Detection and Monitoring of Ransomware Attacks Using Machine Learning and Deep Learning

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfilment of the Requirements

For the Degree of

Masters of Science

in

Computer Science

University of Regina

By

Md Abdullah Al Ahasan

Regina, Saskatchewan

January, 2024

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

SUPERVISORY AND EXAMINING COMMITTEE

Md Abdullah Al Ahasan, candidate for the degree of **Master of Science, Computer Science,** has presented a thesis titled, ***Detection and monitoring of ransomware attacks using machine learning and Deep Learning,*** in an oral examination held on **January 15, 2024**. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner:	Dr. Andrei Volodin, Department of Mathematics and Statistics
Supervisor(s):	Dr. Nashid Shahriar, Department of Computer Science
Committee Member:	Dr. Samira Sadaoui, Department of Computer Science
Committee Member:	Dr. Habib Louafi, Adjunct, Department of Computer Science
Chair of Defense:	Dr. Mohammad Khondoker, Industrial Systems Engineering

ABSTRACT

This thesis presents a comprehensive investigation into the threat of ransomware and explores recent advancements in detection techniques. With the rise in the popularity of ransomware, a unique ecosystem of cybercriminals has emerged, leveraging encryption technology, anonymous cybersecurity, and easily accessible ransomware code. To address this growing concern, this thesis emphasizes the need for a machine learning (ML) and Deep Learning (DL) solution to detect ransomware attacks. Additionally, the study introduces the utilization of Software Defined Networking (SDN) combined with ML and DL for enhanced ransomware detection and mitigation.

In our pursuit of demonstrating ransomware detection capabilities, we introduce an architectural design aimed at providing a highly efficient solution for proactively countering ransomware attacks. Experimental results demonstrate the efficacy of the developed mechanism in promptly detecting and preventing the spread of ransomware. Moreover, considering the significant damage caused by ransomware attacks, the thesis explores the training and testing of various ML and DL models for ransomware detection. A novel and flexible ransomware detection model is proposed, achieving good accuracy and F1-scores on different domains of the dataset.

The proposed method is applicable to any domain of network traffic analysis data. In the context of the dynamic malware landscape, this thesis explores the detection of ransomware attacks by monitoring network traffic between infected computers and command and control servers. By extracting high-level flow features and utilizing a

random forest classifier, a flow-based detection method is developed to identify and classify ransomware without deep packet inspection. The proposed solution demonstrates a high detection rate and low false negative rate, proving its feasibility and accuracy.

The proposed approach significantly improves detection accuracy, making it effective for detecting both ransomware and specific types of malware. The method achieves feature reduction and quick convergence means that our method is attributed to its adept feature reduction capabilities, showcasing its efficiency and efficacy.

ACKNOWLEDGEMENT

I would like to address my sincere gratitude to my supervisor Dr. Nashid Shahriar who has encouraged me to follow this path, has provided me with the needed means and has shared with me his expert knowledge.

I extend my sincere appreciation for the assistance I received from the Department of Computer Science through teaching assistantships. Additionally, I wish to convey my thanks to the University of Regina and the Faculty of Graduate Studies and Research for their invaluable support and guidance throughout my graduate studies.

Gratitude is extended to my lab mates for their indispensable assistance throughout the research project. Special thanks are due to Dr. Habib Louafi for generously dedicating time to review my thesis and providing valuable feedback that significantly enhanced its quality.

Finally, heartfelt thanks go to my parents, my brother, my toddler and my cherished wife and her family for their unwavering support and presence throughout this remarkable journey. Their help has been instrumental in keeping me on track and successfully completing the research work. I would also like to dedicate my master's degree to my mother, Kohinoor Begum, and my father, Md. Bazlur Rashid Khan, and my entire family.

POST DEFENSE ACKNOWLEDGEMENT

I want to convey my appreciation to Dr. Andrei Volodin for serving as the external examiner for my thesis and for providing valuable feedback and suggestions. Additionally, my thanks extend to Dr. Samira Sadaoui and Dr. Habib Louafi for their contributions as part of the Supervisory Committee, and to Dr. Mohammad Khondoker for chairing the defense.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABBREVIATIONS

APTs Advanced Persistent Threats

ARI Attended Recent Inputs

CNN Convolutional Neural Networks

DL Deep Learning

DT Decision tree

KNN K-Nearest Neighbors

LSTM Long Short-Term Memory networks

ML Machine Learning

MLP Multi-Layer Perceptron

NB Naïve Bayes

OS Operating System

RF Random Forest

SDN Software Defined Network

sFlow RT sampled Flow of Real Time

SIEM Security Information and Event Management

UEBA User and Entity Behavior Analytics

VAE Variational Autoencoder

XGBoost eXtreme Gradient Boosting

Chapter 1

INTRODUCTION

The digital world faces a significant threat from malware like ransomware, which has become increasingly harmful and sophisticated over time. Traditional ransomware detection techniques, such as signature and heuristic-based methods, have proven inadequate in combating the latest generations of ransomware with their advanced obfuscation tactics. As a result, there is a growing adoption of *Deep Learning* (DL) in ransomware detection, as DL-based systems outperform conventional approaches by effectively identifying new ransomware family variants. DL techniques offer rapid ransomware prediction, excellent detection rates, and analysis of various ransomware types. Consequently, exploring the evolution of DL-based ransomware detection systems and their application to current trending is crucial [15].

Ransomware is a type of malware that encrypts a victim's data and demands a ransom for decryption. Detecting ransomware is challenging due to its polymorphic nature, encryption, and social engineering tactics. Existing approaches, like signature-based and anomaly detection, have limitations, including vulnerability to false positives and difficulties in keeping up with evolving ransomware variants. To combat ransomware effectively, advanced techniques, such as ML-based models and behavior analysis, are increasingly essential for threat detection. User awareness and

proactive cybersecurity measures are critical to mitigating the impact of ransomware attacks [32].

Ransomware has emerged as a significant threat in recent years, targeting both devices and cloud storage systems during network transfers, resulting in a dire need for effective malware detection techniques. Ransomware attacks typically follow a specific set of stages, starting with the distribution and infection of devices. This malicious software searches for files to encrypt, subsequently demanding ransom payments while threatening to expose sensitive information if demands are not met. Traditional signature-based detection methods prove ineffective against ransomware due to their continuous evolution and the presence of numerous signatures [18].

1.1 Problem Statement and Motivation

Most of the research work has predominantly centered on host-based detection, yielding results akin to static analysis. Detection of ransomware poses unique challenges compared to generic malware detection, primarily due to the constantly growing number of ransomware variants with different signatures. Consequently, new protection mechanisms must focus on identifying the dynamic behaviors of ransomware before it can encrypt files [18].

Without network traffic classification, it isn't easy to detect ransomware in the early stage. Because the evolution of the variant of the ransomware family changed rapidly. *Software Defined Network* (SDN) offers a centralized and programmable approach to network management, enabling more efficient monitoring and control of network traffic. *Machine Learning* (ML) and DL techniques provide the capability to analyze vast amounts of data and identify patterns that can distinguish ransomware behaviors from normal network activities. By leveraging these advanced technologies, there is an opportunity to develop proactive defence mechanisms that can detect and

mitigate ransomware attacks in real-time, minimizing their impact and preventing potential damage [10].

Ransomware is a threat for all types of devices and for all types of Operating systems (OS). So it might be very difficult to develop individual types of solutions for the different types of OS security. In this situation, this research focuses on the one solution for the network level that will solve every OS security problem.

In our proposed architecture, we address the fundamental challenge of ransomware detection. The network traffic collected, as referenced in [29] [25], is guaranteed to be within the specified collection time framework. From our research studies, we have constricted our research to answer the following questions:

1. **(Q1)** How to develop a model that detects ransomware without relying on data synthesis or explicit instructions in the host-based features?

2. **(Q2)** How to classify the different types of ransomware family attacks to identify them in different families?

3. **(Q3)** How can the developed model effectively detect ransomware attacks in real-world scenarios, with better performance?

1.2 Proposed Solution and Contributions

This research aimed to provide an in-depth analysis of recently developed DL-based ransomware detection techniques, with a specific focus on the detection and mitigation of ransomware attacks. The study examines detection techniques for different types of ransomware, including mobile ransomware (Android), Windows ransomware, IoT ransomware, and *Advanced Persistent Threats* (APTs). The goal is to enhance the effectiveness of early detection and monitoring by proposing the integration of SDN and *sampled Flow of Real Time* (sFlow RT) , ML, and DL approaches.

From our research studies, we have proposed solutions for each challenge

1. **(Q1)** How to develop a model that detects ransomware without relying on data synthesis or explicit instructions in the host-based features? One of the key contributions of this thesis lies in its ability to develop a ransomware family detection model that doesn't depend on real data or explicit instructions in the host-based features. By employing a cross-feature selection method in the network traffic features, discriminative features within the dataset are identified, enhancing the model's ability to differentiate between ransomware families. Additionally, the utilization of a *Variational Autoencoder* (VAE) for data reprocessing enables the model to learn ransomware family characteristics directly from processed data, contributing to a more data-driven and effective detection system.

2. **(Q2)** How to classify the different types of ransomware family attacks to identify them in different families? Another contribution of this thesis is centred on its capacity to effectively classify various types of ransomware family attacks. Implementing a multi-class classification methodology using ML and DL models that enables the model to learn and distinguish the unique features and behaviours associated with each ransomware family. This, in turn, facilitates accurate classification based on observed characteristics, offering a valuable tool for the identification and differentiation of ransomware attacks, thereby enhancing overall cybersecurity efforts.

3. **(Q3)** How can the developed model effectively detect ransomware attacks in real-world scenarios, with better performance? The developed model can effectively detect ransomware attacks in real-world scenarios with enhanced performance through the implementation of different Encoders and the VAE for feature extraction, enabling the model to capture essential ransomware charac-

teristics. Additionally, utilizing the optimized cross-features method of Pareto Ensemble learning enhances the model's ability to make accurate predictions and adapt to evolving ransomware behaviours, improving its real-world applicability.

This research began by proposing a realistic testing environment for ransomware attacks using SDN. SDN allowed for flexible network control and management, enabling the implementation of advanced security mechanisms. This utilization of SDN enhanced the detection and mitigation capabilities against ransomware attacks.

Next, ML and DL algorithms were employed to analyze a dataset obtained from reputable sources such as ISOT Ransomware, CICMAL2017, and malwaretraffic-analysis.net. The algorithms were trained to identify patterns and anomalies associated with ransomware attacks. Feature extraction and selection techniques were applied to optimize the models and improve the accuracy of detection.

The study further investigated the performance of different ML and DL algorithms in detecting and categorizing ransomware attacks. Comparative analysis is conducted to evaluate the effectiveness of various algorithms, including *eXtreme Gradient Boosting* (XGBoost), *K-Nearest Neighbors* (KNN), *Decision tree* (DT), *Naïve Bayes* (NB), *Random Forest* (RF), *Convolutional Neural Networks* (CNN), and *Long Short-Term Memory* (LSTM).

Additionally, the research focuses on developing mitigation strategies to counteract ransomware attacks. Leveraging insights gained from the dataset and trained models, proactive measures were proposed to prevent or minimize the impact of ransomware attacks. SDN can play a crucial role in dynamically reconfiguring network resources and isolating infected systems to contain the spread of ransomware.

The experimental results obtained from the analysis of the dataset and the evaluation of ML and DL algorithms demonstrated the effectiveness of the proposed approach. The developed models exhibited high accuracy in detecting and categorizing

ransomware attacks, providing valuable insights for timely response and mitigation.

In conclusion, this research suggests advancements to the realm of ransomware detection and mitigation through the sequential application of VAE, ensemble ML, and DL techniques. The integration of the dataset from CICMAL and malware traffic-analysis.net enhances the reliability and comprehensiveness of the proposed approach. The findings from this study can be utilized to develop robust security systems that proactively detect and mitigate ransomware attacks, ensuring the integrity and security of critical network infrastructures.

1.3 Thesis Structure

The remaining sections of the thesis are structured as follows:

Chapter 2, provides an overview of the relevant background knowledge. It discusses the fundamental concepts of Software Defined Network(SDN), ML, and DL techniques in the context of detecting and mitigating ransomware attacks. The chapter explores the security challenges associated with ransomware attacks and presents a comprehensive understanding of the different types of ransomware and their evolving nature. Additionally, it explains the functioning of LSTM networks and the calculation of various metrics used in the detection and classification of ransomware attacks.

In Chapter 3, outlines the methodology employed for the detection and mitigation of ransomware attacks. It focuses on the utilization of SDN, ML, and DL techniques, specifically LSTM networks, for effectively identifying and mitigating ransomware attacks. The chapter details the approach and techniques used to train and deploy the LSTM, CNN models, including the selection and extraction of relevant features, the training process, and the evaluation of the models' performance using the Pareto ensemble.

Chapter 4, the experiment setup for the detection of ransomware attacks using VAE, ML, and DL techniques is described. The chapter provides details about the laboratory environment for the experiment infrastructure, the selection of relevant tools and technologies, and the setup of virtual machines and docker-based environments. The deployment of necessary components, such as network controllers and security systems, experimental results and comparison with other results are also discussed.

Chapter 5, concludes the thesis by summarizing the key findings and contributions of the research on the detection and mitigation of ransomware attacks using SDN, ML, and DL techniques. The chapter discusses the implications of the study's results, highlights the limitations of the proposed approach, and suggests potential areas for future work. It emphasizes the significance of the newly proposed architecture that enhancements to further improve the effectiveness of ransomware detection and mitigation using SDN, sFlow Rt, ML and DL.

Chapter 2

BACKGROUND AND RELATED WORK

Ransomware attacks have become a significant threat to modern computer networks, causing substantial financial losses and disrupting critical services. Traditional security measures, such as firewalls and antivirus software, often fail to detect and mitigate advanced ransomware attacks. To address this challenge, researchers have explored the integration of SDN with ML and DL techniques for enhanced ransomware detection and mitigation. This thesis presents a comprehensive overview of the related work in this domain, focusing on the use of SDN, ML, and DL to detect and mitigate ransomware attacks

2.1 Ransomware and its Family

Ransomware is malicious software that encrypts data and demands a ransom payment. It is delivered through various methods, including phishing emails, exploit kits, and malicious advertisements. Once infected, the ransomware hides its identity by using a dropper file and masquerading as legitimate processes. It ensures persistence by making registry key additions and adding itself to system startup processes. Ran-

somware exploits human vulnerabilities and uses social engineering tactics to trick users. It can be executed when users click on malicious links or open infected attachments. Ransomware encrypts data to make it inaccessible to users until a ransom is paid for the decryption key. It aims to create fear in victims by threatening real-world consequences or fines. Ransomware operates stealthily, avoiding detection and remaining active even after the system reboots or starts in safe mode. [15]

Ransomware having worm capabilities is very dangerous. According to [6], Worms are self-replicating malware that spread across networks, while ransomware encrypts files or locks systems for ransom. Worms exploit vulnerabilities and autonomously infect systems, while ransomware relies on user interaction to propagate. Worms have a broader focus on infecting as many machines as possible, causing widespread damage and enabling other malicious activities. Ransomware's primary objective is extortion, demanding payment in exchange for decrypting files or unlocking systems. Worms can disrupt network operations and propagate other malware, while ransomware leads to data loss, financial losses, and operational disruptions. Understanding these differences is crucial for effective cybersecurity measures and mitigating risks associated with these malware types.

2.1.1 Different Types of Ransomware Family

We have studied different types of ransomware family and mainly we found 21 different types that have been collected from [25]. Among these families, we identified nine uniques types, which are as follows:

1. Cerber: A highly active ransomware known for its aggressive distribution and advanced evasion techniques. It encrypts files and demands a ransom for decryption.

2. CryptoMix: A ransomware family that emerged in 2016, known for using a com-

bination of RSA and AES encryption algorithms to encrypt files and demand ransom payments.

3. CryptoShield: A ransomware variant that encrypts files and appends a specific extension to the encrypted files. It demands payment in cryptocurrencies for decryption.

4. Crysis: A widespread ransomware family that targets both individuals and organizations. It encrypts files and may also attempt to spread across networked systems.

5. CTB-Locker: One of the early ransomware families, CTB-Locker uses RSA encryption and demands ransom payments in Bitcoin.

6. Locky: A notorious ransomware family that gained significant attention in 2016. It spreads primarily through malicious email attachments and encrypts a wide range of file types.

7. Jaff: Known for its use of the Necurs botnet, Jaff ransomware encrypts files and demands high ransom amounts for decryption.

8. WannaCry: A high-profile ransomware attack that occurred in 2017, exploiting a vulnerability in the Windows operating system. It spread rapidly across networks and caused widespread disruptions.

9. Petya: Another notable ransomware family that targeted organizations by encrypting the master boot record (MBR) or entire hard drives, rendering systems inaccessible.

We found another 10 different types of ransomware families in another dataset [29]. This ransomware generated is based on the Android *Operating System*(OS) .

1. Svpeng: A mobile ransomware family primarily targeting Android devices. It typically locks the device and demands a ransom payment to regain access.

2. Porndroid: A mobile ransomware variant that spreads through malicious apps or websites containing adult content. It locks the device and demands a ransom to unlock it.

3. Koler: Another mobile ransomware family that spreads through malicious apps, particularly disguised as media players or video codecs. It locks the device and displays fake law enforcement messages to intimidate victims into paying a ransom.

4. RansomBO: A ransomware family that primarily targets Linux-based systems. It encrypts files and demands a ransom for decryption.

5. Charger: A mobile ransomware known for targeting Android devices. It encrypts files and displays a ransom message, often masquerading as a legitimate app.

6. Simplocker: A mobile ransomware variant that targets Android devices. It encrypts files on the device's external storage and demands a ransom for decryption.

7. Wannalocker: A ransomware family that gained attention for its similarity to WannaCry. It spreads through network vulnerabilities and encrypts files on infected systems.

8. Jisut: A ransomware family that targets Windows systems. It encrypts files and appends specific extensions while demanding a ransom for decryption.

9. Lockerpin: A mobile ransomware that targets Android devices, specifically focusing on locking the device's screen. It displays a ransom message and demands

payment to unlock the device.

10. Pletor: A ransomware family known for its file encryption capabilities. It encrypts files on infected systems and demands a ransom for decryption.

2.2 Increasing Threat of Ransomware attacks

Ransomware Attacks are very common issues these days [45]. Most industries and educational institutions have been also attacked by different types of ransomware attacks. According to [43], a survey of 350 IT leaders, the following findings were observed regarding ransomware attacks:

1. Lack of Public Discussion: Ransomware is a problem that affects everyone, but organizations are reluctant to discuss it openly.

2. Data Destruction Attempts: A staggering 93% of ransomware attacks aimed to destroy backup data, indicating the malicious intent to render organizations unable to recover their information.

3. Recovery Without Ransom: Only 14% of the surveyed organizations were able to successfully recover their data without paying the ransom. This highlights the challenges and complexity involved in mitigating the impact of ransomware attacks.

4. Ineffective Ransom Payments: Surprisingly, even among organizations that chose to pay the ransom, 19% reported that they still could not retrieve their encrypted data. This suggests that there is no guarantee that paying the ransom will result in data recovery.

The survey reveals the pervasive nature of ransomware attacks, the deliberate targeting of backup data, the low success rate in recovering data without paying the

ransom, and the potential failure of ransom payments to restore access to encrypted data. These findings underscore the critical importance of robust cybersecurity measures and proactive strategies to prevent, detect, and respond to ransomware attacks. This Figure 2.1 illustrates the sequential stages of a ransomware attack scenario, beginning with the user receiving an email, progressing through the attachment and payload delivery, encryption, ransom demand, payment, and concluding with the potential decryption phase, while also highlighting the interaction with a Command and Control (C&C) Server.

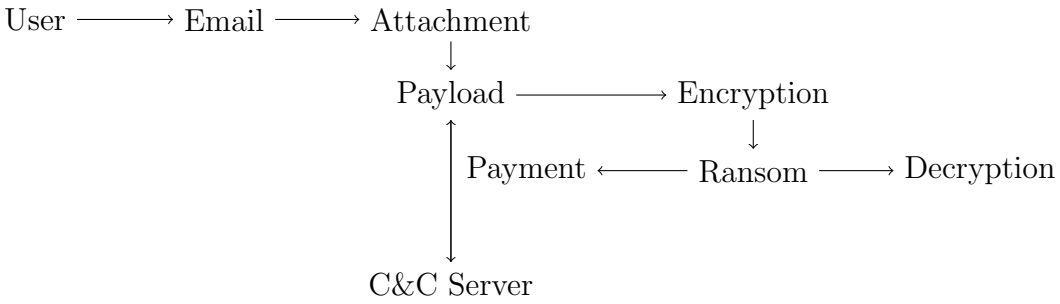

Figure 2.1: The Ransomware Attack Scenario

2.2.1 Different Types of Ransomware attack

There are different types of ransomware attacks [21]. Among them, the seven major types of ransomware attacks are described here:

1. File-Encrypting Ransomware: This type of attack encrypts files on the victim's system, making them inaccessible. The attacker demands a ransom payment in exchange for the decryption key to restore access to the files. Examples include WannaCry and CryptoLocker.

2. Locker Ransomware: Locker ransomware attacks target the system's functionality rather than specific files. It locks the victim out of their device or restricts access to certain features, such as the screen or keyboard. The attacker demands

a ransom to unlock the device. Examples include Police-themed ransomware and Android screen lockers.

3. Master Boot Record (MBR) Ransomware: MBR ransomware infects the master boot record of a computer's hard drive, preventing the system from booting up. The attacker demands a ransom payment to restore the MBR and regain access to the system. Petya and Satana are examples of MBR ransomware.

4. Mobile Ransomware: This type of ransomware specifically targets mobile devices, such as smartphones and tablets. Mobile ransomware can lock the device, encrypt files, or display intimidating messages, demanding a ransom for restoration. Examples include Svpeng and Porndroid.

5. Ransomware-as-a-Service (RaaS): Ransomware-as-a-Service refers to a model where cybercriminals create and distribute ransomware to other individuals or groups who then carry out the attacks. The original creator receives a portion of the ransom payments as profit.

6. Scareware: Scareware involves displaying fake warnings or alerts on a victim's system, tricking them into believing their computer is infected with malware. The attacker then demands payment for a fake antivirus or security software to resolve the issue. Scareware does not actually encrypt or lock files but relies on social engineering tactics.

7. Doxware/Leakware: This type of ransomware not only encrypts files but also threatens to publish or leak sensitive information unless a ransom is paid. Doxware targets individuals or organizations that have sensitive data they want to protect, such as confidential documents or personal information.

This figure 2.3 illustrates the system architecture of diverse ransomware attack vectors, originating from sources such as phishing websites, phishing emails, and exploit

kits, leading to the deployment of malicious payloads including crypto-ransomware, locker-ransomware, and scareware, ultimately culminating in a common endpoint represented by a payment gateway.

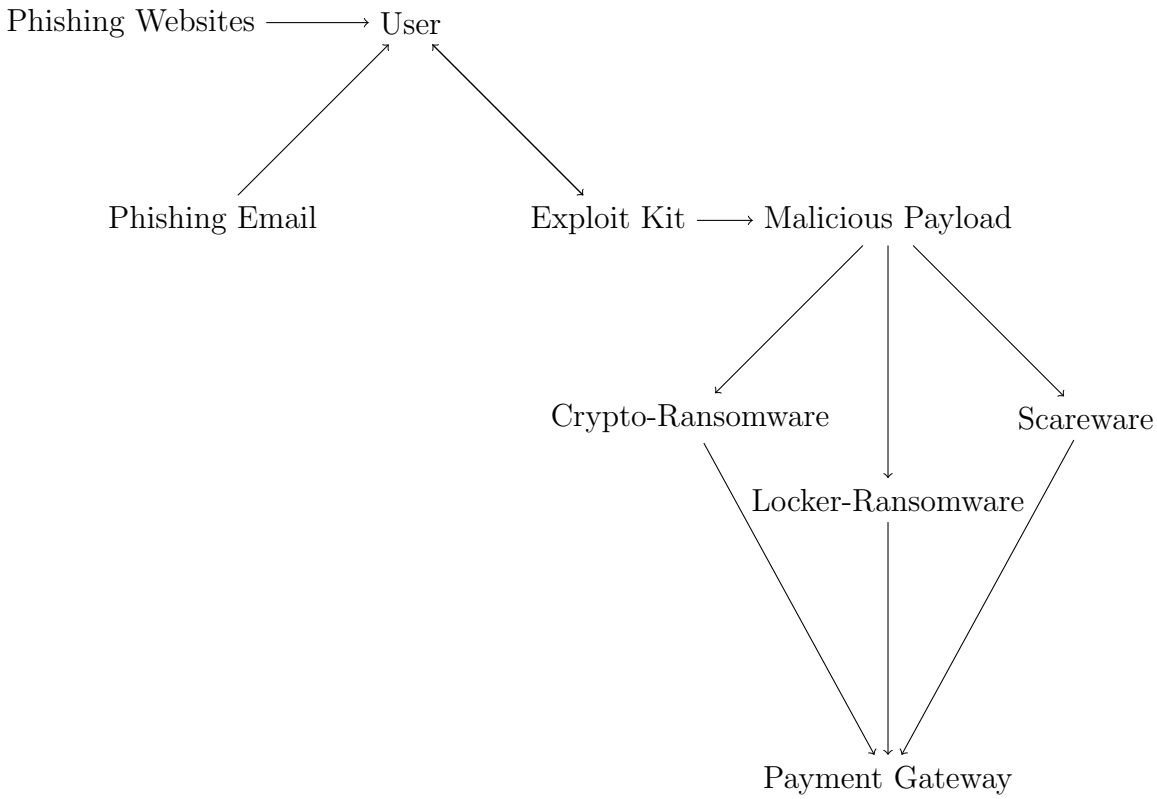

Figure 2.2: System Architecture of Different types of Ransomware Attacks

2.2.2 Need for Effective Detection and Mitigation Strategies

The need for effective detection and mitigation strategies for ransomware attacks is crucial due to their increasing prevalence and severity [26]. Traditional security measures have limitations in combating these evolving threats, leading to financial losses and operational disruptions. The integration of SDN with ML and DL algorithms show promise in enhancing network security. SDN provides dynamic and adaptive security controls, while ML and DL offer the potential to analyze network data and detect ransomware patterns. By combining these technologies, proactive ransomware detection systems can be developed. This research aims to explore this integrated

approach and contribute to the advancement of ransomware defense mechanisms.

2.2.3 Role of SDN

SDN plays a crucial role in the detection and mitigation of ransomware attacks. SDN offers centralized control, programmability, and network virtualization, which enhance network security [41]. SDN enables dynamic and adaptive security controls, allowing administrators to quickly update and customize security measures. It provides real-time visibility into network traffic, facilitating the detection of ransomware activities. SDN's programmability enables the integration of ML and DL algorithms for ransomware detection. It automates security actions, such as reconfiguring network resources and isolating affected devices or segments, to mitigate the spread of ransomware attacks [47]. SDN's ability to collect and analyze network data helps identify ransomware patterns and anomalies. It allows for rapid response and mitigation strategies to minimize the impact of ransomware attacks. SDN's role in the thesis is to provide a foundation for effective ransomware detection and mitigation using ML and DL techniques. By leveraging SDN's capabilities, the thesis aims to enhance network security against ransomware threats. This figure 2.3 illustrates the role of SDN in the detection and mitigation of ransomware attacks. The SDN controller integrates ML and DL components to facilitate ransomware detection, followed by monitoring and mitigation actions to enhance network security

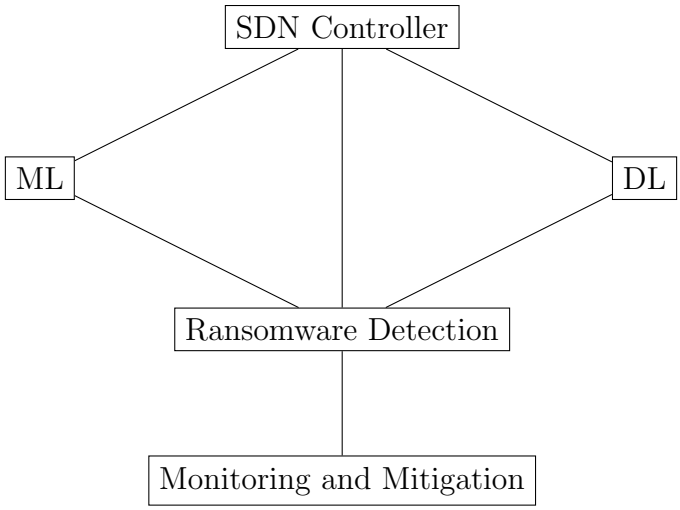

Figure 2.3: SDN role of Ransomware Attacks Detection and Mitigation

2.2.4 Role of ML and DL

ML and DL have a crucial role in the detection and mitigation of ransomware attacks in SDN environments [42]. ML and DL algorithms analyze network traffic and system logs to identify suspicious activities related to ransomware. They can learn from historical attack data and detect new variants by identifying patterns and anomalies in network behavior. ML techniques model normal network activity, enabling the detection of deviations that may indicate ransomware presence. DL algorithms excel in detecting complex patterns and variations, allowing them to identify previously unseen attack patterns. SDN systems can respond in real-time to ransomware attacks by isolating infected hosts, blocking malicious traffic, or alerting administrators. The continuous learning and adaptation of ML and DL models enable SDN systems to stay updated with evolving ransomware threats. ML and DL algorithms aid administrators in making informed decisions regarding attack detection and mitigation strategies. By leveraging ML and DL in SDN, organizations can significantly enhance their ability to effectively detect and mitigate ransomware attacks, thereby ensuring network security and data protection.

2.2.5 Significance of the Study

The significance of the study on the detection and mitigation of ransomware attacks using SDN, ML, and DL is undeniable. Ransomware attacks pose a significant threat to individuals and organizations, making effective detection and mitigation strategies crucial. Integrating SDN with ML and DL algorithms enables real-time analysis of network traffic for accurate ransomware detection. ML and DL algorithms can adapt to new ransomware variants, improving response time and reducing the impact of attacks. The study's findings will contribute to enhanced network security practices, inform policy decisions, and guide the development of future security frameworks.

2.3 Related Work

Cybersecurity threats have become increasingly prevalent in recent years, with ransomware attacks posing a significant threat to businesses and individuals alike. These attacks involve malicious software that takes control of a victim's computer or network, encrypting their files and demanding payment in exchange for decryption keys [3]. As ransomware creators constantly upgrade their products to avoid detection, establishing a solid overcoming methodology for dealing with these attacks has become increasingly difficult [51]. In the past, according to [24], traditional methods for detecting and defending against ransomware attacks have largely relied on signature-based techniques such as SIDS, as well as rule-based static policies including traditional policy-based detection in software-defined networks. However, with the evolution of ransomware capabilities such as encrypted communications and new families that haven't been seen before, these methods are no longer effective for preventing attacks. As a result of the limitations of traditional methods, researchers have proposed utilizing ML and DL techniques for detecting and mitigating ransomware attacks [34]. These approaches are based on the fact that ML/DL require sufficient

ransomware data for training and testing their algorithms [42]. Additionally, [40] conducted a comprehensive review that analyzed existing literature to identify the most effective machine-learning algorithms for detecting ransomware attacks. Their review also highlighted the significance of data quality in ML/DL-based ransomware detection. Furthermore, neural networks are often used due to their ability to analyze the behaviour of ransomware in-depth. Sgandurra et al. proposed an approach that detects and classifies ransomware variants by dynamically analyzing the behaviour of applications during their early stages of infection [49].

2.3.1 Recent Ransomware Attacks and How Business and IT Leaders Addressed These Issues

The frequency of global ransomware incidents continues to surge, with Corvus reporting [23] an 11.22% quarter-over-quarter (QoQ) increase in Q3 and a substantial 95.41% year-over-year (YoY) rise on leak sites. The discernible impact of limited mass exploits on overall ransomware trends is anticipated to persist. In Q2, the ransomware group's exploitation of a zero-day vulnerability in file transfer software contributed to 13% of all ransomware victims in Q3. Q3 would still register a 5% QoQ increase and a noteworthy 70% YoY surge in ransomware incidents

Following the typical seasonal patterns of ransomware, an escalation in attack velocity is expected in Q4. Although ransomware incidents often decline during the summer months, this year's decrease was later and shorter than the usual trend. Law firms and municipalities are witnessing a heightened frequency of attacks, signifying a notable shift in the targeted sectors for ransomware activities.

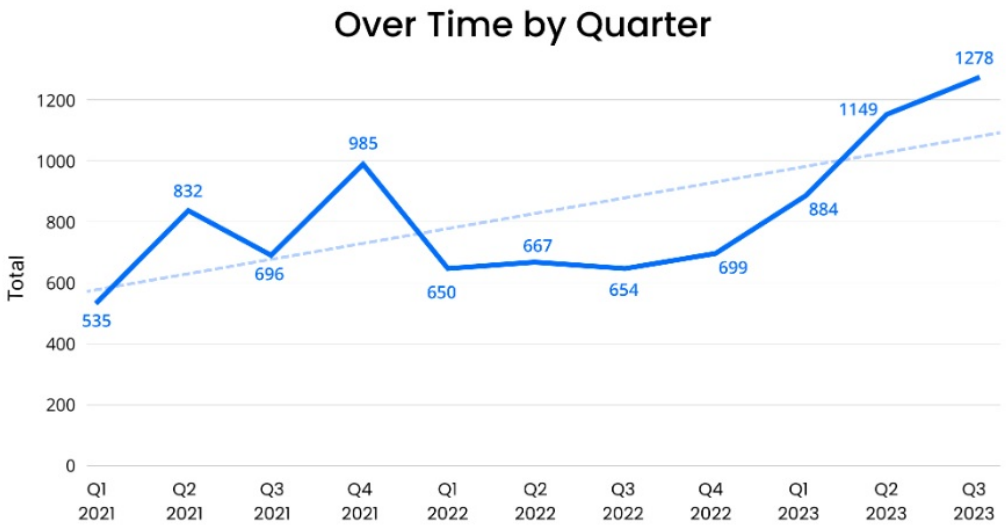

Figure 2.4: Ransomware Attack Over Time by Quarter [23]

2.3.2 Ransomware Detection Techniques

Ransomware Detection Techniques in SDN have also been explored in different experiments. SDN allows for central management of network security policies, making it an attractive option for ransomware detection and defence [24]. SDN was proposed as a promising approach for enhancing network security and mitigating ransomware attacks [50]. Below are some of the major types of ransomware detection techniques.

Signature-based Detection

Signature-based Detection approaches may not be effective in identifying new and unknown ransomware families, but SDN combined with DL/ML can allow for real-time detection and mitigation of such attacks [38]. Additionally, SDN can enable network traffic monitoring and analysis, providing visibility into potential threats.

- Signature matching: Identify ransomware by comparing file signatures with known ransomware signatures stored in a database.

Anomaly-based Detection

Another approach for ransomware detection is anomaly-based detection. This method identifies attacks based on the deviation from normal malicious behaviour, making it effective at detecting unknown ransomware variants [5].

Behavior-based Detection

Lastly, behaviour-based detection is a popular approach that involves monitoring system activities to detect expected behaviours indicative of ransomware attacks. This approach is based on the fact that ransomware can be identified by its behaviour and not just it's signature. Overall, the use of ML and DL for ransomware detection has shown promising results [30].

- Anomaly detection: Monitor system behavior for unusual activities, such as mass file encryption, and trigger alerts.
- Heuristic analysis: Detect ransomware by analyzing file modification patterns and access behaviors.

2.3.3 ML-Based Detection

ML techniques use algorithms and models to learn and detect ransomware based on various features.

- Feature Selection: Select features from files and system activities and use ML models for classification.
- Clustering: Group files based on characteristics and analyze patterns to detect anomalies.

2.3.4 Honeypots

Honeypots are decoy systems or files designed to attract ransomware. By monitoring these systems, we can detect ransomware when it interacts with the decoy.

2.3.5 User and Entity Behavior Analytics (UEBA)

User and Entity Behavior Analytics(UEBA) monitors user and entity behaviours to detect abnormal activities, which can include ransomware actions.

2.3.6 Network Traffic Analysis

Analyze network traffic patterns for signs of ransomware communication or file encryption. Unusual spikes in network activity can be indicative of a ransomware attack.

2.3.7 Backup and Recovery Systems

Regular backups of critical data can help in ransomware detection by comparing encrypted files with their unencrypted counterparts.

2.3.8 Security Information and Event Management (SIEM) Systems

Security Information and Event Management (SIEM) systems aggregate and analyze logs and events from various sources, enabling the detection of ransomware activities through pattern recognition and alerting.

2.3.9 Security Awareness Training

Educating users about ransomware risks and safe online practices can help prevent ransomware infections by reducing the likelihood of users falling for phishing or social

engineering attacks.

Remember that a combination of these techniques is often the most effective approach to ransomware detection. In our research, we propose a combination of these techniques.

2.3.10 SDN-Based Approaches in Ransomware Detection

Several studies have explored the potential of SDN in ransomware detection and defence, with positive results [6]. For example, [49] proposed an approach that uses SDN to detect ransomware attacks by analyzing network traffic and identifying suspicious behavior patterns based on ML algorithms. Their approach achieved high accuracy and reduced false positives compared to traditional signature-based approaches. Another study, by [35], focuses on mitigating ransomware attacks using an SDN-based approach that involved identifying and isolating infected hosts to prevent the spread of ransomware. The authors demonstrated that this approach was effective in mitigating ransomware attacks and reducing the risk of data loss. Additionally, [41] proposed an approach that leverages ML models to train a classifier capable of detecting ransomware attacks based on network traffic features. Their approach uses SDN to monitor network traffic and detect ransomware attacks in real time, making it a promising and effective technique for detecting and mitigating ransomware attacks. Moreover, a different approach was proposed by [4], which utilized SDN to identify the modes of ransomware communication. Their approach observed network communication patterns of CryptoWall and Locky ransomware families to provide a rapid response to ransomware threats.

2.3.11 ML Techniques in Ransomware Detection

ML techniques have been widely used in ransomware detection due to their ability to automatically learn and adapt to changing attack patterns [40].

eXtreme Gradient Boosting (XGBoost)

XGBoost is one of the powerful ML algorithms used for ransomware detection. It offers high accuracy, feature importance analysis, regularization techniques, and efficient handling of imbalanced data. XGBoost excels in capturing intricate patterns and relationships in data, making it effective in differentiating between normal and malicious activities [2]. It provides insights into the most influential features, aiding in the identification of critical factors for ransomware detection. XGBoost incorporates regularization techniques to prevent overfitting and handle noise in the data. It also offers methods to handle imbalanced datasets commonly encountered in ransomware detection. With its speed and efficiency, XGBoost can handle large-scale datasets and perform real-time ransomware detection [13]. By training XGBoost on a dataset comprising both benign and malicious samples, it can learn to accurately classify ransomware instances. Overall, XGBoost is a valuable algorithm in the fight against ransomware, enabling effective detection and mitigation strategies [44].

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is an ML algorithm used for ransomware detection based on the similarity of instances. It compares the characteristics of a sample with its nearest neighbours in the training data to assign a class label [8]. KNN is non-parametric and can handle numerical and categorical features. The choice of the parameter 'k' affects the algorithm's performance, with a small 'k' leading to overfitting and a large 'k' causing underfitting. KNN is computationally efficient during testing but requires significant memory for storing the training dataset. By training KNN on labelled data, it can classify new instances as malicious or non-malicious. KNN is a valuable tool for accurate and interpretable ransomware detection [7].

Decision Tree (DT)

DT is one of the most common ML techniques that have shown promising results. Various studies have demonstrated the effectiveness of this technique in accurately detecting and mitigating ransomware attacks. One study by [37] proposed an ML-based approach for ransomware detection using network traffic features. They extracted features such as packet length and inter-arrival time from network traffic and trained different ML models to classify the traffic as either benign or ransomware-infected.

Naïve Bayes (NB)

Although NB is one of the traditional approaches used in ML. This model achieved an all-time not-high accuracy rate in identifying ransomware attacks based on network traffic patterns. Other studies have focused on the use of SDN in combination with ML and DL techniques for ransomware detection and mitigation. For instance, authors in a study proposed an intrusion detection system based on DL in SDN environment to mitigate Distributed Denial of Service attacks [22]. Additionally, ML techniques have been applied to detect DDoS attacks in SDN environments. Overall, the literature review demonstrates that ML and DL techniques have been successfully used for ransomware detection and mitigation. However, there is a need for further research to explore the potential of combining these techniques with SDN for more effective detection and mitigation of ransomware attacks.

Random Forest (RF)

RF is an ML algorithm that has gained attention for its capability to detect ransomware, among other types of malicious software. RF in ransomware detection is gaining popularity and proving to be highly effective. RF algorithm has emerged as a computationally efficient classification method in various fields. One of the reasons for the increasing popularity of Random Forest in ransomware detection is its proven

effectiveness as a computationally efficient classification method across various fields such as malware and ransomware detection [32]. RF is a popular and effective ML technique that has garnered attention in the field of ransomware detection. RF classifier has several advantages, including its ability to classify large amounts of data with a lower percentage error and its less time-consuming nature, which is a prominent and robust ML technique. RF classifier demonstrates remarkably sensible and preferable results compared to other classifiers in detecting various types of attacks [27].

2.3.12 DL Techniques in Ransomware Detection

DL in Ransomware detection is another area that has gained attention in recent years. In recent years, DL has emerged as another promising approach in the field of ransomware detection for its diversity and robustness in accuracy and adaptability of new types of ransomware detection capacity.

Convolutional Neural Networks (CNN)

CNN is a type of DL algorithm that has gained significant attention in the field of ransomware detection. CNN has shown promise in the field of ransomware detection. The utilization of CNNs in the detection of ransomware has gained traction due to its diversity, robustness, and adaptability to new types of ransomware. CNN is a DL algorithm commonly used for image recognition tasks. It consists of convolutional layers that extract features from input data and pooling layers that reduce the dimensionality of the features. CNN has shown consistent performance in ransomware detection compared to LSTM and other methods [16].

Recurrent Neural Networks (RNN)

RNN have also been explored in the field of ransomware detection. However, RNN is not used as much compared to CNN. It has shown promising results in the detec-

tion of ransomware attacks, particularly in detecting specific types of attacks such as "Brute force cross-site scripting", "In today's rapidly changing world, the significance of accurate ransomware detection cannot be overstated. It is crucial to have efficient and accurate methods for detecting ransomware attacks [16]. The RNN uses a specialized neural component called the *Attended Recent Inputs* (ARI) cell, which performs attention at the input of the RNN. The ARI cell learns attention weights for recent inputs and uses their corresponding significance when processing the input sequence. The RNN with attention mechanisms and the ARI cell aims to capture both the general sequence learning and the short locally repeating patterns present in ransomware sequences, improving the performance of ransomware detection systems [1].

Long Short-Term Memory (LSTM)

LSTMs are commonly used when dealing with long sequences and have the ability to capture long-term dependencies in the data. LSTMs utilize three types of gates (input, output, and forget) along with an explicit cell memory to control the flow of information within the network. This research [1] is adapted with the Attended Recent Inputs (ARI) mechanism, which incorporates attention mechanisms to capture local event patterns in ransomware sequences. The ARI-LSTM variant combines the ARI cell with the LSTM architecture, allowing for improved detection of ransomware by considering both the general sequence learning and the short locally repeating patterns present in ransomware sequences. In the context of ransomware detection, LSTM is used to analyze dynamic features obtained from different views, such as API call sequences, DLLs, enumerated directories, and registry key operations [16]. LSTM enhances the true positive rate (TPR) in ransomware detection, but it can also cause poor accuracy and false positive rate (FPR) compared to other DL models like CNN. *Multi-Layer Perceptron* (MLP) falls behind CNN and LSTM in terms of performance, as it is not designed to handle sequential input, but it can learn from

the registry key feature set. LSTM's ability to learn long-term dependencies makes it suitable for analyzing sequential features in ransomware detection.

2.3.13 Comparative Analysis of Existing Approaches

Various ML techniques have been employed in ransomware detection. For instance, [42] conducted a comprehensive survey on the use of machine-learning algorithms for detecting ransomware attacks. Their study reviewed literature from various perspectives and analyzed the effectiveness of different ML algorithms in detecting ransomware. They found that ML algorithms such as support vector machines, random forests, and neural networks effectively detected ransomware attacks with high accuracy. Similarly, [3] used a ML-based approach in their SDN-based ransomware mitigation technique. They utilized ML algorithms to classify network traffic into normal and ransomware-infected traffic, allowing for early detection and isolation of infected hosts to prevent the further spread of the ransomware. Furthermore, [20] proposed a ML-based approach to detect ransomware attacks based on network traffic features. They extracted various features from network traffic, such as packet length and inter-arrival time, and trained a ML model using these features to classify network traffic as either benign or ransomware-infected.

2.3.14 Research Gaps and Limitations

From the previous studies, this research aims to address the following research gaps:

Limited Integration of SDN and ML: There is a lack of comprehensive research that combines the capabilities of SDN and ML techniques for ransomware detection and mitigation. While both SDN and ML have individually shown promise in enhancing network security, their integration specifically for ransomware detection has not been extensively explored. The research seeks to bridge this gap by leveraging SDN's flexibility in network management and ML's ability to identify patterns and

anomalies associated with ransomware attacks.

Lack of Comprehensive Approach: Existing research often focuses on individual aspects of ransomware detection or mitigation, without considering a holistic and integrated approach. There is a need for a comprehensive framework that combines multiple techniques and methodologies to effectively detect and mitigate ransomware attacks. The research aims to fill this gap by proposing a unified approach that integrates SDN and ML to enhance the accuracy and efficiency of ransomware detection and mitigation.

Limited Evaluation on Real-world datasets: Many existing studies primarily rely on simulated or limited datasets, which may not fully capture the complexity and diversity of real-world ransomware attacks. Evaluating the proposed approach using comprehensive and diverse real-world datasets is essential to assess its effectiveness in practical scenarios. The research aims to address this gap by utilizing reputable and representative datasets to evaluate the performance of the proposed approach in detecting and mitigating real-world ransomware attacks.

Practical Implementation Challenges: While theoretical research on ransomware detection and mitigation techniques exists, practical implementation challenges and considerations are often overlooked. The research aims to bridge this gap by considering the practical aspects of deploying SDN-based solutions with ML capabilities. This includes addressing issues related to network compatibility, scalability, performance, and ethical considerations in the implementation of the proposed approach.

By addressing these research gaps, the thesis aims to contribute to the knowledge and understanding of effective ransomware detection and mitigation techniques, specifically by integrating SDN and ML. The research will provide insights and practical guidelines for designing robust security systems that can proactively detect and mitigate ransomware attacks, thereby enhancing the security and resilience of critical

network infrastructures.

2.4 Summary

In summary, the background chapter provides a solid foundation for the thesis, offering insights into the nature of ransomware, its detection challenges, existing approaches, and the role of ML and SDN in combating ransomware threats. This knowledge sets the stage for the research presented in the subsequent chapters, ultimately aiming to contribute to improved ransomware detection strategies.

Chapter 3

Methodology

3.1 Introduction

This chapter presents the detailed methodology followed in designing, implementing, and evaluating the proposed solution for detecting and mitigating ransomware attacks. The chapter begins with an overview of the research methodology, outlining the key steps involved in this study. This includes data collection, preprocessing, feature extraction, model development, and evaluation. The rationale behind the chosen methodology is discussed, highlighting its suitability for addressing the research problem effectively.

3.2 Overview of the Proposed System

This section provides an overview of the proposed system, outlining its key components and functionalities. The proposed system combines the power of SDN, ML, and DL techniques to enhance ransomware detection, monitoring and mitigation capabilities. SDN allows for centralized control and programmability of the network, enabling efficient monitoring and analysis of network traffic.

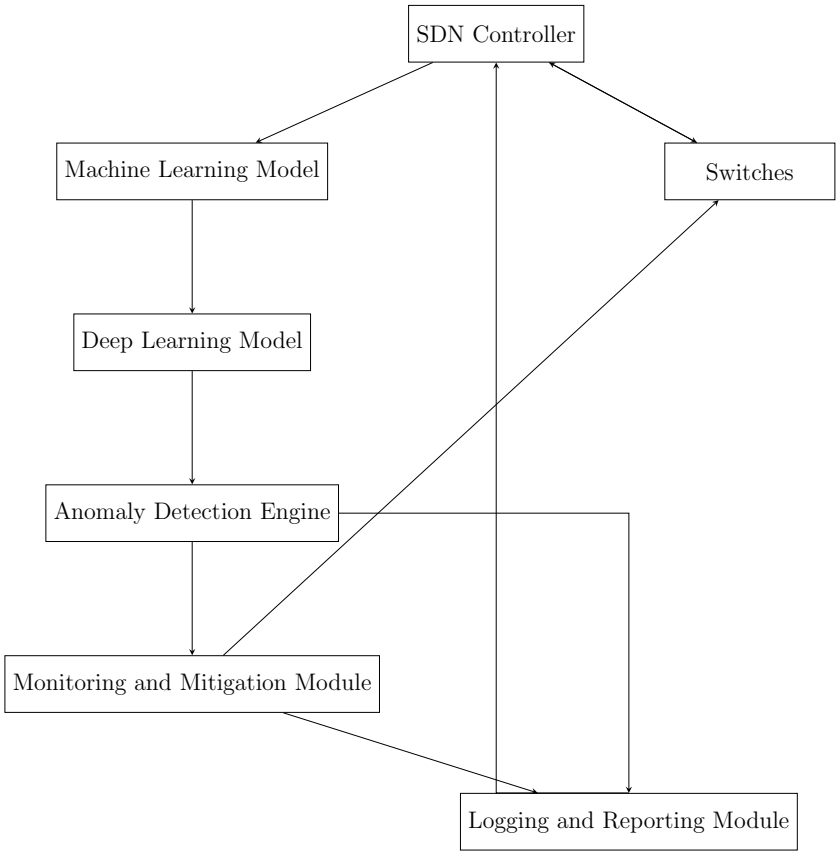

Figure 3.1: Proposed SDN-based Architecture

Our Proposed SDN Architecture, presented in Figure 3.1, consists of several components working together to detect and mitigate ransomware attacks:

SDN Controller: The SDN controller is responsible for managing the network infrastructure and forwarding traffic through the network. It receives traffic flow information from the switches and sends instructions to the switches to forward traffic based on predefined policies.

Switches: The switches are responsible for forwarding traffic in the network based on instructions received from the SDN controller. They also collect traffic flow information and send it to the SDN controller for analysis.

ML Model: The ML model is trained on a dataset of known ransomware attacks and normal traffic. It uses this training data to detect anomalous traffic patterns that could indicate a ransomware attack. The model is periodically updated to improve

its accuracy over time.

DL Model: The DL model in the proposed SDN-based architecture, as depicted in Figure 3.1, is responsible for leveraging advanced DL models. This module processes information from the ML Model and focuses on intricate pattern recognition and analysis. It plays a crucial role in enhancing the system's ability to detect complex and evolving threats, contributing to the overall effectiveness of the intrusion detection and mitigation capabilities within the SDN framework.

Anomaly Detection Engine: The anomaly detection engine receives traffic flow information from the DL Model and feeds it to the Monitoring and Mitigation Module and Logging and Reporting Module. It analyzes the output from the model and triggers an alert if an anomalous traffic pattern is detected.

Monitoring and Mitigation Module: The mitigation module is responsible for taking action to stop the ransomware attack. It receives the alert from the anomaly detection engine and can take several actions to mitigate the attack, such as blocking the source of the attack or isolating the infected device.

Logging and Reporting Module: The logging and reporting module is responsible for collecting logs from all the components in the system and generating reports on network activity, including any ransomware attacks detected and mitigated.

Overall, this architecture uses SDN to enable dynamic traffic forwarding and centralized management of the network, while ML is used to detect ransomware attacks and trigger the mitigation module. The logging and reporting module provides visibility into network activity, helping security teams to identify and respond to ransomware attacks more effectively.

The system leverages a diverse set of ML and DL algorithms, including K-Nearest Neighbors (KNN), Naive Bayes (NB), Decision Trees (DT), Random Forest (RF), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN). These algorithms are trained on labeled datasets containing both ransomware samples

and normal network traffic, enabling them to learn patterns and detect ransomware attacks.

3.3 SDN Architecture

The system architecture is designed to facilitate the integration of SDN with the trained ML and DL models. It encompasses components such as the SDN controller, network switches, data preprocessing module, feature extraction module, and the prediction module. The SDN controller, which acts as the centralized brain of the network.

This architecture 3.2 simplifies network management and allows for dynamic control and configuration of network resources.

At the core of the SDN architecture is the SDN controller acts as the central brain of the system, managing the network and directing the flow of traffic to the appropriate modules for analysis. It is responsible for managing and directing network traffic, enforcing policies, and making intelligent decisions based on network conditions.

The SDN controller communicates with network devices, such as switches and routers, through a southbound interface. This interface allows the controller to gather information about network topology, traffic flow, and performance metrics.

On the other hand, the SDN controller exposes an application programming interface (API) through a northbound interface. This interface allows external applications and services to interact with the SDN controller and make use of its capabilities.

The network devices in an SDN architecture are known as OpenFlow switches. These switches receive instructions from the SDN controller and forward network traffic based on the controller's directives. They provide programmable data forwarding and are capable of implementing network policies and security measures.

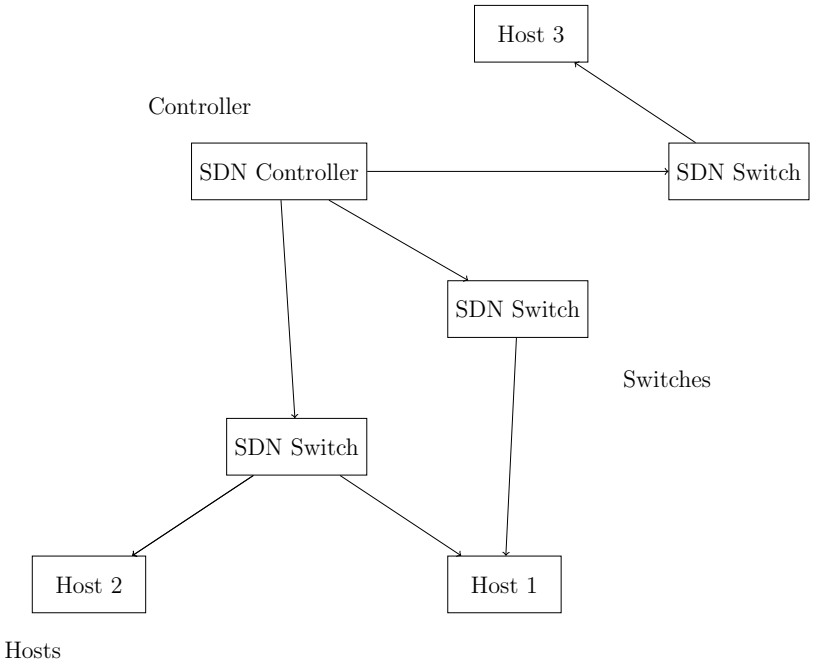

Figure 3.2: SDN Architecture in Complex Scenario

The main difference between the two figures lies in the complexity and level of detail in representing the SDN architecture. Figure 3.2 shows the SDN Architecture in bellows:

Controller: Controller: The figure includes a detailed SDN controller positioned centrally.

Switches: Switches: Multiple SDN switches are present, connected to the controller. They, in turn, connect to various hosts.

Hosts: Several hosts are connected to the switches.

This figure 3.3 provides a more comprehensive view of the SDN architecture, detailing the relationships between the controller, switches, and hosts. Figure 3.3 Simplified SDN Architecture in below:

Controller: Controller: There is a single controller. **Switches:** Two switches are connected directly to the controller. **Hosts:** Four hosts are connected to the switches.

This figure represents a simplified version of the SDN architecture, providing a

high-level overview with fewer components and connections. It focuses on conveying the basic structure of the SDN without delving into intricate details.

In summary, the SDN architecture simplifies 3.3 network management by separating the control plane from the data plane. It centralizes control and decision-making, allowing for flexible network configuration and efficient traffic management. The SDN controller serves as the brain of the network, communicating with network devices and exposing APIs for external applications to leverage its capabilities. OpenFlow switches enable programmable data forwarding based on the controller's instructions.

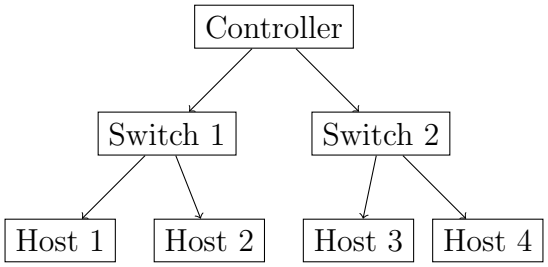

Figure 3.3: Simplified SDN Architecture

This Figure 3.4 illustrates an SDN architecture enhanced for ransomware detection and mitigation, featuring an SDN controller orchestrating communication between switches and hosts while integrating ML and DL modules for advanced threat analysis and response.

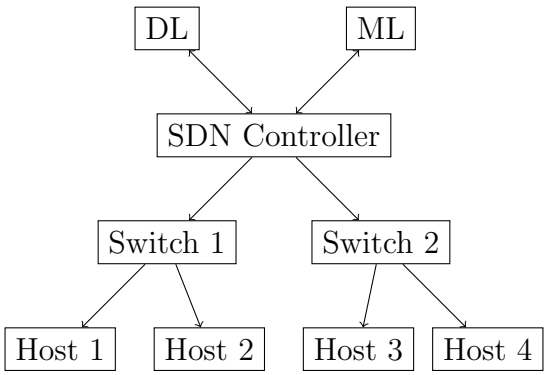

Figure 3.4: SDN Architecture with ML and DL for Ransomware Detection and Mitigation

The data flow within the system involves capturing raw network traffic data,

preprocessing it to remove noise and inconsistencies, extracting relevant features from the preprocessed data, and feeding it into the trained models for prediction. The models generate output predictions indicating the likelihood of a given network traffic being ransomware-related or normal.

The proposed system also includes a feedback loop mechanism to continuously improve its performance. Detected ransomware instances are fed back into the system, enabling it to learn from new patterns and update its detection capabilities.

Overall, the proposed system combines the strengths of SDN, ML, and DL to create a robust and intelligent framework for ransomware detection and mitigation. By leveraging the capabilities of these technologies, the system aims to enhance network security and protect against the growing threat of ransomware attacks.

3.4 Data Collection and Preprocessing

In our research on ransomware detection and mitigation, we primarily utilize network traffic data as our main data source. This data provides valuable insights into device communication patterns within a network. We collect the data from [25], [29] and store it. To handle the large volume of data, we use the CICflowmeter tool [12] [28] to convert network .pcap data into flow-level data and store it in a structured format, such as CSV. We then preprocess the data by handling missing values, removing duplicates, and normalizing numerical features. Feature engineering techniques has also been applied to extract relevant features from the network traffic data. This processed data serves as the foundation for training and evaluating ML and DL models for ransomware detection and mitigation.

3.5 Feature Extraction and Selection

Feature extraction and selection are vital for detecting ransomware attacks. In our study, we collected a dataset with 84 features related to network traffic and system behaviors. We employed the Recursive Feature Elimination (RFE) algorithm and Particle Swarm Optimization (PSO) to enhance our detection model. RFE and PSO both iteratively eliminate less relevant features to improve performance. We ranked the features based on their importance using RFE and PSO. Two experiments were conducted, one with 10 features and another with 20 features. The selected features were chosen for their relevance to ransomware detection. They captured network traffic patterns, resource utilization, and behavioral anomalies associated with ransomware activities. Feature extraction and selection with RFE streamline the detection process, enhance model performance, and reduce computational overhead. Focusing on informative features improves the efficiency and accuracy of our ransomware detection system.

Algorithm 3.1 Recursive Feature Elimination (RFE)

Input : Features X, Target variable y, Number of features to select k
procedure RFE(X, y, k)
    Initialize an empty set S to store the selected features
    Initialize an empty list *scores* to store the feature rankings
    Initialize a ML model *model* (e.g., KNN, DT, Random Forest, etc.)
    while $number_of_selected_features < k$ **do**
        Fit the model using X and y
        Importance Score(i)=$\sum_{k=1}^{N} \frac{w_k}{n_k}$
        Append the scores to *scores* list
        Identify the feature with the lowest score and remove it from X
    end while
    Sort the *scores* list in descending order
    Select the top k features corresponding to the highest scores
    Output: Set of selected features S
end procedure

Recursive Feature Elimination (RFE), shown in Figure 3.1, is a feature selection algorithm used in ML. It iteratively removes less important features based on their

impact on model performance. RFE helps reduce overfitting and improve model generalization by eliminating irrelevant or redundant features. It assigns importance scores to features and eliminates the least important ones until a desired number is reached. RFE is useful for selecting informative features in ransomware detection, enhancing model accuracy and efficiency.

We employed another algorithm that utilizes Particle Swarm Optimization (PSO) in Figure 3.2 as an optimizer, offering a framework for selecting an optimal feature subset in the context of ransomware detection. It iteratively refines the feature selection by updating particle positions and velocities, evaluating fitness, and identifying the best feature subset.

3.6 ML Algorithms for Ransomware Detection

When selecting a ML algorithm for ransomware detection, several factors need to be considered. Firstly, the algorithm should be capable of handling the characteristics and complexities of ransomware data. Secondly, it should provide accurate and reliable results in identifying ransomware patterns. Additionally, the algorithm's computational efficiency is crucial for real-time detection. The interpretability of the algorithm is also important for understanding the decision-making process. Lastly, the scalability and adaptability of the algorithm to evolving ransomware threats should be taken into account. Commonly used algorithms for ransomware detection include K-Nearest Neighbors (KNN), Decision Trees (DT), Random Forests (RF), Support Vector Machines (SVM), and XGBoost algorithms.

Algorithm 3.2 Particle Swarm Optimization (PSO) for Feature Selection

Initialize:
$P_S \leftarrow$ Population size
$M_I \leftarrow$ Max iterations
$w \leftarrow$ Inertia weight
$c_1, c_2 \leftarrow$ Acceleration constants
procedure PSO(X, y, k)
for iteration $\leftarrow 1$ to M_I **do**

$$v_{ij} \leftarrow w \cdot v_{ij} + c_1 \cdot r_1 \cdot (pbest_{ij} - x_{ij}) + c_2 \cdot r_2 \cdot (gbest_j - x_{ij})$$

where v_{ij} is the velocity of particle i on dimension j, x_{ij} is the position of particle i on dimension j, $pbest_{ij}$ is the personal best position of particle i on dimension j, $gbest_j$ is the global best position on dimension j, r_1 and r_2 are random values in the range $[0, 1]$.

Update the Value:

$$x_{ij} \leftarrow x_{ij} + v_{ij}$$

Evaluate particle fitness:
update P_S
Update I_S:
if $P_S > I_S$ **then**
$pbest_{ij} = P_S$
$gbest_j = x_{ij}$
end if
if $P_S < I_S$ **then**
$pbest_{ij} = I_S$
$gbest_j = x_{ij}$
end if
Convergence check:
If the termination condition is met (maximum number of iterations or desired fitness threshold), exit the loop
Output : $pbest_{ij}$ **and** $gbest_j$

3.6.1 Selection of Algorithms (e.g., XGBoost, KNN, DT, NB, RF)

XGBoost (eXtreme Gradient Boosting): XGBoost is a powerful and widely used ML algorithm that excels in handling complex and high-dimensional data. It utilizes an ensemble of weak prediction models, such as decision trees, to create a strong and accurate predictive model. XGBoost is known for its excellent performance in various tasks, including ransomware detection, due to its ability to capture complex patterns and handle imbalanced datasets.

The XGBoost algorithm can be represented as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) + \gamma T_k \tag{3.1}$$

Where, $\hat{y}i$ represents the predicted value for the $i-th$ instance. The summation $\sum k = 1^K$ represents the sum of predictions from each individual decision tree model denoted as f_k, for the $i-th$ instance x_i. This summation is taken over all K trees in the XGBoost model. Additionally, γ represents the regularization term, where T_k represents the structure of the $k-th$ tree.

By summing the predictions from each individual tree and adding the regularization term, the XGBoost algorithm generates the final predicted value $\hat{y}_i$.

K-Nearest Neighbors (KNN): KNN is a simple yet effective algorithm that classifies new instances based on the majority vote of their k nearest neighbors. It is suitable for ransomware detection as it does not require assumptions about the underlying data distribution. KNN is versatile and can handle both numerical and categorical data, making it applicable to a wide range of ransomware detection scenarios.

The mathematical equation for the KNN algorithm can be represented as follows:

$$\hat{y} = \arg\max_{c} \sum_{i=1}^{K} w_i \cdot I(y_i = c) \tag{3.2}$$

Where, $\hat{y}$ represents the predicted value for the query point. The term $\arg\max_c$ indicates that we select the class label (c) that maximizes the following summation. The summation $\sum_{i=1}^{K}$ is taken over the K nearest neighbors, where w_i represents the weight assigned to each neighbor, and $I(y_i = c)$ is an indicator function that returns 1 if the label of the $i - th$ neighbour is equal to c, and 0 otherwise.

By calculating the weighted sum of the labels of the K nearest neighbors and selecting the class label with the highest sum, the KNN algorithm generates the final predicted value $\hat{y}$.

Decision Trees (DT): DT are intuitive and easy-to-interpret models that partition the feature space based on a set of rules. DTs are well-suited for ransomware detection as they can handle both categorical and numerical features, and they capture complex decision boundaries. However, they may suffer from overfitting and instability.

The mathematical equation for the DT algorithm can be expressed as follows:

$$\hat{y} = f(\mathbf{x}) \tag{3.3}$$

Where $\hat{y}$ represents the predicted value for a given input $\mathbf{x}$. The function $f(\cdot)$ represents the decision tree model, which maps the input features to the predicted value.

The decision tree model is built by recursively partitioning the feature space based on the values of the input features. Each internal node in the tree corresponds to a splitting condition on a specific feature, and each leaf node corresponds to a predicted value or class label.

To predict the output $\hat{y}$, the input $\mathbf{x}$ is traversed down the decision tree, following

the appropriate branch at each internal node based on the feature values. Finally, the predicted value or class label associated with the reached leaf node is assigned as the output.

Naive Bayes (NB) : NB is a probabilistic algorithm that assumes independence between features. It is computationally efficient and robust to irrelevant features, making it suitable for ransomware detection. NB performs well with limited training data and can handle high-dimensional feature spaces. However, it may struggle with capturing complex interactions between features.

The mathematical equation for the NB algorithm can be shown as:

$$\hat{y} = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(x_i|c) \tag{3.4}$$

Where, $\hat{y}$ represents the predicted class label for a given input $\mathbf{x}$. C represents the set of possible class labels. P(c) represents the prior probability of class c, which is estimated from the training data. $P(x_i|c)$ represents the conditional probability of feature x_i given class c, which is estimated from the training data as well.

The Naive Bayes algorithm calculates the posterior probability of each class label given the input features and selects the class label with the highest probability as the predicted output.

To make predictions, the algorithm estimates the prior and conditional probabilities based on the training data. Then, for a new input $\mathbf{x}$, it calculates the probability of each class label using the equation above and selects the class label with the highest probability as the predicted output.

Random Forests (RF) : RF is an ensemble learning method that combines multiple decision trees. RF improves the generalization and robustness of individual decision trees and helps reduce overfitting. It is effective in handling high-dimensional data and can provide feature importance rankings for ransomware detection.

The mathematical derivation for the RF algorithm can be shown as follows :

$$\hat{y} = \frac{1}{N} \sum_{j=1}^{N} f_j(\mathbf{x}) \tag{3.5}$$

Where $\hat{y}$ represents the predicted output for a given input $\mathbf{x}$. N represents the number of decision trees in the random forest. $f_j(\mathbf{x})$ represents the prediction of the $j - th$ decision tree.

The Random Forest algorithm works by training each decision tree independently on a random subset of the training data. During prediction, each decision tree in the forest independently makes its own prediction for the input $\mathbf{x}$. The final prediction is obtained by averaging the predictions of all the decision trees.

To summarize, the RF algorithm combines the predictions of multiple decision trees to obtain a more accurate and robust prediction

3.6.2 Model Training and Evaluation

For the ML model training and testing in ransomware detection, we utilize XGBoost, KNN, DT, NB, and RF algorithms. XGBoost employs an ensemble of weak prediction models and optimizes them iteratively to minimize the loss function. It will be trained on the training dataset and evaluated on the testing dataset. KNN algorithm stores the entire training dataset and calculates the distance between new instances and training instances to determine the nearest neighbors. We will fit the training dataset to the KNN model and assess its performance on the testing dataset. DT constructs decision trees by recursively partitioning the feature space based on rules. The DT model will be trained by learning optimal splits for each node using the training dataset and evaluated on the testing dataset. NB is a probabilistic algorithm assuming feature independence. The NB model will estimate conditional probabilities of features given class labels using the training dataset. Its performance

will be evaluated on the testing dataset. RF is an ensemble of decision trees trained on random subsets of data and features. Multiple decision trees' predictions are combined for the final prediction. The RF model will be trained on different subsets of the training dataset and tested on the testing dataset.

During training, ML models learn patterns and relationships in the training data to make predictions on unseen data. The testing phase assesses their performance on unseen instances. Accuracy, precision, recall, and other metrics will be used to compare and select the best-performing ML model for ransomware detection.

3.7 DL Algorithms for Ransomware Detection

Nowadays DL algorithms for ransomware detection consider various factors such as the complexity of the problem, available data, and computational resources. Some commonly used DL algorithms for this task are LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network).

3.7.1 Selection of Algorithms (e.g., LSTM, CNN)

Long Short-Term Memory (LSTM) : LSTM is a type of recurrent neural network (RNN) that can effectively model sequential data by capturing long-term dependencies. It is suitable for analyzing time-series data such as network traffic logs. LSTM networks can learn patterns and relationships in the data over different time intervals, which can be beneficial for detecting ransomware attacks.

Mathematically the LSTM algorithm can be shown as :

$$
\begin{aligned}
f_t &= \sigma_g(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma_g(W_i \cdot [h_{t-1}, x_t] + b_i) \\
o_t &= \sigma_g(W_o \cdot [h_{t-1}, x_t] + b_o) \\
\tilde{C}_t &= \sigma_c(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
h_t &= o_t \odot \sigma_h(C_t)
\end{aligned}
\tag{3.6}
$$

Where, f_t represents the forget gate output, which controls how much of the previous cell state C_{t-1} to forget. i_t represents the input gate output, which determines how much of the input $\tilde{C}_t$ is added to the cell state. o_t represents the output gate output, which controls how much of the cell state C_t is used to compute the hidden state h_t. $\tilde{C}_t$ is the candidate cell state, which represents the new information that could be stored in the cell state. C_t is the updated cell state, which is a combination of the previous cell state and the candidate cell state. h_t is the hidden state, which is the output of the LSTM cell and carries information to the next time step.

W_f, W_i, W_o, W_C are weight matrices, b_f, b_i, b_o, b_C are bias vectors, and σ_g and σ_h represent the sigmoid and hyperbolic tangent activation functions, respectively. The $\odot$ symbol denotes element-wise multiplication.

Convolutional Neural Network (CNN):CNN is a DL algorithm commonly used for image processing tasks. However, it can also be applied to ransomware detection by treating network traffic data as image-like representations. CNNs are adept at capturing spatial features and patterns, making them useful for identifying malicious patterns in network traffic data.

CNN is mathematically expressed as follows:

$$
\begin{aligned}
h_i &= \sigma(W_i * x_i + b_i) \\
h_o &= \sigma(W_o * x_o + b_o)
\end{aligned}
\tag{3.7}
$$

Where, h_i represents the hidden feature map after the convolutional layer, obtained by applying the activation function σ to the element-wise sum of the element-wise multiplications between the input feature map x_i and the corresponding convolutional filter W_i, followed by the addition of the bias term b_i. h_o represents the output feature map after the pooling layer, obtained by applying the activation function σ to the element-wise sum of the element-wise multiplications between the input feature map x_o and the corresponding convolutional filter W_o, followed by the addition of the bias term b_o. In these equations, $*$ denotes the convolution operation, σ represents the activation function (such as ReLU or sigmoid), and W_i, W_o are the convolutional filter weights, while b_i, b_o are the bias terms.

3.7.2 Model Training and Evaluation

LSTM is well-suited for capturing patterns and dependencies in sequential data, making it effective for analyzing network traffic. During training, the LSTM model learns to recognize patterns that are indicative of ransomware activity, enabling it to make accurate predictions during testing.

CNN, originally designed for image processing, can also be utilized for ransomware detection by treating network traffic data as image-like representations. It excels at capturing spatial features and local patterns. The CNN model is trained to extract relevant features from the network traffic data and classify it as either ransomware or benign traffic.

Training the LSTM and CNN models involves feeding them with labeled network traffic data. The models learn from this data and adjust their internal parameters to improve their predictive capabilities. The training process typically involves multiple iterations, with the models continuously refining their understanding of ransomware patterns.

Once trained, the LSTM and CNN models are evaluated on a separate testing

dataset. The models are presented with unseen network traffic data, and their performance is measured using evaluation metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into how well the models generalize to new data and their ability to correctly classify ransomware traffic.The testing phase is crucial for assessing the effectiveness of the LSTM and CNN models in detecting ransomware. It helps determine their overall performance and provides an indication of their reliability in real-world scenarios.

3.8 Integration of SDN, ML, and DL

Figure 3.5 illustrates the system architecture for ML and DL models. The process begins with raw data, which is stored in a data store. The data then undergoes preprocessing, including data transformation and cleaning. Feature extraction is performed to extract relevant information from the data.

Next, the ML and DL models are trained using the extracted features. The models used in this architecture include K-Nearest Neighbors (KNN), Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN).

Once the models are trained, they are tested using a separate testing dataset. The testing phase includes the detection of patterns or anomalies in the data. Finally, the output prediction is generated based on the detected patterns, providing insights or predictions based on the ML and DL models.

This system architecture 3.5 provides a high-level overview of the workflow involved in ML and DL models, from data preprocessing to model training, testing, and prediction.

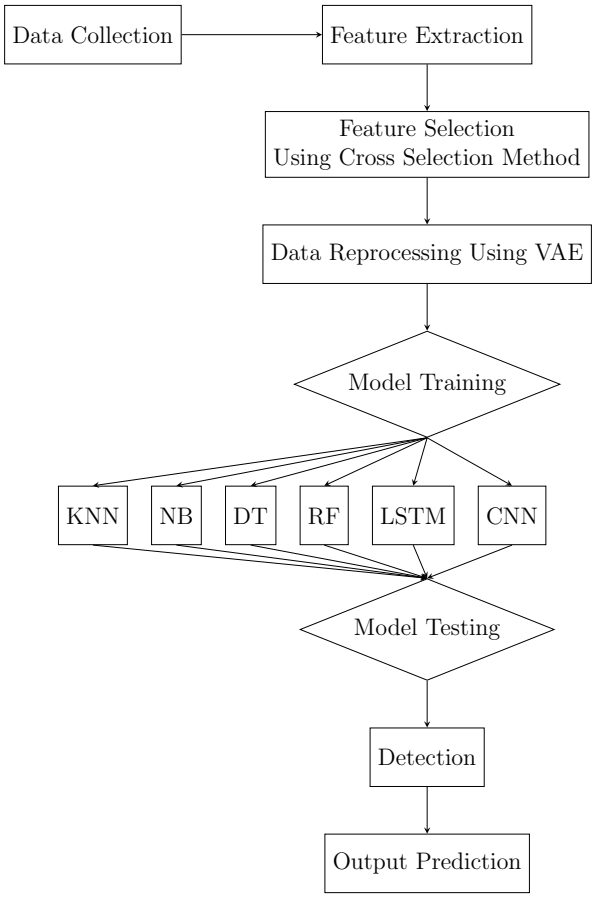

Figure 3.5: System Architecture for ML and DL Models

3.9 Our Proposed Method

In our proposed method we have developed a response to this evolving threat landscape, we propose a comprehensive ransomware detection method that combines VAE. VAE is an algorithmic model employed to learn a probabilistic representation of features in a dataset associated with potential ransomware activities. The VAE is designed to encode the characteristics of normal and anomalous behavior within the data, allowing it to identify subtle patterns indicative of ransomware threats. By leveraging a probabilistic latent space, the VAE not only reconstructs input data but also captures the underlying distribution of normal behavior. This enables the model to detect deviations or anomalies associated with ransomware, making it a valuable tool in the broader spectrum of cybersecurity for identifying malicious patterns and

enhancing threat detection capabilities.

The rationale for using a VAE in ransomware detection is its capability to capture complex data patterns and anomalies by learning a compact data representation. VAEs are like filters that prepare data for computer programs. They help programs (DL and ML) find ransomware by noticing differences from normal data patterns. This makes them good at spotting new types of ransomware. They offer adaptability to evolving threats, flexibility in data processing, and the ability to operate without relying on predefined signatures, making them a valuable tool for robust and proactive ransomware detection using ML, DL, and Pareto Ensemble techniques. This approach aims to enhance the accuracy and robustness of ransomware detection by leveraging the strengths of multiple models.

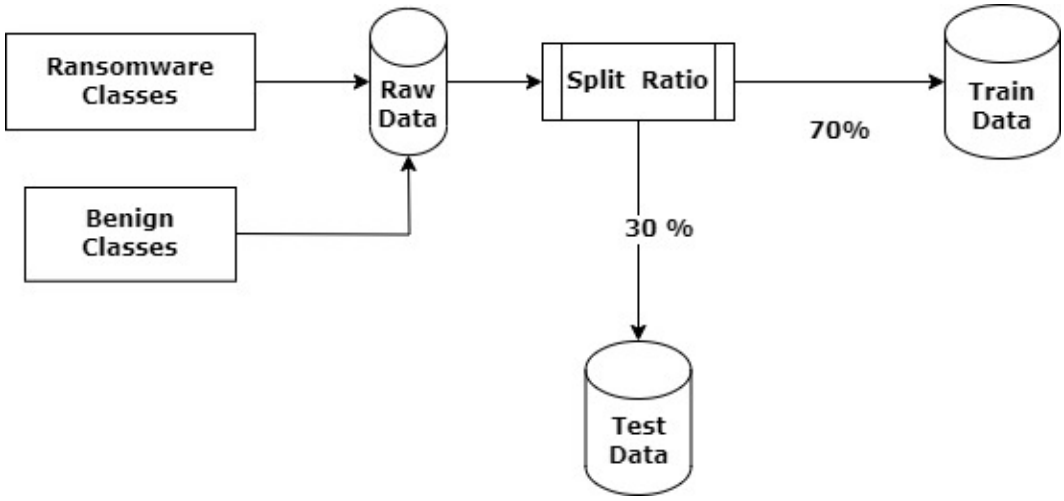

Figure 3.6: Our Proposed Dataset

Figure 3.6 shows the splitting ratio of our proposed dataset, where as Figure 3.7 shows the proposed system architecture.

In the context of ransomware detection using VAE, the output of the encoder is a learned representation of the input data in the latent space. This latent representation is often a lower-dimensional vector that encodes essential features and patterns of the data.

The VAE plays a crucial role in data preprocessing, transforming the raw data into

a meaningful and lower-dimensional representation in the latent space.The CICflowmeter-extracted features, selected using a combined approach of cross feature selection method PSO and RFE algorithms, serve as the input to the VAE encoder, generating a unified feature representation that improves ransomware detection. This hybrid approach leverages the strengths of various methods to improve detection accuracy and adaptability in the face of evolving ransomware threats.

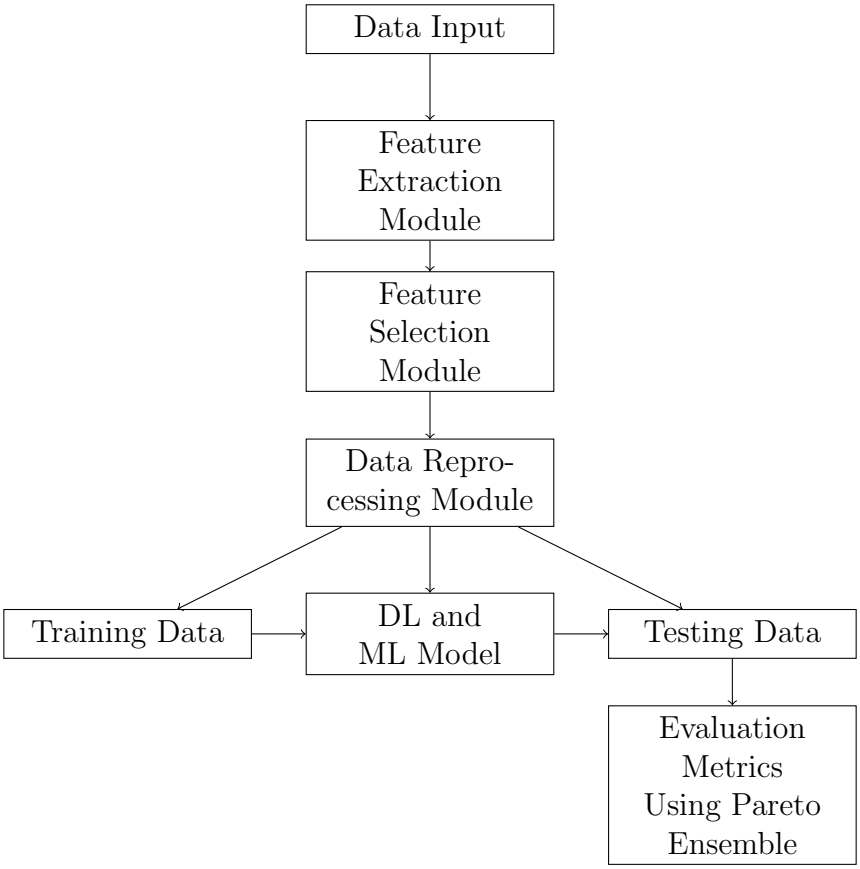

Figure 3.7: Overall Architecture of our Proposed Method

3.9.1 Data Reprocessing Using VAE

VAE in Figure 3.8 is a probabilistic generative model that plays a crucial role in the proposed method. It is utilized for feature extraction and dimensionality reduction from network traffic data. VAEs enable the learning of compact, latent representations of data, which can help uncover hidden patterns in network behaviour. By

training a VAE on a large dataset of network traffic, we can obtain informative, low-dimensional features that are used as input for subsequent models.

Ransomware detection usually involves different techniques, such as anomaly detection, signature-based detection, or ML models trained on features relevant to ransomware behaviour.

A VAE consists of an encoder and a decoder, which are used to learn a probabilistic mapping between high-dimensional input data (in this case, features related to ransomware behavior) and a lower-dimensional latent space. The mathematical equations for a VAE are as follows:

Mathematical Equations and Overall Design of VAE

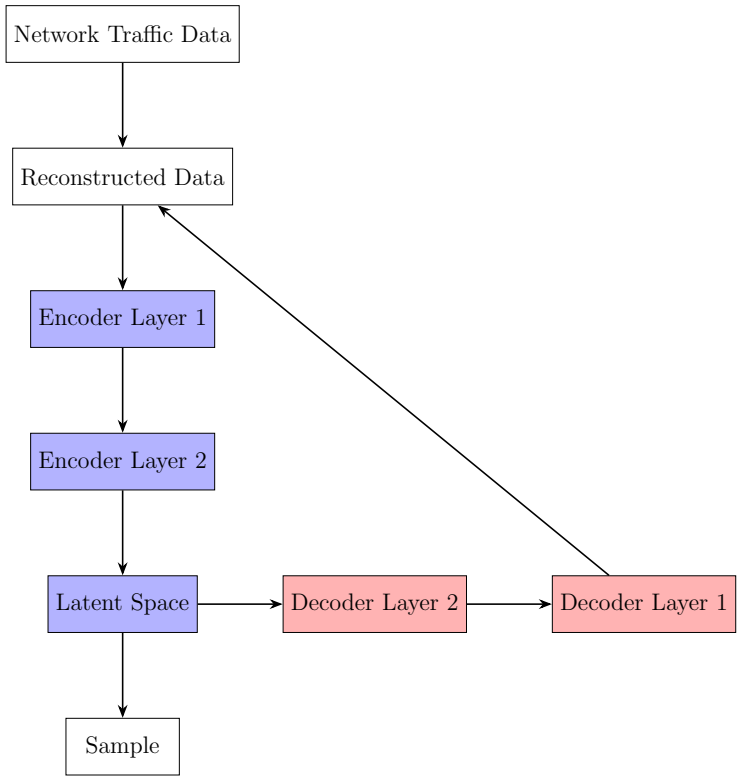

Figure 3.8: Our Simplified design of VAE

Encoder Network

The encoder maps input data, denoted as x (features related to ransomware behavior), to a probability distribution in the latent space, which is typically Gaussian. The encoder outputs two vectors, mean (μ) and standard deviation (σ), representing the parameters of this distribution:

$$q(z|x) = \mathcal{N}(z|\mu(x), \sigma(x))$$

where $\mathcal{N}$ denotes the Gaussian distribution.

Sampling

A point in the latent space (z) is sampled from the distribution $q(z|x)$ using the reparameterization equation or formula:

$$z = \mu(x) + \varepsilon \cdot \sigma(x)$$

Where ε is sampled from a standard Gaussian distribution ($\mathcal{N}(0, 1)$).

Decoder Network

The decoder network takes the sampled latent variable z and maps it back to the data space, attempting to reconstruct the original input data x. The decoder is denoted as $p(x|z)$ and can be any suitable neural network architecture.

Objective Function

The VAE is trained by maximizing a variational lower bound on the data likelihood, known as the Evidence Lower Bound (ELBO):

$$\text{ELBO} = E[\log p(x|z)] - \text{KL}(q(z|x)||p(z))$$

Where:

- $E[\log p(x|z)]$ is the reconstruction term, measuring how well the model reconstructs the input data.
- $\mathrm{KL}(q(z|x)||p(z))$ is the Kullback-Leibler (KL) divergence between the approximate posterior $q(z|x)$ and the prior $p(z)$. This term encourages the learned latent space to be close to a predefined prior distribution, usually a standard Gaussian ($\mathcal{N}(0, 1)$).

In the context of ransomware detection, it's more common to use other ML techniques, such as supervised or unsupervised models, to detect patterns or anomalies in system logs, network traffic, or file behaviour.

3.9.2 ML Models

In addition to VAE, traditional ML models are employed to identify ransomware patterns in the feature space generated by the VAE. These models, including decision trees, random forests, and support vector machines, are well-suited for classification tasks. They are trained on labeled data to differentiate between benign and malicious network traffic patterns. The combination of VAE-derived features with these ML models can significantly improve detection accuracy.

3.9.3 DL Models

DL models, such as CNNs and LSTMs, are utilized to capture intricate patterns and temporal dependencies in network traffic data. CNNs excel at feature extraction from structured data, while LSTMs are effective in modelling sequential information. DL models can effectively operate on feature spaces generated by VAE, benefiting from the informative, low-dimensional representations produced by VAE for tasks such as ransomware detection by integrating these DL models into the proposed method, we

can enhance the detection of complex ransomware behaviors that may be missed by traditional ML algorithms.

3.9.4 Pareto Ensemble Technique

The Pareto ensemble technique is a novel addition to our ransomware detection method. It combines the multiple models to create an ensemble that outperforms individual models. However, unlike traditional ensembles, the Pareto ensemble focuses on optimizing both sensitivity and specificity. It aims to achieve a balance that minimizes the false negative rate (missed ransomware detection) while also minimizing the false positive rate (legitimate traffic misclassified as ransomware). It is achieved by fine-tuning the ML model's decision threshold. Lowering the threshold increases sensitivity to ransomware, reducing false negatives but potentially raising false positives. Conversely, raising the threshold enhances specificity, reducing false positives but potentially increasing false negatives. In our case, we continually adjust this threshold to strike the right balance based on evolving ransomware behaviours and real-world monitoring. This fine-tuned ensemble approach provides a more reliable and robust detection mechanism.

3.9.5 Workflow

The proposed method operates in a series of steps:

1. **Feature Extraction:** CICFlowMeter is employed to extract essential features for ransomware detection from network traffic data.

2. **Feature Selection:** Feature selection in ransomware detection can be achieved effectively through the combination of Recursive Feature Elimination (RFE) and Particle Swarm Optimization (PSO) algorithms. RFE helps identify the most relevant features, while PSO optimizes the feature subset selection, resulting in

improved detection performance. This hybrid approach enhances the efficiency and accuracy of feature selection for ransomware detection.

3. **Data Repossessing:** VAE is used to extract informative, low-dimensional features from network traffic data latent space embeddings that represent abstract traffic patterns, statistical aggregations like mean packet sizes or inter-arrival times, and principal components that capture significant data variations. These low-dimensional features enable efficient data representation, aiding in tasks such as traffic analysis and ransomware detection.

4. **ML Models:** Traditional ML models are trained on the VAE-derived features to perform the initial classification of network traffic.

5. **DL Models:** Convolutional and recurrent neural networks analyze the network traffic data to capture complex patterns and temporal dependencies.

6. **Pareto Ensemble:** The Pareto ensemble technique combines the predictions of the ML and DL models to produce a balanced and optimized output that minimizes both false positives and false negatives.

3.9.6 Our Proposed Deployment Architecture

Our proposed deployment architecture presented in Figure 3.9 is a ransomware detection module using a software-defined network (SDN) infrastructure with a web server, Raspberry Pi SDN switches, an SDN controller with sFlow RT module for network traffic monitoring, and three hosts (Client PC, Server PC, and Test Bot) is an innovative approach to enhance network security. This design allows for dynamic monitoring and response to suspicious network activities that could indicate ransomware attacks.

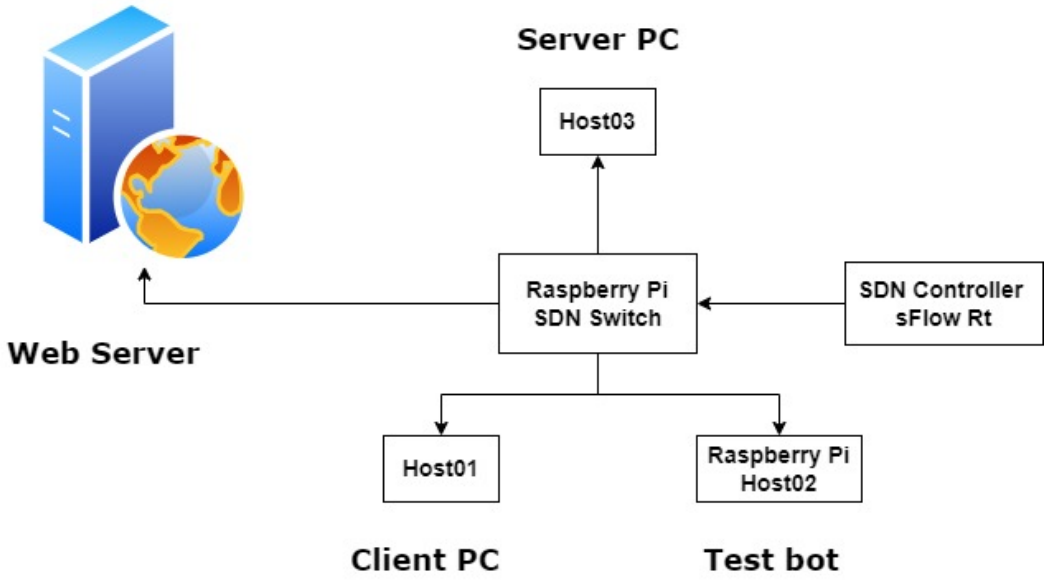

Figure 3.9: Proposed Deployment Architecture

3.9.7 Components and Configuration

Web Server: The web server serves as the endpoint for managing and analyzing network traffic data. It hosts a user interface for controlling the ransomware detection system and displaying real-time insights. It is connected with the SDN switches.

SDN Switches (Raspberry Pi): Raspberry Pi devices are configured as SDN switches to facilitate centralized control of network traffic. It has been chosen for the low cost and ease of installation [14]. They enable efficient routing and forwarding of packets based on instructions received from the SDN controller [17].

SDN Controller with sFlow RT Module: The SDN controller plays a pivotal role in orchestrating the network. The sFlow RT module within the controller is responsible for collecting and analyzing network traffic data in real-time. sFlow is a standard for monitoring network traffic flows, allowing the controller to detect anomalies and patterns.

The SDN Controller with sFlow RT Module leverages sFlow-RT (sFlow Real-Time) for real-time network monitoring and analysis. SFlow is a standard for monitoring network traffic flows, and sFlow-RT is a specific implementation of this stan-

dard. Here's how it's used for monitoring in an SDN environment:

1. **Data Collection:** SFlow-RT continuously collects network traffic data, including flow statistics, packet samples, and other relevant information, from the SDN switches and network devices.

2. **Real-Time Analysis:** The sFlow-RT module within the SDN controller processes this data in real time, enabling immediate analysis of network traffic flows and patterns.

3. **Anomaly Detection:** By monitoring network traffic in real time, the sFlow-RT module can identify anomalies or suspicious activities that may indicate ransomware or other security threats. It can raise alerts or trigger automated responses when unusual patterns are detected.

4. **Traffic Insights:** SFlow-RT provides valuable insights into network traffic behaviour, helping network administrators and security teams understand the normal operation of the network and identify deviations from the norm.

5. **Centralized Control:** The SDN controller can use the information from sFlow-RT to make informed decisions about network traffic routing and resource allocation, improving network efficiency and security.

The integration of the sFlow-RT module with the SDN controller enhances the SDN infrastructure's ability to monitor and respond to network traffic in real time, which is particularly valuable for detecting and mitigating ransomware and other security threats. It helps ensure that network administrators have a clear view of network activity and can take swift actions when anomalies are detected.

Real-time Traffic Analysis with sFlow RT: sFlow is a standard protocol for monitoring network traffic flows. It provides a means to collect and analyze data regarding network communication in real-time [31]. sFlow RT takes this a step

further by offering real-time analytics capabilities. It continually analyzes data flows, allowing for immediate detection of unusual patterns that may indicate ransomware activity [36].

Client PC: One host in the network represents a typical client computer. This host simulates a regular user's network activities and communication.

Server PC: Another host serves as a server, hosting various applications and data. It simulates typical server activities and communication.

Test Bot (Raspberry Pi): The Test Bot is another Raspberry Pi device acting as a testing agent. Every suspected file is installed and ransomware-like network traffic patterns, help test the ransomware detection module's efficacy [39].

3.9.8 Benefits and Expected Outcomes

- **Enhanced Accuracy**: The combination of VAE, ML, DL, and Pareto ensemble ensures a high level of accuracy in ransomware detection.
- **Robustness**: By integrating both ML and DL models, the proposed method can detect a wide range of ransomware variants, including those with evolving behaviours.
- **Balanced Detection**: The Pareto ensemble technique focuses on optimizing the trade-off between false positives and false negatives, resulting in more balanced and reliable detection.
- **Adaptability**: The method can be fine-tuned and updated to adapt to new ransomware threats and evolving network behaviours.

In conclusion, our proposed ransomware detection method represents a holistic and advanced approach to combating ransomware threats. By combining Variational Autoencoder, ML, DL, and the innovative Pareto ensemble technique, we aim to achieve

a high level of accuracy, robustness, and adaptability in identifying ransomware in network traffic data. This approach has the potential to significantly enhance network security and reduce the impact of ransomware attacks.

3.10 Performance Metrics and Evaluation Criteria

Performance metrics and evaluation criteria play an important role in assessing the effectiveness of ransomware detection systems. Here we used these metrics in evaluating the performance of such systems:

Confusion Matrix for Binary Class: The confusion matrix is presented in Table 3.1 which is used to evaluate the performance of a classification model. It provides a comprehensive overview of the model's predictions and the actual values of the target variable. The confusion matrix consists of four key components:

Table 3.1: The confusion matrix for Ransomware and benign

Real \ Predicted	Ransomware (R)	Benign (N)
Ransomware (R)	TP	FN
Benign (N)	FP	TN

Confusion Matrix for Multi class: The multiclass confusion matrix table 3.2 provides a detailed representation of the performance of a multiclass classification model. It visually summarizes the predictions made by the model and compares them against the actual labels of the data.

The confusion matrix consists of a square matrix, where each row represents the true labels of the data for a specific class, and each column represents the predicted labels by the model for that class. Here we have Benign as N and Ransomware classes denoted as R1, R2 and R3. The diagonal elements of the matrix correspond to the correctly classified instances, while the off-diagonal elements represent misclassification according to the table values:

Table 3.2: The confusion matrix for MultiClass Ransomware and Benign

Real \ Predicted	Benign (N)	R1	R2	R3
Benign (N)	TP	FP	TN	TN
R1	FP	TN	FN	FN
R2	FP	FN	TN	TN
R3	TN	TN	TN	TN

True Positive (TP): The number of instances that are correctly predicted as positive (actual positive, predicted positive).

True Negative (TN): The number of instances that are correctly predicted as negative (actual negative, predicted negative).

False Positive (FP): The number of instances that are incorrectly predicted as positive (actual negative, predicted positive).

False Negative (FN): The number of instances that are incorrectly predicted as negative (actual positive, predicted negative).

In our experiment on the multiclass classification we have used the five Ransomware classes as classification, below is the sample table of that classification

Table 3.3: The confusion matrix for Five MultiClass Ransomware and Benign

Real \ Predicted	Benign (N)	R1	R2	R3	R4	R5
Benign (N)	TP	FP	TN	TN	TN	TN
R1	FP	TN	FN	FN	FN	FN
R2	FP	FN	TN	TN	TN	TN
R3	TN	TN	TN	TN	TN	TN
R4	TN	TN	TN	TN	TN	TN
R5	TN	TN	TN	TN	TN	TN

The confusion matrix is typically represented as a square matrix, where the rows represent the actual classes and the columns represent the predicted classes. It provides valuable insights into the model's performance, including measures such as accuracy, precision, recall (sensitivity), and specificity.

Accuracy: It measures the overall correctness of the classification results, repre-

senting the ratio of correctly classified samples to the total number of samples.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TP+FP+FN}} \tag{3.8}$$

Precision: It indicates the proportion of correctly predicted ransomware samples among all the samples predicted as ransomware. It is also known as Positive Predictive Value (PPV). It helps measure the system's ability to minimize false positives.

$$\text{Precision(PPV)} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.9}$$

Recall (Sensitivity): It represents the proportion of correctly predicted ransomware samples out of all the actual ransomware samples. It is also known as True Positive Rate (TPR). It measures the system's ability to detect true positives and minimize false negatives.

$$\text{Recall(TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.10}$$

F1-Score: It combines precision and recall into a single metric that provides a balance between them. It is the harmonic mean of precision and recall, providing an overall evaluation of the system's performance.

$$\text{F1-score} = \frac{\text{2TP}}{\text{2TP+FP+FN}} \tag{3.11}$$

or

$$\text{F1-score} = \frac{\text{2PPV*TPR}}{\text{TPV+TPR}} \tag{3.12}$$

Specificity: It measures the proportion of correctly predicted non-ransomware (benign) samples among all the actual non-ransomware samples. It indicates the system's ability to correctly identify true negatives.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.13}$$

False Positive Rate(FPR): It represents the proportion of non-ransomware samples incorrectly classified as ransomware. It helps assess the system's ability to minimize false alarms.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.14}$$

Macro average: The macro average is a performance evaluation technique used in ransomware detection to assess the overall effectiveness of a model across multiple classes. It provides equal weight to each class, regardless of class distribution or imbalance.

To calculate the macro average first, we compute a performance metric (e.g., precision, recall, F1 score) for each class individually. Then, calculate the average of these individual scores to obtain the macro average.

$$\text{Macro Average} = \frac{1}{n} \sum_{i=1}^{n} \text{Metric}_i \tag{3.15}$$

Where n represents the total number of classes, and Metric_i denotes the performance metric (such as precision, recall, or F1 score) for each class.

The macro average provides a comprehensive assessment of the model's performance across all classes, treating each class equally. It is particularly useful when there is a class imbalance or when we want to evaluate the model's overall effectiveness in detecting different types of ransomware.

Micro average: The micro average is a performance measure used in multi-class classification tasks, including ransomware detection. It calculates the average performance across all classes by considering the total number of true positives, false positives, and false negatives for each class.

$$\text{Micro Average} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{TP}_i + \sum_{i=1}^{n} \text{FP}_i + \sum_{i=1}^{n} \text{FN}_i} \tag{3.16}$$

where n represents the total number of classes. TP_i, FP_i, and FN_i denote the number of true positives, false positives, and false negatives, respectively, for each class i.

It's important to note that the micro average gives equal weight to each instance, regardless of the class. It emphasizes the overall performance rather than the individual class performance

Weighted average : The weighted average is a performance measure used in multi-class classification tasks, including ransomware detection. It calculates the average performance across all classes, considering the class distribution or importance. Each class is assigned a weight based on its frequency or significance, and the weighted average takes into account these weights when calculating the average performance.

$$\text{Weighted Average} = \frac{\sum_{i=1}^{n} (\text{Weight}_i \times \text{Metric}_i)}{\sum_{i=1}^{n} \text{Weight}_i} \tag{3.17}$$

Where n represents the total number of classes. Weight_i represents the weight assigned to class_i, and Metric_i represents the performance metric (such as precision, recall, or F1 score) for class i.

The weights assigned to each class can be based on various factors, such as class imbalance or the importance of certain classes in the context of ransomware detection. These weights should be predefined or determined based on specific considerations.

3.11 Conclusion

The methodology chapter serves as the blueprint for the ransomware detection research, providing a clear path to addressing the research questions. It emphasizes the

fusion of ML, DL, and data-driven techniques, all underpinned by a commitment to ethical research practices. The proposed models and evaluation methods are poised to contribute to the ongoing effort to enhance ransomware detection capabilities in the cybersecurity domain.

Chapter 4

Experimental Results and Analysis

This chapter presents the findings and analysis of the conducted experiments on ransomware detection. It evaluates the performance of ML and DL algorithms in detecting ransomware attacks. The chapter discusses the experimental setup, including the dataset, features, and evaluation metrics. It presents the results in tables and charts, analyzing metrics such as accuracy, precision, recall, F1 score, and specificity. The chapter concludes by summarizing the key findings and providing recommendations for further research.

4.1 Dataset Description

The ISOT ransomware dataset [25] is a comprehensive collection of behaviour data comprising both ransomware samples and benign applications. It consists of 669 ransomware samples, including popular families and variants, obtained from various sources. Additionally, the dataset includes data from 103 benign applications commonly used by Windows users. The dataset's size on disk is 428 GB, making it a substantial resource for ransomware research and analysis. The ransomware and benign samples were analyzed using the Cuckoo sandbox, an open-source software for automated file analysis. Behavior data was gathered by running the samples on a

system equipped with Windows 7 (64-bit). The sandbox architecture involved a host machine for managing the analysis process and an isolated analysis machine. Full Internet access was provided to the analysis machine, while outbound network traffic to the LAN network was restricted to prevent ransomware spread. Table 4.1 shows the number of rows or instances per ransomware family

Table 4.1: Number of instances per ransomware class in the ISOT ransomware dataset

Serial	**Ransomware Family Name**	**Number of Instances**
1	Cerber	621
2	CryptoMix	58
3	CryptoShield	226
4	Crysis	48
5	CTB-Locker	445
6	Flawed	44
7	GlobeImposter	184
8	Jaff	140
9	Locky	9998
10	Mole	191
11	Sage	861
12	Satan	75
13	Spora	383
14	Striked	63
15	TeslaCrypt	33971
16	Unlock26	191
17	WannaCry	49
18	Win32.Blocker	955
19	Xorist	93
20	Zeta	186
21	Petya	90

The dataset directory structure consists of separate subdirectories for each ransomware family and a "Benign" subdirectory for benign applications. Each family subdirectory contains a subdirectory for each sample, identified by a numerical ID. Various behaviour data is collected and stored for each sample, including network traffic dumps, memory dumps (not available for all samples), files metadata in JSON format, raw logs in .bson files, and analysis reports in JSON format. The analysis

reports provide information about the analysis task, memory regions, network connections, processes created by the sample, system calls, extracted strings, file system operations, and Windows registry operations. Table 4.1 represents the 21 different ransomware families found in ISOT ransomware database.

The CICAndMal2017 dataset [29], generated by the Canadian Center for Cyber Security of the University of New Brunswick, includes traffic information for various types of malware (scareware, adware, ransomware, and SMS malware) as well as benign traffic. The dataset was collected from real Android devices to capture the behaviour of malware in an authentic environment. Traffic data was collected in three phases: after app installation, 15 minutes before device reboot, and 15 minutes after device reboot. The ransomware category consists of ten families, including charger, jisut, koler, lockerpin, simplocker, pletor, porndroid, ransomBO, svpeng, and wannalocker. Each instance in the dataset contains 84 features. To focus on ransomware detection, a new dataset was created by merging ransomware traffic with benign traffic. The merged dataset addressed the issue of class imbalance by adding 54,161 instances of benign traffic captured in 2017 and 2016. The final dataset consists of 402,834 samples, including both ransomware and benign traffic. Table 4.2 shows the number of rows or instances per ransomware family.

Table 4.2: Number of instances per ransomware class in the CICAndMal2017 ransomware dataset

Serial	**Ransomware Family Name**	**Number of Instances**
1	Svpeng	54161
2	Porndroid	46082
3	Koler	44555
4	RansomBO	39859
5	Charger	39552
6	Simplocker	36340
7	Wannalocker	32701
8	Jisut	25672
9	Lockerpin	25307
10	Pletor	4715

4.2 Extracting and selecting relevant features

We convert our dataset, which contains 84 traffic features using CICflowmeter [12] [28] into a CSV file. Here for the selection of the features, we use the Recursive Feature Elimination (RFE) algorithm Figure 3.1 and Particle Swarm Optimization (PSO) algorithm 3.2 and the cross method to select the 10 features for the experiment purposes. And these 10 features are the cross-method output of these two algorithms. Our selected 10 features are described in Table 4.3

Table 4.3: List of Selected Features and their Descriptions

Serial	**Feature Name**	**Description**
1	Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
2	Fwd Seg Size Min	Minimum segment size observed in the forward direction
3	Active Mean	Mean time a flow was active before becoming idle
4	Active Std	Standard deviation time a flow was active before becoming idle
5	Active Max	Maximum time a flow was active before becoming idle
6	Active Min	Minimum time a flow was active before becoming idle
7	Idle Mean	Mean time a flow was active before becoming idle
8	Idle Max	Maximum time a flow was idle before becoming active
9	Idle Min	Minimum time a flow was idle before becoming active
10	Idle Std	Standard deviation time a flow was idle before becoming active

4.3 Experimental Setup

The experiments were conducted on a local PC with an 11th Gen Intel(R) Core(TM) i7-11800H processor running at 2.30GHz. The PC equipped with 16.0 GB of RAM

(15.7 GB usable) and operated on a 64-bit operating system with an x64-based processor. The Google Colab platform was utilized for certain experiments and model training tasks. Google Colab provides a cloud-based environment with access to high-performance GPUs, enabling faster computation and training of DL models. The local PC and Google Colab platform operated on different operating systems. The local PC utilized its native operating system, while Google Colab provided a platform-independent environment. The CICflowmeter tool was used for converting .pcap files to .csv format. This tool was installed on a separate machine running Ubuntu 20.04.6 LTS. CICflowmeter facilitates the extraction of flow-based features from network traffic data, which is crucial for subsequent analysis and model training.

4.4 Performance Evaluation of ML Models (XGBoost, KNN, DT, NB, RF)

Table 4.4 presents the experimental results of different ML (ML) classifiers on the ISOT Ransomware Dataset, consisting of 21 ransomware families, with a training set of 70% and a testing set of 30%. Among the classifiers, XGBoost achieved the highest Precision (PPV) of 0.990, True Positive Rate (TPR) of 0.995, F1-score of 0.990, and Accuracy (ACC) of 94.66%. KNN classifier achieved a Precision of 0.970, TPR of 0.990, F1-score of 0.980, and ACC of 93.28%. DT classifier achieved a Precision of 0.964, TPR of 0.977, F1-score of 0.971, and ACC of 92.85%. NB classifier showed lower performance with a Precision of 0.860, TPR of 0.020, F1-score of 0.040, and ACC of 13.73%. RF classifier achieved a Precision of 0.940, TPR of 0.997, F1-score of 0.968, and ACC of 93.98%. These results demonstrate the varying performance of the ML classifiers in detecting and classifying ransomware. XGBoost, KNN, and Decision Tree classifiers showed higher accuracy and precision in identifying ransomware instances. NB had relatively lower performance, while RF demonstrated high True

Positive Rate and Accuracy.

Overall, the results provide insights into the potential of different ML classifiers for accurate ransomware detection.

Table 4.4: Experimental results with different ML classifiers for ISOT Ransomware Dataset

Classifier	PPV	TPR	F1-score	ACC
XGBoost	0.990	0.995	0.990	0.946
KNN	0.970	0.990	0.980	0.932
Decision Tree	0.964	0.977	0.971	0.928
Naive Bayes	0.860	0.020	0.040	0.137
Random Forest	0.940	0.997	0.968	0.939

Table 4.5 presents the experimental results of different ML classifiers trained on the CICAndMal2017 Ransomware Dataset. Analyzing the data in the table, we observe that XGBoost achieves a PPV of 0.830, TPR of 0.900, F1-score of 0.890, and an accuracy of 84.21%. These results indicate that XGBoost performs well in accurately predicting positive instances, achieving a high recall rate and balanced performance. Similarly, KNN demonstrates competitive performance with a PPV of 0.820, TPR of 0.940, F1-score of 0.890, and an accuracy of 80.35%. Decision Tree also yields favorable results with a PPV of 0.825, TPR of 0.940, F1-score of 0.891, and an accuracy of 80.29%.

However, Naive Bayes exhibits relatively lower performance across the metrics, with a PPV of 0.800, TPR of 0.040, F1-score of 0.450, and an accuracy of 79.20%. This indicates limitations in correctly predicting positive instances and overall performance. Finally, Random Forest achieves a PPV of 0.870, TPR of 0.820, F1-score of 0.910, and an accuracy of 82.30%, demonstrating competitive performance.

In summary, based on the results in the table, XGBoost, KNN, Decision Tree, and

Random Forest show promising performance on the CICAndMal2017 Ransomware Dataset, with XGBoost achieving the highest overall performance scores. Naive Bayes, however, exhibits lower performance in accurately predicting positive instances. The table provides valuable insights for selecting an appropriate ML model for ransomware detection, with XGBoost being a potential choice based on its strong performance across the evaluated metrics.

Table 4.5: Experimental results with different ML classifiers for CICAndMal2017 Ransomware Dataset

Classifier	PPV	TPR	F1-score	ACC
XGBoost	0.830	0.900	0.890	0.842
KNN	0.820	0.940	0.890	0.803
Decision Tree	0.825	0.940	0.891	0.802
Naive Bayes	0.800	0.040	0.450	0.792
Random Forest	0.870	0.820	0.910	0.823

4.5 Performance Evaluation of DL Models (LSTM, CNN)

Table 4.6 presents the performance evaluation results of different DL classifiers trained on the ISOT Ransomware Dataset. The classifiers assessed are LSTM and CNN. Analyzing the data in the table, we observe that LSTM achieves a PPV of 0.975, TPR of 0.993, F1-score of 0.984, and an accuracy of 96.38%. These results indicate that LSTM performs well in accurately predicting positive instances, achieving a high recall rate, and maintaining a balance between precision and recall. Similarly, CNN demonstrates excellent performance with a PPV of 0.981, TPR of 0.997, F1-score of 0.997, and an accuracy of 97.85%. These results indicate that CNN excels in correctly predicting positive instances and overall performance, showing a high level

of precision, recall, and accuracy.

In summary, based on the results of the table, both LSTM and CNN show outstanding performance on the ISOT Ransomware Dataset. Both models achieve high precision, recall, F1-score, and accuracy, indicating their effectiveness in ransomware detection. The table provides valuable insights for selecting an appropriate DL model for ransomware detection, with LSTM and CNN being strong candidates based on their superior performance across the evaluated metrics.

Table 4.6: Experimental results with different DL classifiers for ISOT Ransomware Dataset

Classifier	PPV	TPR	F1-score	ACC
LSTM	0.975	0.993	0.984	0.963
CNN	0.981	0.997	0.997	0.978

Table 4.7 presents the performance evaluation results of different DL classifiers applied to the CICAndMal2017 Ransomware Dataset. Analyzing the data in the table, we observe that LSTM achieves a PPV of 0.965, TPR of 0.981, F1-score of 0.960, and an accuracy of 95.80%. These results indicate that LSTM performs well in accurately predicting positive instances, achieving a high recall rate while maintaining a balance between precision and recall. Similarly, CNN demonstrates strong performance with a PPV of 0.971, TPR of 0.987, F1-score of 0.967, and an accuracy of 96.85%. These results suggest that CNN excels in correctly identifying positive instances and exhibits high precision, recall, and accuracy.

In summary, based on the results in the table, both LSTM and CNN showcase impressive performance on the CICAndMal2017 Ransomware Dataset. Both models achieve high precision, recall, F1-score, and accuracy, indicating their effectiveness in ransomware detection. Based on this data CNN is better than LSTM but it mainly depends on the dataset.

Table 4.7: Experimental results with different DL classifiers for CICAndMal2017 Ransomware Dataset

Classifier	PPV	TPR	F1-score	ACC
LSTM	0.965	0.981	0.960	0.958
CNN	0.971	0.987	0.967	0.968

4.6 Performance Evaluation of Pareto Ensemble Learning using Our Proposed Method Performance Evaluation

The performance evaluation for Pareto Ensemble Learning using our proposed method (Our proposed optimized cross-feature selection based on VAE feature extraction method) on the ISOT Ransomware dataset is presented in Table 4.8.

In the context of "RF+XGB+CNN+LSTM+ Proposed Method," the "+" sign typically denotes a combination of multiple ML and DL models. Pareto means combining models for improved predictive performance in ensembles. One model may not always yield the best performance; our investigation revealed that a single model might not be optimal in all cases. So we have to use a combination of different ML and DL to improve ransomware detection by leveraging the strengths of each model type.

The proposed method, incorporating VAE alongside ML and DL models, enhances ransomware detection by integrating an unsupervised VAE for anomaly detection. This approach leverages VAE's ability to learn data distributions and identify deviations from normal behavior in the latent space. By combining these techniques, the system benefits from VAE's adaptability and feature extraction capabilities, providing a comprehensive approach to ransomware detection beyond traditional ML and

DL methods.

The ensemble model consisting of RF, XGB, CNN, LSTM, and our proposed method achieved remarkable performance with a precision of 0.990, indicating that 99.0% of the predicted instances were correctly classified. The true positive rate (recall) and F1-score for this ensemble model were also 0.990, implying that it successfully identified 99.0% of the actual positive instances and achieved a balanced trade-off between precision and recall. The accuracy of this ensemble model was calculated as 97.01%.

Another ensemble model combining CNN, LSTM, and our proposed method demonstrated competitive performance with a precision of 0.970 and a true positive rate of 0.980. The F1-score for this ensemble model was 0.960, indicating a good balance between precision and recall. The accuracy achieved by this ensemble model was 96.98%.

Comparatively, ensemble models without our proposed method, such as DT + RF + XGB and DT + RF, showed slightly lower performance in terms of precision, true positive rate, F1-score, and accuracy. Similarly, the ensemble model NB + RF + DT exhibited lower precision, true positive rate, and F1-score but achieved an accuracy of 87.6%.

Overall, the performance evaluation for Ensemble Learning using our Feature-optimized method highlights the effectiveness of the ensemble models in accurately classifying instances in the ISOT Ransomware dataset. The ensemble model consisting of RF, XGB, CNN, LSTM, and our proposed method achieved the highest performance, showcasing its potential for improving the classification accuracy in ransomware detection tasks.

Table 4.8: Experimental results of ensemble learning for ISOT Ransomware Dataset

Classifier	PPV	TPR	F1-score	ACC
RF+XGB+CNN+LSTM+ Proposed Method	**0.990**	**0.990**	**0.990**	**0.970**
CNN+LSTM+Our Proposed Method	0.970	0.980	0.975	0.969
DT + RF + XGB	0.974	0.966	0.969	0.930
DT + RF	0.957	0.955	0.956	0.904
NB+RF+DT	0.987	0.766	0.862	0.876
RF+XGB+DT	0.879	0.876	0.877	0.879

The Performance Evaluation for Ensemble Learning using our Feature optimized method on the CICAndMal2017 Ransomware Dataset is presented in Table 4.9. The evaluation includes metrics such as PPV, TPR, F1-score, and Accuracy (ACC).

The ensemble model consisting of RF, XGB, CNN, LSTM, and our proposed method achieved significant performance with a precision of 0.980, indicating that 98.0% of the predicted instances were correctly classified. The true positive rate (recall) and F1-score for this ensemble model were also 0.980, implying that it successfully identified 98.0% of the actual positive instances and achieved a balanced trade-off between precision and recall. The accuracy of this ensemble model was calculated as 97.91%.

Another ensemble model combining CNN, LSTM, and our proposed method demonstrated competitive performance with precision, true positive rate, and F1-score of 0.970. This indicates that the ensemble model achieved high accuracy in identifying positive instances. The accuracy achieved by this ensemble model was 93.98%.

Comparatively, ensemble models without our proposed method, such as DT + RF + XGB and DT + RF, showed slightly lower performance in terms of precision, true positive rate, F1-score, and accuracy. The ensemble model NB + RF + DT exhibited lower precision, true positive rate, and F1 score, achieving an accuracy of 77.6%.

Overall, the Performance Evaluation for Ensemble Learning using our Feature-optimized method highlights the effectiveness of the ensemble models in accurately classifying instances in the CICAndMal2017 Ransomware Dataset. The ensemble model consisting of CNN, LSTM, and our proposed method achieved the highest performance, showcasing its potential for improving the classification accuracy in ransomware detection tasks in this dataset.

Table 4.9: Experimental results of ensemble learning for CICAndMal2017 Ransomware Dataset

Classifier	PPV	TPR	F1-score	ACC
RF+XGB+CNN+LSTM+ Proposed Method	**0.980**	**0.980**	**0.980**	**0.979**
CNN+LSTM+Our Proposed Method	0.970	0.970	0.970	0.939
DT + RF + XGB	0.934	0.936	0.935	0.920
DT + RF	0.927	0.955	0.941	0.944
NB+RF+DT	0.847	0.766	0.804	0.776
RF+XGB+DT	0.869	0.836	0.852	0.839

4.7 Performance Evaluation for Multiclass classification

The performance evaluation for the multiclass classification task using LSTM on the ISOT Ransomware Dataset is summarized in Table 4.10. The evaluation is based on various metrics including Precision (PPV), True Positive Rate (TPR), and F1-score.

For the "Benign(N)" class, the model achieved a precision of 0.902, indicating that 90.2% of instances predicted as "Benign(N)" were actually labelled correctly. The true positive rate (recall) for this class was 0.970, indicating that the model successfully identified 97% of the actual "Benign(N)" instances. The F1-score, which combines precision and recall, was calculated as 0.935.

Similarly, for the ransomware classes Cerber, TeslaCrypt, Win32 Blocker, Sage, and Locky (R1 to R5), the model showed varying levels of performance. For instance, the precision for class "R1" was 0.762, indicating that 76.2% of the predicted instances were correctly classified as "R1." The true positive rate for "R1" was 0.952, implying that the model successfully identified 95.2% of the actual "R1" instances. The F1-score for this class was calculated as 0.846.

For the other ransomware classes (R2, R3, R4, and R5), the model demonstrated varying levels of precision, true positive rate, and F1-score.

Overall, this performance evaluation provides insights into the model's ability to classify instances across multiple classes. It highlights the strengths and weaknesses of the LSTM model in accurately identifying different classes within the ISOT Ransomware Dataset.

Table 4.10: Experimental results with LSTM for Multiclass classification for ISOT Ransomware Dataset

Classification	PPV	TPR	F1-score
Benign(N)	0.902	0.970	0.935
R1	0.762	0.952	0.846
R2	0.977	0.934	0.955
R3	0.969	0.805	0.879
R4	0.897	0.960	0.927
R5	0.884	0.718	0.793

The Performance Evaluation for Multiclass classification using LSTM on the CICAndMal2017 Ransomware Dataset is presented in Table 4.11. The evaluation includes various metrics such as Precision (PPV), True Positive Rate (TPR), and F1-score.

For the "Benign(N)" class, the LSTM model achieved a precision of 0.936, in-

dicating that 93.6% of instances predicted as "Benign(N)" were correctly classified. The true positive rate (recall) for this class was 0.986, implying that the model successfully identified 98.6% of the actual "Benign(N)" instances. The F1-score, which combines precision and recall, was calculated as 0.961.

Regarding the ransomware classes Charger, Jisut, Koler, LockerPin, Pletor (R1 to R5), the LSTM model exhibited varying levels of performance. For example, the precision for class "R1" was 0.746, indicating that 74.6% of the predicted instances were correctly classified as "R1". The true positive rate for "R1" was 0.502, suggesting that the model identified 50.2% of the actual "R1" instances. The F1-score for this class was calculated as 0.744.

Similarly, the model demonstrated different precision, true positive rate, and F1-score for the other ransomware classes (R2, R3, R4, and R5).

Overall, the performance evaluation highlights the LSTM model's effectiveness in classifying instances across multiple classes in the CICAndMal2017 Ransomware Dataset. It provides insights into the model's strengths and weaknesses in accurately identifying different classes, thereby aiding in understanding its performance for multiclass classification.

Table 4.11: Experimental results with LSTM for Multiclass classification for CICAndMal2017 Ransomware Data-set

Classification	PPV	TPR	F1-score
Benign(N)	0.936	0.986	0.961
R1	0.746	0.502	0.744
R2	0.683	0.700	0.891
R3	0.800	0.540	0.780
R4	0.840	0.820	0.870
R5	0.929	0.830	0.882

4.8 Comparative Analysis of Results

Table 4.12 provides a comparison of different approaches to ransomware detection, with a focus on the number of features used, the feature approach, and the resulting accuracy.

Table 4.12: Comparison of Our Proposed Method and other works based on the number of features

Approach	Number of features	Feature approach	Accuracy
G Cusack et al [10]	28	Network	87%
Hossain, Md Sakir, et al. [19]	26	Network	81.58%
L. Wang et al. [46]	88	Host	93.6%
Lashkari, Arash Habibi, et al. [29]	9	Network	88.00%
Deepa, K., et al. [11]	30	Host	88.75%
Wen, Long, et al. [48]	26	Host	95.2%
Babaagba, et al [9]	22	Host	77.18%
Noorbehbahani, Fakhroddin, et al [33]	84	Network	88%
Proposed Method	**10**	**Network**	**97.05%**

Now, some specific entries from the table 4.12

G Cusack et al. [10]: This approach employs 28 features related to the network. It achieved an accuracy of 87%.

Hossain, Md Sakir, et al. [19]: This method utilizes 26 network-related features and achieved an accuracy of 81.58%.

L. Wang et al. [46]: Wang et al. use 88 features related to the host, resulting in an accuracy of 93.6%.

Proposed Method (highlighted in bold): The proposed method uses 13 network-related features and achieved an impressive accuracy of 97.05%. This indicates that it uses a relatively smaller number of features compared to some other methods while

maintaining a high level of accuracy.

The table offers a clear comparison of different ransomware detection methods based on the number of features they use and the type of features employed. It shows that the "Proposed Method"stands out with a relatively small number of features and a high accuracy rate, signifying the efficiency and effectiveness of this approach in detecting ransomware.

4.9 Discussion of Findings

This section presents a comprehensive analysis of ransomware detection methodologies, highlighting the strengths and weaknesses of different approaches and providing valuable insights into the effectiveness of the proposed method. The discussion is structured around key themes and research objectives.

Performance Comparison: In this study, a comparative analysis was conducted to evaluate the performance of the proposed ransomware detection method in relation to existing approaches. The results demonstrate that the "Proposed Method" outperforms several other methods in terms of key performance metrics such as True Positive Rate (TPR), False Positive Rate (FPR), and overall accuracy. With a TPR of 99.00% and an FPR of 2.1%, the proposed method excels in correctly identifying ransomware instances while minimizing false alarms. The accuracy of 97.05% indicates a high degree of correctness in classification. This performance is particularly remarkable, considering the relatively small number of features (10) used in the proposed method, highlighting its efficiency and efficacy.

Feature Engineering and Selection: The number of features used for ransomware detection is a critical factor influencing model performance. The findings reveal that the proposed method, which primarily relies on network-related features, achieves exceptional accuracy with a limited feature set. This underscores the signif-

icance of feature engineering and selection in the ransomware detection process. By focusing on a subset of relevant features, the proposed method minimizes computational overhead while maintaining a high detection rate. The choice of network-related features aligns with the dynamic nature of ransomware behaviors that often manifest in network traffic patterns.

Implications for Cybersecurity: The findings of this research have broader implications for the field of cybersecurity. The demonstrated performance of the proposed method, especially in early detection and efficient feature utilization, suggests its potential for practical deployment in security systems. Moreover, the integration of SDN in ransomware detection has the potential to enhance network security and contribute to the proactive defense against emerging threats.

Future Research Directions: While the proposed method shows promise, there are areas for further research and improvement. Future studies can explore the scalability and adaptability of the method to diverse network environments. Additionally, research on the development of hybrid models that combine the strengths of various detection approaches is a promising avenue for enhancing ransomware detection capabilities.

4.10 Conclusion

This thesis demonstrates the effectiveness of the proposed ransomware detection method, particularly in achieving high accuracy, early detection, and efficient feature utilization. These findings contribute to the ongoing efforts to strengthen cybersecurity measures against ransomware threats and lay the foundation for future research and practical implementations in the field.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, this thesis focused on the detection and mitigation of ransomware attacks using SDN, machine learning, and DL techniques. The research aimed to address the growing threat of ransomware attacks and develop effective strategies to detect and mitigate them. Through the implementation and evaluation of various algorithms and models, the study achieved significant progress in enhancing ransomware defence capabilities.

Our proposed deployment architecture demonstrated the efficacy of utilizing SDN, s-Flow RT, ML, and DL in detecting and mitigating ransomware attacks. The combination of SDN's centralized control and programmability with the intelligent capabilities of machine learning and DL algorithms proved to be a promising approach. The developed models exhibited high accuracy in identifying ransomware activities, enabling prompt response and mitigation actions.

Moreover, In the deployment of our system is the integration of SDN provided enhanced network visibility and control, facilitating rapid containment and isolation of infected devices. By leveraging the dynamic nature of SDN, the response to ran-

somware attacks was more efficient and tailored to the specific needs of the network environment.

Overall, this thesis significantly advanced the field of ransomware detection and mitigation by leveraging the capabilities of SDN, machine learning, and DL. The findings and methodologies presented here have practical implications for enhancing cybersecurity measures and protecting critical systems and data from ransomware threats.

5.2 Limitations and Challenges

One of the primary limitations is the availability of diverse and representative datasets for training and evaluation. Obtaining large-scale, real-world ransomware datasets with comprehensive attack scenarios is challenging due to the sensitive and malicious nature of the data.

The generalizability of the developed models to different network architectures, configurations, and attack scenarios also needs further investigation. Ensuring the effectiveness of the proposed approach in various real-world settings is crucial for practical applicability.

As ransomware attackers continuously evolve their techniques, the adaptability of the developed models becomes important. Regular updates and adaptations are necessary to address new ransomware variants and attack vectors. Ongoing research and monitoring of emerging ransomware trends are essential to maintaining the effectiveness of defense mechanisms.

The practical deployment of SDN-based ransomware detection and mitigation systems in large-scale production networks may face implementation challenges. Network compatibility, scalability, and performance considerations need to be carefully addressed to ensure successful deployment.

Ethical considerations arise from the use of real ransomware samples and conducting experiments involving live attacks. Responsible research practices and compliance with privacy and security regulations are necessary to address these concerns.

5.3 Future Research Directions

To overcome these limitations and challenges, further research and collaboration among academia, industry, and policymakers are required. The acquisition of diverse datasets, investigation of generalizability, continuous adaptation to evolving techniques, addressing deployment challenges, and adherence to ethical guidelines are crucial for advancing the field of ransomware detection and mitigation using SDN, machine learning, and DL techniques. By addressing these issues, the effectiveness of ransomware defence mechanisms can be improved, leading to better protection of critical infrastructures. In our upcoming research steps, we have outlined the following plan for the execution at the experimental level.

Early Detection Capability: Early detection of ransomware is a key objective in the realm of cybersecurity. The study showcases that the "Proposed Method" excels in Early Detection (ED). The ability to identify ransomware at an early stage is crucial for mitigating potential damage and loss. This finding underscores the practical applicability of the proposed method in real-world scenarios where rapid response to ransomware threats is essential.

SDN Integration with sFlow Rt: The use of SDN for network traffic monitoring is another significant aspect of the proposed method. The findings highlight the advantages of integrating SDN into the ransomware detection process. SDN facilitates dynamic and centralized control of network traffic, enabling enhanced monitoring, rapid threat response, and adaptability to evolving ransomware tactics. While the results have not been achieved yet, we anticipate that they will demonstrate the

effective utilization of SDN in enhancing network traffic analysis as per the proposed method.

To create a comprehensive dataset: One of our future research plans is also to contribute to the creation of a comprehensive dataset that captures various ransomware behaviours and benign network traffic. This dataset will serve as a valuable resource for training and evaluating the developed models, ensuring their effectiveness in real-world scenarios.

REFERENCES

[1] Rakshit Agrawal, Jack W Stokes, Karthik Selvaraj, and Mady Marinescu. Attention in recurrent neural networks for ransomware detection. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 3222–3226. IEEE, 2019.

[2] Tariq Ahamed Ahanger, Usman Tariq, Fadl Dahan, Shafique A Chaudhry, and Yasir Malik. Securing iot devices running pureos from ransomware attacks: Leveraging hybrid machine learning techniques. *Mathematics*, 11(11):2481, 2023.

[3] Yahye Abukar Ahmed, Baris Kocer, and Bander Ali Saleh Al-rimy. Automated analysis approach for the detection of high survivable ransomware. *KSII Transactions on Internet and Information Systems (TIIS)*, 14(5):2236–2257, 2020.

[4] Maxat Akbanov, Vassilios G Vassilakis, and Michael D Logothetis. Ransomware detection and mitigation using software-defined networking: The case of wannacry. *Computers & Electrical Engineering*, 76:111–121, 2019.

[5] Iman Almomani, Aala AlKhayer, and Walid El-Shafai. Novel ransomware hiding model using hevc steganography approach. *CMC Comput. Mater. Contin*, 70(2):1209–1228, 2021.

[6] Fahad M Alotaibi and Vassilios G Vassilakis. Sdn-based detection of self-propagating ransomware: the case of badrabbit. *IEEE Access*, 9:28039–28058, 2021.

[7] Maram Alsharif and Danda B Rawat. Machine learning enabled intrusion detection for edge devices in the internet of things. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0361–0367. IEEE, 2023.

[8] Femi Emmanuel Ayo, Joseph Bamidele Awotunde, Sakinat Oluwabukonla Folorunso, Matthew O Adigun, and Sunday Adeola Ajagbe. A genomic rule-based knn model for fast flux botnet detection. *Egyptian Informatics Journal*, 24(2):313–325, 2023.

[9] Kehinde Oluwatoyin Babaagba and Samuel Olumide Adesanya. A study on the effect of feature selection on malware analysis using machine learning. In *Proceedings of the 2019 8th international conference on educational and information technology*, pages 51–55, 2019.

[10] Greg Cusack, Oliver Michel, and Eric Keller. Machine learning-based detection of ransomware using sdn. In *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 1–6, 2018.

[11] K Deepa, G Radhamani, and P Vinod. Investigation of feature selection methods for android malware analysis. *Procedia Computer Science*, 46:841–848, 2015.

[12] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414, 2016.

[13] Archit Manishbhai Gajjar et al. Ransomware detection with xgboost hardware acceleration for data centers using high-level synthesis. 2023.

[14] David González, Carlos Mellado, Kevin Waltam, and Adrian Lara. Low-cost sdn switch comparison: Zodiac fx and raspberry pi. In *2019 IV Jornadas Costarricenses de Investigación en Computación e Informática (JoCICI)*, pages 1–5. IEEE, 2019.

[15] M Gopinath and Sibi Chakkaravarthy Sethuraman. A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, 47:100529, 2023.

[16] Sibel Gulmez, Arzu Gorgulu Kakisim, and Ibrahim Sogukpinar. Analysis of the dynamic features on ransomware detection using deep learning-based methods. In *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6. IEEE, 2023.

[17] Vipin Gupta, Karamjeet Kaur, and Sukhveer Kaur. Developing small size low-cost software-defined networking switch using raspberry pi. In *Next-Generation Networks: Proceedings of CSI-2015*, pages 147–152. Springer, 2018.

[18] Juan A Herrera-Silva and Myriam Hernández-Álvarez. Dynamic feature dataset for ransomware detection using machine learning algorithms. *Sensors*, 23(3):1053, 2023.

[19] Md Sakir Hossain, Naim Hasan, Md Abdus Samad, Hossain Md Shakhawat, Joydeep Karmoker, Foysol Ahmed, KFM Nafiz Fuad, and Kwonhue Choi. Android ransomware detection from traffic analysis using metaheuristic feature selection. *IEEE Access*, 10:128754–128763, 2022.

[20] Chia-Ming Hsu, Chia-Cheng Yang, Han-Hsuan Cheng, Paul E Setiasabda, and Jenq-Shiou Leu. Enhancing file entropy analysis to improve machine learning detection rate of ransomware. *IEEE Access*, 9:138345–138351, 2021.

[21] Mamoona Humayun, NZ Jhanjhi, Ahmed Alsayat, and Vasaki Ponnusamy. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1):105–117, 2021.

[22] Fatma Abd-Alhaleem Hussain and Dalia Nashat. Time series similarity for detecting ddos flooding attack. *Assiut University Journal of Multidisciplinary Scientific Research*, 51(3):229–241, 2022.

[23] Corvus Insurance. Ransomware frequency over time, 2023. Image source: Corvus Insurance.

[24] M Izham Jaya and Mohd Faizal Ab Razak. Dynamic ransomware detection for windows platform using machine learning classifiers. *JOIV: International Journal on Informatics Visualization*, 6(2-2):469–474, 2022.

[25] Brijesh Jethva, Issa Traoré, Asem Ghaleb, Karim Ganame, and Sherif Ahmed. Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring. *Journal of Computer Security*, 28(3):337–373, 2020.

[26] Adhirath Kapoor, Ankur Gupta, Rajesh Gupta, Sudeep Tanwar, Gulshan Sharma, and Innocent E Davidson. Ransomware detection, avoidance, and mitigation scheme: a review and future directions. *Sustainability*, 14(1):8, 2021.

[27] Ban Mohammed Khammas. Ransomware detection using random forest technique. *ICT Express*, 6(4):325–331, 2020.

[28] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, Ali A Ghorbani, et al. Characterization of tor traffic using time based features. In *ICISSp*, pages 253–262, 2017.

[29] Arash Habibi Lashkari, Andi Fitriah A Kadir, Laya Taheri, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–7. IEEE, 2018.

[30] Yassine Lemmou, Jean-Louis Lanet, and El Mamoun Souidi. In-depth analysis of ransom note files. *Computers*, 10(11):145, 2021.

[31] Shijin Liu, Hiroaki Fukuda, and Paul Leger. Real-time ddos attack defense system in sdn using lssom. In *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 69–73. IEEE, 2023.

[32] Mohammad Masum, Md Jobair Hossain Faruk, Hossain Shahriar, Kai Qian, Dan Lo, and Muhaiminul Islam Adnan. Ransomware classification and detection with machine learning algorithms. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0316–0322. IEEE, 2022.

[33] Fakhroddin Noorbehbahani and Mohammad Saberi. Ransomware detection with semi-supervised learning. In *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 024–029. IEEE, 2020.

[34] Muhammad Safwan Rosli, Raihana Syahirah Abdullah, Warusia Yassin, MA Faizal, and Wan Nur Fatihah Wan Mohd Zaki. Ransomware behavior attack construction via graph theory approach. *International Journal of Advanced Computer Science and Applications*, 11(2), 2020.

[35] Elpida Rouka, Celyn Birkinshaw, and Vassilios G Vassilakis. Sdn-based malware detection and mitigation: The case of expetr ransomware. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 150–155. IEEE, 2020.

[36] Nadhir Fachrul Rozam and Mardhani Riasetiawan. Xgboost classifier for ddos attack detection in software defined network using sflow protocol. *International Journal on Advanced Science, Engineering & Information Technology*, 13(2), 2023.

[37] Nitin Sharma and AL Sangal. Machine learning approaches for analysing static features in android malware detection. In *2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 93–96. IEEE, 2023.

[38] Pradeep Kumar Sharma and SS Tyagi. Improving security through software defined networking (sdn): an sdn based model. *Int. J. Recent Technol. Eng*, 8(4):295–300, 2019.

[39] Adel Nadjaran Toosi, Jungmin Son, and Rajkumar Buyya. Clouds-pi: A low-cost raspberry-pi based micro data center for software-defined cloud computing. *IEEE Cloud Computing*, 5(5):81–91, 2018.

[40] Konstantinos Tsiknas, Dimitrios Taketzis, Konstantinos Demertzis, and Charalabos Skianis. Cyber threats to industrial iot: a survey on attacks and countermeasures. *IoT*, 2(1):163–186, 2021.

[41] Rusydi Umar, Imam Riadi, and Ridho Surya Kusuma. Mitigating sodinokibi ransomware attack on cloud network using software-defined networking (sdn). *International Journal of Safety and Security Engineering*, 11(3):239–246, 2021.

[42] Umara Urooj, Bander Ali Saleh Al-rimy, Anazida Zainal, Fuad A Ghaleb, and Murad A Rassam. Ransomware detection using the dynamic analysis and machine learning: A survey and research directions. *Applied Sciences*, 12(1):172, 2022.

[43] Veeam. 2023 executive summary ransomware trends. https://www.veeam.com/analyst-reports/ransomware-trends-executive-summary-na$_w$*pp.pdf*, 2023. *Accessed* : [6/11/2023].

[44] Aldin Vehabovic, Hadi Zanddizari, Nasir Ghani, Farooq Shaikh, Elias Bou-Harb, Morteza Safaei Pour, and Jorge Crichigno. Data-centric machine learning approach for early ransomware detection and attribution. *arXiv preprint arXiv:2305.13287*, 2023.

[45] Marco Venturini, Francesco Freda, Emanuele Miotto, Alberto Giaretta, and Mauro Conti. Differential area analysis for ransomware: Attacks, countermeasures, and limitations. *arXiv preprint arXiv:2303.17351*, 2023.

[46] Le Wang, Yuelin Gao, Shanshan Gao, and Xin Yong. A new feature selection method based on a self-variant genetic algorithm applied to android malware detection. *Symmetry*, 13(7):1290, 2021.

[47] Azka Wani and S Revathi. Ransomware protection in lot using software defined networking. *Int. J. Electr. Comput. Eng*, 10(3):3166–3175, 2020.

[48] Long Wen and Haiyang Yu. An android malware detection system based on machine learning. In *AIP conference proceedings*, volume 1864. AIP Publishing, 2017.

[49] Tianrou Xia, Yuanyi Sun, Sencun Zhu, Zeeshan Rasheed, and Khurram Shafique. Toward a network-assisted approach for effective ransomware detection. *arXiv preprint arXiv:2008.12428*, 2020.

[50] Lei Yan, Maode Ma, Dandan Li, Xiaohong Huang, Yan Ma, and Kun Xie. Certrust: An sdn-based framework for the trust of certificates against crossfire attacks in iot scenarios. *CMES-Computer Modeling in Engineering & Sciences*, 134(3):2137–2162, 2023.

[51] Aaron Zimba, Zhaoshun Wang, Hongsong Chen, and Mwenge Mulenga. Recent advances in cryptovirology: State-of-the-art crypto mining and crypto ransomware attacks. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(6):3258–3279, 2019.