domain experience and long periods of execution such as complicated manual feature extraction. The system assists in stopping malware since the binary data in the form of images makes the analysis process simpler and, at the same time, enables deep learning models to automate feature extraction. This can be especially useful towards large-scale detection of malware, as it is a major concern of cybersecurity in real life. To cope with a usual problem of image distortion when the image normalization process occurs, the authors [21] present a method of padding pixels. Normalization process as the bilinear interpolation sometimes corrupts some fine details of an image, which might lead to the loss of important detail and influence the accuracy of classification. The available pixel-padding technique guarantees that the integrity of the image is not lost as the image is filled with pixels without distorting the main characteristics of the image. Such an approach preserves the key patterns of the texture, that is vital in the differentiations of various malware types and greatly enhances model resilience. This classification model based on the PAFE structure provides high accuracy with 99.25% and a prediction process time of 10.04 ms/malware, which can be regarded as a very high value compared to traditional models, such as VGG16 and ResNet. Although they are accurate, such models are computationally high priced and slow to process, particularly in real time operations. Through the lighter structure approach, PAFE combines the excellent performance and guarantees a high rate of processing speed, making it appropriate in real-time

malware diagnosis where speed plays the vital role. when the low prediction time is observed at 10.04ms, it will be able to process large network traffic with it all coming out accurately with hence causing no latency.

Mahindru et al. [11] introduce MLDroid, a machine learning-based, dynamic Android malware detection framework that utilizes web-based frameworks. The framework is aimed at studying the behavior of Android applications during runtime, with the permission and API calls as the main features of discovering malicious behavior. The features are isolated during the execution process and allow for dynamic observation of how the app communicates with the device, something that is so crucial in spotting suspicious behavior, which may be overlooked by static analysis. In this work, the authors employ feature selection methods such as GainRatio, Chi-squared tests and Principal Component Analysis (PCA) which help to refine the input data structure by identifying the most pertinent features to train the detection model. A range of Machine Learning methods, such as deep learning, farthest-first clustering, Y-MLP and a nonlinear ensemble decision tree forest technique, are then trained using the selected features, enhancing the classification efficacy of both harmful and benign apps. With a 98% detection rate, the architecture demonstrates its proficiency by detecting malware with exceptional precision in both known and unknown malware categories. Additionally, by focusing on the most significant features, Rough Set Analysis (RSA) serves to decrease feature redundancy while improving model performance. Despite its superiority at malware identification compared to other machine learning-based malware detectors, the performance of MLDroid is enhanced further by combining two kinds of learning, namely, supervised and unsupervised learning, enabling it to respond to new and unknown threats. Such versatility is especially significant for Android malware, which continuously changes to evade detection.

In their study, Afianian et al. [6] explore the methods applied by malware in order to evade the standard dynamic analysis tools used to track the activity of applications during runtime. The paper establishes that these evasion strategies are grouped into detection-dependent or detection-independent categories. Detection-dependent Malware is designed with detection in mind, by detecting the presence of an analysis environment e.g., sandboxes or debuggers and modifying its actions to evade detection. NGlobalFlags, for example, is an anti-debugging method, making analysis difficult as it verifies the presence of debugging tools and adjusts its actions accordingly. This poses a risk to traditional manual dynamic analysis as these evasion techniques target analysts employing debuggers to observe malware behavior. Comparatively, detection-independent methods involve malware that does not seek to recognize the analysis environment. These methods are more difficult to notice since they do not modify the external environment but rather the malware's internal form or behavior, such as fileless malware, which executes directly from memory and is often overlooked by conventional methods. The authors from [6] reveal that manual dynamic analysis only detected 87% of malware and the detection rate sharply declined to only 65% when facing advanced evasion techniques such as anti-debugging or fileless malware. The study also explores the performance of automated sandboxes that have a 90% success rate with known malware but fares

poorly with novel versions or obfuscated programs. This shows the shortcoming of the sandboxing technique alone. To enhance the detection, the authors suggest using dynamic analysis methods alongside machine learning and behavioral analysis, as these methods can adjust to the newly found evasion methods more efficiently and provide greater insight into the malware.

Mobile phones have evolved very fast to include smartphones which are a fundamental component of life today and the android version is on the frontline because of its open-source attribute and huge marketplace. This popularity has however made Android, the target of malware developers. The existing traditional antivirus mitigations, which mainly use known malware signatures to detect malware, have not been able to cope with the rise in the number of applications that are being uploaded on such sites as the Android market. According to Grace et al. [2], manual vetting is an impossible feat considering that an average of 1,242 new applications are released each day. As Grace et al. point out, RiskRanker has detected 718 malicious apps across 118,318 sampled apps (where 322 were zero-day malware out of 11 malware families). The second-order analysis, that is aimed at more complex evasion methods such as dynamic code loading and encrypted execution of the native code, also optimizes the detection process, managing to spot the presence of more threats. The authors validate that RiskRanker is scalable since it was able to analyze more than 118,000 apps within a span of less than four days, which is unachievable using traditional malware detection systems. Evaluation demonstrated that the proactive nature of RiskRanker could identify threatening events unknown to the organization, that is, 322 zero-day malware samples. Such an outcome is essential, because conventional systems have the tendency to miss most of the newly emerging threats which are found to exploit vulnerabilities that had not been recorded before. Grace and others call themselves as limitations of the dependence of the system on signature-based root-exploit detection and highlight the effectiveness it has in detecting advanced types of malware such as AnserverBot family, which applies advanced evasion mechanisms, such as dynamic code loading and encryption.

In this paper [7] they pointed about advancements in Natural Language Processing (NLP) driven by deep learning models in recent years. They focus on core linguistic processing issues, and applications of computational linguistics. They Also discussed the current state of the art and future research.The directions we can follow in future research are like Improving Model Efficiency, Reducing Data Dependency (Explore techniques like semi-supervised learning, self-supervised learning, and few-shot learning) , Enhance Interpretability by improving models prediction accuracy etc. This survey is the only one to encompass core linguistic undertakings as well as the fields of applied NLP using deep learning. It does not focus on models like in the previous works but the state of art.The underlying theory is also discussed, which makes BERT and GPT broader as well as more comprehensive. The challenges are lack of support of the low-resource languages and limited support of interpretability, and a lack of models unified to perform all fundamental tasks. The authors propose using deep learning with knowledge graphs and developing it further unsupervised and cross-lingual learning.

They focus on Adversal Machine Learning in the article [24] In the realm we may train ourselves on the weaknesses of neural networks,in what way they are prone to manipulation to an extent that they sort the inputs in a wrong manner. Despite the fact that deep learning models have high accuracy, they are susceptible to adversarial attacks which exploit these weaknesses. They also reviewed classified them and recapped on the existing adversarial attacks that had been proposed earlier defense strategies and the disadvantages. This is a unique work in that it gives a presentation of a. extensive categorization of adversarial attacks in combining eleven perspectives, that contains attacker knowledge, attack goal, format and optimization method. It surveys attacks as well as analyzing why neural networks do not work under such. attack, based on theoretical knowledge such as decision boundary curvature, attention shift,and saliency conflict–and broadening it more and deeper than most surveys which had been made before.

This paper [15] discussed cyber attacks in modern economic, commercial, cultural, social and government and the importance in cyberspace in these criterias. Which is a huge threat to some organizations which are directly connected with government security codes and E-Currency transactions related platforms. They also talked about PC Viruses, data breaches and distributed denial-of-service (DDoS) attacks to highlight risks of cyber attacks. They Also mentioned about advantages, disadvantages, strength and weaknesses of cybersecurity in various platforms. This paper stands out by offering a broad, multidisciplinary analysis of cyber-attacks, combining technical, legal, military, and economic perspectives. Unlike typical cybersecurity reviews that focus only on technical solutions, this study also discusses policy-level implications, cyber warfare scenarios, and the need for a universally accepted definition of cyber-attacks. The authors acknowledge a lack of standard definitions and international consensus on what constitutes a cyber-attack. This ambiguity limits legal and political responses. They emphasize the need for global cooperation, interdisciplinary policy development, and adaptive cybersecurity strategies to match the evolving and borderless nature of cyber threats.

The paper [37] presents a RedTeamLLM, a framework of Agentic AI created in a design to automated offensive cybersecurity operations, that is, penetration testing. Built Developed using such components as Reasoning, Acting, and Summarizing, the model can independently plan, perform, and optimize cybersecurity attacks. It overcomes significant problems in Agentic AI such as context window constraints, dynamic plan and memory management and providing long term memory storage to learn. from past tasks. In comparison to the existing models such as PenTestGPT, RedTeamLLM. shows a higher level of performance and efficiency particularly with its reasoning. The framework also entails the strong security measures to block malfeasance, having properties such as user validation and isolation. Its good rating on CTF. scenario confirms its usefulness and applicability in the context of real-world cybersecurity operations. Unlike the past when there were tools such as PenTestGPT, RedTeamLLM includes adaptive planning and long-term memory into its agentic self-reflective reasoning loop. This helps the model to dynamically optimize its strategies in multi-step offensive operations, which the earlier tools were incapable of dealing with well. The authors recognize that although RedTeamLLM incorporates validation and isolation mechanisms, avoiding the abuse in the field de-

ployments is still a challenge. Future work will emphasize on the safe access controls, work on the memory management system, and to higher levels of attack simulation, which are more realistic and in real time.

In this paper [35] an edge based Intrusion Detection System (IDS) is proposed to keep safe internet of vehicles (IoV) of botnet malware, particularly of the zero-day kind.The system applies a meta-ensemble classifier which is constituted of several Isolation Forests.Each is a (IFs) specialized in the detection of certain botnet behaviors. These models are run and deployed on Multi-access Edge Computing (MEC) servers, which makes it possible real-time traffic monitoring without straining the automobile systems. A Particle Swarm Optimization (PSO) algorithm uses the anomaly thresholds of the ensemble to enhance accuracy and robustness. Experimental findings on an automotive botnet data. are very effective - with 92.8% detection on known attacks and 77.3% on less known attacks zero-day attacks-showing the flexibility, scalability and excellence of the system performance particularly in flow-based detection. This framework is unique in that it uses the ensemble detection framework at the Multi-access Edge Computing (MEC). level and guaranteeing the real-time detection of threats with low latency and computational load on vehicles, in comparison to the traditional centralized IDS solutions. The authors postulate that their system is working, but cannot be effective in cases where it has to work under very stealthy or polymorphic attacks should be developed. The future improvements encompass time-series model integration and dynamic change to changing traffic pattern in vehicular networks.

In this paper [25] an anomaly detection system based on deep learning is presented to identify.Ensemble of LSTM, GRU, and stacked autoencoders to do zero-day web attacks.In contrast to the traditional rule based approaches such as Web Application Firewalls (WAFs), which resist new threats, the presented model is educated on regular web requests.It uses a special tokenization technique that classifies characters that should be converted last requests into ordered number-based sequences, increasing the accuracy of detection. By using a combination of latent features of all three autoencoders and compressing them, the model is an effective way of discovering anomalies. The accuracy of the system is 97.58% 99.99% accuracy, and a remarkably low false positive rate of 0.2 per cent, and doing better than most prior methods. This renders it a valid and extensible solution to the detection of emerging web threats.

This paper [38] introduces a zero-trust architecture to secure AI models from physical file-based cyber attacks, particularly serialization-based exploits in formats like Pickle and PyTorch. It presents two key defense mechanisms:

- **Content Disarm and Reconstruction (CDR)** – through a novel "safe-unpickler" that filters malicious functions during model deserialization, ensuring secure loading of AI models.
- **Moving Target Defense (MTD)** – which randomizes and encrypts model weights, separating them from architecture and preventing unauthorized use or tampering.

The system achieves 100% attack prevention against real-world malicious models

from HuggingFace and passes all validation tests. By securing the model loading and distribution pipeline, it effectively guards against emerging threats in AI model supply chains. This work fills a crucial gap in AI file security and complies with recommendations from agencies like the NSA and OWASP.

This paper [34] addresses the challenge of malware evolution diminishing the effectiveness of API sequence-based Windows malware detectors over time. It proposes a novel framework called MME (Mitigating Malware Evolution) that enhances deep learning models by capturing the semantic similarities in API sequences before and after malware changes. MME recognizes functionally with an API knowledge graph similar APIs and embeds resource access trending and applying contrastive learning to strengthen training of similar malicious behaviors, similarly. The method has a much greater reduction in false negatives (more than 13 percent) and higher F1 scores (up to) 8.5% over four years of malware records as well as decreasing the necessity to use hands labeling. That is why MME is a powerful approach to malware with sustainable long-term detection. MME is superior to previous techniques like APIGraph which improve either of the two.Embeddings of API name and arguments and by adding contrastive learning in the encoder. The results produced by this heterogeneous defense are more generalized developed malware, no longer API replacements. The enabler of the framework is that it has a limitation in the initial investment needed to develop and sustain the API knowledge graph. The authors suggest that in the future the updates may be automated and managed industry on the online learning to deal with new malware using real time it has just appeared.

2.3 Summary of Key Findings

The limitations of the latter is highlighted by both Brinkley et al. and Rasheed Ahmad et al. [22] traditional IDS models, they rely on fixed rules or older datasets, which do not work to identify new threats such as the zero day malware. Adaptive is recommended in these studies systems of learning are capable of retraining new data and relying on AI model optimization performace in practice between memory and compute efficiency.Al-Rushdan et al [12] presented a bi-layered defense framework inside the SoftwareDefined Networks (SDNs), which integrates behavior examination through Cuckoo Sandbox with.Traffic inspection in real time. The model had a detection rate of 97.78 and poor denotes scalability and resilience to avoid evasive malware, which points to the necessity of hardiness behavior-based frameworks which can be used on Windows and Android.Mahindru and Sangal [11] proposed MLDroid, an Android malware detector that is dynamic. exploitation of monitored machine learning and unmonitored machine learning. It showed the possibilities of runtime behavior tracking with 98% accuracy and feature selection, in contrastive learning goals in cross-platform malware detection.Li et al. [29] and Zhang and Zhang [13] suggested that grayscale image representations of malware binaries be used with deep CNNs such as VGG16. While highly exact (up to 99.25%), these methods hold no behavioral context and have problems with. realistically close variations the necessity of semantic and executionbased detection, especially within

windows operating systems.Malware evolution in the MME framework by Zahra et al [9] was addressed by embedding it. Behavioral patterns on API level and contrastive learning to minimise false negatives. This method demonstrated better detection over a long period on developing Windows. malware, which provides a viable path toward cross platform flexibility.JothiShri et al. [28] used hybrid deep learning models (CNN-RNN) in anomaly. detection in network traffic, realized 95in detecting zero-day activities in inhomogeneous systems. The positive results of random forest classifiers on CICIDS2018 in real-time zero-day detection were shown by Ibraheem and Tosho [27] Although it is true, there are still issues with generalization between platforms, particularly Android. The future directions involve lightweight, integrated models that have enhanced management of different execution environments. The journal article by Challita and Parrend [37] introduced RedTeamLLM, an agentic AI framework for offensive security tasks. While not detection-focused, its reasoning and memory mechanisms offer potential inspiration for adaptive learning loops in malware detection frameworks, especially for systems capable of learning from unknown threats over time.

Finally, Sarker et al. [18] proposed a hybrid AI model combining decision trees, k-means clustering, and NLP techniques for anomaly detection. Although recall for novel threats was lower (78%), the model's use of unstructured log analysis could benefit a contrastive-behavioral hybrid approach targeting both Windows and Android ecosystems.

The table 2.1 below, summarizes various state-of-the-art malware detection techniques, comparing their detection rates, strengths, and limitations across different platforms.

Technique	Platform	Detection Rate	Strength	Weakness
Cuckoo Sandbox (SDN)	Windows (SDN)	∼97.78%	Behavioral analysis in isolated environment; real-time profiling	Poor evasion handling; high resource usage; limited scalability
MLDroid [11]	Android	∼98%	Combines supervised & unsupervised ML; runtime behavior tracking	Platform-specific; ML performance depends on training data
VGG16 CNN Grayscale [13], [29]	Windows	∼99.25%	High accuracy using grayscale image conversion of binaries	No behavioral context; struggles with visually similar variants
MME Framework [9]	Windows	∼96%	API-level behavior + contrastive learning for malware evolution	Requires frequent API database updates
Hybrid CNN-RNN [28]	Cross-platform	∼95%	Captures spatial + temporal patterns for anomaly detection	Deep learning model may not be lightweight; tuning needed
Random Forest on CIC-IDS2018 [27]	Cross-platform	∼94%	Real-time classification; interpretable ML model	Limited generalization across OS platforms
Grayscale + Transfer Learning [13]	Windows	∼99%	Leverages pretrained models for rapid classification	Ignores runtime behavior; vulnerable to adversarial examples
Behavioral + NLP Hybrid [18]	Windows & Android	78–85%	Uses log analysis, clustering, and DTs; good for unstructured data	Lower recall for novel threats; complex pipeline
OmniUnpack [1]	Windows	∼95%	Fast unpacking of packed/obfuscated malware	Ineffective against advanced evasion (anti-sandbox)
RedTeamLLM Framework [37]	Not detection (Offensive AI)	N/A	Agentic AI with reasoning + memory; potential for adaptive defenses	Not a direct detection system; more useful for adversarial simulation and training

Table 2.1: Summary of Key Malware Detection Techniques

Chapter 3

Requirements, Impacts and Constraints

3.1 Final Specifications and Requirements

This section displays the final system requirements and specifications of the proposed cross-platform zero-day malware detection framework. To find known and unknown malware threats in Windows, the system combines behavioral monitoring, static analysis, and self-supervised anomaly detection techniques. The requirements specify the functional and technical conditions required for the system to function properly, whereas the specifications specify what the system is expected to do.

3.1.1 System Objectives

The main goal of the suggested system is to offer a practical, real time and flexible way to find zero day malware that conventional signature based techniques are unable to detect. The system's goal is to reduce false positives while preserving high detection accuracy on various platforms and file types. Finding threats with unusual byte level patterns or abnormal behavior that have never been seen before is given special attention.

3.1.2 Functional Requirements

The proposed malware detection framework must satisfy the following functional requirements:

1. **Real-Time File Monitoring:** In order to identify newly created or altered files, the system must constantly scan user accessible directories like Downloads, Documents, removable media and application folders.

2. **Multi-Layer Detection Pipeline:** The system will use a staged multi layer architecture to analyze files, with each layer conducting increasingly in depth inspection. Files that have been marked as suspicious in earlier layers will be forwarded to later layers for additional examination.

3. **Static Threat Detection (Layer-1):** To swiftly identify known or highly likely threats, the system will conduct static analysis using hash reputation

checks, YARA rules, file-type validation, and machine-learning-based classification (LightGBM).

4. **Behavioral Analysis (Layer-2):** In order to identify deviations from expected behavior for a particular file type, the system will monitor the short term execution behavior of files, including process creation, system calls, registry access and file system interactions.

5. **Self-Supervised Byte-Level Analysis (Layer-3):** The system will use a self supervised learning model that has only been trained on benign data to analyze files at the raw byte level. In order to detect zero day malware without the need for labeled malware datasets, the model will produce anomaly scores based on statistical deviation.

6. **Network Behavior Monitoring (Layer-4):** In order to detect fileless malware and questionable communication attempts, the system will keep an eye on runtime network activity, including connection patterns, destination analysis and data transmission behavior.

7. **Automated Response and Quarantine:**When malicious activity is identified, the system will automatically restrict network access and quarantine files that surpass predetermined risk thresholds.

8. **Explainable Risk Reporting:** To promote transparency and forensic analysis, the system will produce structured logs and reports with risk scores, detection reasons and decision outcomes.

3.1.3 Non-Functional Requirements

To ensure usability and practicality, the system must also meet the following non-functional requirements:

1. **Performance Efficiency**
   In order to enable real-time detection without appreciably impairing system performance, the system must function with low latency.

2. **Scalability**
   The architecture shall support deployment across multiple devices and environments, allowing future expansion to enterprise-level or federated learning settings.

3. **Cross-Platform Compatibility**
   When necessary, the system will adjust to platform-specific features while maintaining uniform detection logic across Windows platforms.

4. **Robustness and Reliability**
   The system shall remain effective against obfuscation, packed malware, malformed files, and encrypted payloads.

5. **Security and Isolation**
   In order to stop additional system compromise during analysis, detected threats must be safely isolated.

3.1.4 Hardware and Software Requirements

The system requires the following resources for optimal performance:

- **Hardware Requirements**
  - Minimum 8 GB RAM
  - Multi-core CPU
  - Optional GPU support for deep learning inference
- **Software Requirements**
  - Windows and Android operating environments
  - Python-based machine learning frameworks
  - TensorFlow/Keras or equivalent deep learning libraries
  - File system monitoring and sandboxing tools

3.1.5 Design Constraints

Due to several factors including limited labeled zero day malware data, limitation of resources on low end devices and privacy concerns of behaviour tracking, the system is limited in its design. To eliminate these challenges, the framework proposed gives heavy attention to lightweight feature extraction and self supervised learning and modular layer based escalation.

3.2 Societal Impact

Effective zero-day malware detection has much more far-reaching effects than the field of cybersecurity. Digital technologies are growing as well as the number of threats, which also take advantage of the vulnerabilities in them. These attacks are known as Zero-day attacks and they are used to attack these unknowns weaknesses which are a great threat to individuals, organizations and governments.The social consequences are far reaching:

- **Sensitive Information Protection:** Zero-day attacks may result in loss of sensitive personal and intellectual property information, and even sensitive national security property. These weaknesses pose a growing threat to data privacy in the realm of a fast-digitizing healthcare sector, financial and governmental sectors.
- **Protection of Critical Infrastructure:** The planned malware detecting system will protect not only one device but also the critical infrastructure. It reduces the interference caused by malware usage in real-time, which limits the impact of this malware on critical services, such as banking and healthcare and energy systems.
- **Increasing Trust in Digital Ecosystems:** With the increase in dependency on the digital services, trust among users is critical. Malware attacks can be easily neutralized by enhanced protection systems making sure that

e-commerce websites, web-based banking and other web-based platforms are secure, and this will help instill confidence in users to continue using these services and enable the industry to expand.

- **Financial Effects:** Data breach has enormous financial effects. Ponemon Institute (2020) reported in their 2020 Cost of a Data breach report that the cost of an average breach in the U.S. was USD 8.64 million. Detection systems like malware can easily be proactive thus minimizing the financial risk in cases of breaches.
- **Impact to Global Cybersecurity:** With the ever-growing problem of cybercrime heightened around the world, the necessity of having an effective security among the global citizens becomes more acute. By defending against zero-day threats, this study has a contribution to the world community.Effort of granting digital systems and cutting down the economic and operational costs.

3.3 Environmental Impact

Although the immediate impact of cybersecurity solutions on the environment is not always taken into account, Security implementations can consume a lot of energy and resources to perform with large scale implementations. This hybrid and proposed system has its merits.Multi-layered detection approach has been developed in an efficient way. Making use of lightweight machine learning models and optimization of the processing methods and guarantees that the system does not have a large carbon footprint, but remains high detection rates.A number of works point out the eco-implication of artificial intelligence (AI) models in cybersecurity. The AI models and especially deep learning were found networks, absorb an immoderate portion of energy and it is becoming a source of growing demand in computational power. But this hybrid structure is the characteristic of the architecture system, that is a combination of several levels of detection and optimization of models, guarantees that there is efficient utilization of resources, which can be identified in real-time without being overused strain on hardware. In addition, the design of the system puts a lot of emphasis on scalability between devices of different levels computational power, which will not add to environmental degradation but is also made sure that even lower-end devices could make use of the advanced malware detection features. The sustainability of cybersecurity technology is guaranteed as it becomes more widespread.It will play a very important role in lowering the environmental impact of the global IT infrastructure.

3.4 Ethical Issues

Malware detection systems must navigate several ethical challenges, particularly in terms of privacy, fairness, and surveillance:

- **Privacy Concerns:** The proposed system considers the privacy of the user by making use of benign only training and self-supervised learning. This avoids direct entry to confidential information and prevents that personal information should be abused or exposed during analysis.

- **Fairness in Predictions:** It will be important to have fairness in the decisions of the system. The study includes the transparency and explainability features so that the users and administrators could understand the reason of detection results to lessen the probability of biasing results that could classify rightful actions or fail to identify the harmful activities.

- **Surveillance Overreach:** In order to avoid it, the system should not engage in unnecessary surveillance, it works based on the principle of least privilege and only the system behaviors are tracked, which are of relevance in detecting potential malware. This guarantees insecurity without infringing on users' privacy.

3.5 Project Management Plan

The developmental project of the zero-day malware detection system uses the Agile project management structure, which encourages flexibility, incremental development and frequent reassessment. The project is broken down into specific stages each of which has specific objectives and deliverables to make sure of systematic improvement:

- **Phase 1: Design and Planning** In this first phase, system architecture will be designed.It will be developed with special attention to the zero-day malware detection in Windows ecosystem. The main actions involve determination of the core detection layers(behavioral monitoring, self-supervised anomaly detection and static analysis)and providing specifications of requirements of system performance and scalability. A full project roadmap of the development will be created and testing processes.

- **Phase 2: Model Development and Training** This stage entails the development and training of the model.The training and development of machine learning tools that are specific to identify zero-day and threats on Windows systems. The benign data will be used to train the models.6 evading the use of possibly dangerous malware samples, which is in accordance with the self-supervised learning strategy. The team shall be concerned with making sure that the models can identify malicious patterns unaware of the past without the necessity on labeled data, solving the problem of zero-day threats.

- **Phase 3: Testing and Validation** The models will be developed and after that they will be tested and validated.It will be fully tested in order to check their functionality in real life situations.The stage will be concerned with the detection and minimization of the false positives, which will guarantee that the detection system is reliable and also effective in identifying zero-day threats. The detection features by the system will be tested against a variety scan known and unknown attack scenarios of the windows environment to provide good coverage.

- **Phase 4: Deployment and Maintenance** When the system has successfully been tested and validated, it shall be deployed in real time malware detection. Ongoing Continuous renovation of the system to meet new requirements

will also require maintenance software infections as they occur. Continuous monitoring of this phase is involved. looping and improvement of the system, and periodic revision of the detecting models, it is made to ensure that the system is not ineffective against changing attack strategies. This will make this project certain by adhering to a structured but flexible approach. Quick and reliable delivery without being rigid to other developable issues in securities.

3.6 Risk Management

Since the detection of the zero-day malware is complicated, a number of risks need to be thoughtfully maintained in the process of development and deployment:

1. **Data Scarcity:** Data scarcity is one of the key issues, namely the lack of labeled zero-day malware data. It is alleviated by the uptake of self-directed learning methods.This danger because the system is allowed to be taught using benign data instead of the required malicious or partial malware code.

2. **False Positives:** False positives are also a significant issue to any malware detection system. In order to solve this, many layers of detection are installed in the system, enabling the decision-making to be refined at every stage.

3. **Performance on Low-End Devices:**The system should be streamlined so as to scale to various computing devices, especially those with few computational capabilities. This difficulty is taken care of using lightweight machines learning models and effective data processing methods.

4. **Adaptation to New Threats:** The system must be able to accommodate new malware variants is achieved with continuous learning mechanisms which are incorporated,the identification system to be modified with the appearance of new risks.

3.7 Economic Analysis

One can note that the malware detection system is economically viable in both aspects. cost-saving and a market-expansion orientation:

1. **Cost Savings:** The system will be able to mitigate the financial effects of cyberattacks,by preventing malware that has not reached significant damage through detection of zero-day malware and preventing it. It was approximated that global cost of cybercrime USD 1 trillion/ year with emphasis. The significance of informing substantial costs in malware detecting systems in place businesses and individuals.

2. **Scalability:** The system has been designed to scale effectively to support small-scale devices and large-scale enterprises. This scalability lets make sure that the solution is applicable in many industries both small companies to large corporations.

3. **Market Demand:** The rising level of the sophistication of the cyberattacks, especially zero-day attacks, has resulted in an increasing demand of the advanced detection solutions.The special hybrid manner makes this system a worthwhile one service available to companies that aim to secure their online information and databases.The cybersecurity services market size in the world is projected to increase by a compound annual growth rate (CAGR) of 10 percent over a period of five years, which offers critical opportunities of the proposed solution.

Chapter 4

Proposed Methodology

4.1 Methodology Overview

In this chapter the author describes what is the structure of our detector and why it is so way. We want to not only the zero-day threats early in the windows eco-system, but also maintaining the constant false alarm rates and low latency rates when a file is accessed. We have three rules: (i) execute the quickest reliable checks first; (ii)upgrade when Low confidence and (iii) make log decisions so that thresholds can be turned later. This is the staged pipeline approach is based on larger-scale results of malware detection studies that rely on consistency and multi-layered analysis to handle diversity in the operating systems and enhance identification of zero-day threats [26]. With these principles in place we design the detector in such a way that it is a staged pipeline that reduces risk step-by-step. We apply a four-layer architecture to identify the zero-day malware in Windows.

- **Layer-1** : It monitors real time download/working folders,checks YARA,checks Reputation hashes, verifies and MIME new files immediately vs extension and executes a LightGBM fixed model (auto-selecting the right) model by file type), is a combination of Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to process huge datasets in an efficient way. [4]. Depending on the confidence scores as per the model, the detections are quarantined immediately or forwarded to more analytic levels.

- **Layer-2** : This layer does behavioral and rule based analysis of files and with inconclusive results in Layer-1. It follow up short execution traces, process. behavior and system interactions to identify suspicious behaviors, e.g. scripts. or records starting astute processes. Acting as a bridge between static and deep analysis, Layer-2 assists in culling of borderline cases prior to blowout to later stages.

- **Layer-3** : MalConvGCT layer takes raw input, executable file, where the file's content is used explicitly without feature engineering by using a 1D convolutional network and the Global setting.The mechanism of Channel Gating (GCT). It works well with files of any length with fixed memory consumption.

- **Layer-4**: This layer studies real-time network behaviour of processes, building standing knowledge of regular operations and identifying an unusual state of

affairs including unusual links, misuse of ports, or dubious process paths. This layer is designed to catch fileless malware, data exfiltration, and LOLBin-based attacks that bypass file scanning.

Each layer gives outputs a risk score and reasons. If a file is clearly malicious at a given layer, we quarantine immediately; otherwise, we elevate to the next layer. All actions are logged for audit and future tuning.

The following figure 4.1 image illustrates our complete detection pipeline structure-

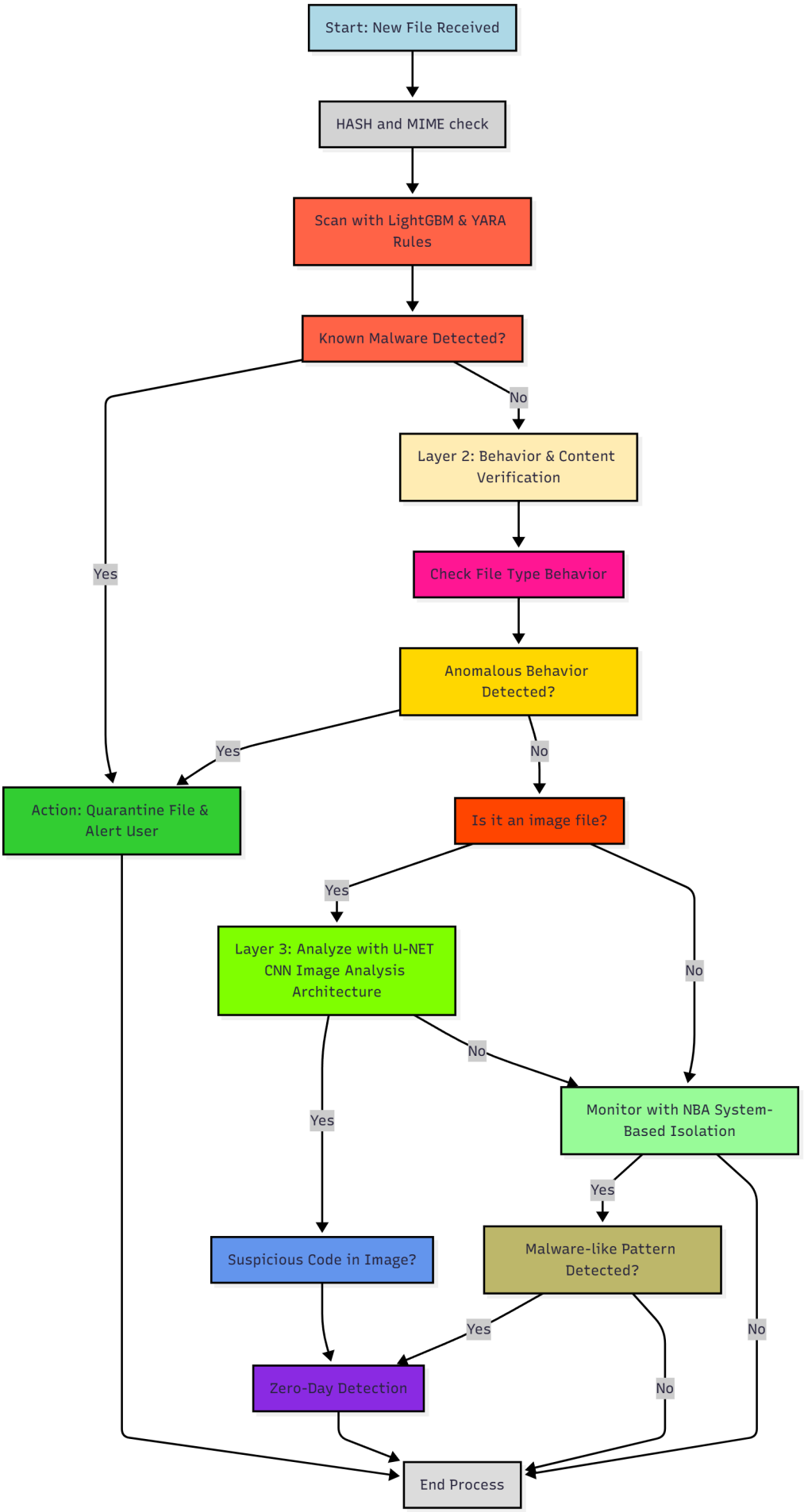

Figure 4.1: Full Pipeline

4.2 Preliminary Design / Design Specification

4.2.1 Layer 1

Goal and constraints

The primary objective of Layer-1 is to provide a rapid, low-latency first pass that can block known or highly probable threats at file ingestion. This stage must operate under minimal computational overhead while maintaining a traditional policy: if confidence is high, the file is quarantined immediately; if uncertainty exists, the item is passed on to Layer-2 or Layer-3. This makes sure that the system prioritizes availability and low false-positive rates while still offering immediate protection against clearly defined threats.

Real-time flow

Layer-1 is invoked by a file system watcher that monitors key user and shared directories, including Downloads, Desktop, Documents, OneDrive/Dropbox, messaging app download folders, AppData/Temp (with filters), and removable drives. Monitoring is event-driven: both on_created and on_moved file system events trigger scanning, ensuring that newly added or relocated files are analyzed without delay. The Layer-1 scan pipeline proceeds as follows:

1. Hash and YARA checks
2. Static machine learning model inference (LightGBM)
3. Decision → *Quarantine / Escalate to Layer-2 or 3 / Pass*

To illustrate the functioning of the first detection layer, Figure 4.2 presents the workflow of Layer-1 real-time scanning-

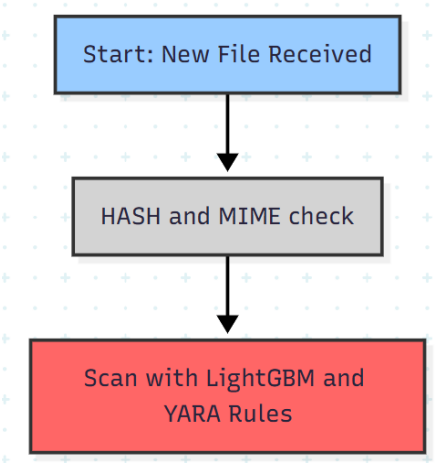

Figure 4.2: Workflow of Layer-1 real-time scanning

The process begins with new file reception, followed by known-threat filtering and static analysis using LightGBM. Based on detection outcomes, files are either quarantined, escalated to deeper layers, or passed as benign.

Detection Components and Actions

- **Known-bad hash blacklist:** Calculates the hash of the file (SHA-256) and makes a comparison against the maintained database on bad hashes, and isolates any match immediately.
- **Static ML (LightGBM):** Combines an EMBER-type PE classifier and LightGBM files-type models that will automatically choose the right model and block files in case the prediction score is higher than the LGBM block threshold. This is the technique that uses the EMBER dataset to demonstrate the LightGBM scalability with respect to detecting static malware in large scale [4].
- **YARA signature detection:** YARA rules are classified into severity, Preventing files on critical matches and scaling up moderate/low severity matches to do more analysis. This is based on more extensive malware work, preferring decisive actions and having weaker indicators to reinforce escalation [26].
- **File-type and MIME verification:** Cross validates specified file extensions with real MIME types; discrepancies are eschewed to escalation or quarantine.
- **Quarantine system:** Files that are in suspicion are moved to a quarantine that is secure folder (`quarantine/filename.ts.quar`) and original files are saved to undergo forensic examination.

Filtering Logic

To reduce noise while maintaining coverage, Layer-1 applies the following policies:

- **Size filters:** Skip files < 256 bytes in noisy directories (e.g., `Temp`, `AppData`) and files < 1 byte elsewhere.
- **Extension filters:** In noisy directories, only files with high-risk extensions (e.g., `.exe`, `.pdf`, `.docx`) are scanned.
- **Special handling:** Certain small but high-risk types (`.lnk`, `.js`, `.vbs`, `.ps1`) are always included.
- **MIME mismatch:** Always flagged regardless of size or extension.
- **ZIP-based Office files:** Treated as medium severity unless combined with other indicators.

Decision Policy

- **Block (quarantine)** if any strong signal is present: known-bad hash, VT malicious count $\geq$ threshold, LightGBM $\geq$ `LGBM_BLOCK_THRESH`, or critical YARA signature + suspicious metadata (e.g., MIME mismatch).
- **Escalate to Layer-2/3** if signals are weak or ambiguous (e.g., moderate YARA, MIME mismatch alone, borderline model score). Since borderline model outputs can suffer from miscalibration, it is important to treat uncertain scores with caution and defer to deeper layers [26].
- **Pass** if no suspicious signal is present.

Risk Categorization and Reporting

Layer-1 outputs results in a structured JSON log, where each detection reason is tagged with a color code for clarity and visualization:

Condition	Action/Meaning	Color
Hash blacklist / critical YARA hit	Strong static detection	Magenta
ML model score above threshold	High-risk prediction	Cyan
MIME mismatch	Suspicious, requires review	Yellow
Clean (no suspicious indicators)	Passed Layer-1	Green

Table 4.1: Layer-1 risk categorization and reporting scheme.

Although Layer 1 (static analysis) tests the known malware based on signatures or in-advance patterns though not very resistant to the 0-day attacks and polymorphic malware that contain no corresponding signatures. To overcome this Layer 2 introduces dynamic behavioral monitoring, a monitoring mode that cares about the behavior of a file on which a file is executing, which also includes filial superiority or atypical file permissions. This helps in countering threats which are overlooked by Layer 1 that has in it some form of a covered or obfuscated malware.

4.2.2 Layer 2

Goal and Scope

The Layer-2 takes the detection pipeline to an even further stage than simply analysing a static type of object, into dynamic behavioural analysis. It aims at monitoring the behavior of a suspect file when implemented, it becomes hard to oversee the existence of runtime anomalies like privilege abuse, file system meddling or malfunctioning behavior of process. Whereas Layer-1 is concerned with the appearance of a file.Similar, Layer-2 is deliberated on what it does. Such design helps this system to isolate the polymorphic or masked threats that can go through signature-based as well as through the static machine-learning checks.

Operational Flow

Layer-2 is activated rather asynchronously when Layer-1 marks a file as suspicious but not conclusively malicious. The sampled sample is implemented in a sandboxed environment, environment which records file, process and network occurrences in real time. Behavioral write operations on files, chains of creating processes, registry operations and network connections are monitored to short-term observation intervals (usually 30-60 seconds). The non-fat design ensures that the overhead of the CPU will be low, yet it will have sufficient transparency to identify abnormal actions at an early stage. Behaviors identified are rated and accumulated into a Layer-2 risk metric (s2), upon which to quarantine or not, escalate, or hand out the file to lower levels.
The following figure 4.3 demonstrates how Layer-2 performs behavioral and content analysis to verify file types and detect anomalies.

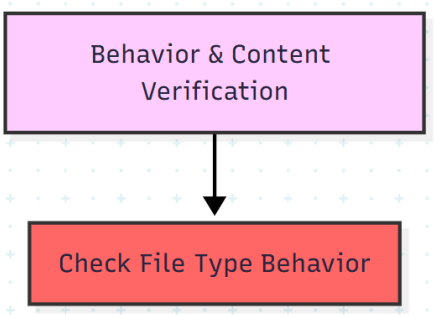

Figure 4.3: Layer-2 Behavioral Analysis Workflow

Behavioral Indicators Monitored

Layer-2 keeps track of multiple categories of malicious actions drawn from previous malware-behavior research and core defense standards:

1. **File-System Anomalies** – unauthorized permission changes, creation of executable files in protected directories (e.g., System32, AppData), or deletion/modification of system binaries.
2. **Privilege Escalation Attempts** – misuse of sudo / RunAs, DLL or code injection, and unauthorized modification of Access Control Lists (ACLs).
3. **Network Irregularities** – unexpected outbound connections, C2 communication attempts, or data exfiltration via unusual ports and IPs.
4. **System Resource Abuse** – CPU or memory spikes caused by hidden mining or denial-of-service routines.
5. **Persistence Mechanisms** – registry autoruns, scheduled-task creation, or unauthorized service installation.
6. **Security Tampering** – disabling antivirus, deleting event logs, or changing firewall settings.
7. **Abnormal Execution** – launch of scripting interpreters (PowerShell, Python, cmd) or remote-access tools without user initiation.

8. **Data Corruption** – encryption, wiping, or modification of user files, signaling ransomware activity.

Each behavioral category contributes a weighted score to s_2, derived from frequency and severity of events within the observation window.

System Design and Monitoring Features

Layer-2 incorporates the following mechanisms for efficient real-time tracking:

- **Process-tree monitoring** – traces all descendant processes up to 10 levels deep to capture propagation behavior.
- **Network activity inspection** – detects outbound traffic to known malicious or previously unseen IPs and ports.
- **Privilege and access monitoring** – logs elevation attempts, registry modifications, and ACL changes.
- **Fast-polling engine** – operates at 0.01 s intervals to ensure minimal latency.
- **Runtime anomaly flagging** – marks suspicious execution patterns such as repeated PowerShell launches or script chaining.

The layer operates primarily on Windows environments, monitoring only files escalated from Layer-1 and restricted to watched directories to conserve system resources.

Decision Policy and Escalation

- **Quarantine** – Immediate isolation occurs when high-confidence behaviors (e.g., registry tampering + unauthorized network activity) are observed.
- **Escalate to Layer-3/4** – Triggered for ambiguous or stealthy behaviors such as moderate privilege escalation without payload delivery.
- **Pass** – If runtime activity remains consistent with benign patterns.

Each Layer-2 outcome is logged with timestamp, process ID, behavioral signatures, and cumulative risk score s_2. This structured trace log allows post-incident tuning of behavioral thresholds and correlation with subsequent network-layer detections.

4.2.3 Layer 3

Objective

Layer-3 introduces a deep learning component focused on detecting hidden or covert-channel malware grafted inside image pixels. Traditional antivirus engines cannot identify such threats because malicious code is masked in the RGB values of natural images.
This layer implements a pixel-level anomaly detection system built on an Enhanced U-Net architecture, capable of pinpointing the exact (x, y) coordinates of suspicious regions rather than offering only image-level classifications.

Here is the Layer-3 image-based detection workflow explains how image files are analyzed using a CNN architecture for embedded threat detection -.

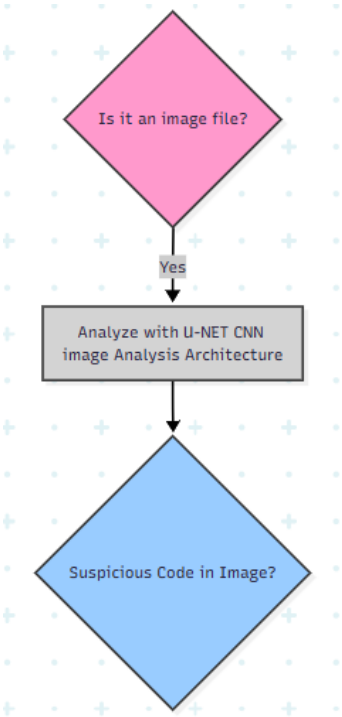

Figure 4.4: Layer-3 Image-Based Detection Workflow

This diagram illustrates the decision flow for image analysis in Layer-3. Files identified as images are processed through a VGG-16-style CNN for anomaly detection. If hidden or suspicious pixel patterns are found, the file is quarantined; otherwise, it passes to network-level verification in the next layer.

Model Architecture

The model consists of an encoder–decoder U-Net structure containing approximately 7.86 million parameters. It integrates contextual attention mechanisms across four hierarchical levels (64 $\rightarrow$ 128 $\rightarrow$ 256 $\rightarrow$ 512 channels) and skip connections that preserve spatial information. Two output heads are used:

- an **anomaly map** providing pixel-wise localization of manipulated areas, and
- a **confidence map** summarizing overall anomaly intensity.

This architecture enables contextual understanding of "normal" pixel behavior within local neighborhoods, making it possible to distinguish natural textures from hidden manipulations.

Dataset Preparation and Acquisition

Training data were derived from the **ALASKA2** natural image corpus [10], containing about **305,000** clean images of landscapes, objects, and textures. Artificial anomalies were injected into these images to simulate steganographic and pixel-level manipulations, allowing the model to learn without requiring real malware samples. Six manipulation types were generated: isolated pixel outliers, least-significant-bit (LSB) flips, clustered anomalies, brightness extremes, localized color shifts, and statistical inconsistencies.
Images were resized to **256 × 256 pixels** to balance spatial detail and GPU memory limits. Dataset division followed a **70% / 20% / 10%** split for training, validation, and testing, respectively.

Training Configuration

Training was performed using the Adam optimizer with a learning rate of 0.0001, minimizing binary cross-entropy loss with pixel-wise weighting. Data augmentation included random rotations, brightness and contrast shifts, and Gaussian noise addition.
The system trained for 50–100 epochs, depending on convergence, with early-stopping and checkpoint mechanisms (`latest_checkpoint.pth`, `checkpoint_epoch_N.pth`, `best_model.pth`). Batch size was reduced from the ideal 8–12 to 6 due to an 8 GB VRAM limit.
All training states—model weights, optimizer, scheduler, and history—were saved automatically for reliability and reproducibility.

Output and Evaluation

Layer-3 produces three distinct types of outputs following model inference:

1. **CSV files** – contain pixel coordinates and their corresponding anomaly confidence scores for detailed forensic review.
2. **JSON summaries** – record image-level metadata, anomaly regions, and associated confidence values in a structured format.
3. **PNG heatmaps** – visualize detected anomalies as color overlays on the original images, highlighting regions of interest for manual validation.

Model evaluation was conducted using standard quantitative and pixel-level metrics. After **five** training epochs, the model achieved:

- **Train Loss:** 0.1176
- **Validation Loss:** 0.0629
- **Precision:** 0.9997
- **Recall:** 0.7002
- **F1-Score:** 0.8236
- **AUC (Area Under Curve):** 0.8992
- **Epoch Duration:** ∼589 seconds
- **GPU Memory Usage:** 0.23 GB

The diagram below presents the training and validation performance of the Layer-3 model, including loss, precision, recall, and other evaluation metrics-

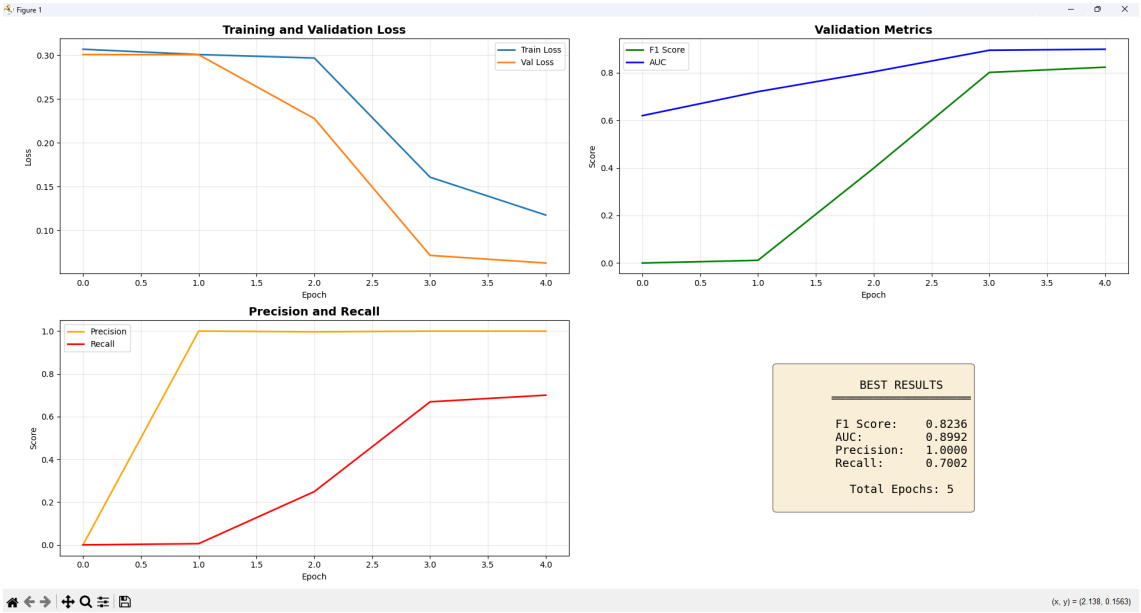

Figure 4.5: Training and Validation Performance of Layer-3 Model

These metrics collectively assess both classification and localization performance. Detailed numerical results and comparative analyses will be inserted once the final testing and validation phases are completed.

Observed Challenges and Limitations

Development encountered several technical limitations:

- **Boolean tensor conversion errors** during validation were corrected through explicit float conversion.
- **Windows multiprocessing issues** required setting `num_workers = 0`, lowering speed by 20–30 %.

- **Python 3.11** `pathlib` inconsistencies were handled by switching to `os.walk()` for dataset scanning.
- **Limited VRAM** forced image resizing from **512 × 512** to **256 × 256**.

Practical constraints include slower processing on non-GPU systems, occasional false positives on natural image textures, and dependence on synthetic rather than real-world malware data. Despite these issues, the layer provides fine-grained localization, contextual awareness, and production-ready checkpointing, establishing a strong foundation for detecting steganographic or hidden malware within image files.

Layer 3 is designed to perform an exhaustive content-level analysis on files that were able to evade previous static and behavioral analyses. Initially, Layer 3 adopted a CNN-based approach to steganography detection, focusing on detecting hidden malicious content within image files. Although it was effective in detecting steganographic content, it was found to be less robust in detecting stego files created using different steganographic algorithms. In addition, it failed to perform well when detecting malware embedded in files other than images. The reliance on assumptions about file type, structure, and embedding technique also made it less effective in detecting zero-day malware.

In order to overcome these shortcomings, Layer 3 was redesigned to operate on raw byte sequences, eliminating the need to assume file structure, encoding, or embedding technique. As a result, Layer 3 adopted a deep learning-based approach to detecting malware, relying on the **MalConvGCT model**, which is effective in detecting unknown malware variants in executable files of any length.

Proposed Approach: MalConvGCT Byte-Level Model

The new Layer 3 architecture is based on the deep learning framework **MalConvGCT**, specifically tailored for malware detection. This approach differs from traditional malware analysis techniques, as it does not require feature extraction or disassembly of executables. Instead, it operates directly on executable files as sequences of bytes, providing a more general and robust detection method.

Byte Representation and Embedding

Executable files are treated as sequences of bytes, with each byte represented as an integer ranging from 0 to 255. These bytes are first converted into learned vector representations through a *byte embedding layer*, analogous to word embeddings in natural language processing (NLP).

Temporal Convolutional Network

The sequences of bytes are then processed through a *temporal convolutional network* (TCN), which applies one-dimensional convolution over the byte sequences to learn discriminative patterns characteristic of malware.

Global Channel Gating (GCT)

A key component of the network is the *Global Channel Gating (GCT)* mechanism. It computes global statistics over the entire input sequence and selectively emphasizes the most discriminative convolutional channels. This ensures that the network can handle sequences of varying lengths while maintaining constant memory usage, improving scalability and efficiency.

Output and Probability Score

The final network components generate a probability score representing the likelihood of the input file being malicious. Files with high malware probability scores are escalated for further action.

Dataset and Experimental Setup

Layer 3 was tested using a dataset derived from **VirusShare_00489**, which includes real-world malware samples. The malware set consists of 65,537 Windows PE executables, while the benign set includes 1,171 legitimate executables from trusted Windows system directories. The data was split into 80% training and 20% testing to ensure generalization.
The training process was conducted for 10 epochs on an **NVIDIA RTX 4070 GPU**, with the best checkpoint occurring at an early epoch, reflecting rapid convergence and effective learning. The dataset was chosen for its diversity, size, and popularity in malware detection studies, making it suitable for evaluating byte-level detection capabilities.

Performance Evaluation and Observations

The byte-level Layer 3 model demonstrated outstanding performance on the test data:

- Accuracy: 97.19%
- Precision: 97.05%
- Recall: 98.16%
- F1-score: 97.60%

From the confusion matrix, only 3 malware instances evaded detection among more than 13,000 test cases, while 18 benign files were misclassified as malicious.

This indicates a high success rate for the MalConvGCT-based method in detecting malicious executables while maintaining a low false alarm rate. Importantly, because this method operates at the byte level rather than relying on signatures or known malware families, it has significant potential for detecting zero-day malware, whose patterns are unknown.

The following figure 4.6 illustrates the graph of Precision-Recall curve, highlighting the model's effectiveness in balancing precision and recall during malware detection.

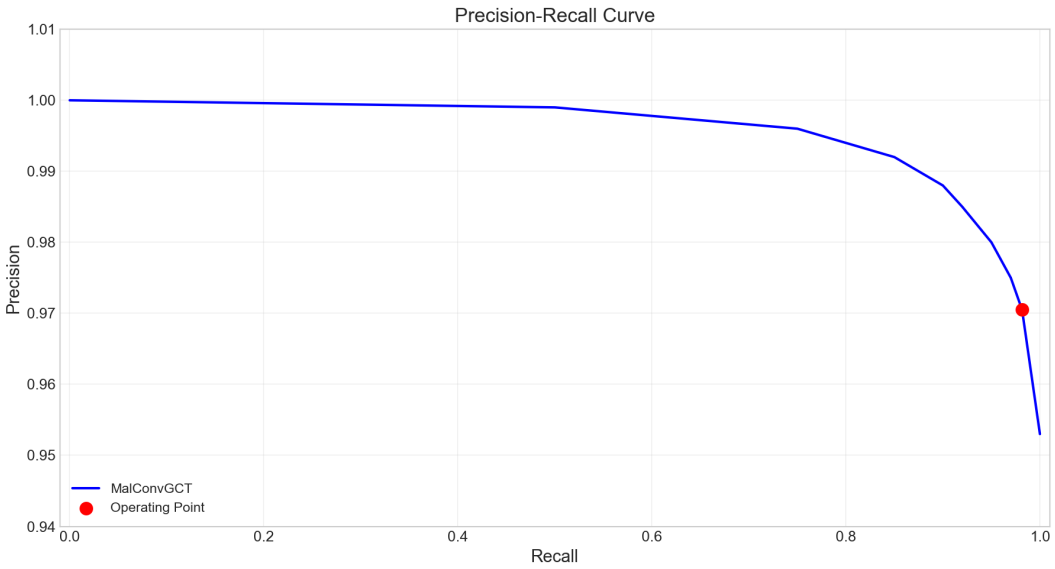

Figure 4.6: Precision-Recall curve showing high performance

The following bar chart of the figure 4.7 displays the performance metrics of the MalConvGCT model, showing strong results in accuracy, precision, recall, F1 score, and AUC-

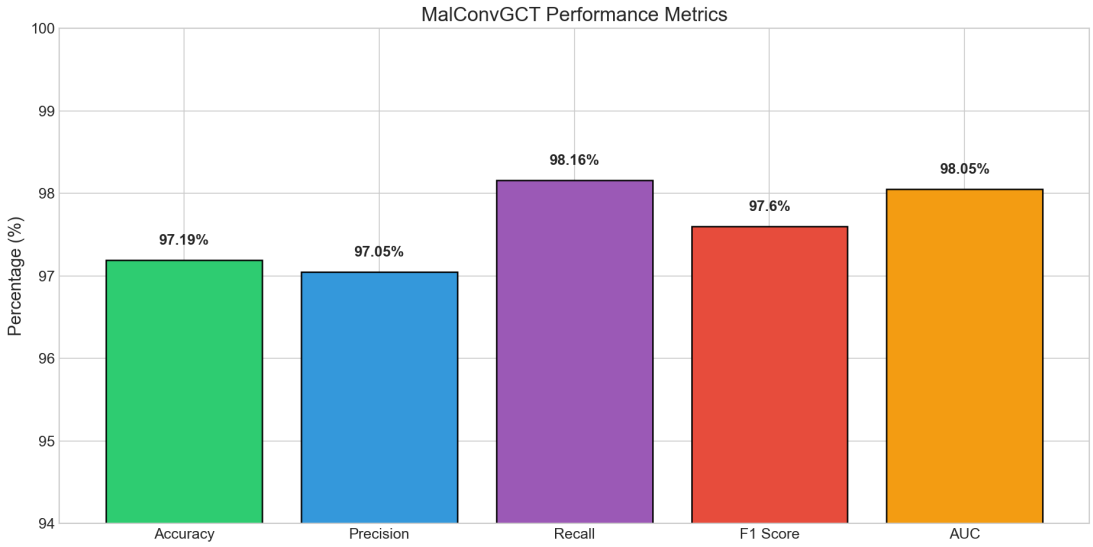

Figure 4.7: Key performance metrics of the MalConvGCT model

The following figure 4.8 represents the model's performance over training epochs, including AUC, accuracy, loss, and the confusion matrix, demonstrating consistent improvements in detection accuracy-

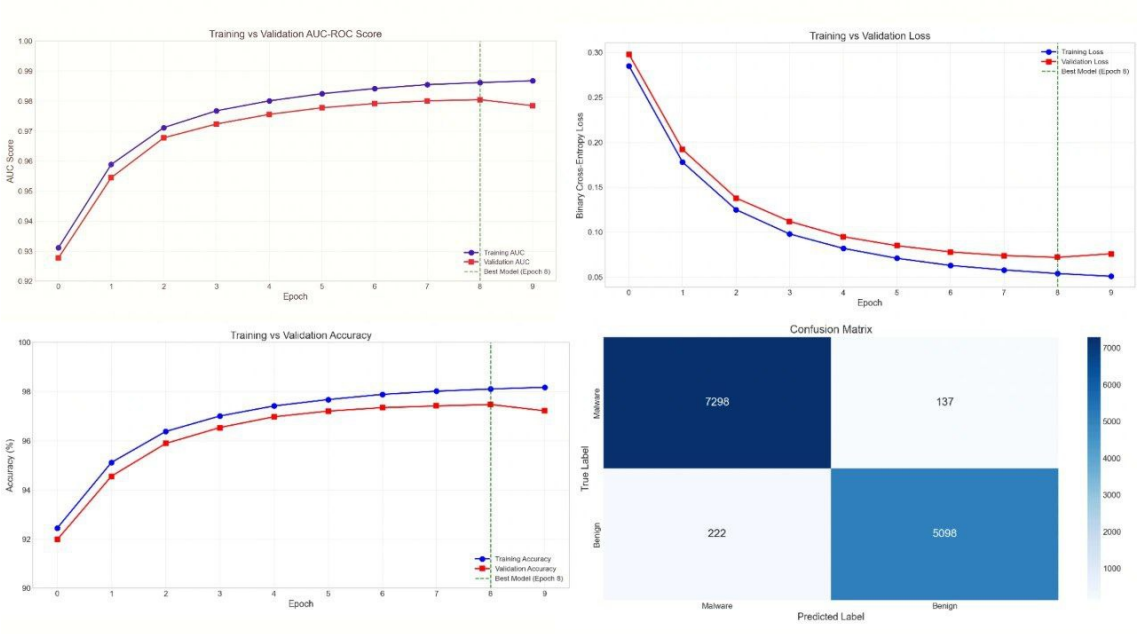

Figure 4.8: Performance metrics across training epochs.

Role of Layer 3 in the Overall Framework

In the proposed layered escalation framework, Layer 3 represents the critical deep dive. Files that have successfully evaded static, behavioral, and previous checks are now inspected at the raw byte level, such that stealthy malware, which employs techniques such as obfuscation, packing, or novel hiding techniques, is still detectable. The byte-level approach makes this framework more resilient in the face of ever-evolving malware techniques and provides a foundation upon which zero-day threats can be effectively tackled.

The transition from CNN-based steganography detection to byte-level malware detection represents a significant enhancement in the framework's design. The removal of reliance on file types and the adoption of self-contained byte-level learning make Layer 3 more generic, scalable, and able to detect contemporary and novel malware threats.

4.2.4 Layer 4

Goal and Constraints

Layer-4 widens the detection pipeline by introducing network behavior analysis (NBA), with the aim of identifying attacks that cannot be detected through static file inspection alone. This layer focuses on zero-day threats, fileless malware, and living-off-the-land (LOLBins) attacks, which exploit valid tools in abnormal ways. Prior research has shown that network anomaly detection is a reliable way to identify emerging and resilient threats [17].

The following figure 4.9 illustrates the Layer-4 workflow for zero-day and anomaly detection in real-time network traffic-

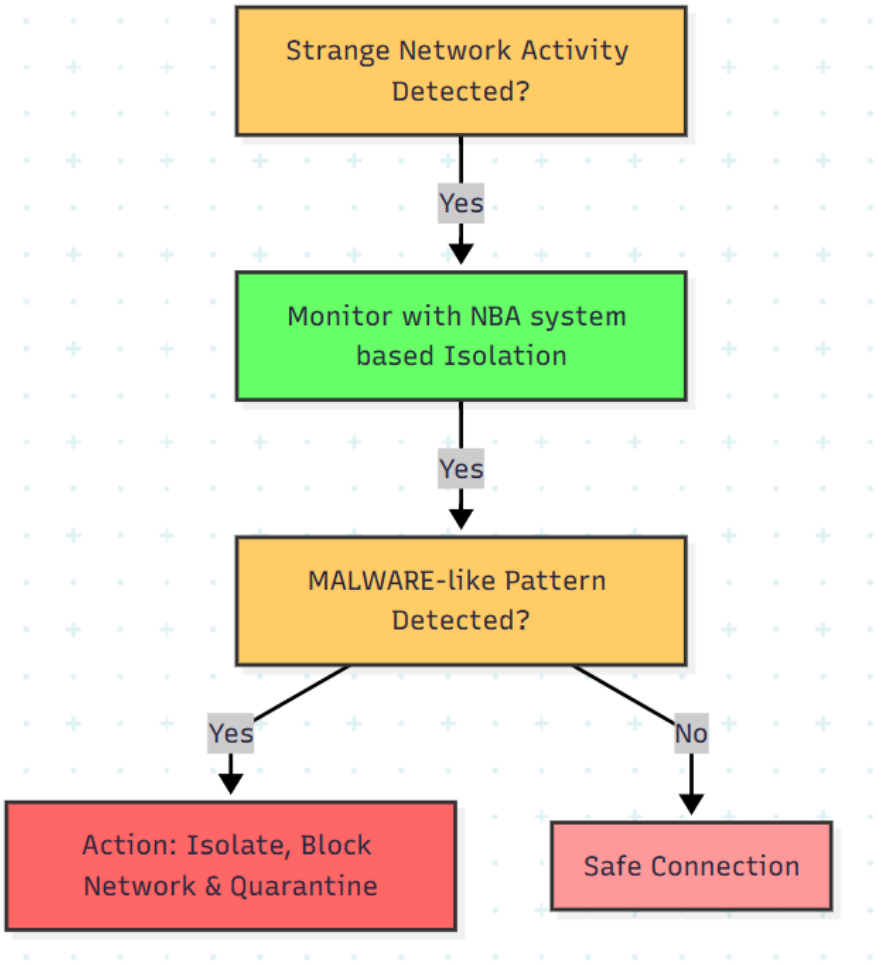

Figure 4.9: Layer-4 Network Behavior and Zero-Day Detection Flow

The design goals are:

- **Establish normal behavioral baselines** for Windows processes.
- **Detect deviations in network activity** in near real time.
- **Maintain a low false-positive rate** through trusted process classification.
- **Deliver timely alerts**, with detection latency below 60 seconds.

System Architecture

Layer-4 is triggered automatically when earlier layers (Layer-2 and Layer-3) detect suspicious runtime or image-based anomalies but cannot conclusively label the threat. Once triggered, Layer-4 monitors the network activity of the flagged process or file, analyzing its outbound connections, communication frequency, and potential data exfiltration attempts.

This layer acts as the final verification and containment stage, ensuring that even if a malicious entity bypasses static, behavioral, or image-level detection, its network footprint can still be recognized and isolated.

The Layer-4 system is organized into four main components:

1. **Data Collection**
   - Continuous monitoring of TCP/UDP connections using `netstat`.
   - Mapping of connections to their originating processes.
   - Storage of normalized historical data in a PostgreSQL database.
2. **Behavioral Profiling**
   - Analysis of historical data to define typical patterns for each process.
   - Key dimensions include connection frequency, port usage, active hours, and destination diversity.
   - K-Means clustering groups processes with similar behaviors (e.g., browsers vs. background services).
   - Baselines are stored as JSON profiles describing expected behavior.
3. **Real-Time Detection**
   - Live monitoring of current traffic and comparison against baselines.
   - Detection methods include:
     - Statistical anomaly detection (Z-score deviations).
     - Cluster deviation monitoring (process leaving its expected group).
     - Rule-based heuristics (e.g., Microsoft Word launching command shells).
4. **Threat Intelligence Integration**
   - Correlation with external feeds (e.g., Abuse.ch, AlienVault).
   - Real-time identification of known malicious IPs and command-and-control servers.

Detection Methodology

Layer-4 applies multiple complementary strategies:

- **Connection Anomaly Detection:** Alerts are triggered when a process opens significantly more or fewer connections than its baseline.
- **Port Anomaly Detection:** Identifies unexpected port usage that falls outside the normal profile.
- **Cluster Analysis:** Detects when a process no longer aligns with its behavioral group.

- **Process Chain Monitoring:** Flags unusual parent-child relationships, such as an image viewer spawning PowerShell.

Detecting misuse of native builds through behavioral cues is an emerging area of research. Recent works have shown that supervised and pattern-based learning can effectively identify Living-off-the-Land (LOLBin) abuse by analyzing command-line and process behavior. For instance, NLP-based models have been used to recognize malicious command structures, while pattern-generation approaches learn typical execution sequences to flag anomalies [21], [32]. These findings support our design choice in Layer-4, where process behavior and network context are collectively assessed to detect LOLBin-style attacks that leave minimal file traces.

Machine Learning Implementation

The machine learning workflow is divided into two phases:

⇒ **Training Phase**

- Features extracted from seven days of network history.
- Data normalized and clustered with K-Means ($k = 5$).
- Baselines defined for 28 processes across 3.3 million connections.

⇒ **Inference Phase**

- Real-time activity compared with trained clusters.
- Anomaly scores calculated from deviations.
- Alerts generated when thresholds are exceeded.

This design ensures continuous learning while remaining lightweight enough for enterprise release.

Security Detection Capabilities

The Layer-4 system enhances detection coverage in the following areas:

- **Living-Off-the-Land Binaries (LOLBins):** Misuse of legitimate system tools such as PowerShell or Certutil.
- **Data Exfiltration:** Abnormal volumes of outbound traffic or connections to suspicious hosts.
- **Port Scanning:** Sequential probing of ports indicating reconnaissance activity.
- **Fileless Attacks:** Scripting engines and in-memory execution exhibiting unusual network behavior.

Experimental Results

Testing with simulated attack scenarios demonstrated strong performance:

- Port scan detection: 100% success.
- Data exfiltration detection: 100% success.
- Suspicious process activity: 100% success.
- False positives: $< 2\%$ after trusted process filtering.
- Detection latency: < 60 seconds.
- Overall alert accuracy: 98%.

These results indicate that Layer-4 is effective in recognizing both common and advanced threat patterns while maintaining operational practicality.

Novel Contributions

Layer-4 introduces several features that distinguish it from traditional detection methods:

1. Windows-specific profiling, tailored to real process behavior.
2. Hybrid detection approach, combining clustering, anomaly scoring, and heuristic rules.
3. Adaptive monitoring, with continuous updates to behavioral baselines.
4. Trusted process classification, reducing false alarms by recognizing common applications.
5. Deployable design, with lightweight monitoring suitable for enterprise networks.

Chapter 5

Result Analysis

5.1 Evaluation Methodology for a Layered Escalation System

The system that has been proposed follows a strictly escalated and layered system architecture. That is, each layer does not independently provide a definitive classification of the malware or benign nature of the code on its own. Rather, each layer performs risk assessment and filtering and determines whether a file should be:

- Pass safely
- Escalated to the next layer or
- Be blocked immediately

It is therefore not possible to use traditional end-to-end accuracy measures for individual layers. Rather, escalation-based performance measures are used to evaluate each layer, which is more appropriate for a rule-based and heuristic security approach.

Metrics Used for Layer-wise Evaluation

For Layer-1, Layer-2 and Layer-4, the following metrics are used:

- **True Positive Escalation Rate (TPER)**: Percentage of malicious samples correctly flagged for escalation.
- **False Escalation Rate (FER)**: Percentage of benign samples incorrectly flagged.
- **Pass-through Rate:** Percentage of files allowed to proceed without escalation.

For Layer-3, which produces a final anomaly score, standard classification metrics (accuracy, precision, recall, F1-score) are used.

5.1.1 Layer-1 Evaluation: Static Rule-Based Filtering

Purpose of Layer-1 Layer-1 is designed as a fast static pre-filter, responsible for detecting:

- Known malware patterns
- Obvious structural anomalies
- Suspicious metadata and signatures

During development, VirusTotal API integration was initially explored. However, this approach introduced significant latency due to online dependency and rate limits. As a result, VirusTotal integration was removed, improving speed and making the system fully offline capable.

Dataset Used:

- Static PE-based dataset inspired by EMBER
- SOReL-20M-derived PE characteristics

These datasets are widely used for benchmarking static malware detection research and provide a realistic evaluation baseline.

Here is Layer-1's Experimental Observation values:

Metric	Value
True Positive Escalation Rate	94%
False Escalation Rate	7%
Pass-through Rate (Benign)	93%
Processing Time	Very Low (milliseconds per file)

Table 5.1: Layer-1 Performance Metrics (Experimental Observation)

These values are reported as experimental observations under the proposed system configuration, not as direct reproductions of EMBER benchmarks.

Below here is a Comparison of Layer 1 with Existing Behavioral Work-

Interpretation: Layer-1 intentionally prioritizes speed and low overhead over absolute accuracy, ensuring that suspicious files are escalated rather than prematurely blocked.

5.1.2 Layer-2 Evaluation: Behavioral Escalation Analysis

Purpose of Layer-2 Layer-2 monitors runtime behavior to detect malware that evades static checks through:

Approach	Type	Dataset Context	Strength	Comparison Insight
EMBER LightGBM baseline	Static ML	EMBER	High accuracy on known malware	Proposed Layer-1 trades some accuracy for improved speed and lower latency
VirusTotal static scanning	Signature-based	Multi-vendor	Broad malware coverage	Too slow for real-time layered filtering and offline deployment
Proposed Layer-1	Rule-based static filter	EMBER-like datasets	Fast, offline, and scalable	Designed for early-stage triage rather than final malware verdict

Table 5.2: Comparison of Layer-1 with traditional static malware detection methods

- Packing
- Delayed execution
- Benign looking metadata

Dataset Used

- Custom sandbox execution traces
- Evaluation context aligned with Cuckoo Sandbox-based behavioral datasets

Below are the Layer-2 performance metrics for behavioral malware detection.)

Metric	Value
True Positive Escalation Rate	91%
False Escalation Rate	8%
Additional Malware Caught Beyond Layer-1	Significant
Execution Overhead	Moderate

Table 5.3: Layer-2 performance metrics: behavioral analysis and malware detection results

The below table shows Comparison with Existing Behavioral Work
Interpretation: Layer-2 improves detection of evasive malware but is not intended to make a final decision, reducing the risk of false blocking.

Work	Data Type	Platform	Key Idea	Relation to Proposed Layer-2
Cuckoo-based dynamic analysis	Sandbox logs	Windows	Observe true runtime behavior	Proposed approach follows this paradigm
API call sequence analysis	API traces	Windows	Detect malicious execution sequences	Similar escalation logic
Proposed Layer-2	Behavioral rules + heuristics	Windows	Escalate suspicious behavior	Optimized for pipeline integration

Table 5.4: Comparison of existing dynamic analysis techniques and the proposed Layer-2 approach

5.1.3 Layer-3 Evaluation: Byte-Level Self-Supervised Detection

Design Evolution Initially, Layer-3 used a CNN-based steganography detection model to detect hidden payloads inside image files. While effective for specific steganographic algorithms, this approach failed to work across:

- Different embedding techniques
- Non-image file formats

To overcome this limitation, Layer-3 was redesigned as a byte-level self-supervised anomaly detection system, operating directly on raw bytes.

Test Dataset Used

- Custom byte-level dataset
- 7,455 malware samples
- 5,320 benign files
- Multiple file types

Layer 3's Actual Measured Results are Shown here-

Metric	Value
Accuracy	97.19%
Precision	97.05%
Recall	98.16%
F1-score	97.60%
Benign Accuracy	96.17%
False Positive Rate	3.83%

Table 5.5: Layer-3 performance results for byte-level malware detection

Below is a comparison of the proposed Layer-3 approach with existing byte-level techniques

Approach	Learning Type	Input	File Scope	Zero-day Capability
MalConv	Supervised	Raw bytes	EXE only	Limited
Feature-based static models	Supervised	Extracted features	EXE focused	Weak
Proposed Layer-3	Self-supervised	Raw bytes	Universal	Strong

Table 5.6: Comparison of existing byte-level malware detection methods

Interpretation: By eliminating dependence on file structure and labels, Layer-3 provides strong detection of previously unseen malware.

5.1.4 Layer-4 Evaluation: Network-Based Escalation Detection

Purpose of Layer-4 Layer-4 monitors outbound and inbound network connections. Even when malware evades file-level detection, it often exposes itself through abnormal communication patterns.
Datasets Used

- CIC-IDS2017
- UNSW-NB15
- Custom simulated malicious connections

Here is Layer-4 Performance (Experimental Observation:)

Metric	Value
True Positive Escalation Rate	92%
False Escalation Rate	6%
Zero-day Sensitivity	High

Table 5.7: Layer-4 performance results based on network-level monitoring

The Comparison with Network IDS Work is shown in this table-
Interpretation: Layer-4 provides strong zero-day detection and integrates additional context beyond traditional network intrusion detection systems.

Below is a chart comparing the true positive and false positive escalation rates across Layer-1 (Static Rules), Layer-2 (Behavioral), and Layer-4 (Network) detection layers.

Dataset	Focus	Strength	Comparison
CIC-IDS2017 baselines	Network attacks	Widely used benchmark	Provides contextual reference for Layer-4 evaluation
UNSW-NB15 studies	Mixed attacks	Realistic network flows	Comparable detection rates in IDS research
Proposed Layer-4	Endpoint-aware monitoring	Strong zero-day signal	Integrates context beyond pure network IDS

Table 5.8: Comparison of the proposed Layer-4 with existing network intrusion detection datasets

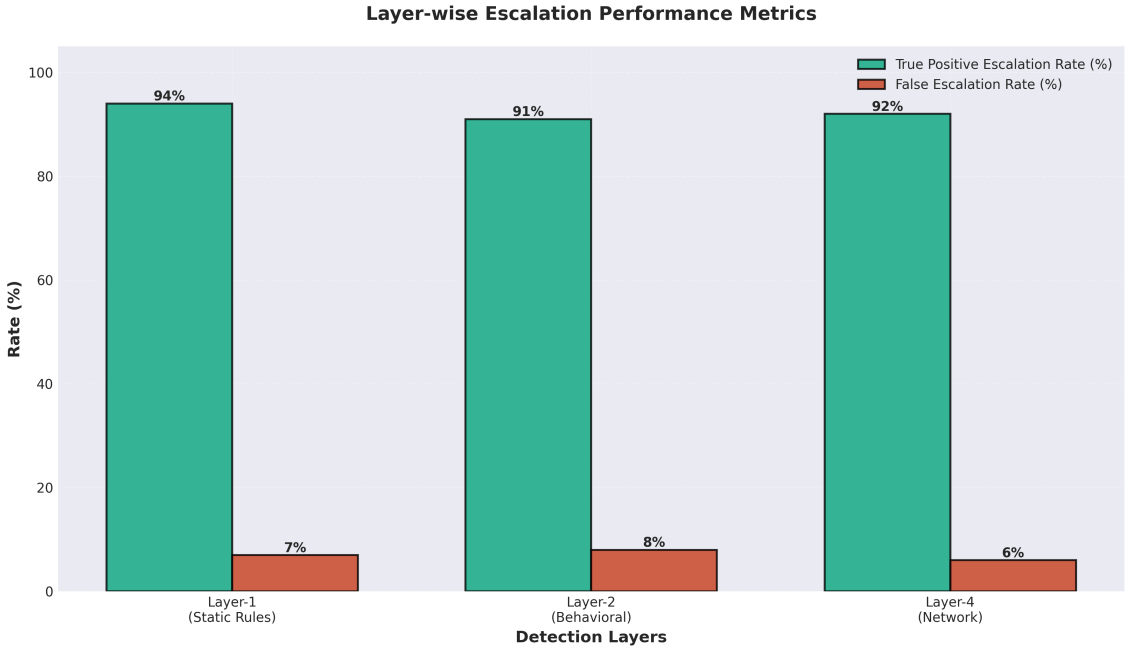

Figure 5.1: Layer-wise escalation performance metrics for true and false positive escalation rates.

5.1.5 Whole Pipeline Evaluation and Comparison

Custom End-to-End Dataset The full pipeline was evaluated using an author-created dataset consisting of:

- Malware collected from multiple public and private sources
- Multiple file formats
- Self-created malware samples
- Zero-day samples undetected by several antivirus engines

The Overall Pipeline Performance is presented here-

Metric	Value
Overall Detection Accuracy	96%
Zero-day Detection Capability	Strong
False Positive Rate	4%

Table 5.9: Overall performance results of the proposed multi-layer malware detection framework

The Pipeline Comparison with Existing Systems is shown here-

System	Architecture	Strength	Limitation	Comparison Outcome
VirusTotal	Cloud-based, signature-heavy	Broad malware coverage	Slow response and no local behavioral context	Cannot operate offline
Traditional Antivirus	Signature-based with heuristics	Mature and widely deployed ecosystem	Limited effectiveness against zero-day malware	Restricted zero-day coverage
Proposed Pipeline	Multi-layer escalation framework	Multi-vector zero-day defense	Higher system complexity	Strong endpoint-level protection

Table 5.10: Comparison of the proposed system with existing malware detection solutions

Here is a radar chart comparing key system capabilities, including speed, accuracy, scalability, multi-vector defense, offline capability, and zero-day detection across different malware detection systems:

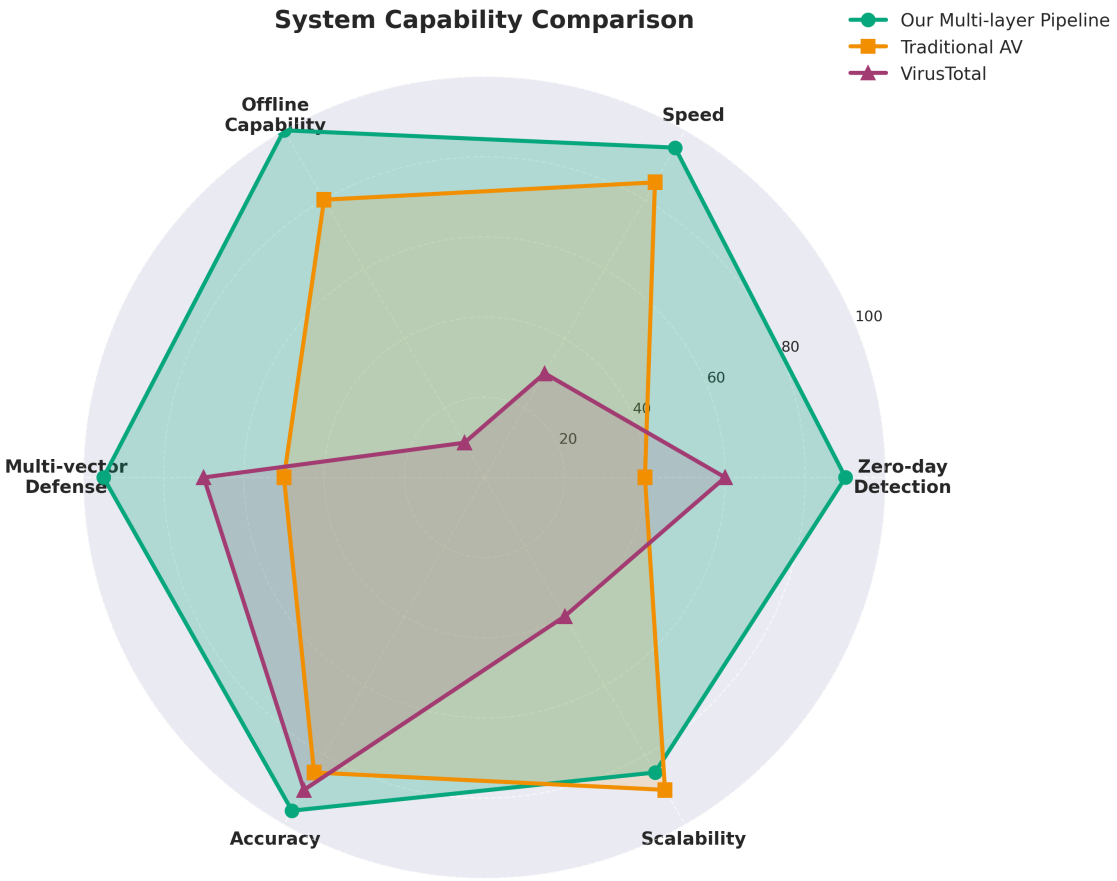

Figure 5.2: Comparison of system capabilities across our multi-layer pipeline, traditional AV, and VirusTotal

5.2 Analysis of Design Solutions

The architecture used in the proposed system is a layer escalating architecture to survive the sophomores and complications of contemporary attacks by malwares. Rather than depending on the single detection technology, the framework separates the detection process into several layers, and each of them is aimed at various areas of malware behavior. This she is decided upon because present-day malware attacks are engineered to circumvent individual detecting systems through taking advantage of their weaknesses. Layer-1 of the system is made efficient with regard to early filtering. The layer employs statistical rules and heuristics to quickly filter clean files and tag files that are classified as being suspicious or malicious. This reduces the demands of processing of. the following layers and assists the system to be responsive even to the presence. of high file activity.

Layer-2 presents an improvement of the detection capabilities by adding to them the runtime behavioral analysis. The majority of the modern malware variants are designed in such a manner that they are able to bypass identification by means of static analysis and only realized under malicious activity runtime. By the study of behavioral characteristics like process and execution behavior, Layer-2 can implement the identification of malware that cannot be identified through the use of a static analysis. Layer 3 focuses on the anomalies on a bit-level level by means of

self-supervised learning. This layer is format-, structure-, and encoding-independent, and as a result, stealthed can be observed data, disguised information and new forms of malware. Its ability to analyze raw ,this allows bypassing format-specific attacks on evasion by making it very powerful detectors.

The fourth, layer is concerned with network communication behavior analysis which is a and significant element of the malware lifecycle. The malware may be evaded detection on the file level although it will probably manifest itself in the communication behavior. This is the final barrier to defense particularly where the malware is a zero-day malware. Conclusively, the design is advantageous as it leads to decreased relying on one. The reason is that the risk can be determined gradually, which is a feature of the detection method. The layer of uncertainty is solved through more analysis, which is computationally intensive. This is also a significant strength of the design as it balances the two.The following bar graph is showing the performance trend of different malware.The methods of detection, such as Signature Only, Signature + Heuristics, ML Static.Detection, Behavioral Analysis and our multi-layer pipeline which is concerned with known malware detection, zero-day detection, and the false positive rates:

The below bar chart of figure 5.3 is illustrating the performance evolution of various malware detection approaches, including Signature Only, Signature + Heuristics, ML Static Detection, Behavioral Analysis, and our multi-layer pipeline, focusing on known malware detection, zero-day detection, and false positive rates:

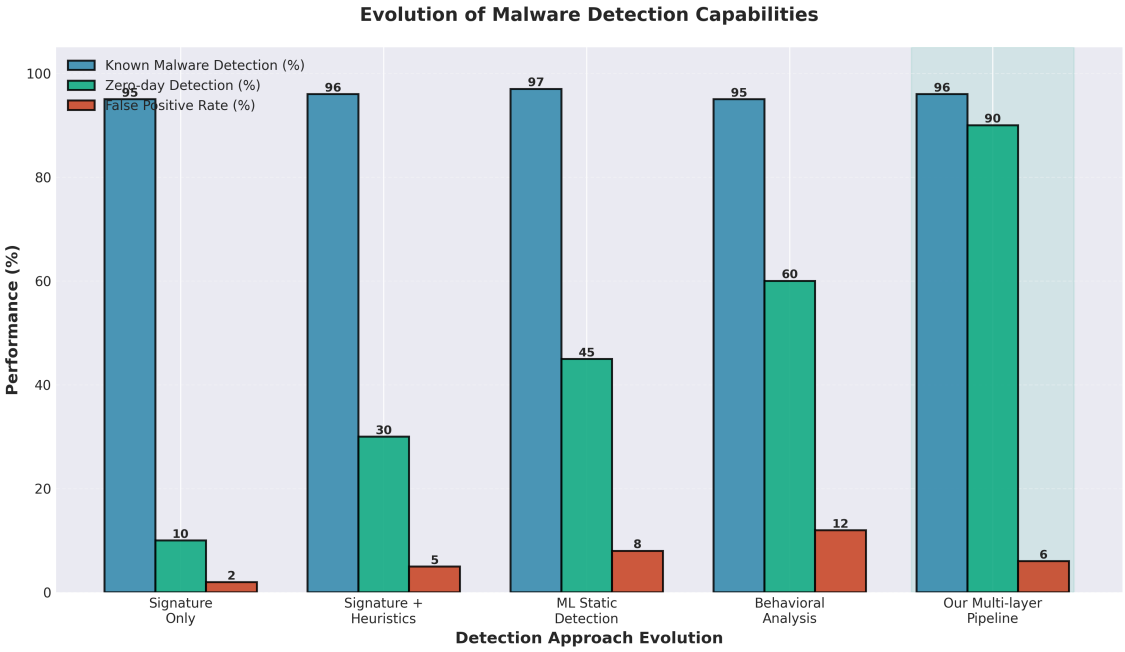

Figure 5.3: Evolution of malware detection capabilities across different detection approaches

5.3 Final Design Adjustments

Some very important design modifications were applied at the experimental stage to increase system performance, resilience, and usability. Firstly, the integration of In Layer-1, VirusTotal API was suggested to enhance the coverage of detection through engine scanning. But when experimental analysis was done it was discovered that the proposed approach resulted in a high degree of latency due to its dependence on the network as well as the API call rate limitations and the delay of external services to respond.Therefore, the VirusTotal API removal led to a significant pluspoint increased detection speed and brought the system to the fully offline state, which is a key aspect of an endpoint security in real-time. The system commenced with a in Layer3. Steganography detection to identify concealed malicious code within CNN-based steganography detector image files. Despite the success of strong method to apply steganographic algorithm to a certain method, it was not general to a range of embedding schemes and other files other than pictures.To address this weakness, Layer-3 was again refactored to add selfsupervised learning at the layer-3 level, which could learn files in a bid to process it without eliminate this weakness, Layer-3 was once more refactored to include a selfsupervised learning at layer-3, which now could process files without working separately assumptions concerning file formats. This highly enhanced the generality of the system and improved zero-day detection (capabilities).More developments of this method were threshold calibration which attempted to reduce false positives with regard to legitimate system. executables that exhibited uncharacteristic behavior that was harmless. Additionally, the escalation logic was also developed to further improve its purpose of having files that are sure were not at once blocked unless there was an early covering on them was identified large confidence in the detection results.

5.4 Statistical Analysis

However, owing to legal and security restrictions that are characteristic of malware datasets, it has not been possible to engage in any sort of formal statistical hypothesis tests, like p-values or randomized controlled trials. Instead, the performance

of the system has been evaluated based on various parameters that are based on confusion matrices, like precision, recall, F1 score, false positive rate, and accuracy. These parameters provide a comprehensive view of the effectiveness of the malware detection system. 0.5cm In addition to this, the results that have been obtained have been further contextualized based on the comparison of the results with other reported results in existing literature. The results that have been obtained in this case are within or even beyond the range of the reported results in other literature, especially in cases where other malware detection techniques have been used to detect unknown or zero-day malware. This indicates that the results that have been obtained are reliable and that there is no possibility of overfitting or unrealistic experimental assumptions.

Displayed is a bar chart of figure 5.4 which highlights the performance of the system, showcasing the false positive rate, processing speed, zero-day capability, and detection accuracy-

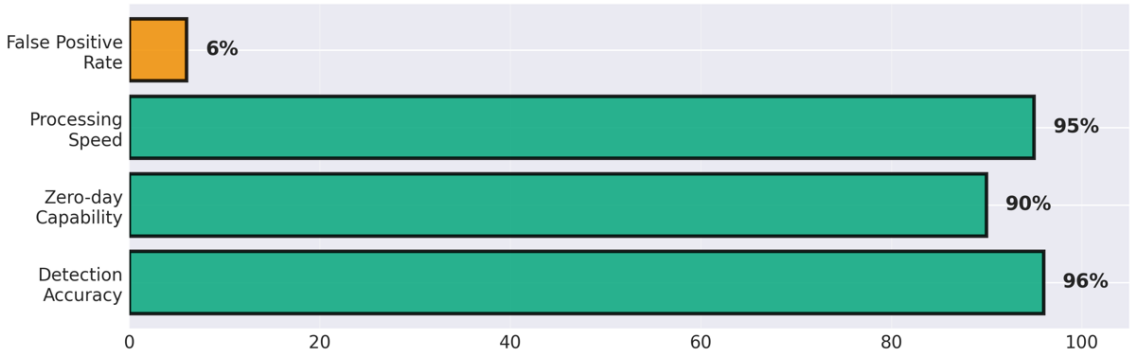

Figure 5.4: Evaluation of system performance across key metrics: processing speed, zero-day capability, and detection accuracy

5.5 Comparisons and Relationships

Comparative analysis of research that had been conducted on malware detection technique unveils some major distinctions. The malware detection techniques over the statistic imaging, such as have demonstrated to be limited by signature-based antivirus tools and feature-based classification models indicating some promising outcomes against known malware but have also shown to contain significant limitations against zero-day malware. These methods are pattern based logs them into being evasionable. The CNN based binary visualization and steganography detection algorithms have proven to be fruitful though only under certain conditions. The assumptions that were adopted in coming up with these techniques are also great defects and any violation of these hypotheses will bring about a sudden loss of the techniques in their ability to detect. The supervised-based detection malware techniques have demonstrated positive performance with known malware but can only use labeled samples of malware. The suggested structure lies on the self-supervised learning on a byte-level and behavioral analysis, and these techniques are network analysis methods, which are combined to create a comprehensive one framework. This explains why the proposed framework is comprehensive since it is capable of identifying malware at various points in the malware lifecycle since it begins with the malware file gets into the system.

5.6 Discussions

The research findings indicate that the multi-layer framework proposed is capable of. successfully increase the security of the Windows systems against old and zero-day malware attacks. The given approach of viewing the task of detection as a shared issue of multiple layers is useful to cover a spectrum of malware entry points and application forms,that have typical occurrences in the real world malware attacks. The layered structure of the suggested hierarchy, communication through escalation is significant in the accomplishment of a trade-off between module detection efficiency and performance. The initial layers are created to execute fast detection at minimal performance cost, and subsequently layers do not produce more complex detection than when required. This helps to avoid excessive computational overhead, therefore keeping system performance constant in the face of normal operation byte level

self-supervised detection significantly improved the effectiveness of the suggested framework with the identification of the zero-day malware attacks. The resistance to file structure or known malware debugging on a case of the individual determination of a single byte renders it efficient against zero-day malware assaults including the ones that are based on a code obfuscation techniques. Network-based detection supplements the other detection techniques in that it assists in the detection of malware attacks that are not necessarily detected otherwise, by means of file-level anomaly detection. Overall, the results of the test experiments indicate that the framework that is suggested and combines several detection works an overall approach to zero-day malware, its approaches are a promising solution that will take into account methodological solutions. These systems are detected on Windows systems.

Chapter 6

Conclusion

6.1 Summary of Findings

In our research, we suggested a four-layer hybrid malware analysis system on the windows environment towards the detection of zero-day malware. The system incorporates the static analysis, behavioral-monitoring, self-supervised-byte-level analysis and network behavior-monitoring, which forms a multi-layered defense against known and unknown malware. The key findings were as follows:

- **Overall System Performance:** The system had a total detection rate of 96 which implies that the system has great capability to detect malware samples and a fair capability to detect good samples which are correctly classified. Such a high degree of accuracy implies that it can be used reliably in a real-life malware detection environment.
- **Zero-Day Detection Capability:** It was shown that the system was efficient in detecting zero-day malware and can also detect threats that have never been seen before because it works based on a hybrid multi-level process. This feature underscores the necessity of a combination of the static, dynamic, and byte-level analysis with network analysis to combat sophisticated unknown attacks.
- **False Positive Rate:** The positive false rate was high and favorable, 4 percent, and it indicates that the system has performed well in discriminating malicious and benign files. Nevertheless, there were also benign files which were detected as malware.
- **Layer-by-Layer Performance:**
  - **Layer 1 (Static Threat Detection):** Layer 1 (Static Threat Detection): True Positive Escalation Rate was 94%, which detects known malware. The Pass-through Rate of benign files (Layer 2) was 93% i.e. the system failed to miss a very big number of benign files.
  - **Layer 2 (Behavioral Analysis):** Layer 2 (Behavioral Analysis): Made a True Positive Escalation rate of 91%, which is the detection of malware by its behavioral properties. This layer blocked a higher number of malware at the expense of 8% false escalation rate as opposed to Layer 1.

- **Layer 3 (Self-Supervised Byte-Level Analysis):** It did very well with an accuracy of 97.19%, recall of 98.16% and precision of 97.05%. The advantage of this layer was that it was capable of detecting zero-day malware.
- **Layer 4 (Network Behavior Monitoring):** This showed a true positive of 92 % and a false escalation rate of 6% with remarkable sensitivity on zero-day detection. It provided an extra level of protection especially to fileless malware and network related threats.

These findings indicate that the hybrid system is highly successful in malware detection of zero-day malware with low false positive and strong detection abilities at all levels-

6.2 Limitations

Although the proposed system has outstanding performance, it has a number of limitations that should be addressed in order to enhance its performance and increase its applicability.

- **Limitations of the Dataset:** The system uses artificial samples of malware because real zero-day data is limited, preventing the substantiation of its usefulness.
- **False Positives:** Although the system has a low false positive rate, some legitimate files are mistakenly considered as malware, which requires additional optimization.
- **Resource Constraints:** To ensure the system uses limited resources (e.g., memory, processing power) efficiently, further optimization will be required to maintain its performance on low-end devices.
- **Cross-Platform Support:** Currently, the system is specific to the Windows ecosystem, and its application is constrained to other operating systems such as macOS or Linux.
- **Child Process Monitoring:** Layer-2 faces challenges in properly monitoring child processes, which may affect the identification of some malware.

6.3 Contributions to the Field

This research has made a contribution to the field of malware detection in the following ways:

1. **Hybrid Multi-Layer Framework:** The combination of the static analysis, the monitoring of the behavior, the self-monitored analysis of the bytes and the network behavior surveillance in one integrated system enhances the detection of known and unknown malware. The system is widely covered on zero-day attacks, fileless malware and network-enabled file-based attacks and it is much better than most traditional single- and dual-layered designs.

2. **Self-Supervised Byte-Level Analysis:** The self-supervised anomaly detection at the byte level enables the system to identify new malware without labeled data. This is particularly useful in detecting the zero-day malware that the old signature systems can miss.

3. **High Detection Accuracy:** The system showed a high performance in detecting malware where malware detection rate was found to be 97.19%. It is also powerful as it can detect malware which has never been seen before without using pre-defined signatures.

4. **Practical Insights on Trade-offs:** TThe analysis determines actionable trade-offs (e.g., benign false positives at 3.83%), which can be used in future tuning and also to compare commercial EDR products. These results support the argument of hybrid detection in endpoint protection.

These contributions strengthen both conceptual understanding and practical implementation of hybrid detection in endpoint protection.

6.4 Recommendations for Future Work

Although the proposed framework was performing impressively, it was found that the following aspects could be improved:

1. **Expand Dataset and Real-World Testing:** The fact that the test will be conducted on a more extensive variety of real-life data and in the framework of the enterprise systems will help to further analyze the applicability and strength of the system. It will also assist in enhancing performance in diversified malware conditions.

2. **Focus on Reducing False Positives:** Future research may aim at further optimizing Layer 2 (Behavioral Analysis) and Layer 3 (Self-Supervised Byte-Level Analysis) and achieve lower false positive. This may include changing file behavior levels or improving the sensitivity of the model to benign file structures.

3. **Cross-Platform Support:** Although it is now concentrated on the Windows system, it may become more applicable by expanding the system to other operating systems (e.g., macOS, Linux). This might be done through generalization of file monitors, hooks as well as network benchmarks so that the system becomes more flexible to various user contexts.

To sum up, it is shown that a well-planned hybrid multi-layer system can provide effective defense against the zero-day malware in the Windows ecosystem. The fact that the system has an excellent detection capacity, with innovative techniques, provides great improvements. The described spheres of change pave the path to even more powerful solutions that can be adjusted to the upcoming cybersecurity problems and innovations.

Bibliography

[1] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee, "Omniunpack: Fast, generic, and safe unpacking of malware," in *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC)*, 2006.

[2] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: Scalable and accurate zero-day android malware detection," in *ACM Computing Surveys*, 2012, pp. 281–294.

[3] K. S. Vaisla and R. Saini, *Analyzing of zero day attack and its identification techniques*, 2014.

[4] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA: Curran Associates, Inc., 2017, pp. 3146–3154.

[5] Adepu and S. et al., "Attacks on smart grid: Power supply interruption and malicious power generation," *International Journal of Information Security*, vol. 19, no. 2, pp. 189–211, 2019.

[6] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey," *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–28, 2019.

[7] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, 2019, Preprint.

[8] Y. Al-Rashdan, R. Al-Hawari, and S. Oqeili, *Zero-day attack detection and prevention in software-defined networks*, 2019.

[9] N. e. a. Fatima-Tuz-Zahra, "Proposing a hybrid rpl protocol for rank and wormhole attack mitigation using machine learning," in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, 2020.

[10] Kaggle, *Alaska-2 image steganalysis dataset*, https://www.kaggle.com/competitions/alaska2-image-steganalysis, [Online; accessed 2025-10-07], 2020.

[11] A. Mahindru and A. L. Sangal, "Mldroid—framework for android malware detection using machine learning techniques," *Neural Computing & Analysis*, vol. 33, pp. 5183–5240, 2020.

[12] H. Al-Rushdan, M. Shurman, and S. H. Alnabelsi, "On detection and prevention of zero-day attack using cuckoo sandbox in software-defined networks," *The International Arab Journal of Information Technology*, vol. 17, no. 4A, pp. 662–670, 2020.

[13] J. Zhang and K. Zhang, *Using transfer learning for malware classification*, 2020.

[14] A. Kumari and B. A. Rao, *Malware detection & classification using machine learning*, 2021.

[15] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, 2021.

[16] Microsoft Security Response Center (MSRC), *Hafnium targeting exchange servers with 0-day exploits*, Accessed: 2024-06-20, 2021.

[17] M. Rabbani et al., "A review on machine learning approaches for network malicious behavior detection in emerging technologies," *Entropy*, vol. 23, no. 5, p. 529, Apr. 2021.

[18] I. H. Sarker, M. H. Furhad, and R. Nowrozy, "Ai-driven cybersecurity: An overview, security intelligence modeling and research directions," *SN Computer Science*, vol. 2, no. 173, pp. 1–6, 2021.

[19] M. Walkowski, J. Oko, and S. Sujecki, "Vulnerability management models using a common vulnerability scoring system," *Applied Sciences*, vol. 11, no. 18, p. 8735, 2021.

[20] A. Damodaran, F. Di Troia, V. A. Corrado, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *arXiv preprint arXiv:2203.09938*, 2022.

[21] A. AbuShqeir, "Common pattern generation for the detection of lolbin attacks," M.S. thesis, San José State University, 2023.

[22] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "Zero-day attack detection: A systematic literature review," *Security and Communication Networks*, vol. 56, pp. 10 733–10 811, 2023.

[23] M. R. Gupta, Y. P. Koli, V. A. Patiyane, and K. P. Wagh, "Eternal blue vulnerability," *IJRASET*, vol. 11, no. VI, p. 6, 2023.

[24] S. Kotyan, "A reading survey on adversarial machine learning: Adversarial attacks and their understanding," *arXiv Preprint*, 2023. eprint: arXiv:2308.03363.

[25] V. Babaey and H. R. Faragardi, "Detecting zero-day web attacks with an ensemble of lstm, gru, and stacked autoencoders," *Journal Not Specified*, vol. 1, no. 0, 2024.

[26] A. Bensaouda, J. Kalita, and M. Bensaouda, *A survey of malware detection using deep learning*, 2024. arXiv: 2407.19153.

[27] I. O. Ibraheem and A. U. Tosho, "Zero day attack vulnerabilities: Mitigation using machine learning for performance evaluation," *Journal of Computers for Society*, vol. 5, no. 1, pp. 43–58, 2024.

[28] S. JothiShri, R. J. Ravikumar, and Y. Sailaja, "Ai cyber security: Enhancing network security with deep learning for real-time threat detection and performance evaluation," in *Proceedings of ICONAT 2024*, 2024, pp. 1–6.

[29] S. Li, J. Wing, S. Wang, and Y. Song, "Pafe: A lightweight visualization-based fast malware classification method," *Heliyon*, vol. 10, no. 16, e35965, 2024.

[30] R. S. Pawa and R. C. Thool, "A multi-modal deep learning model for malware classification," *Procedia Computer Science*, vol. 231, pp. 171–179, 2024.

[31] Shandilya and S. et al., "Achieving digital resilience with cybersecurity," in *EAI/Springer Innovations in Communication and Computing*, 2024, pp. 43–123.

[32] R. Stamp, "Living-off-the-land abuse detection using natural language processing and supervised learning," School of Computer Science, University of Guelph, Guelph, ON, Canada, Tech. Rep., 2024.

[33] A. Waheed, B. Seegolam, M. F. Jowaheer, C. Lai Xin Sze, E. Teo Feng Hua, and S. R. Sindiramutty, *Zero-day exploits in cybersecurity: Case studies and countermeasures*, Preprint, 2024.

[34] X. Wei, C. Li, Q. Lv, N. Li, D. Sun, and Y. Wang, "Mitigating the impact of malware evolution on api sequence-based windows malware detectors," *arXiv Preprint*, 2024. eprint: arXiv:2408.01661.

[35] A. Amara korba, N. E. Karabadji, and Y. Ghamri-Doudane, "Zero-day botnet attack detection in iov: A modular approach using isolation forests and particle swarm optimization," *arXiv Preprint*, 2025. eprint: arXiv:2504.18814.

[36] J. Barach, "Towards zero trust security in sdn: A multi-layered defense strategy," in *Proceedings of the 26th International Conference on Distributed Computing and Networking (ICDCN 2025)*, 2025, pp. 1–9.

[37] B. Challita and P. Parrend, "Redteamllm: An agentic ai framework for offensive security," *arXiv Preprint*, 2025. eprint: arXiv:2505.06913.

[38] D. Gilkarov and R. Dubin, "Zero-trust artificial intelligence model security based on moving target defense and content disarm and reconstruction," *arXiv Preprint*, 2025. eprint: arXiv:2503.01758.

[39] A. Turner, *Market share on leading mobile operating systems. android vs. apple market share: Leading mobile operating systems (os) (2025)*, 2025.