Declaration of Authorship

I, Nithya SHIKARPUR, declare that this Undergraduate Thesis titled, 'Detection Of Malware For IoT Devices' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research project at this Company.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this Company or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Nithya S

Date: 12/12/2019

Declaration of Authorship

I, Nithya SHIKARPUR, declare that this Undergraduate Thesis titled, 'Detection Of Malware For IoT Devices' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research project at this Company.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this Company or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Nithya S

Date: 12/12/2019

Detection Of Malware For IoT Devices

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Nithya Shikarpur
ID No. 2016A7TS0670G

Under the supervision of:

Dr. Abhishek Tripathi
&
Prof. Hemant Rathore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS
December 2019

Certificate

This is to certify that the thesis entitled, "*Detection Of Malware For IoT Devices*" and submitted by Nithya SHIKARPUR ID No. 2016A7TS0670G in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by her under my supervision.

Supervisor
Dr. Abhishek TRIPATHI
Machine Learning Researcher,
McAfee
Date: December 13, 2019

Co-Supervisor
Prof. Hemant RATHORE
Professor,
BITS-Pilani Goa Campus
Date:

Declaration of Authorship

I, Nithya SHIKARPUR, declare that this Undergraduate Thesis titled, 'Detection Of Malware For IoT Devices' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research project at this Company.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this Company or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Nithya S

Date: 12/12/2019

"You have to fight for your privacy or else you lose it."

Anonymous

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS

Abstract

Bachelor of Engineering (Hons.)

Detection Of Malware For IoT Devices

by Nithya Shikarpur

This thesis studies methods of malware detection, specifically for IoT devices. IoT technology has seen rapid development due to which it has become indispensable in our lives. However, its security standards have not evolved as fast, resulting in serious security concerns. To add to the problem, a large percentage of the network traffic is encrypted. This traffic includes malicious communication which cannot directly be analysed due to encryption. Hence, this is a study of the challenges and possible solutions for malware detection on IoT devices using TLS encrypted network packets. Joy [9] is a software developed by Cisco to extract data features from network capture files. It also uses a simple logistic regression model to identify malicious packets. For classification, we have used only the TLS features (extracted using Joy) due to their presence in both encrypted and unencrypted traffic. Unlike in Anderson et al. [1], we have studied malware detection specifically for IoT devices. Due to a lack of computing resources on these devices, detection of malware has to either be done at a network gateway or in the cloud. We have implemented both types of models: one, a lighter weight logistic regression model (reduced the number of features drastically by using a feature ranking system) implemented on a router; second, a deep learning based model comprised of 2 autoencoders trained on malware and benign data respectively.

Acknowledgements

I would like to sincerely thank my project supervisor Dr. Abhishek Tripathi for his insights and suggestions during the course of my thesis. I would also like to thank Dr. Dattatraya Kulkarni, head of Technology Pathfinding group for this wonderful opportunity. I thank Tirumaleswar Reddy for his guidance on the malware-related aspects of this project. I would like to thank the entire Technology Pathfining group at McAfee for always helping me and answering my questions. I also thank Prof. Hemant Rathore, my co-supervisor for his support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Malware threats have always been a concern in both the personal and professional world. Given the abundance of IoT devices and the amount of personal data that they have access to, malware detection has become a very important aspect of IoT technology. The motivation behind studying malware detection for these devices, in particular, is two-fold: first, the gaping vulnerabilities in the security of IoT devices; second, the need to perform malware detection over encrypted network traffic.

Internet of Things (IoT) has found a use case in almost every setting. Smart devices encompass a large spectrum of gadgets ranging from small wearable smart watches to bigger air conditioning systems. IoT devices have found their way into several fields including medicine, entertainment and governance. A report by Business Insider, states that there will be 30 billion IoT devices by the year 2020 [5].

This surge in IoT devices has developed too quickly for the security standards to be maintained [8]. This has made these devices highly vulnerable to malware attacks. Due to a lack of computational resources on these devices, it is difficult to use traditional malware detection methods for the same.

TLS (Transport Layer Security) is a cryptographic protocol that allows data being transmitted over the internet to be encrypted. According to a report that Fortinet Networks generated in 2018, almost 72.2% of the network traffic is encrypted with HTTPS (which uses TLS) [16]. This number has significantly increased from 50% in 2016. Given the sharp increase in the use of TLS encryption, it is safe to assume that its use will keep going up in the next few years.

While it is true that an increased use of HTTPS over the internet ensures greater security, it has also allowed malware to encrypt its traffic, thus making it hard to detect. Earlier methods of

detection used the port numbers, IP addresses[10] or patterns in the payload[13]. These features cannot be used with encrypted traffic as they become unavailable with simple feature extraction. Even if we use 'deep packet inspection' to extract these features, it becomes too costly in terms of both time and memory. Even metadata including packet lengths, inter-packet arrival time etc. as used in Anderson et al. [1] may not be completely reliable due to packet padding and other packet encryption techniques.

Hence we rely on features that can be extracted from encrypted network traffic, i.e., TLS Features. Each TLS encrypted exchange of packets begins with a TLS handshake involving the exchange of a few unencrypted packets. We extract our features from this series of packets.

Joy [9] is a software developed by Cisco to extract data features from live network traffic or network packet capture files(pcap). They have extracted 5 types of features including TLS information, Flow Metadata, Byte Distribution, Sequence Of Packet Lengths and Sequence Of Packet Inter-Arrival Times for a series of packets called a 'flow'. Their analysis on the importance of these feature sets for malware detection show that TLS features are vital for the classification. Hence, based on their results and our experiments, we have used the TLS information alone for classification.

In this project, we specifically study malware detection in the context of IoT devices. Anderson et al. [1] uses a logistic regression model for classification of flows. In an attempt to reduce the size of the logistic regression model so that it can be implemented on the router (as it cannot be run on an IoT device), we propose feature reduction based on a feature ranking system. In addition, we also use a more complex deep learning model: a model comprising of 2 autoencoders, each trained on malware and benign data respectively. The output of the model is based on the output of both autoencoders.

The following report is organised as follows: Chapter 2 is a brief overview of malware detection and TLS encrypted communication. Chapter 3 is about malware detection for IoT devices. Chapter 4 is about previous related work in the field of malware detection (Joy software). Chapter 4 and 5 are a description of the experiments carried out and the results generated.

Chapter 2

Malware Detection and TLS

2.1 Malware Detection

Malware or Malicious Software refers to harmful software aimed to infect individual computers or enterprise networks. They exploit vulnerabilities in the systems to gain access to personal data or other systems connected to the network.

The first documented virus in the early 1970s, called the 'Creeper Worm', was a self-replicating program [4]. Having started as a fun game in Bell labs, malware has now turned into an omnipresent threat.

Malware authors use malware to steal financial and personal information that can either be used directly, or held as leverage for a ransom (ransomware). In 2018, SonicWall recorded a record-breaking 10.58 billion malware attacks [12]. Hence, it is important to detect malwares and protect the privacy of one's data.

2.1.1 Malware As An Anomaly

Anomaly detection is the identification of rare items or observations which raise suspicions by differing significantly from the majority of the data [2] .

Malicious software is not as common as benign software in the real world. Hence, for several detection models, malware can be thought of as an anomaly rather than as a different class type. We will view malware as both a class type and an anomaly in the models that we present.

2.2 Encrypted Network Traffic

According to a report generated in 2018 by Fortinet Networks, almost 72.2% of the network traffic is encrypted with HTTPS (HTTPS uses TLS encryption) [16]. This percentage has steeply increased from 50% in 2016. This increased use of encrypted network traffic has not only sheltered benign data from criminals but has also sheltered malware traffic from the eyes of several innocent individuals.

Having established the abundance of encrypted network traffic in today's world, it becomes imperative that we use features that can be extracted from this encrypted traffic. Methods used in [10] and [13] use IP addresses, ports and patterns in the payload as features for detection. However all these features are either unavailable or too expensive (in terms of memory used and time taken to extract the features) with encrypted traffic.

Since most of the encrypted traffic uses TLS (Transport Layer Security), we use features extracted from this protocol for classification.

2.2.1 TLS

Transport Layer Security is a cryptographic protocol used to secure communications over the internet. Every time a connection is made between a server and a client, a TLS handshake takes place between the two parties. Although, the data exchanged during the communication between client and server is encrypted because of TLS, the exchange of parameters during the TLS handshake can give us meaningful insight into the type of communication taking place.

2.2.2 TLS Handshake

The TLS handshake takes place in order to establish a TLS connection. During a TLS handshake, the following are achieved by client and server:

- Version of TLS to be used is specified.
- A cipher suite (asymmetric encryption algorithm used for key exchange) is decided upon.
- Identity of the server is authenticated.
- Same session keys are generated at both ends. These keys are used for the symmetric encryption of data between both parties.

During the TLS handshake, a series of packets are exchanged between client and server to ensure a secure connection. The steps involved are [15]:

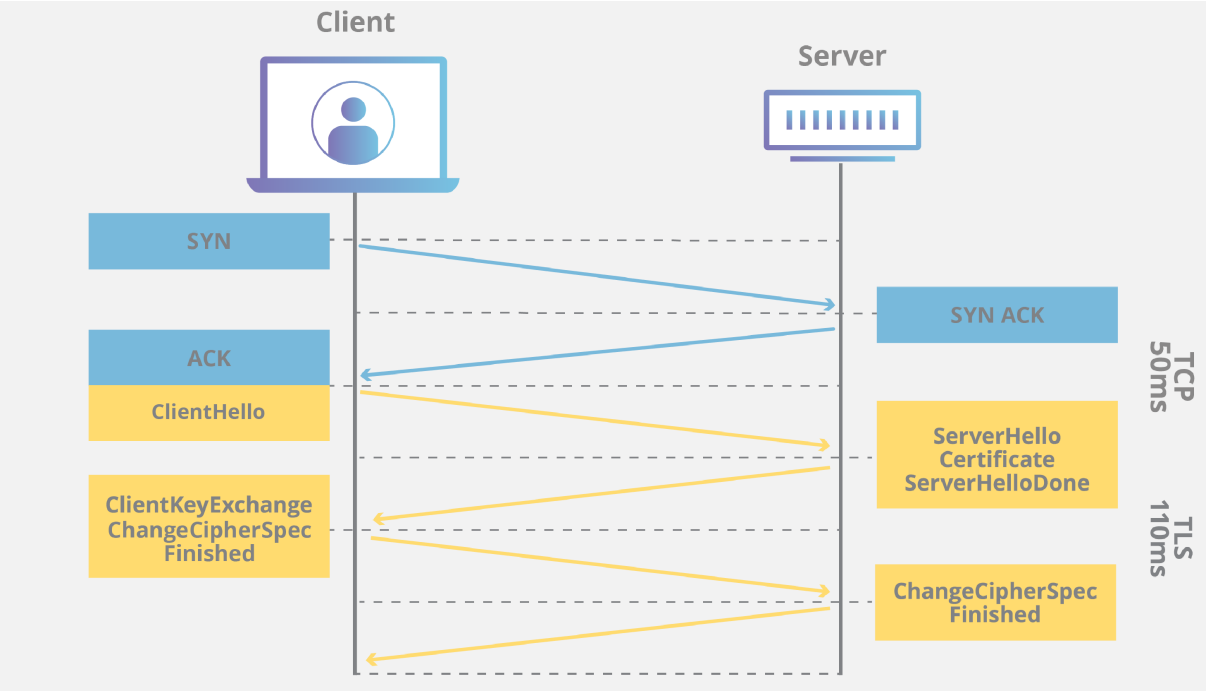


FIGURE 2.1: Basic TLS Handshake
Source: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/

1. **Client Hello**
   This message from the client to the server includes the TLS version and cipher suites supported by the client and a client random (string of random bytes).

2. **Server Hello**
   The server responds with a message including an SSL certificate (used by client to check server authenticity), the selected cipher suite and a server random (another string of random bytes).

3. **Server Authentication**
   The client verifies the server's SSL certificate with the certificate authority that issued it.

4. **Premaster Secret**
   The client sends a premaster secret (another string of random bytes) encrypted with the server's public key (present in the certificate). This can be decrypted only with the corresponding private key (present only with the server).

5. **Private Key Used**
   Server decrypts the premaster secret.

6. **Session Keys Created**
   Using the premaster secret, client and server generate the same session keys.

7. **Client Ready**
   A 'finished'message, encrypted with the session key is sent by the client.

8. **Server Ready**
   A ‘finished’message, encrypted with the session key is sent by the server.

9. **Secure Symmetric Encryption Achieved**
   Encrypted communication begins. Data is symmetrically encrypted with the session key generated.

Chapter 3

Malware Detection For IoT Devices

3.1 IoT Devices

Internet of Things devices are nonstandard computing devices that connect to the internet and have the ability to transmit data[6]. IoT devices can be broadly classified into the following sectors:

- **Consumer**: smart watches, smart TVs, smart speakers etc.
- **Enterprise and Industry**: smart city technologies used to monitor traffic or weather conditions

3.1.1 Importance of Malware Detection For IoT devices

The use of IoT devices has become increasingly popular in an individual's personal and professional life. The ease with which these 'smart' devices can simply connect to the internet and make decisions based on the data that they receive from other such devices make them very lucrative for daily use.

Security standards have not been able to keep up with the rapid developments in IoT technology [8]. A few challenges that IoT devices face in the field of security include[7]:

1. **Lack of proper manufacturing standards of IoT devices**
   Several IoT devices are manufactured without security being a primary concern. For instance, data might be sent from a device with unencrypted connections or the device may still be discoverable even after pairing. Ultimately, the manufacturers are not required to maintain any particular level security due to a dearth of any security standards.

2. **Update management**
   As new vulnerabilities are discovered, new software updates are released for devices. However, if these updates aren't installed (maybe due to the consumer's unawareness), these devices become open targets for attacks.

3. **Lack of Physical Security**
   IoT devices can be physically tampered with giving rise to more security concerns. IoT devices are known to be low-cost devices. Hence adding sensors to physically protect the device from any threat is not cost-effective for the manufacturer.

4. **Lack of Consumer Awareness**
   Due to the novelty of IoT devices, some users are still unaware of basic security measures that should be taken. T installing regular software updates and resetting all passwords on the device.

3.1.2 Challenges Faced For Malware Detection On IoT Devices

While trying to build a malware detection model for the IoT devices, we came across certain constraints that we had to work with. These constraints are listed below:

1. **Lack of computing resources**
   Malware detection systems, whether rule-based or machine learning based, require a significant amount of computing resources. Since IoT devices are built with low computing capacity, we cannot use traditional malware detection techniques for these smart devices. We thus use a simple machine learning model in a network gateway (router) and a more complex one in the cloud.

2. **Resources on a router are also limited**
   Although better than IoT devices, the router also has a limited amount of memory. For this reason, we have tried to reduce the size of a simple logistic regression model. This is done by reducing the number of features based on a feature ranking mechanism. The highest ranked features are retained to make a decision while the others are discarded.

3. **Decision has to be made in milliseconds**
   Once a TLS handshake is established it becomes impossible to stop the connection. Hence, the prediction has to be generated in real time, before the handshake is complete (within a few milliseconds). As mentioned before, feature reduction helps in generating a faster prediction from the model.

Chapter 4

Related Work

4.1 The Joy Package

Joy is a software developed by Cisco to extract data features from live network traffic or network traffic capture files (pcap)[9]. Researchers from Cisco analysed the detection of malware and its types using these features. These findings are presented in their paper: Deciphering Malware's Use of TLS(without decryption)[1].

With the advent of computer security and privacy, most of the network communications are encrypted. Malware samples are taking advantage of this encryption to go undetected. Hence, Joy allows us to use features from encrypted traffic to detect the type of flow.

4.1.1 Flow

The data features extracted using Joy are condensed using a flow-oriented model. A flow is defined as a series of packets containing the same:

1. Source/Destination port
2. Source/Destination address
3. Layer 3 protocol (Network Layer Protocol)
4. Class of Service (classification of specific traffic at layer 2)

4.1.2 Features Extracted

The features extracted for each network flow using Joy can be put into 5 broad classes:

1. **TLS information (186 features)**
   These features were binary (except for *length of public key* which was numeric).
   (a) Cipher suites (176)
   (b) TLS extensions (9)
   (c) Length of Public Key (1)
2. **Flow Metadata (7 features)**
   (a) Number of inbound bytes
   (b) Number of outbound bytes
   (c) Number of inbound packets
   (d) Number of outbound packets
   (e) Source port
   (f) Destination port
   (g) Total duration of flow in seconds
3. **Byte Distribution (256 features)**
   Normalized frequency of each of the 256 bytes present in the flow
4. **Sequence of Packet Lengths (100 or 3600 features)**
   A flattened Markov chain representation of packet lengths is used. The number of features is determined by the value of the compact variable in the software.
5. **Sequence of Packet Inter-arrival times (100 or 900 features)**
   A flattened Markov chain representation of packet inter-arrival times is used. The number of features is determined by the value of the compact variable in the software.

4.1.3 Classification Model

One of the classification models run in their study, pertaining to this problem statement, was a binary classification model. A logistic regression model (with l1 penalty) was trained and tested on a dataset containing approximately equal number of malware and benign flows. The model was trained and tested using different combinations of the feature sets.

The results pertaining to binary classification can be broadly summarised as follows:

- The classifier works best (highest accuracy) with all features considered.
- The performance of the classifier falls significantly if the TLS features are removed.

4.2 Feasibility Study

To check the feasibility of using the features produced with Joy for detection of malware, a few experiments were carried out on an open source dataset of malware and benign samples.

4.2.1 Data

The data used was collected by Stratosphere Lab, CVUT University, Prague [14]. The data included malware as well as benign packet captures.

- **Malware Data**
  Collected as a part of the Malware Capture Facility Project. It includes long-term captures of malware samples. These samples are mostly botnets run in commercial sandbox environments. All samples are run on a Windows-XP based environment.
- **Benign Data**
  These captures include benign network traffic captured from different operating systems including Linux and Windows.

4.2.2 Experiments Conducted

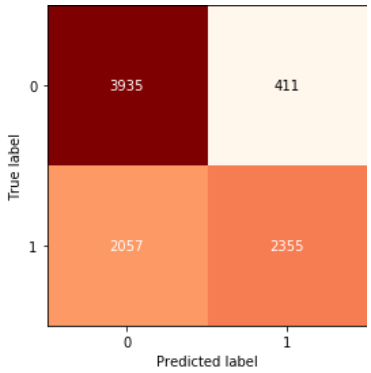
1. **Classification With Balanced Dataset**
   The data consisted of 43,786 flows. We define the ratio of malware flows is to benign flows in the data distribution (train and test sets) as the **MB ratio**. The MB ratio, here, was 1:1. The data was split into an 80-20 train-test split. Logistic Regression Classifier and Random Forest Classifier were used on the dataset.

   There were 2 experiments carried out:

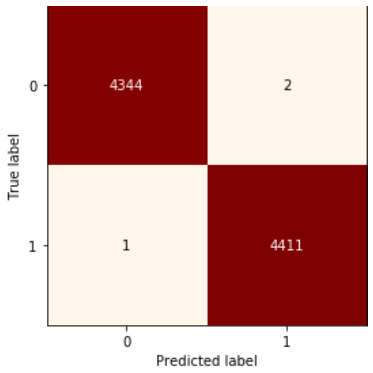
   (a) No Feature Selection
   All 649 features generated using joy were used for the classification.

   (b) With Feature Selection
   Features were assigned a feature importance score using the Random Forest Classifier. Features with an importance of 0 were dropped and the other features were retained.

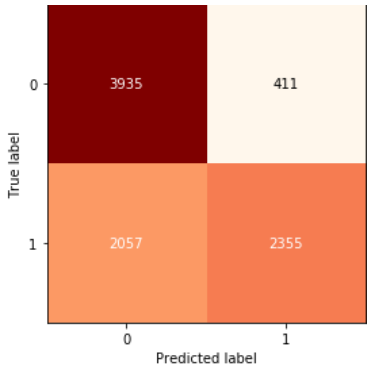
   Result
   With no hyperparameter tuning and using all the features extracted from joy, Random Forest performs much better than Logistic Regression. In addition, it can be noted that there is no significant change in classification using the above method of feature selection.



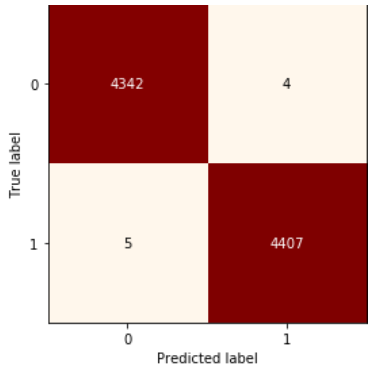
a. Logistic Regression with no Feature Selection



b. Random Forest with no Feature Selection



c. Logistic Regression with Feature Selection



d. Random Forest With No Feature Selection

TABLE 4.1: Confusion Matrices Of Experiment 1

2. **Classification With Imbalanced Dataset**

   The dataset in this experiment was purposely made to have a highly skewed MB ratio in the training and test data. This was done to simulate real-world data, where the percentage of malware samples is very low.

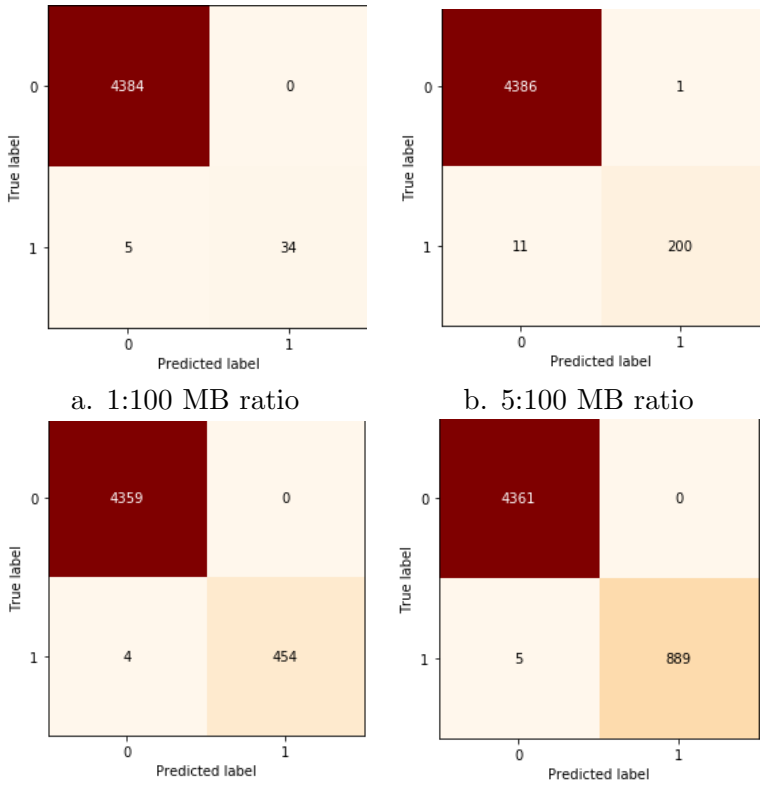
   Data used consisted of 21,893 benign flows. MB ratios of 1:100, 5:100, 10:100 and 20:100 were used. The number of malware flows were decided according to the MB ratio. The data was split into an 80-20 train-test split. All the features were used and the Random Forest Classifier was used for classification.

   Result

   Random Forest Classifier, as is evident from the confusion matrices, works well with the classification of highly imbalanced datasets.

3. **Classification Using Different Feature Sets**

   We next test the importance of each feature set in the identification of malware. We used each of the 5 feature sets - TLS, Byte Distribution, Sequence of Packet Lengths, Sequence of Packet Inter-Arrival Times and Flow Metadata. In addition to this, we also tried classification with all features except TLS (referred to as 'No TLS').



a. 1:100 MB ratio b. 5:100 MB ratio

c. 10:100 MB ratio d. 20:100 MB ratio

TABLE 4.2: Confusion Matrices Of Experiment 2

The data consisted of 43,786 flows including malware and benign flows in a 1:1 MB ratio. The data was split into an 80-20 train-test split. Both the Logistic Regression and Random Forest models were used.

Feature Set Used	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
TLS	96.66	93.32	99.7	96.73
No TLS [1]	72.73	85	55.27	66.98
Byte Distribution	60.99	72.4	35.67	47.79
Sequence Of Packet Lengths	83.28	94.28	70.89	80.93
Sequence of Packet Inter-Arrival Times	65.42	89.59	34.97	50.31
Flow Metadata	71.49	83.96	53.2	65.13

TABLE 4.3: Results using different features sets with logistic regression

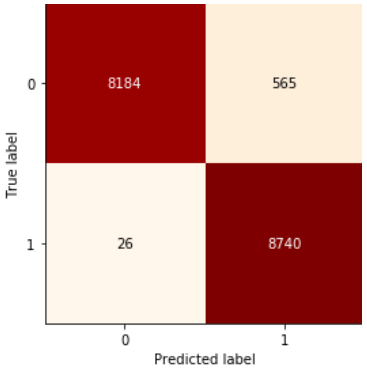
Result

It is evident from the confusion matrices and score metrics that Logistic Regression and Random Forest perform equally well using only the TLS feature set. This proves that we can use only TLS features for classification with a (light weight) Logistic Regression model. In addition, we can see a significant fall in the performance of Logistic Regression once we remove the TLS features. This finding agrees with the results in Anderson et al. [1].

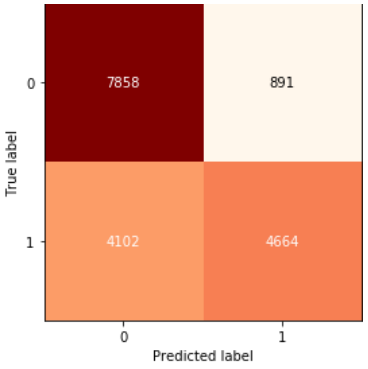
Since we are specifically interested in TLS features, we have only presented the confusion matrices using the TLS features and all features except TLS features for both models.

Feature Set Used	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
TLS	96.63	93.92	99.72	96.74
No TLS	99.44	99.87	99	99.43
Byte Distribution	99.1	99.52	98.68	99.1
Sequence Of Packet Lengths	83.84	95.45	71.15	81.5
Sequence of Packet Inter-Arrival Times	68.36	94.7	38.96	55.21
Flow Metadata	96.3	97.38	95.17	96.26

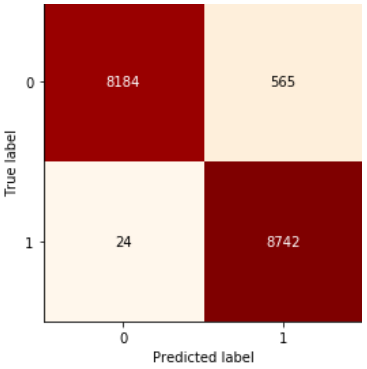
TABLE 4.4: Results using different features sets with random forest classifier



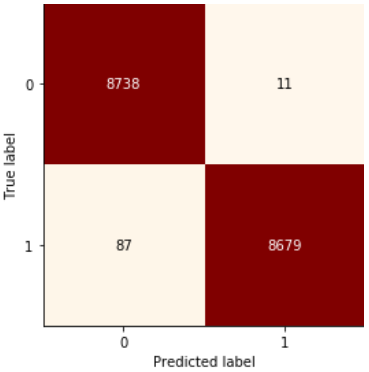
a. Logistic Regression With Only TLS Features



b. Logistic Regression With All Features Except TLS Features



c. Random Forest Classifier With Only TLS Features



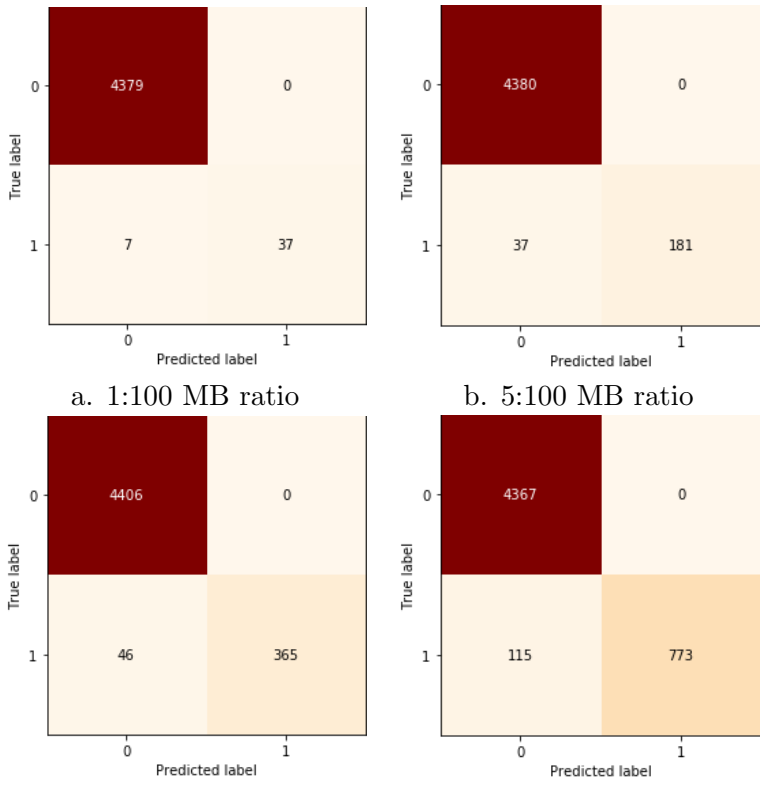
d. Random Forest Classifier With All Features Except TLS Features

TABLE 4.5: Confusion Matrices Of Experiment 3

4. **Classification Of Imbalanced Dataset With Only TLS Features**

   Having established that TLS features are good enough for classification with a logistic regression model, in this experiment, we tried to simulate the real world data by skewing the malware:benign ratio.

   Data used consisted of 21,893 benign flows. MB ratios of 1:100, 5:100, 10:100 and 20:100 were used. The number of malware flows were decided according to the MB ratio. The data was split into an 80-20 train-test split. Only the TLS features were used with a Logistic Regression model.



a. 1:100 MB ratio b. 5:100 MB ratio

c. 10:100 MB ratio d. 20:100 MB ratio

TABLE 4.6: Confusion Matrices Of Experiment 4

Result

The Logistic Regression model is fairly robust to the imbalance in the dataset. The F1 scores for the 1:100, 5:100, 10:100 and 20:100 MB ratio experiments were 91.36%, 90.73%, 94.07% and 93.08% respectively.

This fesibility study proves that it is possible to build a light weight Logistic Regression model which can be implemented with low computing resources to classify TLS encypted packets using only features from the TLS handshake. Using this idea we implemented 2 models, explained in the next chapter.

Chapter 5

Experiments Conducted

5.1 Data

5.1.1 Data Source

1. Stratosphere Lab, CVUT University, Prague

   (a) Malware data
   Collected as a part of the Malware Capture Facility Project. It includes long-term captures of malware samples. These samples are mostly botnets run in commercial sandbox environments. All samples are run on a Windows-XP based environment.

   (b) Benign data
   These captures include benign traffic from different operating systems including Linux and Windows.

2. McAfee LLC
   Benign data flows collected from a few IoT devices.

5.1.2 Feature Extraction

All the data collected was in the form of pcap (Packet CAPture files). These files capture the network packets coming to and going from a device. TLS Features were extracted from the pcap file into a csv file using a python script.

Features used include:

- Cipher suites (176)

- TLS extensions (9)
- Length of Public Key (1)

5.1.3 Data Imbalance

There are two aspects to the data imbalance that we face for this dataset.

- Desired Imbalance
  Malware traffic is significantly less frequent than benign traffic in the real world. Since, ideally, a model should be trained on the same distribution as the test set, we try to keep the MB ratio as close to the real world as possible.
- Actual Imbalance In Dataset
  In the dataset collected, there was an unusual problem of having a greater number of malware samples than benign samples. This was because:
  - The Malware Capture Facility Project (conducted by Stratosphere Lab) captured a greater number of malware samples than benign samples.
  - At McAfee LLC there were a fixed number of IoT devices (7 devices) present. Each device has a fixed number of profiles (meaning that each device can only produce a fixed number of unique flows). Hence, collecting data from these devices over a longer period of time wasn't practical as flows would keep getting repeated in an unrealistically high frequency.

  Hence, despite a great number of malware samples present in our dataset, we had to use only a small subset of these samples for training and testing. To ensure that our model doesn't overfit on the malware data (due to its small size), we implemented a bootstrapping technique during training. In this, multiple models were trained with random sampling (with replacement) of malware data. An average of these models was taken as the final trained model.

  One point to note is that due to the high imbalance in the dataset, accuracy is not a good metric to measure a model. Hence, throughout this project we have relied on F1-score[1] (harmonic mean of precision[2] and recall[3]) to judge how good a model is.

[1]
$$\text{f1-score} = \frac{2 * precision * recall}{precision + recall} \tag{5.1}$$

[2]
$$precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{5.2}$$

[3]
$$recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{5.3}$$

Two models were implemented on the data. One is a simple, light-weight logistic regression model. The other is a more complex deep learning based model composed of 2 autoencoders trained on benign and malware traffic respectively.

5.2 Logistic Regression

In this model, data is fit into a linear regression model, which is then acted upon by a sigmoid function to predict the probability of the target categorical dependent variable being a certain value[11].

5.2.1 About The Model

- Linear Model
  This is a model where all the features (X) are added up based on a given set of weights (W).
  $$z = w_1 * x_1 + w_2 * x_2 + ... + w_n * x_n = W * X \tag{5.4}$$

- Sigmoid Function
  This function has a range of (0, 1). It is used to convert the output of the linear model into a probability (value between 0 and 1).
  $$sigmoid(z) = \frac{1}{(1 + e^{-z})} \tag{5.5}$$

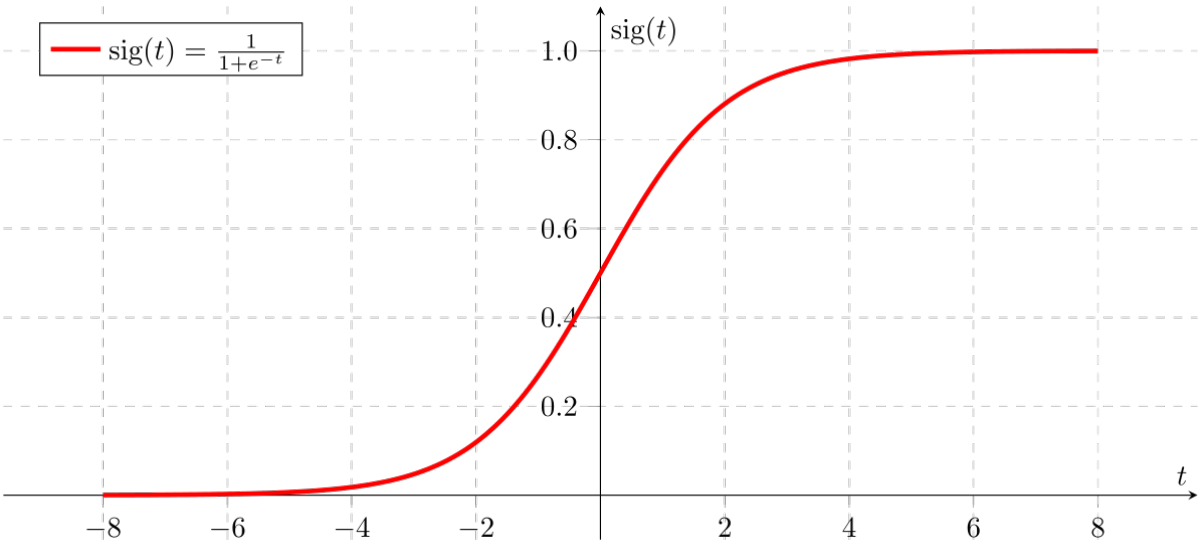


FIGURE 5.1: Sigmoid Function
Source: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

- Conditional Probability
  Conditional probability can be thought of as the probability of an event A occuring, given that a set of events B has already occured.
  For this problem, the target values are {0, 1} (benign or malicious). Hence the conditional probability that the target value is 1, given a set of feature values and weights is modeled to be the output of the sigmoid on the linear sum.
  Mathematically, the prediction is calculated as,

$$\hat{y} = P(y = 1|X; W) = sigmoid(W * X) \tag{5.6}$$

  where X = feature vector and W = weight vector

5.2.2 Training The Model

The question now is, 'how are the weights calculated?'. These weights are obtained by minimizing a loss function on the train data set. In this case, the loss is calculated as binary cross-entropy.

$$L(\hat{y}, y) = y * log(\hat{y}) + (1 - y) * log(1 - \hat{y}) \tag{5.7}$$

where $\hat{y}$ = predicted probability and y = true class

5.2.3 Weights Generated

The logistic regression model generates a weight corresponding to each feature. The weight tells us about:

- **The importance of the feature**
  The absolute value of the feature is directly proportional to how important that particular feature is in making a decision. Hence, arranging features according to the absolute value of their weights can help us rank the features. This helps us reduce the size of the model by cutting down on the less important features.
- **Contribution of the feature towards a prediction as malware**
  A positive weight implies that the feature contributes towards a prediction as malware (a conditional probability closer to 1). Similarly, a negative weight implies that the feature contributes towards a prediction as benign (a conditional probability closer to 0).

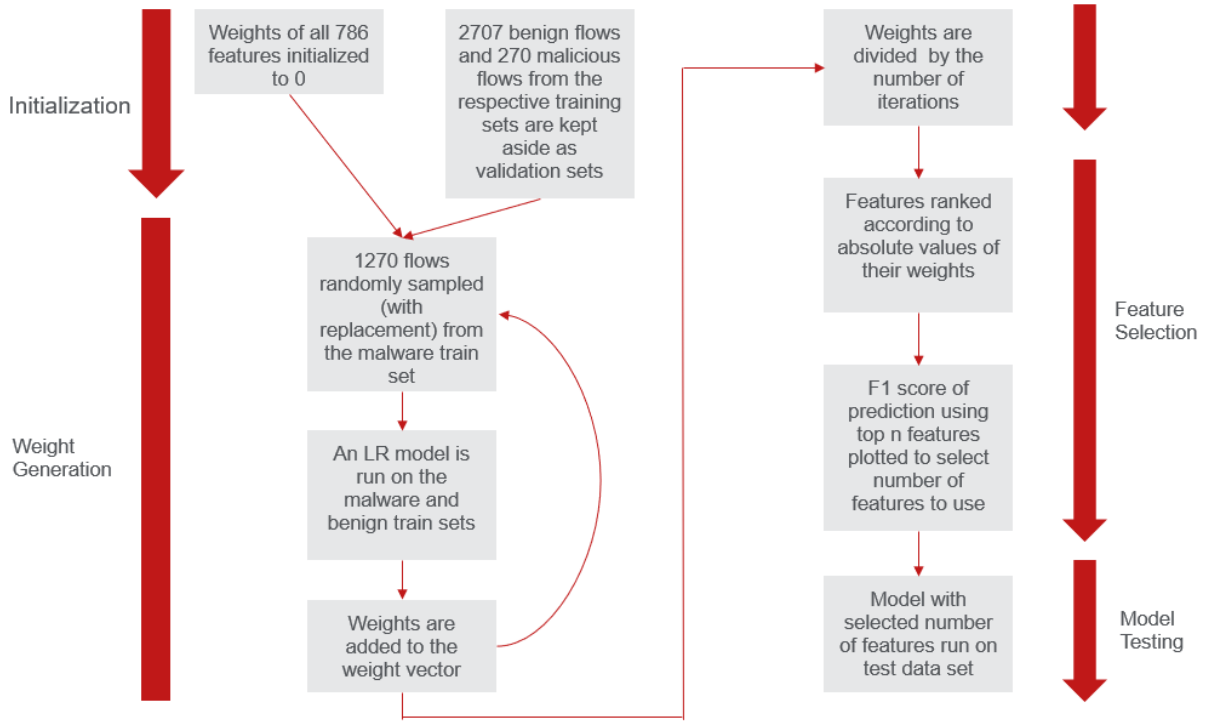


FIGURE 5.2: Algorithm For The Logistic Regression Model

5.2.4 Basic Logistic Regression Model (Experiment 1)

For this experiment we considered 10,000, 2,707 and 7,000 benign train, validation and test flows and 1,270, 270 and 700 malicious train, validation and test flows respectively.

To model true data, we ensured an MB ratio of 10:100. This meant that only a small subset of malware samples could be used to train the model. In order to prevent overfitting for the selected samples of malware, we trained the model 10 times and averaged the weights produced each time. For each training, a new subset of malware samples was randomly sampled with replacement.

All the features were ranked according to the absolute value of their weights as discussed above. By plotting the F1 score on the validation data against the number of features used, it is evident that around 200 features are enough for the classification. The F1 score stagnates at 98.92% after this point.

Results

The above model gave us an F1 score of 98.92% with an accuracy of 99.8%, precision of 99.7% and recall of 98.1%.

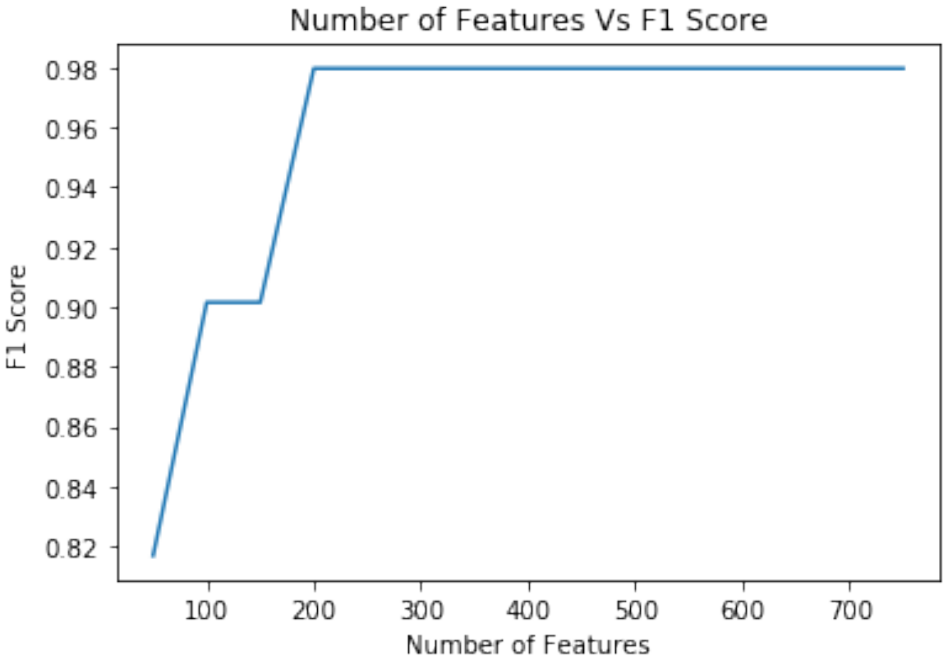


FIGURE 5.3: Number of features Vs F1 score

Feature	Feature Type
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	Cipher Suite
signature algorithms	Extensions
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	Cipher Suite
renegotiation info	Extension
TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256	Cipher Suite
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	Cipher Suite
application layer protocol negotiation	Extension
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	Cipher Suite
TLS_RSA_WITH_AES_128_GCM_SHA256	Cipher Suite
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Cipher Suite

TABLE 5.1: Top 10 Weighted Features

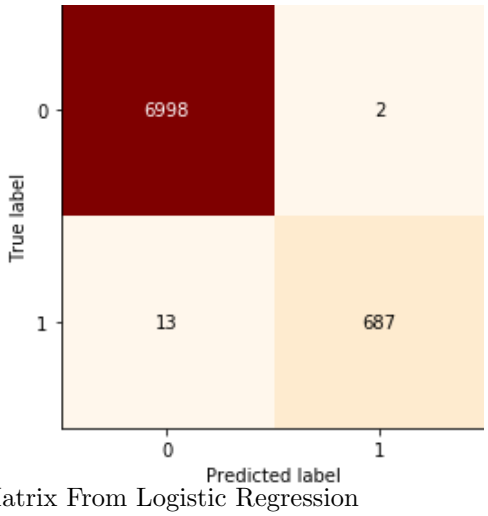


FIGURE 5.4: Confusion Matrix From Logistic Regression

5.2.5 Logistic Regression With Thresholds (Experiment 2)

To increase the confidence with which the model makes a prediction, we introduce 2 thresholds. If the predicted probability is above the malware threshold (th_m), then the sample is predicted as malware (1). If the predicted probability is below the benign threshold (th_b), then the sample is predicted as benign (0). If the predicted probability lies in the middle of the two thresholds, then the sample is labelled as undecided (-1).
Mathematically, for a sample X with predicted probability $\hat{y}$, the predicted class p is

$$p = \begin{cases} 1 & \hat{y} > th_m \\ 0 & \hat{y} < th_b \\ -1 & th_b \leq \hat{y} \leq th_m \end{cases}$$

Results

Empirically, we set $th_m = 0.99$ and $th_b = 0.01$. This gave us an F1 score of 84.1%, precision of 100%, recall of 72.57% and accuracy of 70.29%. A 100% precision implies that this classifier never misclassifies a benign sample as malware. In other words, there are 0 false positives. The drop in the values of other metrics is due to the high number of unclassified samples. This is evident in confusion matrix and scatter plot of test samples presented. All the samples between the two threshold lines in the scatter plot are unclassified.

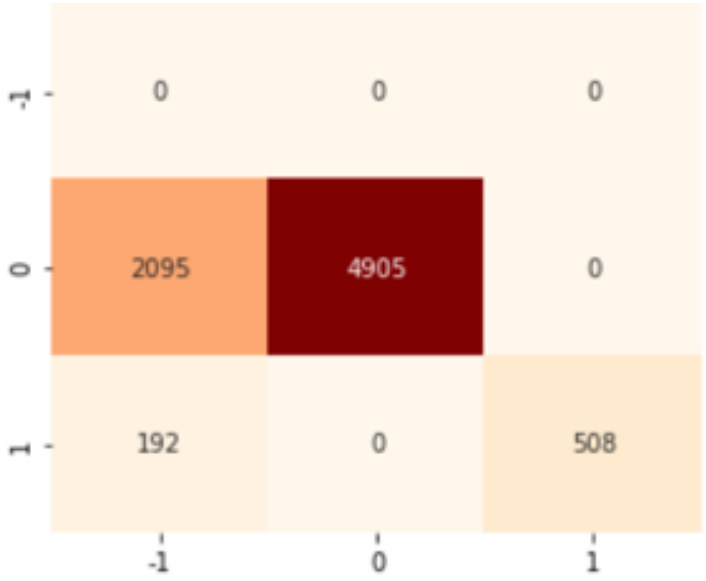


FIGURE 5.5: Confusion matrix for logistic regression with thresholds

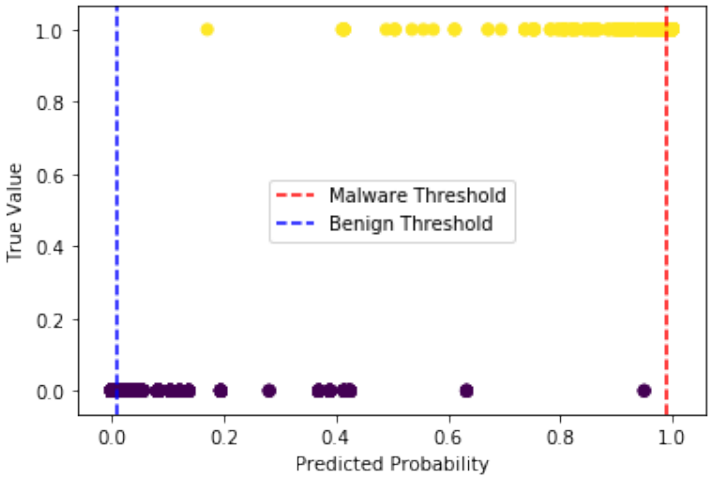


FIGURE 5.6: Scatter plot of prediction probability of the test samples

5.3 Autoencoders

Autoencoders are a type of unsupervised deep learning model. In this model, the input is encoded into a smaller size and then decoded into the original size. This process helps with removal of noise in the input data.

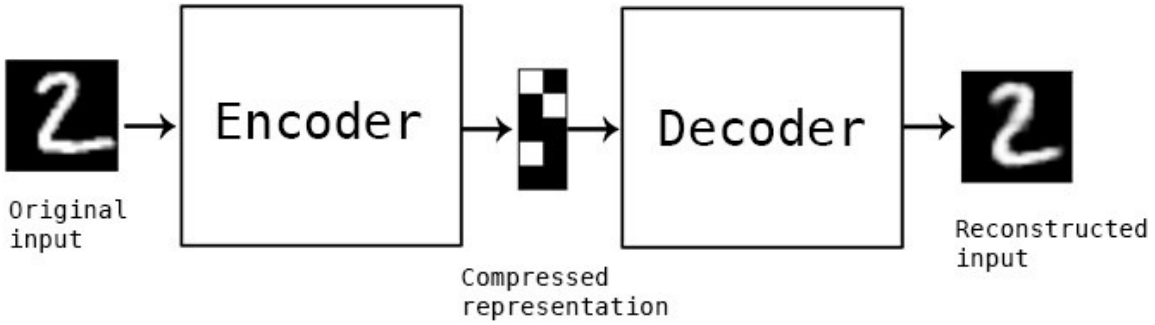


FIGURE 5.7: Representation of autoencoder on digit 2
Source: https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

5.3.1 Components Of An Autoencoder

Broadly speaking, an autoencoder can be divided into 4 components[3]:

1. **Encoder**
   This part of the model converts the input into a lower dimension.

2. **Bottleneck**
   This layer contains the compressed representation of the data.

3. **Decoder**
   This part of the model converts the low dimension representation into the input size.

4. **Reconstruction Loss**
   This is the loss that is minimized during back propogation.

5.3.2 Autoencoders For Anomaly Detection

Autoencoders are useful for anomaly detection. Since autoencoders have the ability to learn a type of object, they are used to catch any anomalies.
The autoencoder model is trained on the majority class. Hence, when an anomaly is passed through the model, the reconstruction error should be significantly higher than usual.

5.3.3 Architecture

In the case of malware detection, malware is the anomaly. Hence, ideally, the autoencoder should be trained on a set of benign data and should consequently recognise malware samples as anomalies. However, due to a significantly lesser amount of benign data, we have used 2 autoencoders.
One autoencoder was trained on the benign dataset and the other trained was on the malicious dataset. If both predict a sample to be of the same class (malware/benign) then the predicted class becomes that prediction. However, if both autoencoders provide different classes as the predicition, then the sample is classified as undecided.

Both autoencoders consisted of only one hidden layer with a relu activation function. The benign autoencoder had 8 neurons whereas the malware autoencoder had 16 neurons in the inner layers respectively. These values were determined empirically.

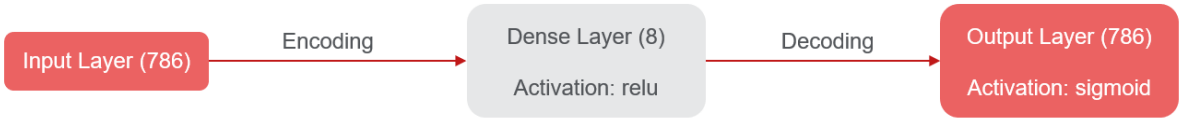


FIGURE 5.8: Benign Autoencoder Architecture

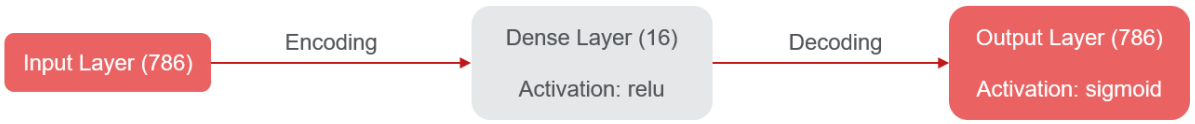


FIGURE 5.9: Malware Autoencoder Architecture

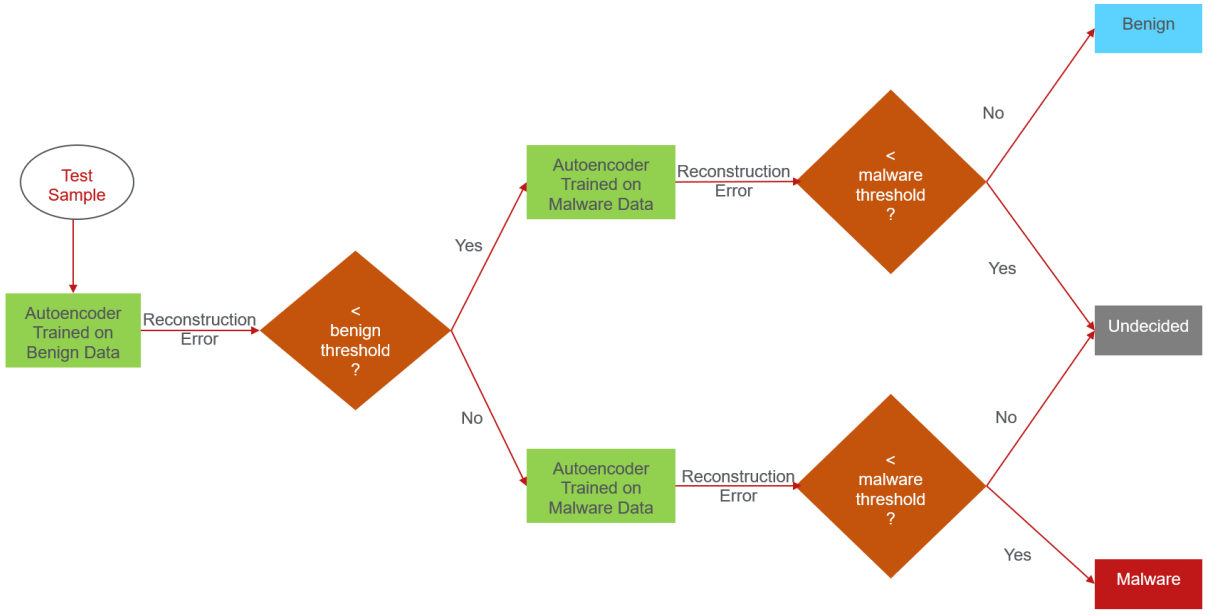
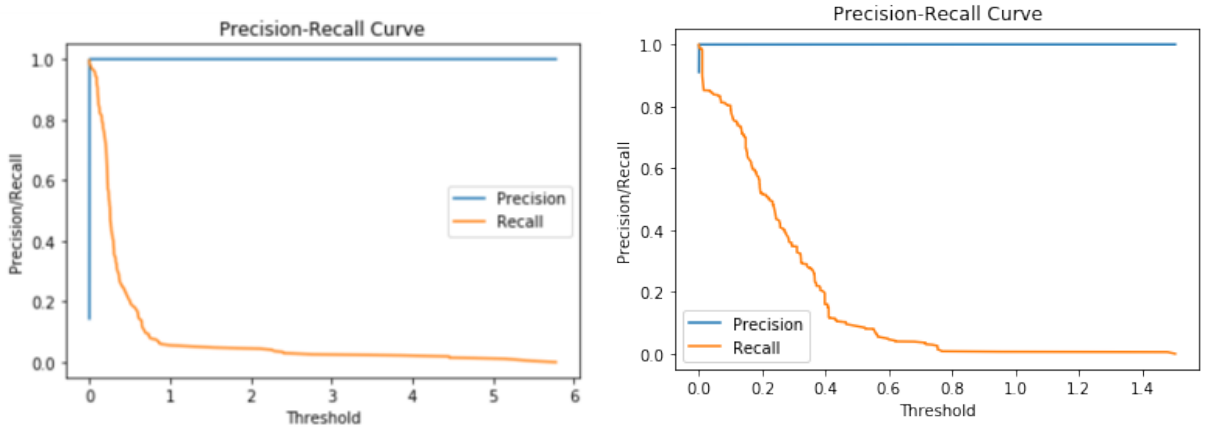


FIGURE 5.10: Complete Architecture Of Autoencoder

5.3.4 Experiment

There were 10,000, 2,707 and 7000 benign flows in the train, validation and test datasets. There were 49,104, 12,276 and 700 malware flows in the train, validation and test sets. The loss used was binary crossentropy with an Adam optimizer.

The threshold for both autoencoders was decided by plotting a precision-recall curve. The thresholds were set as 0.01 and 0.001 for the benign and malignant autoencoders respectively.

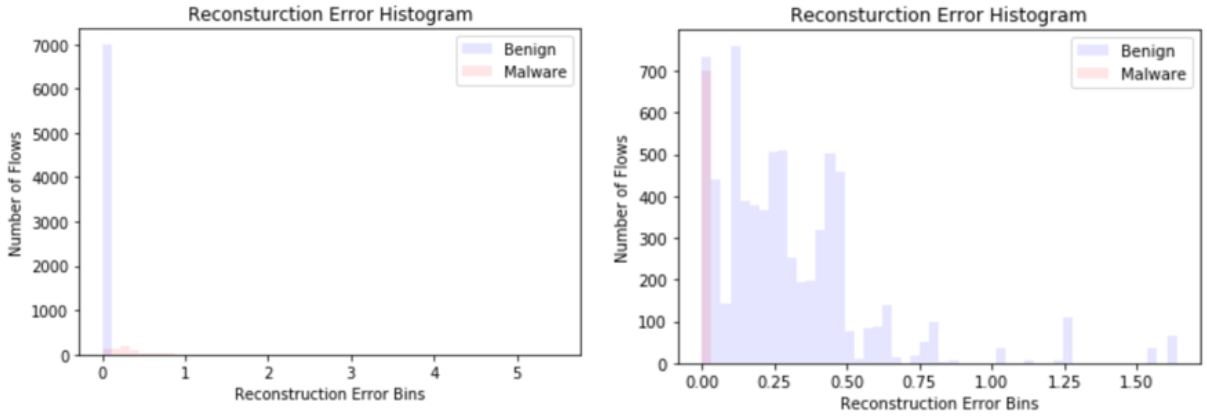


(A) Precision-Recall Curve For Benign Autoencoder (B) Precision-Recall Curve For Malware Autoencoder

FIGURE 5.11: Precision-Recall Curves For Autoencoder

The reconstruction error histograms for both the benign and malware autoencoders help us analyse how effective the models are. As is evident from the figures, the models are able to

detect the classes that they were trained on (i.e., the reconstruction error for benign samples is low in the benign autoencoder; same for malware).



(A) Reconstruction Error Histogram For Benign Autoencoder

(B) Reconstruction Error Histogram For Malware Autoencoder

FIGURE 5.12: Reconstruction Error Histograms For Autoencoders

Results

This model gave us an F1 score of 99.13% with an accuracy of 98.02%, precision of 100% and recall of 98.28%. Just like the logistic regression model with thresholds, this model also has 0 false positives (100% precision). However, the number of unclassified samples has reduced drastically in this model.

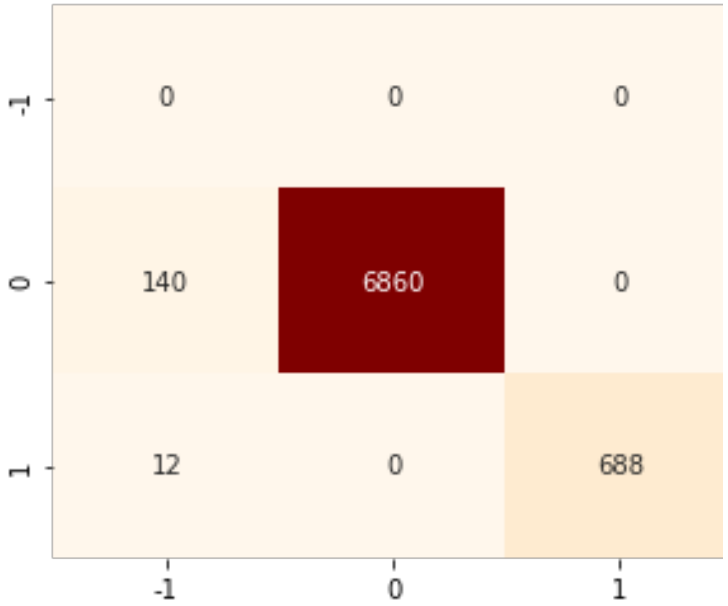


FIGURE 5.13: Confusion Matrix From Autoencoder

Chapter 6

Wraping It Up

6.1 Observations

- The basic logistic regression model is able to classify the samples with a very high F1-score. Since this model requires only some scalar multiplications and a sigmoid function for prediction, it is not very resource intensive. Hence this can be easily implemented on the router to detect malicious network packets.
- Once thresholds are introduced in the logistic regression model, almost 30% of our test samples went unclassified. Even though this reduces the number of false positives to 0, it shows us that in our logistic regression model, several samples aren't classified with a high enough confidence.
- The autoencoder is able to classify the samples with 100% precision (no false positives). This means that it predicts the malware class with a very high confidence. However, this autoencoder does take up a lot of memory for both training and classification. Hence it will not be feasible to implement this model on an IoT device or router. This can be used as a classifier (implemented in a cloud) on samples that a simpler model is not able to confidently classify.

6.2 Conclusion

In this report we were able to establish 2 methods for malware detection on IoT devices. This is necessary due to the large number of IoT devices present today and a severe lack of security standards in these devices. Due to encryption of a large proportion of network packets, we use features from the TLS handshake for classification.

We proposed a simple light-weight Logistic Regression model to be implemented on the router to classify network packets (as the model cannot be implemented on the smart device due to resource constraints). This model was made lighter by reducing the number of features using a feature ranking technique. We also studied the performance of this model with the introduction of two thresholds for classification. Even though this reduced the number of false positives to 0, it left a lot of samples unclassified. We also proposed a model comprising of two autoencoders. Given the constraints on the amount of data we have, this model performed exceptionally well. This model predicted a value if the output of both autoencoders were in agreement, else it could not predict a value. From our experiments we learned that this model never misclassified any flow. Being a heavier model, this can be implemented in a cloud used to classify certain suspicious flows.

In conclusion, data extracted from the TLS features can be used for malware detection in IoT devices. These features can be used with encrypted as well as unencrypted network traffic. These features work well with both simple models like logistic regression, random forest classifier and with more complex deep learning based models.

Bibliography

[1] Blake Anderson, Subharthi Paul, and David McGrew. "Deciphering Malware's use of TLS(without Decryption)". In: (2016).

[2] *Anomaly Detection*. URL: https://en.wikipedia.org/wiki/Anomaly_detection.

[3] *Auto Encoder*. URL: https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726.

[4] *History Of Malware*. URL: https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/.

[5] *IoT BI*. URL: http://www.businessinsider.com/%20how-the-internet-of-things-market-will-grow-2014-10.

[6] *IoT Definition*. URL: https://internetofthingsagenda.techtarget.com/definition/IoT-device.

[7] *IoT Security*. URL: https://www.intellectsoft.net/blog/biggest-iot-security-issues/.

[8] *IoT Security Standards*. URL: https://analyticsindiamag.com/iot-security-standards-hackers/.

[9] *Joy repo*. URL: https://github.com/cisco/joy.

[10] Thomas Karagiannis et al. "Is P2P dying or just hiding?" In: (2004).

[11] *Logistic Regression*. URL: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

[12] *Malware Statistics*. URL: https://www.comparitech.com/antivirus/malware-statistics-facts/.

[13] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures". In: (2004).

[14] *Stratosphere Lab*. URL: https://www.stratosphereips.org/.

[15] *TLS Handshake*. URL: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake.

[16] *TLS stats*. URL: https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/threat-report-q3-2018.pdf.