Detection of Crypto-Ransomware Attack Using Deep Learning

Muna Jemal

Detection of Crypto-Ransomware Attack Using Deep Learning

A Thesis Presented to

The Faculty of the Computer Science Department

by

Muna Jemal

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

May 2023

Detection of Crypto-Ransomware Attack Using Deep Learning

Approved:

Dr. Dan Lo – Advisor

Dr. Yong Pei – Department Chair

Dr. Sumanth Yenduri – Dean

Your Name

Notice To Borrowers

The author of this thesis is:

Muna Jemal
3021 Hidden Forest Ct
Marietta, Georgia, 30066

The director of this thesis is:

Advisor Name
Office address,
City, State, Zip code

Detection of Crypto-Ransomware Attack Using Deep Learning

An Abstract of

A Thesis Presented to

The Faculty of the Computer Science Department

by

Muna Jemal

Bachelor of Science in Computer Science, Kennesaw State University, 2021

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

May 2023

Abstract

The number one threat to the digital world is the exponential increase in ransomware attacks. Ransomware is malware that prevents victims from accessing their resources by locking or encrypting the data until a ransom is paid. With individuals and businesses growing dependencies on technology and the Internet, researchers in the cyber security field are looking for different measures to prevent malicious attackers from having a successful campaign. A new ransomware variant is being introduced daily, thus behavior-based analysis of detecting ransomware attacks is more effective than the traditional static analysis. This paper proposes a multi-variant classification to detect ransomware I/O operations from benign applications. The deep learning models implemented in the proposed approach are Bi-directional Long Short-Term Memory (Bi-LSTM) and Convolutional Neural Networks (CNN). The deep learning models are compared against a classic machine learning model such as Logistic Regression (LR), Support Vector Machine (SVM), and Random Forest (RF). The ransomware samples contain 70 binaries from 30 different ransomware extracted during the encryption of an extensive network shared directory. The benign samples came from network traffic traces recorded in a campus LAN where staff users access files from shared servers. A sample contains I/O operations (short Control Commands, bytes being read, and written) per second over a period of T seconds. The proposed deep learning models are tested with Zero-day ransomware samples as well. Both Bi-LSTM and CNN achieved above 98% in accurately classifying ransomware and benign samples.

Detection of Ransomware Attack Using Deep Learning

A Thesis Presented to

The Faculty of the Computer Science Department

by

Muna Jemal

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Advisor: Dr. Dan Lo

Kennesaw State University

May 2023

Acknowledgment

Foremost, I would like to give thanks to my supervisor and advisor Dr. Dan Lo for his guidance and encouragement in partaking in this research.

I would like to also give thanks to my parents for supporting and providing for me to pursue my education in the United States of America. My siblings and friends also played a key role in being there for me emotionally to keep pursuing my dreams

Contents

1 Introduction **1**

1.1 Types of Ransomware Attacks 2

1.2 Prevention and Recovery 3

1.3 Types of Ransomware Analysis 4

1.4 Research Question 5

2 Related Works **6**

2.1 Static Analysis 6

2.2 Dynamic Analysis 9

3 Proposed Approach and Architecture **12**

3.1 Classic Machine Learning Models 12

3.1.1 Logistic Regression 12

3.1.2 Support Vector Machine 13

3.1.3 Random Forest 14

3.2 Proposed Deep Learning Models 15

3.2.1 Convolutional Neural Network 15

3.2.2 Bi-directional LSTM 17

4 Experiments **19**

4.1 Dataset And Feature Selection 19

4.2 Evaluation Metrics 22

5 Results and Analyses 24

5.1 Testing Results and Comparison 24

6 Conclusion and Future Work 32

Bibliography 34

List of Figures

3.1 Visualization of SVM 13

3.2 Visualization of RF 14

3.3 Visualization of CNN layers. 16

3.4 CNN model summary for T = 20s. 17

3.5 Visualization of Bi-LSTM 18

3.6 Bi-LSTM model summary for T = 20s. 18

4.1 Visualization of mirroring network traffic 20

4.2 I/O of Maze ransomware 20

4.3 I/O of TeslaCrypt ransomware 21

4.4 Visualization of I/O features in T seconds 22

5.1 Confusion Matrix for T=10s 25

5.2 Confusion Matrix for T=20s 26

5.3 Confusion Matrix for T=30s 27

5.4 Confusion Matrix for T=40s 28

5.5 Confusion Matrix for T=50s 29

5.6 Confusion Matrix for T=60s 30

List of Tables

4.1 Dataset distribution of ransomware/benign samples for T seconds. 22

5.1 Performance Metric for T= 10s 25

5.2 Performance Metric for T= 20s 26

5.3 Performance Metric for T= 30s 27

5.4 Performance Metrics for T=40s 28

5.5 Performance Metrics for T=50s 29

5.6 Performance Metrics for T=60s 30

Chapter 1

Introduction

Ransomware is a type of malware that prevents victims from accessing their own resources by locking the Operating System (OS) or encrypting their data until a ransom is paid. The ransom is usually demanded to be paid in untraceable forms of payment or decentralized digital currency like Bitcoin. Ransomware infects a victim's device by various means ranging from email phishing to brute force attacks (Cventicanin, 2023). The first known ransomware attack was in 1989, before the age of the internet, using a floppy disk, called AIDS Trojan (Humayun, Jhanjhi, Alsayat, & Ponnusamy, 2021). Early ransomware attacks required external devices to be connected to the victim's device; however, malicious attackers have now progressed to the usage of the internet to expand their attacks. Once the use of the internet expanded, the dependency of individuals and businesses on technology increased exponentially, and so have ransomware attacks (Richardson & North, 2017). In an article by Cyber Magazine, they estimate that ransomware attacks occur about every 11 seconds, which is about 3 million unique attacks a year (Savage, Coogan, & Lau, 2015). The reason for such a high amount of ransomware attacks is due to groups of people with no programming experience operating these attacks through Ransomware as a Service (RaaS). RaaS is a business model in which ransomware operators charge affiliates by developing and creating the ransomware code, while affiliates conduct the ransomware attacks that the operators have developed (S. H. Kok, Abdullah, Jhanjhi, & Supramaniam, 2019).

In recent years, ransomware attacks have been targeting bigger businesses like healthcare entities, governmental organizations, industrial companies, and so on. Since these big organizations have high-value data, once they are breached, ransomware groups can demand higher ransom payments. The impact of ransomware attacks is felt at a higher scale for big institutions; 66% of organizations reported a significant loss of revenue, 55% of organizations had their brand and reputation damaged, and 29% reported being forced to lay off employees due to financial pressures following a ransomware attack (Freed, 2021). On 7 May 2021, Colonial Pipeline, an oil pipeline system, experienced one of the most disruptive digital ransomware attacks by a group called Darkside (also operates as a RaaS) (Alqahtani & Sheldon, 2022). The attackers stole 100 GB of data within two hours and infected the Colonial Pipeline IT network. The hacker group DarkSide was paid 75 Bitcoin (equivalent to 4.4 million USD at the time) by Colonial Pipeline to get the decryption key in order to restore the disruption. This attack disrupted the shipping and airline industry of approximately 5,500 miles of distribution network. As a result, these and other occurrences made ransomware a national security threat, leading the United States Department of Justice (US DOJ) to classify such attacks as terrorist attacks. Thus, a ransomware attack is a serious threat to the digital era and it is critical to find measures to prevent such attacks.

1.1 Types of Ransomware Attacks

The two main forms of ransomware attacks are Locker ransomware and Crypto ransomware. Locker ransomware is a type of ransomware that locks computers or other devices and prevents the victim from using them. Even if the malware is difficult to remove, the data can usually be recovered by transferring the storage device to another working device, which makes this type of malware less effective for attackers. Crypto ransomware, on the other hand, encrypts data so that it cannot be accessed even if the malware is removed from the computer or the storage media is relocated to another device (Savage et al., 2015). Double-extortion is a more recent and popular type of ransomware attack where attackers steal the victim's data before encrypting it and demand a second ransom by threatening to release the data on the Internet (Kerns, Payne, & Abegaz, 2022).

In this paper, ransomware refers to Crypto ransomware unless specified otherwise. As malicious groups are creating new strains of ransomware each day, they generally follow a similar approach to execute such attacks in the steps below :

1. Find vulnerabilities to infiltrate a targeted victim.
2. Collect information about the victim's system.
3. Distribute infection vector.
4. Install ransomware code.
5. Retrieve/generate encryption key.
6. Access/steal victim's files to encrypt.
7. Encrypt the victim's data and make it inaccessible.
8. Demand ransom in exchange for the victim to have access to their data.

Once a malicious group has successfully taken control of a victim's system and encrypted all their data, in most cases it is too late for the victim to gain back control without paying the ransom or at all. Thus, security professionals and researchers are looking for ways to detect, prevent and mitigate ransomware attacks before victims completely lose access to their data.

1.2 Prevention and Recovery

To reduce the risk of ransomware attacks, there must be security measures to prevent access to a system by unauthorized groups. This could be done by having a regular backup, having the latest software with better security, not opening links from unknown sources, protecting all devices and networks with firewalls, and educating individuals on common patterns of ransomware attacks. Big institutions have a dedicated security team to monitor all devices and their whole network system. Even with all these security measures, malicious groups are committed to finding new ways to infiltrate a victim's system. Thus, having a detection system set up in every step before all victim's data is encrypted (mentioned in Section 1.1 before Step 7).

Later sections will discuss how to detect ransomware attacks on a network-shared file system that is similar to a corporate setting. Once ransomware is detected, all infected devices must be disconnected from the network to prevent it from spreading to other devices in the network. Berrueta, Morato, Magaña, and Izal (2018) proposed the architectural design of a network-shared file system that can recover more than 5GB of files encrypted by 48 out of 54 ransomware samples detected promptly.

1.3 Types of Ransomware Analysis

In the cybersecurity field, ransomware analysis can be broadly classified into two types: static analysis and dynamic analysis. Static analysis, also known as signature-based methods, involves analyzing a malicious file without executing it. However, due to the increase in ransomware variants and anti-forensic techniques like packing and obfuscation, these methods have limitations (Urooj, Al-rimy, Zainal, Ghaleb, & Rassam, 2021). In contrast, dynamic analysis, also known as behavior-based approaches, involves running a malicious program and observing its behavior within the system. This method provides a more comprehensive understanding of ransomware behavior by observing its actions in real-time. Signature-based analysis can be used to detect ransomware attacks until the installation of ransomware code (until Step 4 mentioned in Section 1.1). Hybrid Analysis is a multi-level ransomware analysis using both signature-based and behavioral-based approaches.

Dynamic analysis can be used to determine how ransomware behaves. This is done by observing how applications behave in a controlled environment, typically when obtaining an encryption key or gaining access to the victim's data (in Step 5 and Step 6 before Step 7 as mentioned in Section 1.1). One drawback of dynamic analysis when analyzing ransomware is that the malware may attempt to hide its true actions by checking its environment, which could lead to the ransomware avoiding detection in a controlled environment. Specifically, the ransomware might perform environmental mapping to ensure that it is running in the victim's system and not in a controlled environment, making it more difficult for analysts to accurately observe the behavior of ransomware and identify its malicious actions (Aldauiji, Batarfi, & Bayousef, 2022). Static analysis is more efficient and quicker compared to dynamic analysis, but a simple addition to the ransomware feature will cause a mismatch and thus not be effective. Dynamic analysis, on the other hand, is better at detecting Zero-day ransomware attacks.

1.4 Research Question

This paper explores effective defense strategies for businesses and essential organizations against ransomware attacks, despite the constant emergence of new ransomware variants. The study uses a dataset from a network scenario on Windows computers, similar to a corporate setting, to detect ransomware attacks that exhibit abnormal behavior while accessing, writing, reading, and deleting files in a network-shared file system. This study proposes a multi-variant deep learning classification technique, leveraging the capacity of deep learning to comprehend complex patterns to detect malicious behavior by ransomware from benign application behavior. The evaluation of the proposed models is based on their accuracy in differentiating ransomware behavior from benign applications, minimizing the False Negative (FN) rate of ransomware, and detecting Zero-day ransomware.

Chapter 2

Related Works

In the modern world, machine learning is used to optimize various processes in different sectors such as healthcare, finance, transportation, and cybersecurity. Thus, researchers in the cybersecurity field are increasingly using machine learning to detect, prevent and respond to cyber threats. This chapter will categorize previous works that focus on detecting ransomware attacks based on the types of analysis mentioned in 1.3.

2.1 Static Analysis

An online malware scanning tool called VirusTotal includes its own clustering and similarity-matching algorithms for various sample sets. In order to compare various ransomware families and determine the similarity matrix between those samples, Yamany, Azer, and Abdelbaki (2022) suggested their own malware indexing approach, which depends on hybrid data from static feature extraction. This study's shortcoming is finding similarities between 10,000 different ransomware samples for clustering and classification.

Static features can be collected from VirusTotal to detect ransomware attacks; VirusTotal is a popular service that scans malicious files and web URLs (Peng, Yang, Song, & Wang, 2019). In M., S., P., and Sandhya (2020), they extracted about 400 features from both ransomware and benign executable files using N-gram and Term Frequency-

Inverse Document Frequency (TF-IDF) methods. The machine learning models used to classify ransomware attacks are Gradient Boosting Tree, Decision Tree, Navies Bayes, and Adaboost. Gradient Boosting Tree performed better with 99.997% accuracy and a False Positive rate of 0.01.

The core function of ransomware attacks is similar, such as finding new vulnerabilities and distributions. Medhat, Gaber, and Abdelbaki (2018) proposed extracting static features such as Application Programming Interface (API) functions, file keywords, cryptography signatures, and file extensions. YARA language is used to make ransomware decision rules, that make decisions by grouping the static features. This approach achieved an accuracy of 94.15% in classifying ransomware samples. Their approach is extended in Medhat, Essa, Faisal, and Sayed (2020), YARAMON, which is adding a dynamic analysis where the memory dump is extracted and scanned. It increased the accuracy to 96.2% in classifying ransomware samples.

Ransomware attacks can occur on multiple operating systems and devices, such as mobile devices with Andriod applications. In Kanwal and Thakur (2017), proposed an approach that detects ransomware by static analysis of Android applications. They extracted codes from applications and the .apk files for code review and text analysis. The analysis is done by comparing the percentage of permission/keywords used by applications to previously known ransomware. The approach was tested across 10 devices and also developed their own application that has suspicious code. The limitation of this paper is that it might not be able to keep up with the complexity of both benign and ransomware applications.

Deep learning models are able to extract meaningful features thus, in Zhang et al. (2020) proposed a static approach based on N-gram instruction of machine codes (opcodes) using deep learning. The deep learning models used to classify the opcode sequence are a combination of a Self-Attention Convolutional Neural Network (CNN) and a Bi-Directional Self-Attention Network. The models achieved an accuracy of 89.5%, precision of 87.5%, recall of 87.6%, and F1-measure of 87.3%. This approach fails to handle new variants of ransomware.

Using dynamic analysis on a program is ineffective as it is uncertain how long a program should be monitored to detect a ransomware attack (Manavi & Hamzeh, 2020). Thus, they proposed a static analysis by extracting features from Portable Executable (PE) header files and constructing a 32*32 gray-scale image of the malicious executable files. Then used a CNN to extract features from the image and classify the constructed image. This approach achieved an accuracy of 93.33%. The drawback to this approach is that converting the PE header into an image would require a network with more layers to extract its features. In (2021), Manavi and Hamzeh proposed to improve the processing of the sequence of header bytes using a Long Short Term Memory (LSTM) network and classify the ransomware samples from benign samples. The LSTM network achieved an accuracy of 93.25%.

Byte-level static analysis from an executable file to detect ransomware attacks is proposed in (Khammas, 2020). They extracted the features from raw bytes of an executable file using 32-bit sliding windows (4-gram) and frequent pattern mining. Random Forest (RF) machine learning technique is used to detect ransomware attacks. The approach achieved an accuracy of 97.7% in just 1.37s time of detection.

Pre-Encripytion Detection Algorithm (PEDA) two-level detection of ransomware attacks is proposed in (S. Kok, Abdullah, & Jhanjhi, 2022). The first level matches ransomware signatures using SHA-256 (Secure Hashing Algorithm) from the stored Signature Repository and the second level detects the behavior of ransomware using Learning Algorithm (LA), based on API calls. In this research, they identified fourteen important APIs that can differentiate between ransomware and good ware. The limitation of this research is that it will not be able to detect ransomware that uses its encryption code without an API.

2.2 Dynamic Analysis

The current signature-based detection is not able to address the growing number of distinct ransomware variants, hence in Sgandurra, Muñoz González, Mohsen, and Lupu (2016) proposed a framework called EldeRan that used a machine learning model, Regularized Logistic Regression, to classify ransomware by using dynamic features such as API calls, Registry Key Operations, and File System Operations. The ransomware and good ware dataset are obtained by analyzing 582 binaries from 11 ransomware families and 942 good ware applications in a sandbox environment for 30 seconds. EldeRan achieved an accuracy of 96.3% detection rate and also 93.3% detection rate on Zero-day ransomware. The dataset in this paper is frequently used by researchers for dynamic analysis of ransomware attacks, which is one of the few publicly available datasets. An example use of the dataset, in Sethi, Kumar, Sethi, Bera, and Patra (2019), proposed an approach to use two different feature selections to extract the most relevant features and increase the accuracy of ransomware detection. They also compared different machine learning models such as K-Nearest Neighbor (KNN), Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF). DT gives a high accuracy of 99.11%.

Deep Contractive Autoencoder Zero-Shot Learning (DCAE-ZSL) as well as Heterogeneous Voting Ensemble (DCAE-ZSL-HVE) to extract features against Zero-day ransomware attacks is proposed in (Zahoora, Rajarajan, Pan, & Khan, 2022). The DCAE-ZSL technique learns the uniform latent semantic embedding and concentrates on key similarities between known and unknown ransomware attacks while penalizing the input for minor changes. In the DCAE-ZSL-HVE technique, ZSL-train data are trained on heterogenous learning models such as RF, SVM, Gaussian Naive Bayes (GNB), and Logistic Regression (LR). To infer the unknown class label, they combined two decision spaces. DCAE-ZSL-HVE has a Recall of 0.95 and a reduced False Negative (FN).

Hardware Performance Counters (HPCs) are monitored dynamically to detect malicious programs targeted for a particular system within some time interval in Alam et al.

(2020). In this study, they proposed a two-step unsupervised detection tool called Ransomware Prevention via Performance Counters (RAPPER). RAPPER utilized an LSTM-based autoencoder to generate an intermediate feature vector related to the time-series multivariate input sequence and Fast Fourier Transformation (FFT) to understand repetitive patterns of ransomware traces. The limitation of this approach is that ransomware and benign samples are generated by observing system activities for 10ms.

Dynamic Ransomware Detector based on improved TextCNN (DRDT) to obtain semantic information features from an API call sequence is proposed in (Qin, Wang, & Ma, 2020). They used the frequency of API calls, which are the specific operation of ransomware or benign application such as creating/deleting, reading/writing files, or modifying registry keys provided by Sangfor Technologies. DRDT reached an accuracy of 95.9% in comparison to LSTM with an accuracy of 88.7% and CNN-LSTM hybrid with an accuracy of 90%.

Time-series micro-architectural information is collected from 80 ransomware executables and 76 benign programs executed at random for 2ms from embedded HPCs in (Maniath et al., 2017). LSTM-based Recurrent Neural Network (RNN) is used for the time-series classification of micro-architectural event signatures of 20 timestamps. RanStop has an accuracy of 97% for 50 random trials. This method's drawback is that runtime hardware data processing is susceptible to corruption.

Since fog nodes collect and process a lot of sensitive data, such as personal data, they are a prime target for ransomware attacks. Thus, Homayoun et al. (2019) utilized two deep learning models CNN and LSTM to classify ransomware samples from benign samples based on system call sequence in 10s. The proposed models achieved an F-measure of 99.6% with a true positive rate of 97.2% and a false positive rate of 0.027% in the classification of ransomware variants.

Malicious attackers find it easy to target Supervisory Control and Data Acquisition (SCADA), which is the underlying control system of most critical infrastructures such as power, energy, water, traffic lights, and nuclear plants. Thus, Basnet, Poudyal, Ali, and Dasgupta (2021) proposed three deep learning models, Deep Neural Network (DNN), CNN, and LSTM-RNN to detect ransomware attacks on the framework in the SCADA-controlled Electric Vehicle Charging Station (EVCS). 561 ransomware samples and 447 benign samples are collected from VirusTotal extracted from Windows OS to train and test the deep learning models. A dynamic binary instrumentation technique is used to extract frequency analysis of assembly instructions. The accuracy of the three proposed deep learning models is 98.30% for DNN, 98.73% for CNN, and 97.59% for LSTM-RNN.

It is noted that the number of unique files and total files accessed by benign users is considerably lower than by ransomware attacks. Thus, the I/O Request Packet (IRP), a low-level file system I/O logs of 272 ransomware samples from 18 different families is analyzed in (Ayub, Continella, & Siraj, 2020). They extracted features such as ransomware process IDs, and process names for each sample's IRP logs. Artificial Neural Network (ANN), a deep learning model, is used to classify ransomware and benign samples. The model achieved an accuracy of 99.86%. The weakness of this study is that the ransomware IRP logs are captured within 90 minutes post-encryption, thus it would not be an early detection of ransomware attacks.

Ransomware has a higher activity such as opening, closing and modifying files than a benign application (Morato, Berrueta, na, & Izal, 2018); thus, they extracted these features from file-sharing network traffic during the execution of different ransomware binaries. They proposed a framework called Ransomware Early Detection from FIle SHaring traffic (REDFISH), designed to detect ransomware that overwrites the original file or produces an encrypted version in the same file system path. REDFISH is tuned to achieve high accuracy. Berrueta, Morato, Magaña, and Izal (2022) proposed another approach by training and testing a Tree Ensembles and a 3-layer Neural Network deep learning model. Their approach achieved an accuracy of over 99% in detecting ransomware activity.

Chapter 3

Proposed Approach and Architecture

This section will introduce the proposed deep learning models; Convolutional Neural Networks, and Bi-Directional Long Short Term Memory for detecting crypto-ransomware attacks. The deep learning models are built using a Python library called Keras that runs on top of TensorFlow (Chollet et al., 2015). The proposed models will be compared against the classic machine-learning models Logistic Regression, Support Vector Machine, and Random Forest. The Scikit-learn library is used to implement a logistic regression model for this paper (Pedregosa et al., 2011).

3.1 Classic Machine Learning Models

3.1.1 Logistic Regression

A Logistic Regression model (LR) is a supervised machine learning algorithm used for binary classification problems. It uses a logistic function called the Sigmoid function (equation below 3.1) to model the relationship between the input features and the output class probability.

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.1}$$

During the training phase, the LR algorithm learns the weights of the features through an optimization process called maximum likelihood estimation. Once trained, the model can be used to predict the output class of new input instances by computing their predicted probabilities using the learned weights and applying a threshold to classify them into the positive or negative class (Kanade, 2022).

3.1.2 Support Vector Machine

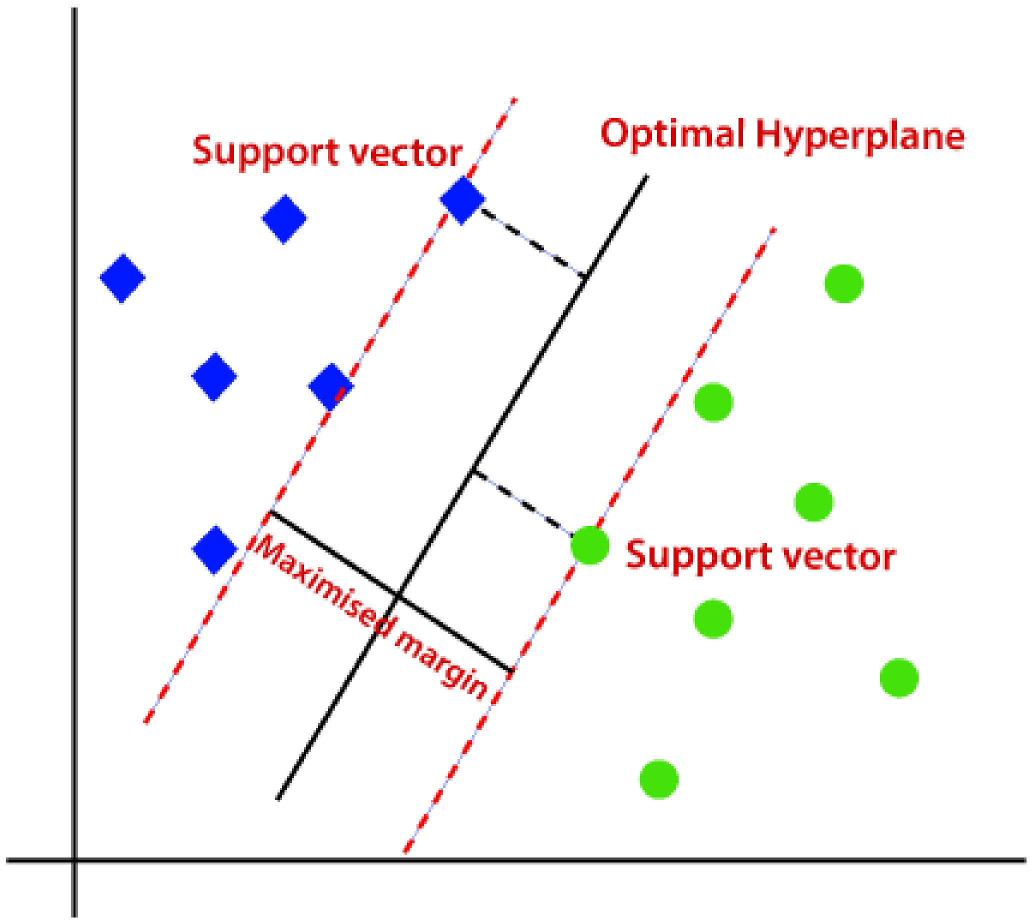


Figure 3.1: Visualization of SVM

Support Vector Machine (SVM) is a machine learning algorithm mainly used for classification. The goal of SVM is to find a line (or a hyperplane in higher dimensions) that maximally separates the two classes of data points as shown in Figure 3.1 (Saini, 2021).

The hyperplane should be equidistant from the closest data points of each class, and this distance is called the margin. The data points that are closest to the margin are called Support Vectors and are used to define the hyperplane. Once the hyperplane is found, new data points can be classified as belonging to one of the two classes depending on which side of the hyperplane they fall in. SVM is a powerful algorithm that works well with high-dimensional data and can handle non-linearly separable data using kernel functions. It has been successfully applied in various fields, including computer vision, text classification, and bio-informatics.

3.1.3 Random Forest

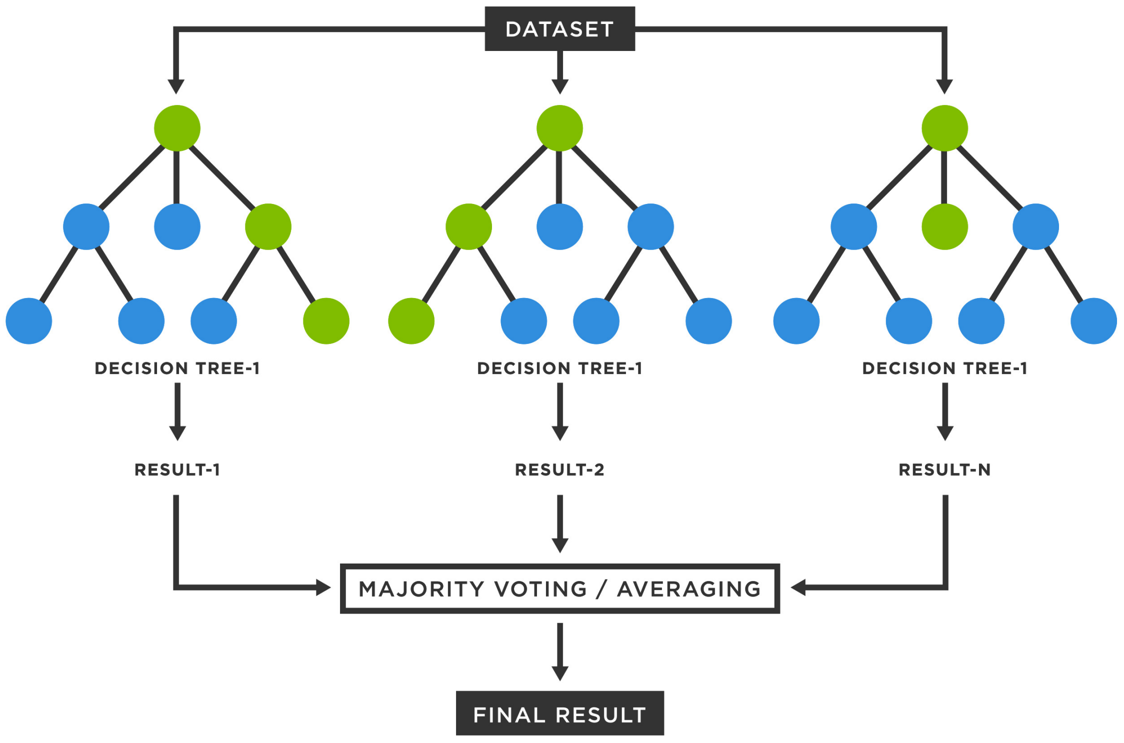


Figure 3.2: Visualization of RF

A Decision Tree is a supervised machine-learning algorithm that has a tree-like structure widely used for classification and prediction. By learning straightforward decision rules, it predicts a class by organizing the data from the root to the leaf/terminal node of the tree. A Random Forest (RF) combines multiple decision trees to create a more accurate model. Each decision tree is built on a subset of the training data and each split decision tree is chosen at random to reduce over-fitting. The final decision in RF is from the class that receives the most votes from individual trees as shown in figure 3.2 . Features of RF have high accuracy and they can handle large datasets but they fail to comprehend the significance of each variable.

3.2 Proposed Deep Learning Models

3.2.1 Convolutional Neural Network

Convolutional Neural Networks (CNN) are variants of neural networks which are mainly used for image classification, facial recognition, object detection for self-driving cars, image analysis in healthcare, automatic speech recognition, and so on. CNNs are composed of layers and each layer applies a function to its input tensors and passes the transformed version of the data tensors as input to the next layer.

The three main types of layers of CNN are convolutional layers, pooling layers, and Fully Connected layers. Convolutional layers use feature map equation in Equation 3.2 a filter (kernel) to extract features from input data.

$$\mathbf{h}_i = f((\mathbf{W} * \mathbf{x})_i) \tag{3.2}$$

Pooling Layers are used to reduce the number of parameters and to avoid over-fitting. There are different types of pooling techniques such as max pooling, average pooling, and so on (Kiranyaz, Gastli, Ben-Brahim, Al-Emadi, & Gabbouj, 2019).

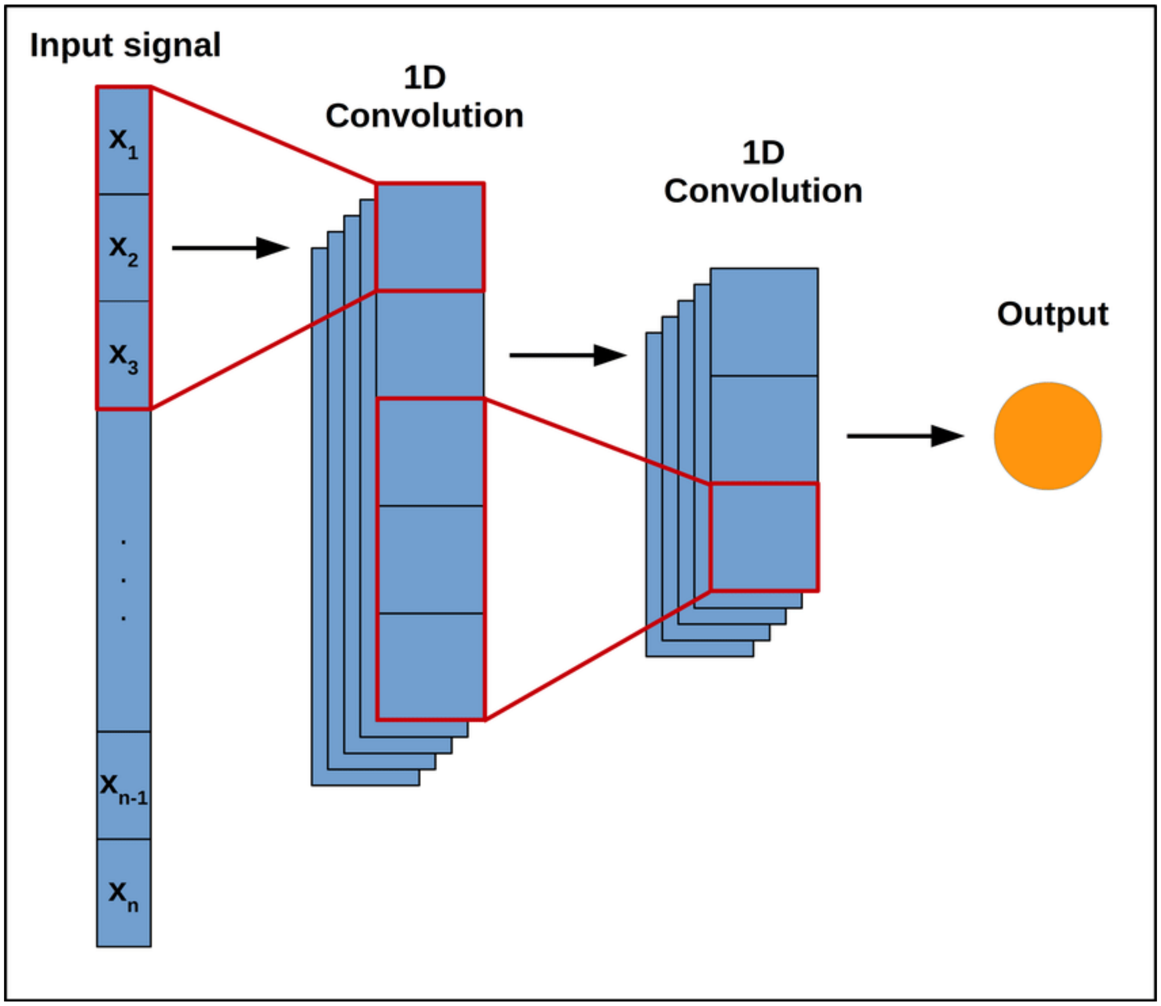
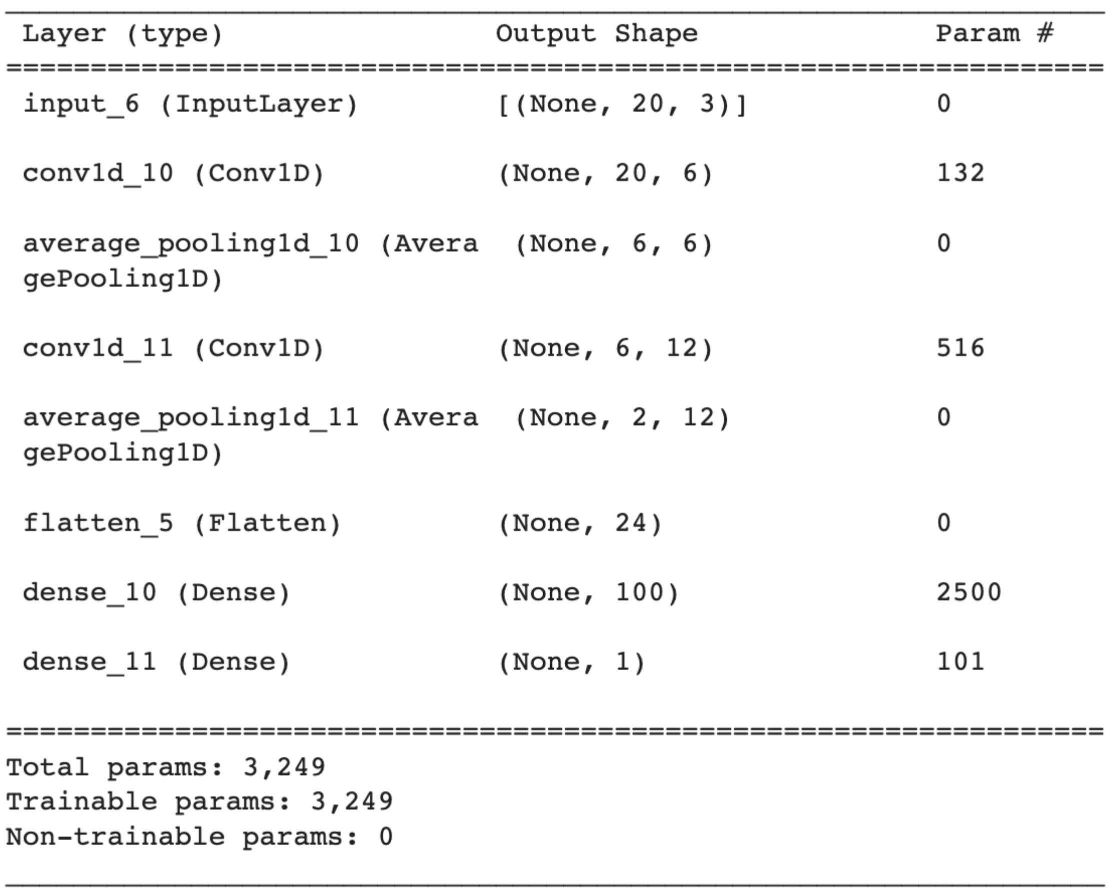


Figure 3.3: Visualization of CNN layers.

For this research, the CNN model is composed of two Convolutional blocks. Each block has a 1D convolution layer with Sigmoid as an activation function and a 1D averaging pooling layer. Then it is flatted to be an input for the fully connected layer which applies linear transformation followed by an activation function for an output layer. The CNN model is compiled with Mean Squared Error for the loss function, Adam optimization, and Binary Accuracy for the metrics as shown in Figure 3.4.

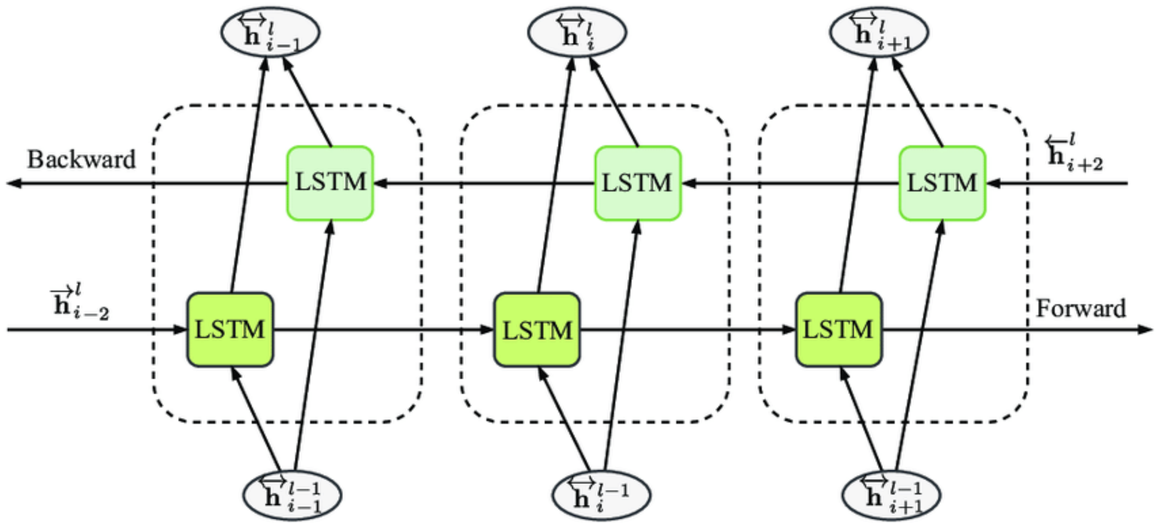


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 20, 3)]           0

 conv1d_10 (Conv1D)          (None, 20, 6)             132

 average_pooling1d_10 (Avera  (None, 6, 6)             0
 gePooling1D)

 conv1d_11 (Conv1D)          (None, 6, 12)             516

 average_pooling1d_11 (Avera  (None, 2, 12)            0
 gePooling1D)

 flatten_5 (Flatten)         (None, 24)                0

 dense_10 (Dense)            (None, 100)               2500

 dense_11 (Dense)            (None, 1)                 101

=================================================================
Total params: 3,249
Trainable params: 3,249
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.4: CNN model summary for T = 20s.

3.2.2 Bi-directional LSTM

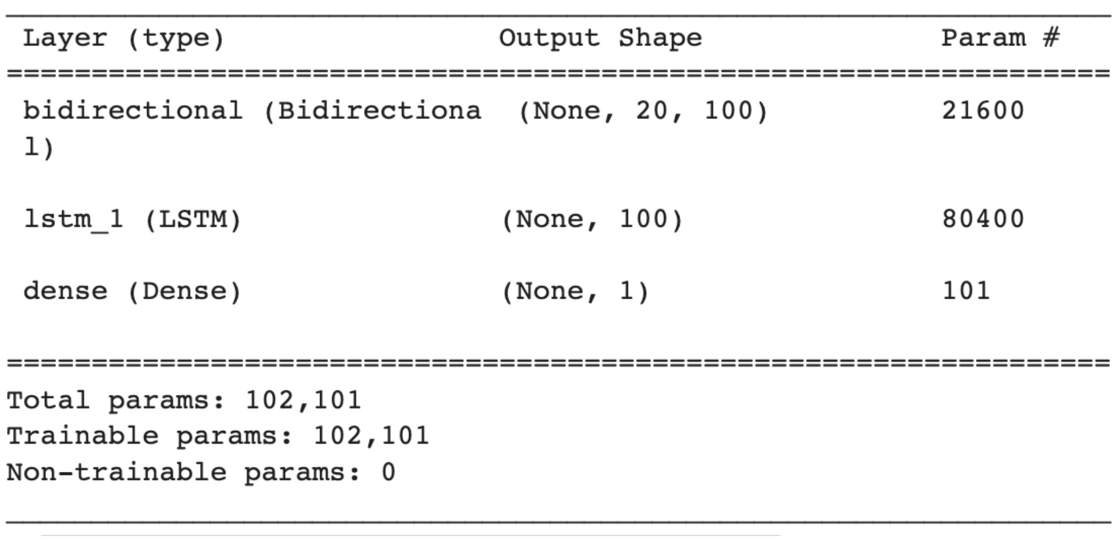
Unlike traditional neural networks where inputs and outputs are independent of each other, in Recurrent Neural Network(RNN) provides some gates to store output from a previous step to use as input in the current step. This enables RNN to leverage previous sequential information for arbitrary long sequences, however, in practice, it has memory limitations and it is also unable to capture long-term dependencies due to 'vanishing gradients' (Siami-Namini, Tavakoli, & Namin, 2019). Long Short-Term Memory (LSTM) is introduced to avoid the long-term dependencies problem of RNN. LSTM is able to remember long sequences of input data furthermore, it is able to describe the relationship between input and output data in terms of further dimensions such as time. Bi-directional LSTM (Bi-LSTM) is a variation of normal LSTM, where each step will have input from both forward LSTM and backward LSTM about the sequence. Figure 3.5 shows the illustration of the Bi-LSTM model (Verma, 2021).



$\overleftrightarrow{h}_{i-1}^{l}$
$\overleftrightarrow{h}_{i}^{l}$
$\overleftrightarrow{h}_{i+1}^{l}$
Backward
$\overleftarrow{h}_{i+2}^{l}$
LSTM
LSTM
LSTM
$\overrightarrow{h}_{i-2}^{l}$
LSTM
LSTM
LSTM
Forward
$\overleftrightarrow{h}_{i-1}^{l-1}$
$\overleftrightarrow{h}_{i}^{l-1}$
$\overleftrightarrow{h}_{i+1}^{l-1}$

Figure 3.5: Visualization of Bi-LSTM

In this research, the Bi-LSTM is composed of forward LSTM and backward LSTM. Both LSTMs use Sigmoid for the activation function. The Bi-LSTM model is compiled with Mean Squared Error for the loss function, Adam optimization, and Binary Accuracy for the metrics as shown in Figure 3.6.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 20, 100)          21600
 l)

 lstm_1 (LSTM)               (None, 100)               80400

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 102,101
Trainable params: 102,101
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.6: Bi-LSTM model summary for T = 20s.

Chapter 4

Experiments

4.1 Dataset And Feature Selection

In Berrueta, Morato, Magaña, and Izal (2020), it contains file access operations of more than 70 samples from 30 distinct crypto-ransomware retrieved during the encryption of an extensive network share directory which is comparable to a corporate setting. The ransomware samples are executed on a Microsoft Windows machine running in a virtual box environment with a Network Attached Storage (NAS) file server network configuration to extract information from the network traffic. The network traffic is captured and analyzed using a network probe which mirrors network traffic between a user and NAS file server. In Microsoft Windows, Server Message Block (SMB) protocols are transported over TCP/IP in network file-sharing scenarios and developed their own tool to extract input/output (I/O) operations from SMB commands as shown in Figure 4.1. In Morato et al. (2018), the I/O operations of ransomware and benign dataset are made available to train machine learning models. The benign samples came from network traffic traces recorded in a campus LAN, where staff users access files from shared servers.

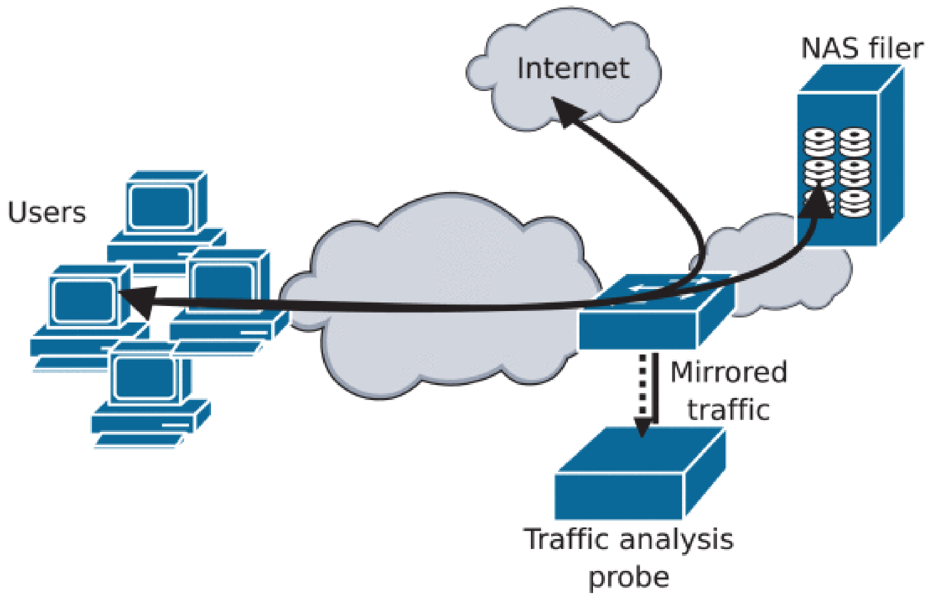


Figure 4.1: Visualization of mirroring network traffic

Even though there is a variation during a ransomware attack, in most cases, files are opened and read to be encrypted. Thus I/O operations extracted from benign and ransomware samples are vital to detecting ransomware attacks. A sample contains I/O features such as short Control Commands (such as opening/closing, renaming, or deleting a file), bytes being read, and bytes being written per second.

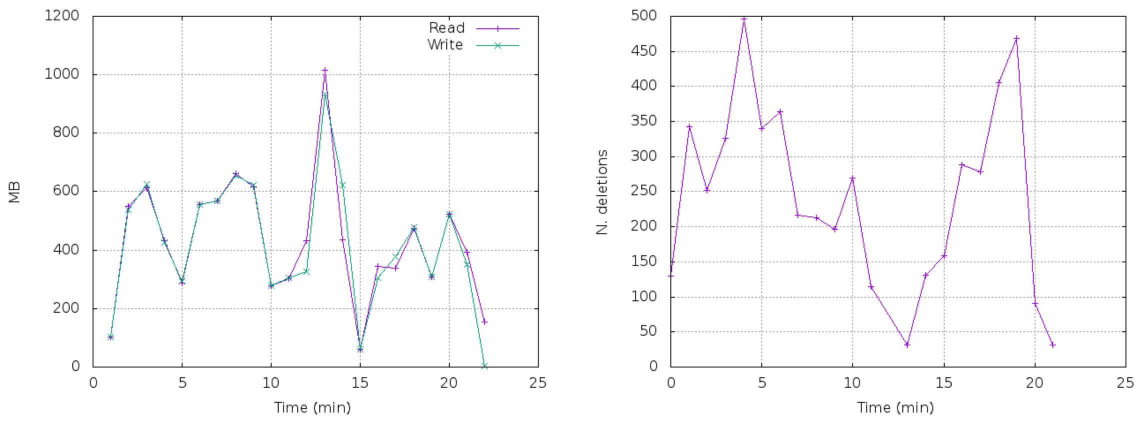


Figure 4.2: I/O of Maze ransomware

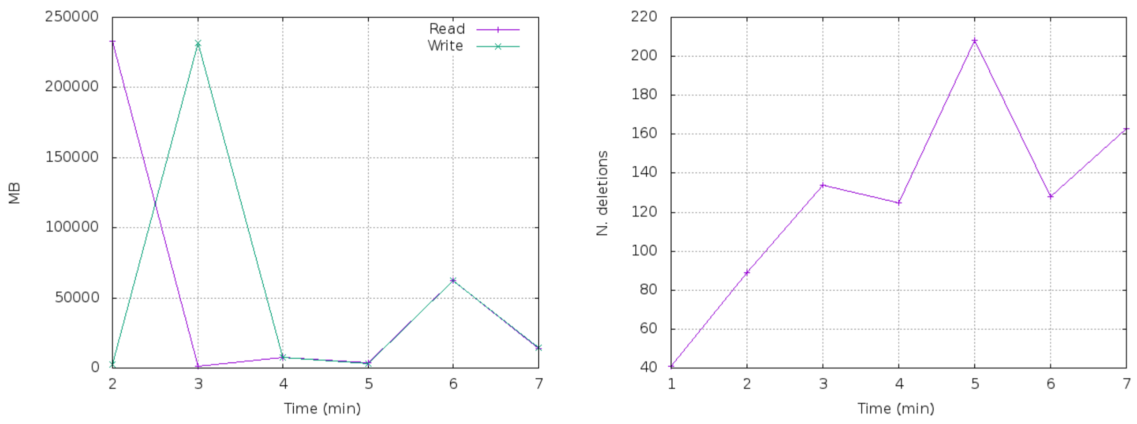


Figure 4.3: I/O of TeslaCrypt ransomware

Maze ransomware uses the double extortion method, it connects to a file transfer protocol(FTP) and moves the data by copying files and encrypting them. Figure 4.2 shows the I/O operations of Maze ransomware, the rate of bytes read and written are the same, and it co-relates to the number of delete commands(Berrueta et al., 2020). In Figure 4.3 I/O of a different ransomware called TeslaCrypt also encrypted the victim's images, and documents, and demanded a ransom in exchange for the decryption key. Different ransomware exhibits different I/O operations and sometimes benign applications might showcase similar behaviors thus training the deep learning models with a variety of ransomware and benign samples yield a better result.

Figure 4.4 shows the visualization of I/O features captured over T seconds from execution time, meanwhile, T represents 10, 20, 30, 40, 50, and 60. Since different ransomware exhibits different behavior over time, having a detection system spread across T seconds would yield successful detection of malicious activities. In this study 0 represents a benign sample and 1 represents a ransomware sample. Table 4.1 shows the number of ransomware and benign samples for each T second period in the dataset.

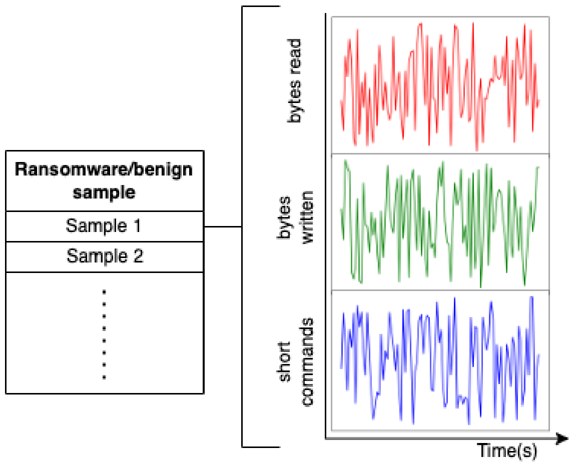


Figure 4.4: Visualization of I/O features in T seconds

	10s	20s	30s	40s	50s	60s
Ransomware	17,618	9,097	6,111	4,632	3,687	3,055
Benign	17,618	9,097	14,537	12,512	11,159	10,228

Table 4.1: Dataset distribution of ransomware/benign samples for T seconds.

4.2 Evaluation Metrics

In the context of classification models, there are four possible outcomes: True Positive (TP), True Negative (TN), False Negative (FN), and False Positive (FP). In this particular study, TP refers to when a sample is correctly classified as ransomware, while TN refers to when a sample is correctly classified as benign. On the other hand, FP is when a sample is incorrectly classified as ransomware, and FN is when a sample is incorrectly classified as benign.

In order to assess the performance of the machine learning models, several evaluation metrics were employed including accuracy, sensitivity (recall), specificity, precision, F1-score, and Matthews Correlation Coefficient (MCC). Accuracy refers to the percentage of correct predictions out of the total number of predictions made by the model.

$$\text{Accuracy} \;=\; \frac{\text{TP}}{(\text{TP + TN + FP + FN})} \tag{4.1}$$

Sensitivity assesses the model's ability to predict true positives in each available category.

$$\text{Sensitivity(recall)} \;=\; \frac{\text{TP}}{(\text{TP + FN })} \tag{4.2}$$

While specificity measures the model's ability to predict true negatives in each category.

$$\text{Specificity} \;=\; \frac{\text{TN}}{(\text{TN + FP})} \tag{4.3}$$

Precision is the number of accurate positive results divided by the number of positive results expected by the classifier.

$$\text{Precision} \;=\; \frac{\text{TP}}{(\text{TP + FP})} \tag{4.4}$$

The F1 score showcases how accurate and robust your classifier is. High precision but poor recall yields a highly accurate result, but it also misses a significant number of difficult-to-classify instances. The model's efficiency increases with a higher F1 Score.

$$\text{F-score} = 2\frac{\text{Precision} \,*\, \text{Recall}}{\text{Precision} \,+\, \text{Recall}} \tag{4.5}$$

Matthews Correlation Coefficient(MCC) is a very important evaluation metric as it shows the correlation between the four outcomes(TP, TF, FN, FP). The value is between -1 and 1. So the closer the value of MCC is to 1, it signifies the predicted class and true class are strongly related. In this study, the measured value of MCC has been scaled between 0 to 100%.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4.6}$$

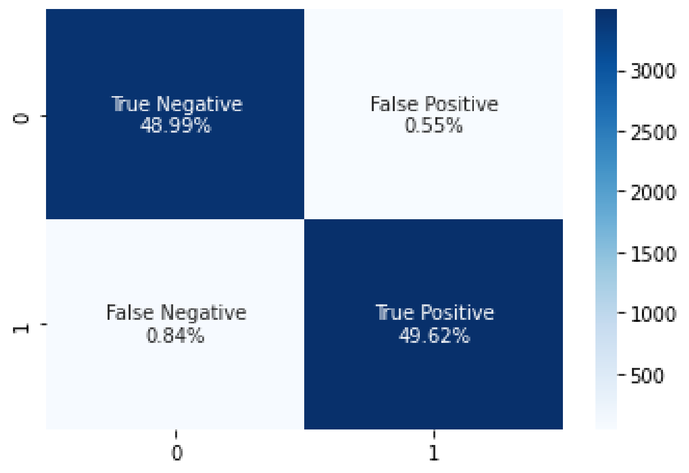
Chapter 5

Results and Analyses

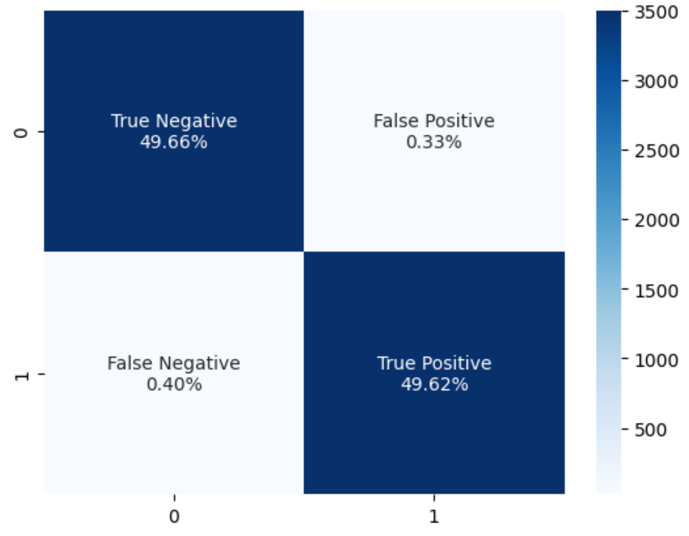
This section will discuss the evaluation and comparison of the baseline and proposed models. Logistic regression and Random Forest, process features independently which means it does not take the leverage of information about the sequence of events. Hence, multi-variant classification using CNN and Bi-LSTM is proposed in this paper. The models are trained and tested on Google Colab, which is an online Jupyter Notebook that provides computing resources through a browser. The programming language used to implement and evaluate these models is Python and open-resource Python libraries such as Keras and Scikit-learn.

5.1 Testing Results and Comparison

80% of the samples are used for training the model meanwhile, both CNN and Bi-LSTM are trained over 100 epochs. After training both models, they are tested with the remaining 20% of samples used to test and evaluate the deep learning models. A Confusion Matrix (CM) is a that reports the number (in this study, it will be shown in percentage) of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).



(a) CNN

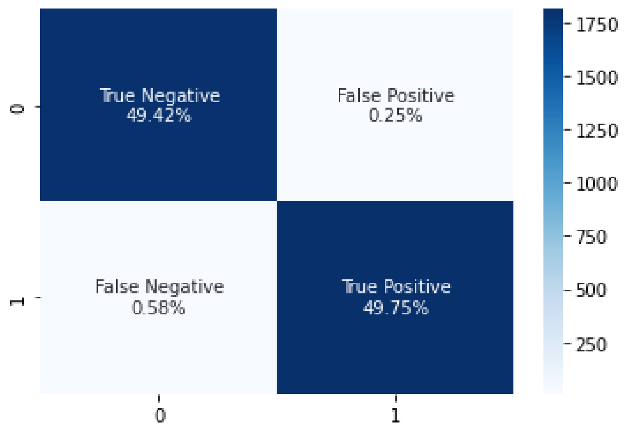


(b) Bi-LSTM

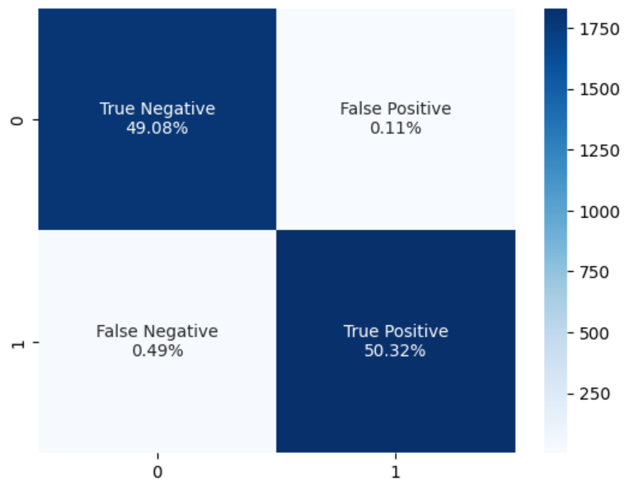
Figure 5.1: Confusion Matrix for T=10s

	Accuracy	Sensitivity	Specificity	F1-Score	MCC
Logistic Regression	89.65%	86.04%	93.27%	89.27%	79.52%
Support Vector Machine	92.42 %	90.24%	94.60%	92.25%	84.92%
Random Forest	99.58%	99.77%	99.4%	99.58%	99.17%
Convolutional Neural Networks	98.62%	98.48%	98.76%	98.63%	97.24%
Bi-directional LSTM	99.27%	99.2%	99.34%	99.27%	98.55%

Table 5.1: Performance Metric for T= 10s



(a) CNN

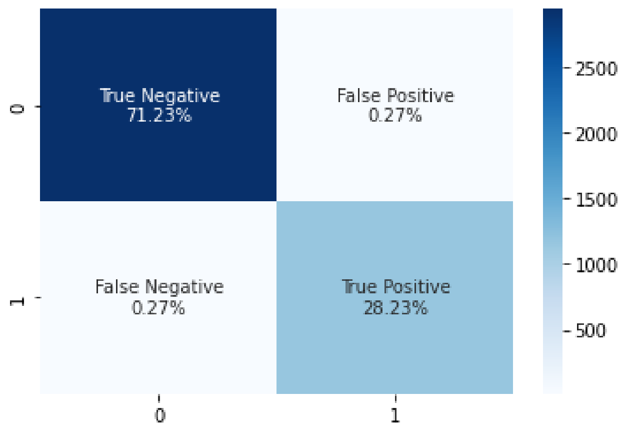


(b) Bi-LSTM

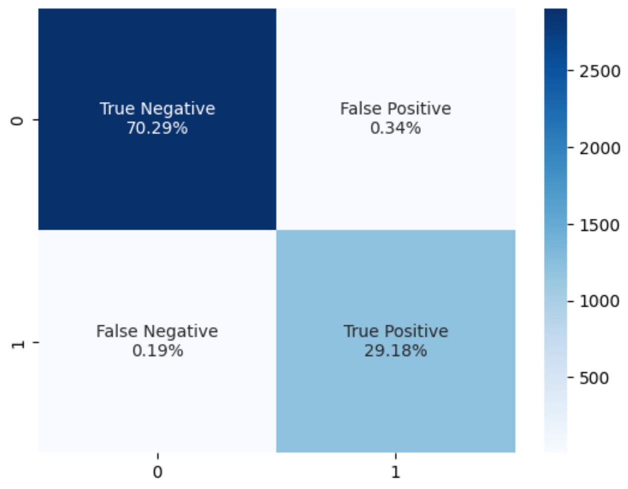
Figure 5.2: Confusion Matrix for T=20s

	Accuracy	Sensitivity	Specificity	F1-Score	MCC
Logistic Regression	90.38%	86.26%	94.63%	90.11%	81.09%
Support Vector Machine	92.22%	90.26%	94.24%	92.18%	84.52%
Random Forest	99.67%	99.62%	99.72%	99.67%	99.34%
Convolutional Neural Networks	99.23%	98.9%	99.55%	99.23%	98.4%
Bi-directional LSTM	99.39%	99.02%	99.77%	99.40%	98.79%

Table 5.2: Performance Metric for T= 20s



(a) CNN

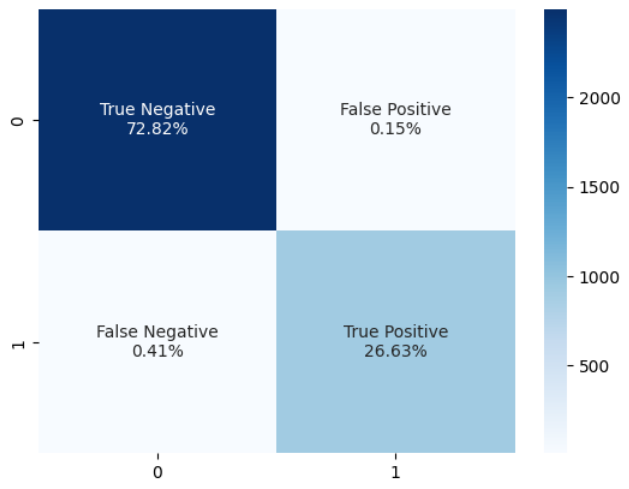


(b) Bi-LSTM

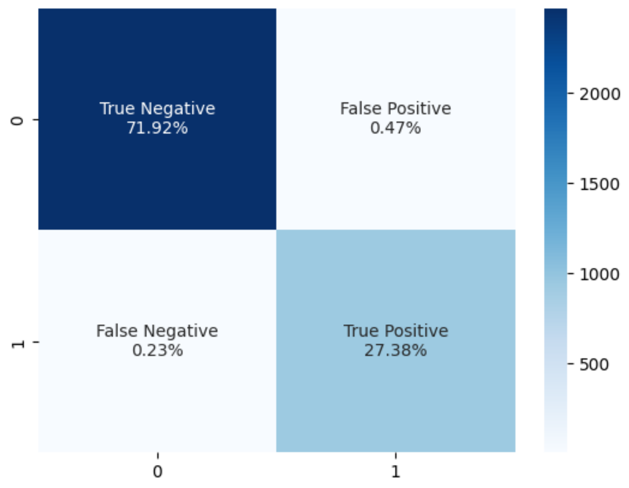
Figure 5.3: Confusion Matrix for T=30s

	Accuracy	Sensitivity	Specificity	F1-Score	MCC
Logistic Regression	93.7%	84.08%	97.73%	88.73%	84.65%
Support Vector Machine	95.42%	90.02%	97.66%	92.03%	88.87%
Random Forest	99.7%	99.25%	99.89%	99.5%	99.29%
Convolutional Neural Networks	99.46%	98.38%	99.89%	99.05%	98.69%
Bi-directional LSTM	99.46%	99.34%	99.52%	99.09%	98.71%

Table 5.3: Performance Metric for T= 30s



(a) CNN

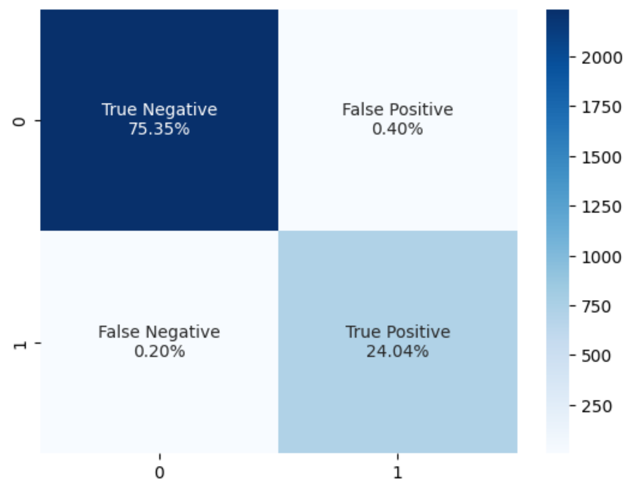


(b) Bi-LSTM

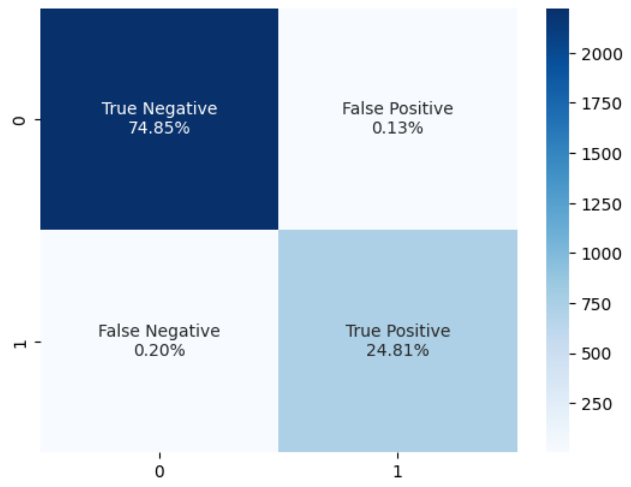
Figure 5.4: Confusion Matrix for T=40s

	Accuracy	Sensitivity	Specificity	F1-Score	MCC
Logistic Regression	94.75%	86.27%	97.98%	90.07%	86.67%
Support Vector Machine	96.29%	91.86%	97.98%	93.19%	90.67%
Random Forest	99.7%	99.25%	99.89%	99.5%	99.29%
CNN	99.44%	98.48%	99.8%	98.97%	98.59%
Bi-directional LSTM	99.56%	98.83%	99.83%	99.2%	98.90%

Table 5.4: Performance Metrics for T=40s



(a) CNN

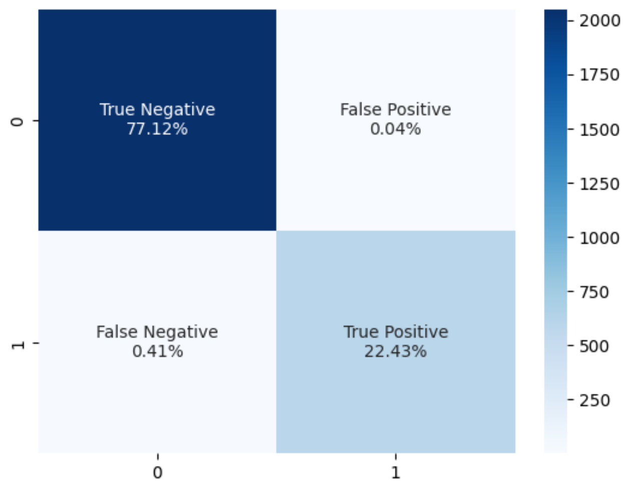


(b) Bi-LSTM

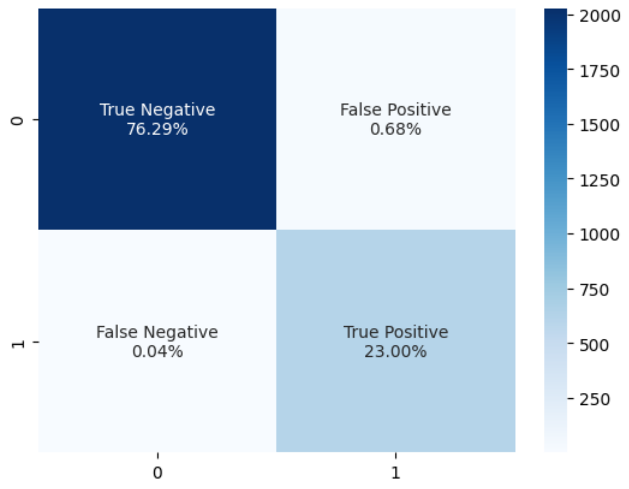
Figure 5.5: Confusion Matrix for T=50s

	Accuracy	Sensitivity	Specificity	F1-Score	MCC
Logistic Regression	94.64%	85.46%	97.7%	88.87%	85.47%
Support Vector Machine	96.19%	91.38%	97.79%	92.31%	89.79%
Random Forest	99.6%	98.92%	99.91%	99.32%	99.1%
Convolutional Neural Networks	99.39%	99.16%	99.46%	98.75%	98.35%
Bi-directional LSTM	99.66%	99.19%	99.82%	99.32%	99.1%

Table 5.5: Performance Metrics for T=50s



(a) CNN



(b) Bi-LSTM

Figure 5.6: Confusion Matrix for T=60s

	Accuracy	Sensitivity	Specificity	F1-Score	MCC
Logistic Regression	95.48%	86.27%	98.23%	89.79%	87.02%
Support Vector Machine	97.13%	94.11%	98.04%	93.81%	91.95%
Random Forest	99.62%	98.85%	99.85%	99.1%	98.93%
Convolutional Neural Networks	99.54%	98.18%	99.95%	99%	98.71%
Bi-directional LSTM	99.66%	99.55%	99.7%	99.26%	99.04%

Table 5.6: Performance Metrics for T=60s

Looking at the performance metrics in Table 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6 for the different models, it can be observed that all models performed quite well in terms of accuracy, with Random Forest achieving the highest accuracy score followed by CNN. In terms of sensitivity, which is a measure of how well the model identifies true positive cases, Bi-directional LSTM performed the best followed by CNN. Specificity, which measures how well the model identifies true negative cases, was highest for Random Forest, followed by CNN. F1-score is a weighted average of precision and recall and gives an overall measure of the model's accuracy. Random Forest achieved the highest F1-score followed by Bi-directional LSTM. Finally, Matthews Correlation Coefficient (MCC) is a measure of the quality of binary classification, taking into account true and false positives and negatives. Random Forest had the highest MCC followed closely by CNN. Overall, the results suggest that Random Forest, CNN, and Bi-directional LSTM are all strong performers for detecting ransomware attacks, with high accuracy, sensitivity, specificity, F1-score, and MCC.

Figure 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6 displays the Confusion Matrix results of Convolutional Neural Network (CNN) and Bi-directional LSTM (Bi-LSTM). Overall, it can be observed that both CNN and Bi-LSTM false negative rate decreases as T seconds increase. Bi-LSTM achieved a low false negative rate of 0.04% and CNN achieved a low false negative rate of 0.2% when T = 50 seconds. The deep learning models are tested with a ZeroDay ransomware to dictate if it will be able to detect ransomware strains that were not used to train the models.CNN and Bi-LSTM detected ZeroDay ransomware with a high accuracy of 98.66% and 99.53% respectively at T= 60s.

Chapter 6

Conclusion and Future Work

In this study, a multi-variant classification using deep learning models is proposed to detect ransomware attacks from a network scenario that is similar to a corporate setting. The ransomware samples were extracted from 70 binaries from 30 different ransomware during the encryption of an extensive network shared directory. The benign samples came from network traffic traces recorded in a campus LAN where staff users access files from shared servers. The samples were collected T seconds after the program executions, T = 10, 20, 30, 40, 50, and 60. The deep learning models proposed are CNN and Bi-LSTM. The deep learning models are compared against classic machine learning models, LR, SVM, and RF. CNN and Bi-LSTM achieved an accuracy above 98% and 99% respectively. Meanwhile, LR, SVM, and RF achieved an accuracy of above 89%, 92%, and 99% respectively. The proposed deep learning models overall performed well in classifying ransomware samples from benign samples; they are also able to classify Zero-day ransomware with high accuracy. Since this is a dynamic approach, the ransomware must execute, and some files might be lost in the process, which can be retrieved if a backup system is set in place.

This paper showed that deep learnings are able to extract features from I/O operations in a network file-sharing scenario in a corporate setting to detect ransomware attacks. In the future, adding more features of ransomware behavior would increase the accuracy of the model and minimize the rate of false negatives. Once a victim's system has been infiltrated, the future direction of this study is also to detect a ransomware attack using a hybrid model from multiple1 stages. Overall, many essential businesses and organizations have valuable data in which malicious actors can encrypt/steal to ask for ransom. This study shows using CNN and Bi-LSTM trained over increasing time to detect ransomware attacks once a system in a corporate setting has been infiltrated.

Bibliography

Alam, M., Sinha, S., Bhattacharya, S., Dutta, S., Mukhopadhyay, D., & Chattopadhyay, A. (2020). *Rapper: Ransomware prevention via performance counters.*

Aldauiji, F., Batarfi, O., & Bayousef, M. (2022). Utilizing cyber threat hunting techniques to find ransomware attacks: A survey of the state of the art. *IEEE Access*, *10*, 61695-61706. doi: 10.1109/ACCESS.2022.3181278

Alqahtani, A., & Sheldon, F. T. (2022). A survey of crypto ransomware attack detection methodologies: An evolving outlook. *Sensors*, *22*(5). Retrieved from `https://www.mdpi.com/1424-8220/22/5/1837` doi: 10.3390/s22051837

Ayub, M. A., Continella, A., & Siraj, A. (2020). An i/o request packet (irp) driven effective ransomware detection scheme using artificial neural network. *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, 319-324.

Basnet, M., Poudyal, S., Ali, M. H., & Dasgupta, D. (2021). Ransomware detection using deep learning in the scada system of electric vehicle charging station. , 1-5. doi: 10.1109/ISGTLatinAmerica52371.2021.9543031

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2018). Ransomware encrypted your files but you restored them from network traffic. , 1-7. doi: 10.1109/CSNET.2018.8602978

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2020). Open repository for the evaluation of ransomware detection tools. *IEEE Access*, *8*, 65658-65669. doi:

10.1109/ACCESS.2020.2984187

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2022). Crypto-ransomware detection using machine learning models in file-sharing network scenarios with encrypted traffic. *Expert Systems with Applications*, *209*, 118299. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0957417422014312` doi: https://doi.org/10.1016/j.eswa.2022.118299

Chollet, F., et al. (2015). *Keras.* GitHub. Retrieved from `https://github.com/fchollet/keras`

Cventicanin, N. (2023, Jan). *The biggest ransomware attacks in history.* Retrieved from `https://dataprot.net/guides/ransomware-attacks/`

Freed, A. M. (2021, Nov). *A brief history of ransomware evolution.* Retrieved from `https://www.cybereason.com/blog/a-brief-history-of-ransomware-evolution`

Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R., Choo, K.-K. R., & Newton, D. E. (2019). Drthis: Deep ransomware threat hunting and intelligence system at the fog layer. *Future Gener. Comput. Syst.*, *90*, 94-104.

Humayun, M., Jhanjhi, N., Alsayat, A., & Ponnusamy, V. (2021). Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, *22*(1), 105-117. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1110866520301304` doi: https://doi.org/10.1016/j.eij.2020.05.003

Kanade, V. (2022, Apr). *What is logistic regression? equation, assumptions, types, and best practices.* Retrieved from `https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/`

Kanwal, M., & Thakur, S. (2017). An app based on static analysis for android ransomware. , 813-818. doi: 10.1109/CCAA.2017.8229907

Kerns, Q., Payne, B., & Abegaz, T. (2022). Double-extortion ransomware: A technical analysis of maze ransomware.

Khammas, B. M. (2020). Ransomware detection using random forest technique. *ICT Express*, *6*(4), 325-331. Retrieved from `https://www.sciencedirect.com/science/article/pii/S2405959520304756` doi: https://doi.org/10.1016/j.icte.2020.11.001

Kiranyaz, S., Gastli, A., Ben-Brahim, L., Al-Emadi, N., & Gabbouj, M. (2019). Real-time fault detection and identification for mmc using 1-d convolutional neural networks. *IEEE Transactions on Industrial Electronics*, *66*(11), 8760-8771. doi: 10.1109/TIE.2018.2833045

Kok, S., Abdullah, A., & Jhanjhi, N. (2022). Early detection of crypto-ransomware using pre-encryption detection algorithm. *Journal of King Saud University - Computer and Information Sciences*, *34*(5), 1984-1999. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1319157820304122` doi: https://doi.org/10.1016/j.jksuci.2020.06.012

Kok, S. H., Abdullah, A. B., Jhanjhi, N. Z., & Supramaniam, M. (2019). Ransomware, threat and detection techniques: A review.

M., M. J. M., S., U., P., M. B., & Sandhya, S. G. (2020). Detection of ransomware in static analysis by using gradient tree boosting algorithm. , 1-5. doi: 10.1109/ICSCAN49426.2020.9262315

Manavi, F., & Hamzeh, A. (2020). A new method for ransomware detection based on pe header using convolutional neural networks. , 82-87. doi: 10.1109/ISCISC51277.2020.9261903

Manavi, F., & Hamzeh, A. (2021). Static detection of ransomware using lstm network and pe header. , 1-5. doi: 10.1109/CSICC52343.2021.9420580

Maniath, S., Ashok, A., Poornachandran, P., Sujadevi, V., Sankar A.U., P., & Jan, S. (2017). Deep learning lstm based ransomware detection. , 442-446. doi: 10.1109/RDCAPE.2017.8358312

Medhat, M., Essa, M., Faisal, H., & Sayed, S. G. (2020). Yaramon: A memory-based detection framework for ransomware families. , 1-6. doi: 10.23919/ICITST51030.2020.9351319

Medhat, M., Gaber, S., & Abdelbaki, N. (2018). A new static-based framework for ransomware detection. , 710-715. doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00124

Morato, D., Berrueta, E., na, E. M., & Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications*, *124*, 14-32. Retrieved from `https://www.sciencedirect.com/science/article/pii/S108480451830300X` doi: https://doi.org/10.1016/j.jnca.2018.09.013

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peng, P., Yang, L., Song, L., & Wang, G. (2019). Opening the blackbox of virustotal: Analyzing online phishing scan engines. New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3355369.3355585` doi: 10.1145/3355369.3355585

Qin, B., Wang, Y., & Ma, C. (2020). Api call based ransomware dynamic detection approach using textcnn. , 162-166. doi: 10.1109/ICBAIE49996.2020.00041

Richardson, R., & North, M. M. (2017, Jan). *Ransomware: Evolution, mitigation, and prevention.*

Saini, A. (2021, Nov). *Support vector machine(svm): A complete guide for beginners.* Retrieved from `https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/`

Savage, K., Coogan, P., & Lau, H. (2015, Aug). *The evolution of ransomware.* Retrieved from `https://its.fsu.edu/sites/g/files/imported/storage/images/information-security-and-privacy-office/the-evolution-of-ransomware.pdf`

Sethi, K., Kumar, R., Sethi, L., Bera, P., & Patra, P. K. (2019). A novel machine learning based malware detection and classification framework. , 1-4. doi: 10.1109/CyberSecPODS.2019.8885196

Sgandurra, D., Muñoz González, L., Mohsen, R., & Lupu, E. C. (2016, Sep). *Automated dynamic analysis of ransomware: Benefits, limitations and use for detection.* Retrieved from https://arxiv.org/abs/1609.03020

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). The performance of lstm and bilstm in forecasting time series. , 3285-3292. doi: 10.1109/BigData47090.2019.9005997

Urooj, U., Al-rimy, B. A. S., Zainal, A., Ghaleb, F. A., & Rassam, M. A. (2021). Ransomware detection using the dynamic analysis and machine learning: A survey and research directions. *Applied Sciences*.

Verma, Y. (2021, Aug). *Hands-on guide to bi-lstm with attention.*

What is a random forest? (n.d.). Retrieved from https://www.tibco.com/reference-center/what-is-a-random-forest

Yamany, B., Azer, M. A., & Abdelbaki, N. (2022). Ransomware clustering and classification using similarity matrix. , 41-46. doi: 10.1109/MIUCC55081.2022.9781655

Zahoora, U., Rajarajan, M., Pan, Z., & Khan, A. (2022). Zero-day ransomware attack detection using deep contractive autoencoder and voting based ensemble classifier. , *52*(12). Retrieved from https://doi.org/10.1007/s10489-022-03244-6 doi: 10.1007/s10489-022-03244-6

Zhang, B., Xiao, W., Xiao, X., Sangaiah, A. K., Zhang, W., & Zhang, J. (2020). Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes. *Future Generation Computer Systems*, *110*, 708-720. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167739X19315912 doi: https://doi.org/10.1016/j.future.2019.09.025