A MACHINE LEARNING CLASSIFICATION APPROACH TO DETECT TLS-BASED MALWARE USING ENTROPY-BASED FLOW SET FEATURES

KINAN KESHKEH

UNIVERSITI SAINS MALAYSIA

2022

A MACHINE LEARNING CLASSIFICATION APPROACH TO DETECT TLS-BASED MALWARE USING ENTROPY-BASED FLOW SET FEATURES

by

KINAN KESHKEH

Thesis submitted in fulfilment of the requirements
for the degree of
Master of Science

November 2022

ACKNOWLEDGEMENT

First of all, I thank **Allah** for giving me the power and perseverance to complete my Master’s work. Nothing could have been done without his will.

I would like to sincerely thank my supervisors and mentors, Associate Professor Dr. Aman B. Jantan (main supervisor) and Dr. Nur Intan Raihana Ruhaiyem (co-supervisor), for their invaluable notes and guidance coming from their vast research experience during the entire period of my research work.

Most importantly, I would like to thank my wife, Riwa. Her support, encouragement, patience, and endless love were fundamental and the cornerstone in continuing my master’s study journey, especially during predicaments.

And I would extend my gratitude to Dr. Kamal Ibrahim Alieyan, who supported me in my academic journey. He shared many from his vast experience with me. Also, a big thank you to my brother Dr. Alaa Kadi for his support, especially when I first came to Malaysia.

I thank my parents, Nizar and Raghda, for their faith in me and for encouraging me to be as ambitious as I wished. Despite the distance between us, my mother was close to me, supporting me in all aspects and constantly reminding me to trust in Allah Almighty and that I have the ability to complete my studies in the presence of many life challenges.

I would also like to thank my brother, Besher, and my sister, Farah, for providing unconditional support, especially when I moved out to a new flat in Nov 2020. And last but not least, my Jordanian friend Dr. Fadi Saqallah for his always advice and suggestions during my study. He was like a big brother to me.

TABLE OF CONTENTS

ACKNOWLEDGEMENT .... **ii**

TABLE OF CONTENTS .... **iii**

LIST OF TABLES .... **ix**

LIST OF FIGURES .... **x**

LIST OF SYMBOLS AND ABBREVIATION .... **xiii**

LIST OF APPENDICES .... **xvi**

ABSTRAK .... **xvii**

ABSTRACT .... **xix**

CHAPTER 1 INTRODUCTION **1**

1.1 Background 1

1.2 Research Problem 3

1.3 Research Motivation 5

1.4 Research Questions 5

1.5 Research Objectives 6

1.6 Research Contribution 6

1.7 Research Scope and Limitations 7

1.8 Research Methodology 8

1.9 Thesis Outline 9

CHAPTER 2 LITERATURE REVIEW 11

2.1 Introduction 11

2.2 Background 11

2.2.1 Malware Taxonomy 12

2.2.2 TLS-based Malware 15

2.2.2(a) TrickBot 15

2.2.2(b) Dridex 16

2.2.3 Malware Analysis Techniques 17

2.2.3(a) Static Analysis 17

2.2.3(b) Dynamic Analysis 18

2.2.4 TLS Protocol Overview 19

2.2.5 Classification Assessment 27

2.3 Related Work 30

2.3.1 Features used in TLS-based Malware Detection 32

2.3.1(a) TLS/SSL Features 32

2.3.1(b) Contextual HTTP features 44

2.3.1(c) Contextual DNS Features 51

2.3.1(d) TCP Features 54

2.3.1(e) Packet Features 55

2.3.1(f) Flow and Flow-set Features 61

2.3.2 Feature Selection 68

2.3.3 ML Usage in TLS-based Malware Detection 69

2.3.3(a) Linear Regression 70

2.3.3(b) Classification 71

2.3.3(c) Neural Network 80

2.3.3(d) Summary of ML Usage 83

2.3.4 Critical Analysis of Strengths and Weaknesses 88

2.4 Summary 92

CHAPTER 3 RESEARCH DESIGN AND METHODOLOGY 94

3.1 Introduction 94

3.2 Research Methodology Flow 94

3.2.1 Literature Review and Problem Identification 96

3.2.2 TLSMalDetect Approach Design 96

3.2.3 Analyzing Results and Discussion 98

3.3 TLSMalDetect Approach ... 98

3.3.1 Phase 1: Data Collection ... 99

3.3.1(a) Sources ... 99

3.3.1(b) Malware Families ... 101

3.3.2 Phase 2: Feature Processing ... 105

3.3.2(a) Metadata Generation ... 105

3.3.2(b) Flow-set Features Generation ... 105

3.3.2(c) Merging New PCAP Metadata Files ... 109

3.3.2(d) Specifying the Feature Set and Cleansing Data ... 109

3.3.2(e) Categorial Variables Processing and Feature Scaling ... 112

3.3.2(f) Labeling the Final Data ... 114

3.3.3 Phase 3: Machine Learning ... 115

3.3.3(a) Feature Importance and Selection ... 115

3.3.3(b) Applying ML Algorithms ... 118

3.3.3(c) ML Algorithm Selection ... 119

3.4 Experiment Environment ... 119

3.4.1 Physical Environment ... 119

3.4.2 Logical Environment 120

3.5 Summary 124

CHAPTER 4 EXPERIMENTAL RESULTS AND DISCUSSION 126

4.1 Introduction 126

4.2 Differences between Malware and Benign Feature Values 126

4.2.1 EFS and Outliers Percent Flow-set Features 127

4.2.1(a) Duration 127

4.2.1(b) Number of Packets Received and Sent 128

4.2.2 Transport Layer Security (TLS) 131

4.2.2(a) Ciphersuites and Extensions Offered 131

4.2.2(b) TLS Record Protocol Message Types 135

4.2.3 Transmission Control Protocol (TCP) 137

4.2.4 Flow 138

4.3 Comparing EFS, Outliers Percentage, and Flow Features 141

4.4 EFS Features Effect on Classification Performance 147

4.5 Evaluating TLSMalDetect Detection Performance 150

4.6 Results Comparison with Existing Studies 152

4.7 Summary 154

CHAPTER 5 CONCLUSION AND FUTURE WORKS 156

5.1 Overview 156

5.2 Conclusion.. 156

5.3 Future Work 159

REFERENCES 161

APPENDICES

LIST OF PUBLICATIONS

LIST OF TABLES

Page

Table 2.1 Malware generic groups 13

Table 2.2 Malware characteristics 14

Table 2.3 Some server certificate fields 24

Table 2.4 A comparison of different TLS server certificate features used in related work 42

Table 2.5 A comparison of HTTP features used in the related work 46

Table 2.6 Packet features in the literature 58

Table 2.7 A Comparison of flow feature usage 62

Table 2.8 Summary of the ML performance in the literature 84

Table 2.9 Related work strengths and weaknesses 88

Table 3.1 Mapping Research Questions, Objectives, and TLSMalDetect stages 97

Table 3.2 The details of the dataset malware families 103

Table 3.3 The initial feature set 109

Table 3.4 "tcpAnomaly" and "tcpFlags" features 113

Table 3.5 136 Basic features 116

Table 3.6 Top 23 features 117

Table 4.1 The metric performance results of seven classification algorithms applied to the 136 basic & top 23 features 151

Table 4.2 Similarities between this research and the two selected existing studies 153

Table 4.3 Comparing precision and recall with the related work 153

LIST OF FIGURES

Page

Figure 1.1 AV-TEST Malware Growth 1

Figure 1.2 Percentage of Malware info stealers using TLS 2

Figure 1.3 Research Scope 8

Figure 1.4 Research methodology overview 8

Figure 2.1 Literature review background 12

Figure 2.2 Malware Properties 15

Figure 2.3 TrickBot functions to collect credentials from Outlook, FileZilla, and WinSCP 16

Figure 2.4 Dridex attack stages 17

Figure 2.5 TLS protocol stack 20

Figure 2.6 TLS record structure 21

Figure 2.7 TLSv1.2 handshake messages 22

Figure 2.8 Example of a DV Certificate 26

Figure 2.9 Example of an EV Certificate 27

Figure 2.10 Confusion Matrix for two classes 28

Figure 2.11 AUC-ROC 30

Figure 2.12 Related work chart 31

Figure 2.13 TLS v1.2 handshake messages that are sent in clear text 33

Figure 2.14 TLS handshake client features utilization in related work 34

Figure 2.15 Variation between the malicious and benign client hello fields reported by Cisco Team 35

Figure 2.16 TLS handshake Server Hello features utilization in the literature. 37

Figure 2.17 Variation between malicious and benign server hello certificate fields reported 38

Figure 2.18 The apps ratio communicating with each server kind........................40
Figure 2.19 Contextual HTTP flow differences...49
Figure 2.20 Variations of HTTP client header...51
Figure 2.21 DNS contextual flow difference between benign and malware traffic...53
Figure 2.22 Comparison of TLS packet lengths and inter-arrival times between Bestafera Malware exfiltration and Google search............................55
Figure 2.23 Grouping flows into sets...65
Figure 2.24 Qakbot malware regularity of flow duration values without periodicity. 67
Figure 3.1 Research methodology flow...95
Figure 3.2 TLSMalDetect Approach...99
Figure 3.3 No. of unidirectional flows - Stratosphere malware captures..........100
Figure 3.4 No. of unidirectional flows – Stratosphere & CICIDS2017 benign captures 101
Figure 3.5 Malware families...102
Figure 3.6 Flow-set Features Generation 106
Figure 3.7 Binary encoding of value list features...112
Figure 3.8 One-hot-encoding of SSL Version..113
Figure 3.9 Decoding of "tcpAnomaly" and "tcpFlags" into new binary features 113
Figure 3.10 Data flow of the experiment...121
Figure 3.11 Tranalyzer bash script used for parsing PCAP files.........................122
Figure 3.12 A snapshot of the script used to add the entropy and outliers to metadata files for each flow set...123
Figure 4.1 Outliers% & EFS of flow duration..128
Figure 4.2 Different numPktsRcvdEntropy values with the same outliers% value 129

Figure 4.3 Outliers% & EFS of number of packets sent 130
Figure 4.4 Outliers% & EFS of number of packets received 131
Figure 4.5 Differences in the no. of client offered ciphersuites and extensions. 133
Figure 4.6 Differences in client offered ciphersuites and extensions. 134
Figure 4.7 Differences in TLS record protocol message types 136
Figure 4.8 The differences between malware and benign TCP anomaly & flags 137
Figure 4.9 No. of packets received & packets per second variations 139
Figure 4.10 No. of bytes received & standard deviation of IAT variations 140
Figure 4.11 Top 50 Features using Mutual Information – Filter Method. 142
Figure 4.12 Top 40 Features using LR Regularisation L1– Embedded Method. 144
Figure 4.13 The Average Feature Importance of RF and XGBoost with 10-fold CV. 145
Figure 4.14 Classification Performance of the top 23 features with & without EFS of numPktsSnt & Rcvd. 148
Figure 4.15 Classification Performance of The 136 Basic Features With & Without EFS of numPktsSnt & Rcvd. 149
Figure 5.1 Mapping the hex codes to ciphersuites used in figures 178
Figure 5.2 Mapping the hex codes to extensions used in figures 178

LIST OF SYMBOLS AND ABBREVIATION

AA	Adversarial Autoencoders
ADTree	Alternative Decision Tree
API	Application Programming Interfaces
AUC-ROC	Area Under a Receiver Operating Characteristic
C&C	Command & Control
CA	Certificate Authority
CART	Classification And Regression Trees
CFS	Correlation Based Feature Selection
CNN	Convolutional Neural Network
CV	Cross-Validation
DMZ	Demilitarized Zone
DNS	Domain Name System
DT	Decision Tree
DV	Domain Validation
EC	Elliptic Curve
EV	Extended Validation
EFS	Entropy-based Flow Set
F1	F1-score
FM	Filtering Model
FN	False Negatives
FNR	False Negative Rate

FP	False Positives
FQDN	Fully Qualified Domain Names
FTP	File Transfer Protocol
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IP	Internet Protocol
KNN	K-Nearest Neighbor
LR	Logistic Regression
LSTM	Long Short-Term Memory
MD5	Message Digest Algorithm 5
MFCM	Malware Family Classification Model
ML	Machine Learning
MML	Multi-layer Perceptron Classifier
NB	Naïve Bayes
NIDS	Network-based Intrusion Detection System
OS	Operating System
OV	Organization Validation
RBF	Radial Basis Function
RF	Random Forest
RNN	Recurrent Neural Network

RSA	Rivest, Shamir, & Adleman (public-key encryption technology)
SAN	Subject Alternate Name
SHA1/256	Secure Hash Algorithm 1/256
SMTP	Simple Mail Transfer Protocol
SMTPS	Simple Mail Transfer Protocol Secure
SNI	Server Name Indication
SPL	Sequence of Packet Length
SPT	Sequence of Packet Time
SS	Self-Signed
SSL	Secure Sockets Layer
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLSMalDetect	TLS Malware Detection Approach
TN	True Negatives
TP	True Positive
TPR	True Positive Rate
TTL	Time To Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator

LIST OF APPENDICES

Appendix A THE HEX CODES FOR CIPHERSUITES AND EXTENSIONS

PENDEKATAN KLASIFIKASI PEMBELAJARAN MESIN UNTUK MENGESAN MALWARE BERASASKAN TLS DENGAN MENGGUNAKAN CIRI SET ALIRAN BERDASARKAN ENTROPI

ABSTRAK

Memandang penyulitan Internet berkembang untuk melindungi privasi pengguna, malware memanfaatkan protokol penyulitan seperti Transport Layer Security (TLS) untuk menyembunyikan sambungan berbahayanya. Kesukaran menyahsulit trafik rangkaian TLS sebelum Sistem Pengesanan Pencerobohan (IDS) mendorong banyak kajian, untuk memberi tumpuan kepada pengesanan malware berasaskan anomali tanpa penyahsulitan menggunakan pelbagai ciri dan algoritma Pembelajaran Mesin (ML). Walau bagaimanapun, beberapa kajian ini menggunakan ciri aliran dengan nilai kepentingan ciri yang rendah dan keupayaan yang lemah untuk membezakan aliran berniat jahat, seperti bilangan paket yang dihantar dan diterima dalam aliran atau tempohnya. Tambahan pula, outlier dan transformasi ciri aliran berasaskan frekuensi (FTT) yang digunakan untuk mengurangkan ciri aliran lemah mempunyai beberapa kelemahan. Tesis mencadangkan pendekatan pengesanan malware berasaskan TLS (TLSMalDetect) berdasarkan klasifikasi ML untuk menangani had penggunaan ciri Flow dalam kerja yang berkaitan. TLSMalDetect termasuk ciri set aliran berasaskan entropy (EFS) berkala yang dihasilkan oleh teknik FFT. Kecekapan ciri EFS dinilai dalam dua cara: (1) membandingkan ciri Outliers dan aliran kerja menggunakan empat kaedah kepentingan, dan (2) menganalisis prestasi klasifikasi dalam senario dengan dan tanpa ciri EFS. Prestasi pengesanan TLSMalDetect menggunakan tujuh algoritma klasifikasi ML dan mengenal pasti yang mempunyai ketepatan tertinggi. Penemuan penyelidikan menunjukkan keunggulan ciri EFS pada bilangan paket yang dihantar dan diterima kepada ciri Outliers dan Flow yang sepadan. Kes ketepatan algoritma Support Vector Machine (SVM) menunjukkan keupayaan meningkatkan prestasi sehingga 42%. TLSMalDetect melalui Naïve Bayes menggunakan ciri asas mencapai ketepatan klasifikasi tertinggi sebanyak 93.69%.

A MACHINE LEARNING CLASSIFICATION APPROACH TO DETECT TLS-BASED MALWARE USING ENTROPY-BASED FLOW SET FEATURES

ABSTRACT

As internet encryption has grown to safeguard users' privacy, malware has evolved to leverage encryption protocols such as Transport Layer Security (TLS) to conceal its hazardous connections. The difficulty and impracticality of decrypting TLS network traffic before it reaches the Intrusion Detection System (IDS) has driven numerous research studies to focus on anomaly-based malware detection without decryption employing various features and Machine Learning (ML) algorithms. Nonetheless, several of these studies used flow features with low feature importance value and poor capability to distinguish malicious flows, such as the number of packets sent and received in a flow or its duration. Furthermore, the outliers and frequency-based flow feature transformations (FTT) applied to mitigate the poor flow feature have several flaws. This thesis proposes a TLS-based malware detection (TLSMalDetect) approach based on ML classification to address flow feature utilization limitations in related work. TLSMalDetect includes periodicity-independent entropy-based flow set (EFS) features produced by an FFT technique. The efficiency of EFS features is assessed in two ways: (1) by comparing them to the relevant related work's features of outliers and flow using four feature importance methods, and (2) by analyzing the classification performance in the scenarios with and without EFS features. This study also investigates TLSMalDetect detection performance using seven ML classification algorithms and identifies the one with the highest accuracy. The findings of this research demonstrated the superiority of EFS features of the number of packets sent and received to the corresponding outliers-based and flow features and showed their ability to improve performance by up to 42% in the case of the Support Vector Machine (SVM) algorithm accuracy. Additionally, among the ML algorithms applied, TLSMalDetect achieved the highest classification accuracy of 93.69% via Naïve Bayes utilizing the basic features.

CHAPTER 1

INTRODUCTION

1.1 Background

Malware is any program intended to harm a particular device, server, or computer network, whether it is a virus, spyware, or anything else (*Defining Malware: FAQ | Microsoft Docs*, 2021). It is the most destructive cyber problem threatening businesses of all sizes. Based on the AV-TEST Institute report, the total number of discovered malware has been tremendously growing over the last ten years, as illustrated in Figure 1.1 (*AV-TEST | Antivirus & Security Software & AntiMalware Reviews*, 2022).

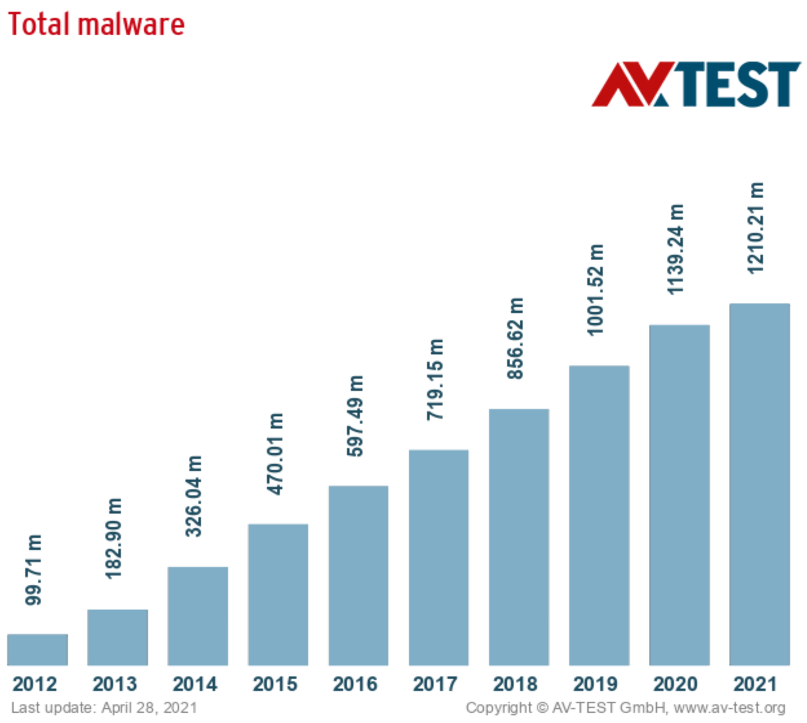


Figure 1.1 AV-TEST Malware Growth (*Malware Statistics & Trends Report | AV-TEST*, 2021)

With the massive proliferation of malware, malware authors opted to embrace the ongoing development of encryption methods by using the Transport Layer Security

(TLS) protocol and communicating over Hypertext Transfer Protocol Secure (HTTPS) connections to obfuscate the contents of malicious communication (Nagy, 2020). According to Sophos Labs, roughly 23% of all malware categories and 44% (almost half) of information-stealing malware employ TLS while transmitting or receiving orders from the Command and Control (C&C) server, installing harmful payloads, or accessing data provided by that payload (Figure 1.2) (Nagy, 2020).

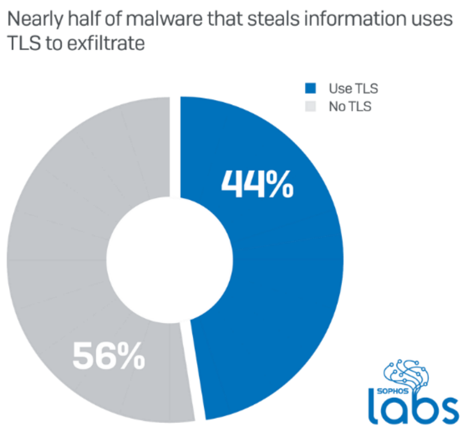


Figure 1.2 Percentage of Malware info stealers using TLS (Nagy, 2020)

The malware expansion necessitated the installation of data security mechanisms in both private and public networks. An Intrusion Detection System (IDS) is a network security system that scans for unusual behaviour and warns users if malware is discovered. There are two types of IDS: Network-based IDS (NIDS), which is responsible for detecting network threats from the captured traffic, and Host-based IDS (HIDS), which detects threats on hosts such as computers (Rezek, 2020). IDSs also detect attacks using two methods: (1) signature-based detection, which uses databases with predefined known attack patterns, and (2) anomaly detection, which detects unknown suspicious behaviour by comparing it to normal activity (Rezek, 2020).

Despite the fact that IDS technology continues to be the best security practice, no security strategy is perfect. IDS researchers are still seeking solutions for identifying malware that uses TLS encryption to mask its connections information (Anderson et al., 2018; Jenseg, 2019; Liu et al., 2019; Maroušek, 2017; Roques et al., 2019).

1.2 Research Problem

Although IDS is a viable technology, signature-based IDS still have several flaws regarding TLS-based malware detection due to the unavailability of clear-text data in the encrypted traffic (Strasák, 2017). Decrypting network traffic before it reaches the signature-based NIDS could be an option for detecting TLS-based malware (*Decryption Overview- PAN-OS® Guide*, 2022). Nevertheless, decryption is impractical as it increases the network infrastructure complexity and causes all devices to trust the certificate used by the decryptor (Durumeric et al., 2017).

Using anomaly-based NIDS with Machine Learning (ML) algorithms does not involve traffic decryption, and it is considered a solution to detect TLS-based malware (Anderson et al., 2018). ML applies a mathematical modelling approach to the available traffic features to learn past data patterns before predicting the possible malicious anomaly behaviour using new data (Kok et al., 2019).

Several feature types have been utilized for ML in the area of TLS-based malware anomaly detection in a network. Flow features are among the common features in the literature that were extracted and directly passed to train ML models (Anderson et al., 2018; Jenseg, 2019; Liu et al., 2019; Maroušek, 2017; Roques et al., 2019). However, several research studies revealed that certain flow features, such as the number of packets sent/received in a flow or the duration, have low feature importance value and can poorly differentiate malicious flows (Jenseg, 2019; Roques et al., 2019). Consequently, flow features could negatively impact the performance as the classifier is trained on less important or uninformative features (Kumar, 2021).

Feature engineering overcomes the issue of using poor flow features by applying domain knowledge (Dong & Liu, 2018). Feature transformation is a feature engineering aspect that generates new features from existing ones through arithmetic and aggregation operations (Dong & Liu, 2018). Based on the literature investigation, different feature transformation methods were applied to flow features in sets to produce additional flow-set features that tackle the issue of using low-importance flow features but, unfortunately, with several drawbacks.

One method is to apply frequency-based analysis (converting from a time domain to a frequency domain) like Fourier Analysis to discover the periodicity of a flow-set feature, assuming that malware is more likely to display periodic values (Fehrman et al., 2020). Although this approach often works, malware can show a non-periodically continuous anomaly pattern, as in the flow duration feature of Qakbot malware, rendering ML detection ineffective based on frequency (*Malware-Traffic-Analysis.Net - Qbot (Qakbot) Infection*, 2020).

Another method discovered in the literature has utilized statistics to distinguish outliers in flow feature values (Dai et al., 2019; Strasák, 2017). The method works by calculating Standard Deviation and Mean, then finding the per cent of all values out of range *mean+/- standard deviation* (outliers). However, the method’s disadvantage is that the outlier values significantly influence the mean and standard deviation, making the number of outliers of a malicious flow set susceptible to being as in normal traffic (*Outlier Detection Methods*, 2021). It is also quite unlikely to find outliers in small samples (Frost, 2022).

1.3 Research Motivation

Malware is multiplying, representing a severe threat to the networks of the entire world. For instance, according to Malwarebytes 2020 report, the number of malware families has increased from 2018 to 2019 with a minimum of 6% (Trojan Emotet) and up to 60.69% (Adware Yontoo) (*Malwarebytes - 2020 State of Malware Report*, 2020). This malware growth has also driven around a quarter (23%) of malware to adapt TLS protocol to mask malware connections hindering signature-based IDS detection and producing a significant problem to overcome (Nagy, 2020).

Since decryption of the TLS traffic is inapplicable, researchers have been trying to propose ways to detect TLS-based malware using ML anomaly detection with appropriate features (Durumeric et al., 2017). However, according to the literature, certain features related to flows were found inefficient and needed a proper feature transformation technique within a flow set (Jenseg, 2019; Roques et al., 2019). Although some research studies have proposed feature transformation methods to solve this issue, these methods still have drawbacks (Dai et al., 2019; Fehrman et al., 2020; Strasák, 2017).

In view of the foregoing, it was motivating to perform this research, suggesting a detection approach that addresses the significant gap in the employment of flow features and contributes to the detection of TLS-based malware.

1.4 Research Questions

A lot of information is needed to build an ML classifier that can distinguish TLS-based malware traffic and fill up the literature gap related to flow feature usage. This information is based on both the research problem and literature, and it reflects the following questions:

1. What periodicity-independent flow set features have higher feature importance than the related outliers based and flow features?
2. What will happen to the ML classification performance when the features that have higher importance than outliers based and flow features are included and excluded?
3. What ML classification algorithm that the TLSMalDetect approach uses to attain the highest performance accuracy?

1.5 Research Objectives

This research's main goal is to propose an ML classification approach, TLSMalDetect, to detect TLS-based malware and overcome the drawbacks of the flow feature usage of the existing work. This overall goal can be broken into the following detailed objectives:

1. To propose periodicity-independent entropy-based flow set (EFS) features with higher feature importance than the related outliers based and flow features.
2. To analyze the impact of EFS features on classification performance by comparing the classification performance metric results with and without EFS features.
3. To investigate TLSMalDetect detection performance using seven ML classification algorithms to identify the best accuracy achieving one.

1.6 Research Contribution

The contribution of this is summarized as follows :

1. Two periodicity-independent entropy-based flow set (EFS) features, numPktsSntEntropy and numPktsRcvdEntropy, which have higher feature importance than the related outliers based and flow features.
2. A two-scenario analysis by which the two EFS features, numPktsSntEntropy and numPktsRcvdEntropy, are proven to have the ability to boost the classification performance metrics up to ~42% in the case of Support Vector Machine (SVM) ML algorithm.
3. An ML classifier that can detect TLS-based malware with the highest Naïve Bayes accuracy in the related literature using MCFP and CICIDS 2017 datasets.

1.7 Research Scope and Limitations

This research focuses on TLS-based malware anomaly detection using ML classification and studying the efficiency of EFS flow-set features. The reason for scoping the study into ML classification is the availability of the two types of dataset: malware and normal, which can be labelled eventually (*Supervised vs Unsupervised - Seldon*, 2021). Also, due to resource limitations, specific flow features like the number of packets sent and received and the duration are transformed to generate EFS features. Apart from EFS and outliers% features, particular TLS Handshake, Transmission Control Protocol (TCP), and flow features that have shown promising results in the literature are only included because of resource limitations too. Furthermore, the most popular ML classification performance evaluation metrics used are F1-score (F1), Accuracy, Precision, Recall, and AUC (Agrawal, 2021). Figure 1.3 illustrates the scope of this thesis research.

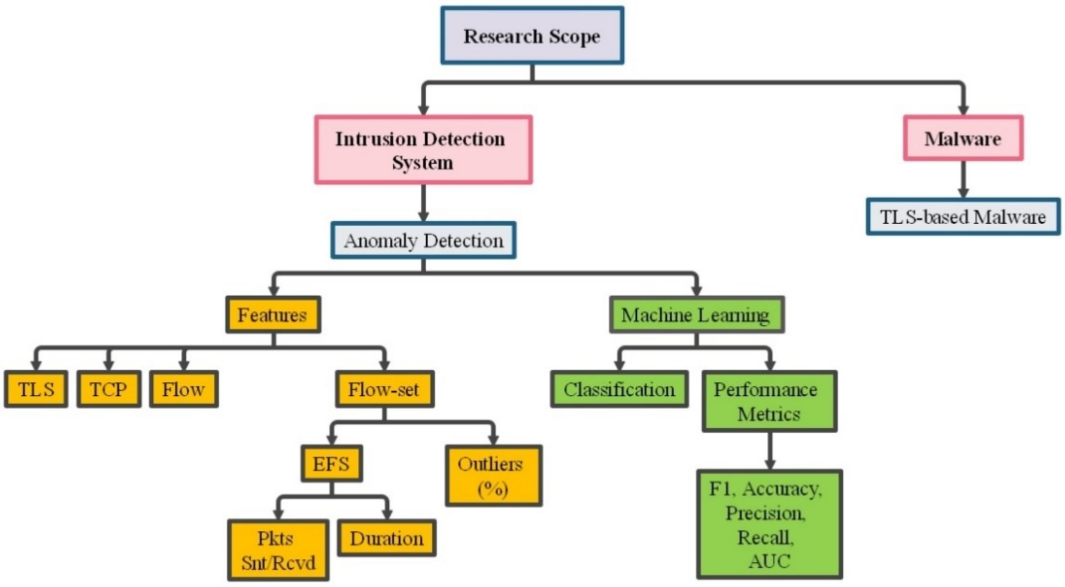


Figure 1.3 Research Scope

1.8 Research Methodology

This research methodology is divided into four main steps, as shown in Figure 1.4. **The first step** was to comprehensively review the related studies and identify the research problem as a result.

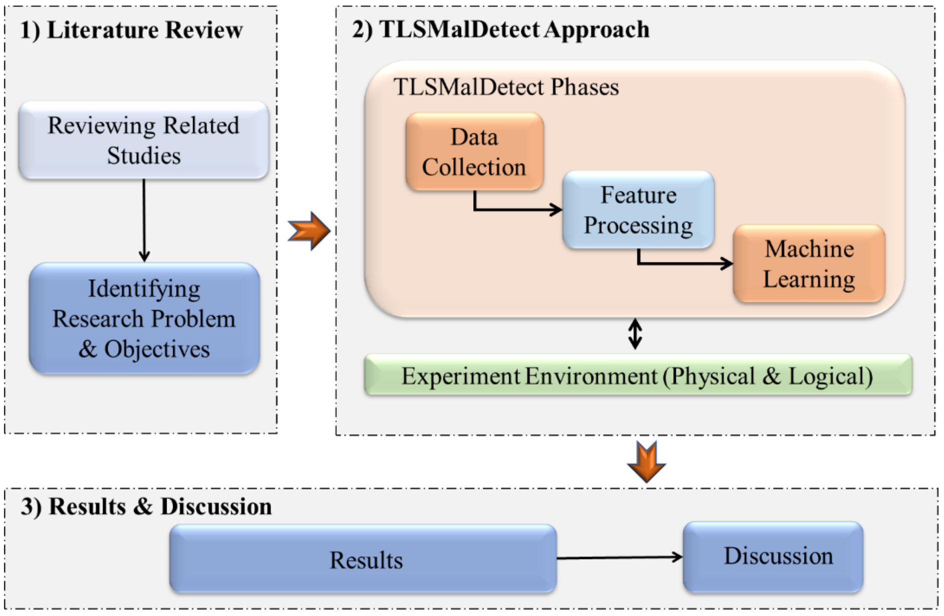


Figure 1.4 Research methodology overview

In **the second step**, the TLSMalDetect approach to solving the related work gap and achieving the research objectives was designed and implemented with the proper setup of the physical and logical experiment environments. TLSMalDetect has three phases: (1) Data collection, where the experiment's dataset is obtained; (2) Feature processing, in which features are processed and made ready for ML use; (3) Machine Learning, where feature importance methods and ML algorithms are employed.

Finally, the results of the statistical analysis, feature importance, and ML algorithms were presented and discussed in **the third step**.

1.9 Thesis Outline

The thesis consists of five chapters and a references section with appendices. The chapters are as follows:

Chapter one starts with a background followed by the research problem, motivation, questions, objectives, contributions, scope, and limitations. Finally, it summarizes the thesis organization.

Chapter two offers the literature review. First, it provides the reader with the essential concepts of malware taxonomy, malware analysis, TLS protocol, and classification assessment metrics. After that, it discusses the related work features, ML, and feature selection usage. It also highlights the weaknesses found in each study, especially those relevant to the flow feature usage.

Chapter three thoroughly explores the research methodology and the phases of the proposed TLSMalDetect approach, clarifying these phases' relations and data flow. It also explains the experiment's physical and logical environments used.

Chapter four reports and discusses the experiment results. It statistically evaluates the differences between malware and benign in some feature values. The chapter also discusses the findings of feature importance methods concerning the superiority of EFS features and their effect on classification performance. In addition, it compares TLSMalDetect detection classification performance with other related studies.

Chapter five concludes the thesis with a summary and comments on the contributions. It also suggests possible future directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Malware has been developed to leverage encryption protocols such as Transport Layer Security (TLS) to conceal the contents of destructive connections as internet encryption has grown to protect users' privacy. However, decrypting network traffic before it arrives at the signature-based Intrusion Detection System (IDS) to detect TLS-based malware is impracticable since it complicates infrastructure and threatens user privacy (Durumeric et al., 2017). As a result, several research works have been conducted to study anomaly detection without decryption utilizing various traffic features and methodologies such as Machine Learning (ML).

This chapter aims to review the related TLS-based malware detection works and examine the use of different features and ML in these works to better understand the field's present condition. Furthermore, it emphasizes some of the literature's strengths and makes various recommendations on its weaknesses for future successful detection methods. The chapter is organized as follows: Section 2.2 presents some essential background concepts. After that, Section 2.3 discusses the related work usages of features, feature selection, and ML and also shows the related work strengths and weaknesses. Finally, Section 2.4 ends the chapter with a summary.

2.2 Background

This section presents some essential concepts that help get the necessary knowledge to comprehend better the related work of TLS-based malware detection. It discusses malware taxonomy, some TLS-based malware types, malware analysis techniques, an overview of TLS protocol, and classification assessment. Figure 2.1 shows the background topics which are discussed in the following sub-sections.

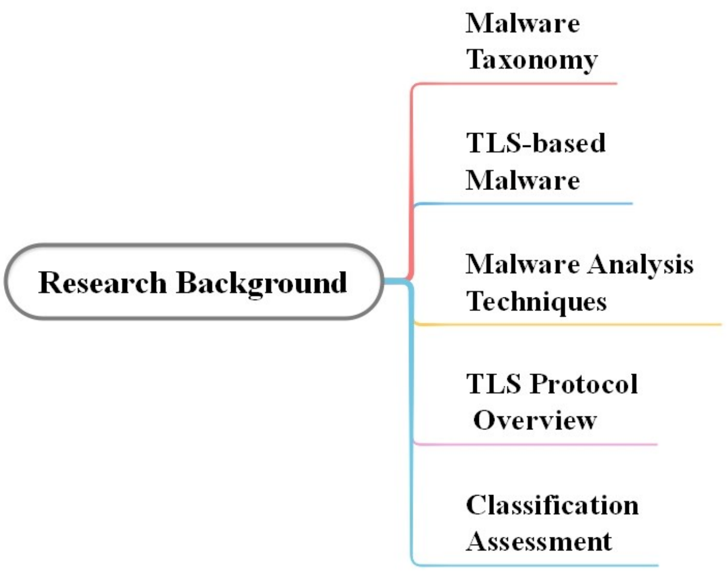


Figure 2.1 Literature review background

2.2.1 Malware Taxonomy

Malware is a harmful code or program that causes damage to the system resources or steals credentials (*Defining Malware: FAQ | Microsoft Docs*, 2021). Malware has lately expanded in quantity, and it is critical to understand its taxonomy, including classes and attributes, to identify it (*Malware Statistics & Trends Report | AV-TEST*, 2021). Microsoft has grouped malware into different generic groups, summarised in Table 2.1, to better understand its taxonomy (*Understanding Malware & Other Threats - Windows Security | Microsoft Docs*, 2020).

Table 2.1 Malware generic groups (Understanding Malware & Other Threats - Windows Security | Microsoft Docs, 2020)

Malware Generic Group	**Description**
Coin miners	Allow criminals to penetrate a corporation and illegally mine coins.
Exploits and exploit kits	Take advantage of software vulnerabilities to bypass the security defence of the machine and compromise the computer.
Phishing	Tries to capture sensitive information through means that appear legal via emails, directories, text messaging, or other types of electronic contact.
Ransomware	Encrypts files and attempts to extortion victims' money by demanding money, usually in the form of cryptocurrency, in return for the key to decryption.
Rootkits	Can hide both themselves and their malicious behaviour on a computer.
Trojans	Appear like legitimate apps.
Worms	Spread over the network by exploiting vulnerabilities.
Virus	Harmful software that replicates from one device to another.
Bots	Automated processes connected with other network services to deliver information that a human being would otherwise carry out.

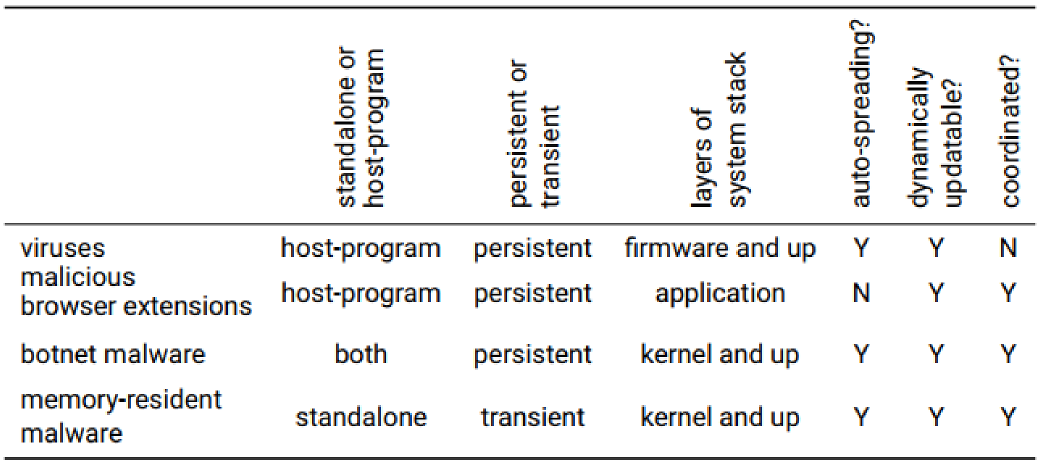
In fact, if the communication traffic is encrypted using the TLS protocol, any of the malware kinds listed above can be considered TLS-based malware (Nagy, 2020). The most common types of TLS-based malware include Trojans (such as "TrickBot," "Dridex," and "IcedID"), Ransomeware, and Botnets (Nagy, 2020).

Furthermore, generic malware groups exhibit various characteristics throughout their life cycles, such as individual interaction, collective behaviour, and the site at which they attack (Lee et al., 2019). The malware characteristics are explained in Table 2.2.

Table 2.2 Malware characteristics (Lee et al., 2019).

Characteristic	**Description**
Dependency	Malware can be independent, such as worms and botnets, or a component of a program code that runs while that program is executed, such as text macro viruses and malicious browser plug-ins.
Persistency	Malware can be persistent, installed in persistent storage like a file system, or transient (temporary), which operates in memory. Transient malware can avoid detection by several anti-virus systems that depend on file scanning, and it has the advantage of being simple to clean up (or cover-up) the attack actions.
Spreading	Malware can run and reproduce itself automatically by exploiting vulnerabilities, resulting in rapid expansion. In contrast, certain malware forms, such as email attachment malware, run and propagate solely by the victim's behaviour (Lee et al., 2019).
Updating Process	Most malware kinds do updates that allow them to escape detection. The update is sent to the malware dynamically by the malware's author via a server. Other malware, on the other hand, just runs once and never gets updated.
Coordinating	Malware may attack independently or as part of an organized network such as botnets. Although botnets are responsible for many cyberattacks such as DDoS, spam, phishing, and so on, isolated malware is becoming more prevalent in targeted attacks.
Targeted System Stack Layers	In ascending order, the system stack layers on which the malware is built and executed are firmware, boot-sector, operating system kernel, drivers, Application Programming Interfaces (APIs), and user applications (Lee et al., 2019). Malware that operates at the lower stack stage is usually more difficult to program and detect.

Besides, a summary of malware categories and the according properties is illustrated in Figure 2.2.



	standalone or host-program	persistent or transient	layers of system stack	auto-spreading?	dynamically updatable?	coordinated?
viruses	host-program	persistent	firmware and up	Y	Y	N
malicious browser extensions	host-program	persistent	application	N	Y	Y
botnet malware	both	persistent	kernel and up	Y	Y	Y
memory-resident malware	standalone	transient	kernel and up	Y	Y	Y

Figure 2.2 Malware Properties (Lee et al., 2019)

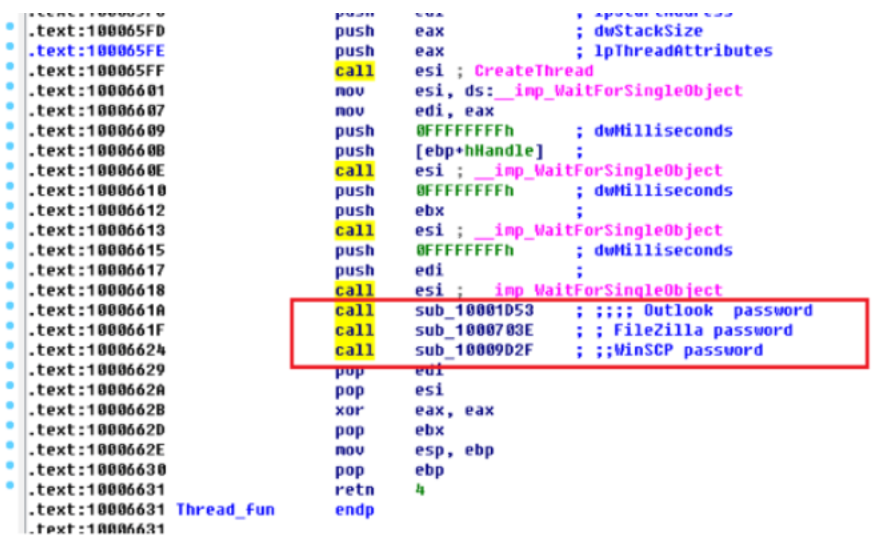
2.2.2 TLS-based Malware

Many known malware categories use TLS to communicate with Command & Control (C&C) servers and act harmfully, such as trojans (Trickbot, Emotet, Dyre) and ransomware (Jigsaw, Locky). To better understand this dangerous behaviour, some types of well-known TLS-based malware are explored in the following sections.

2.2.2(a) TrickBot

TrickBot is a distant descendant of the ZeuS banking Trojan, which first appeared in 2005, but it is most often associated with Dyre or Dyreza, which disappeared in 2015 (*TrickBot: Not Your Average Hat Trick – A Malware with Multiple Hats*, 2021). TrickBot can do various criminal activities, such as targeting foreign banks through its web injects, stealing from Bitcoin wallets, and collecting emails and credentials using the Mimikatz tool (*Trojan.TrickBot - Malwarebytes Labs | Malwarebytes Labs | Detections*, 2020).

TrickBot has a configuration file that contains modules. Each module is assigned a particular task: gaining persistence, propagation, stealing passwords, or encryption. (*Trojan.TrickBot - Malwarebytes Labs | Malwarebytes Labs | Detections*, 2020). This kind of malware relies on TLS to communicate with its C&C server by POST requests reporting credentials of many applications such as Google Chrome, File Transfer Protocol (FTP) clients, and Outlook. Figure 2.3 displays some TrickBot functions to collect credentials from Outlook, FileZilla, and WinSC (*Deep Analysis of TrickBot New Module Pwgrab*, 2020).

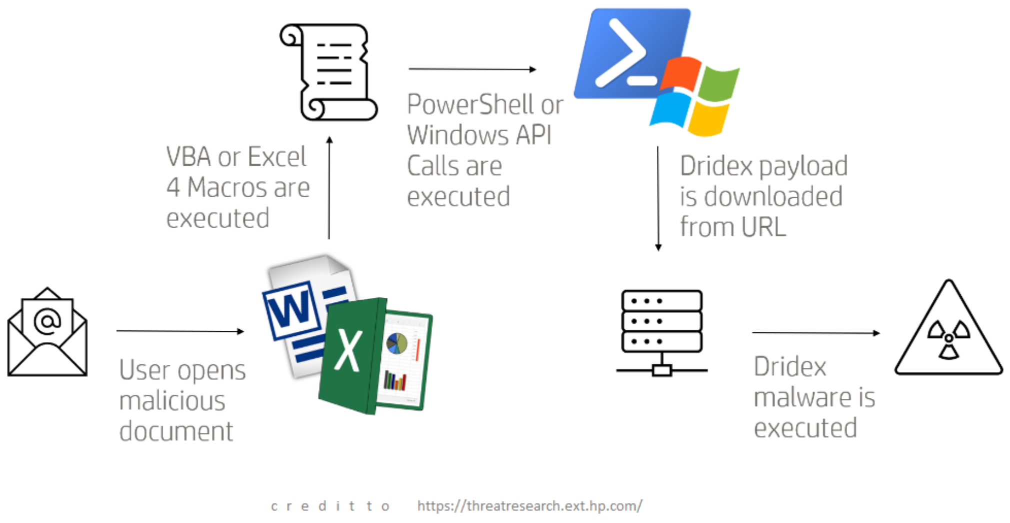


```
.text:100065FD             push    eax             ; dwStackSize
.text:100065FE             push    eax             ; lpThreadAttributes
.text:100065FF             call    esi ; CreateThread
.text:10006601             mov     esi, ds:__imp_WaitForSingleObject
.text:10006607             mov     edi, eax
.text:10006609             push    0FFFFFFFFh      ; dwMilliseconds
.text:1000660B             push    [ebp+hHandle]   ;
.text:1000660E             call    esi ; __imp_WaitForSingleObject
.text:10006610             push    0FFFFFFFFh      ; dwMilliseconds
.text:10006612             push    ebx             ;
.text:10006613             call    esi ; __imp_WaitForSingleObject
.text:10006615             push    0FFFFFFFFh      ; dwMilliseconds
.text:10006617             push    edi             ;
.text:10006618             call    esi ; __imp_WaitForSingleObject
.text:1000661A             call    sub_10001D53    ; ;;;; Outlook  password
.text:1000661F             call    sub_100070 3E    ; ; FileZilla password
.text:10006624             call    sub_10009D2F    ; ;;WinSCP password
.text:10006629             pop     edi
.text:1000662A             pop     esi
.text:1000662B             xor     eax, eax
.text:1000662D             pop     ebx
.text:1000662E             mov     esp, ebp
.text:10006630             pop     ebp
.text:10006631             retn    4
.text:10006631 Thread_fun  endp
.text:10006631
```

Figure 2.3 TrickBot functions to collect credentials from Outlook, FileZilla, and WinSCP (*Deep Analysis of TrickBot New Module Pwgrab*, 2020)

2.2.2(b) Dridex

Dridex is a banking malware classified as Trojan that was initially spread in late 2014 via a spam campaign that generated upwards of 15,000 emails each day and was primarily focused on systems in the United Kingdom (*What Is Dridex Malware? - Spambrella*, 2021). Cybercriminals distribute it through spam emails disguised as official, prompting the victim to open an attached Microsoft Word or Excel file containing an embedded macro (Gillis, 2020). This macro would import the Dridex payload and then steal information using a keylogger, which records every keystroke typed on the keyboard; TLS will also encrypt all subsequent messages to the C&C server, including the stolen data (Gillis, 2020). The stages until Dridex execution are illustrated in Figure 2.4.



VBA or Excel 4 Macros are executed
PowerShell or Windows API Calls are executed
Dridex payload is downloaded from URL
User opens malicious document
Dridex malware is executed
c r e d i t t o https://threatresearch.ext.hp.com/

Figure 2.4 Dridex attack stages (*Dridex Malware Analysis [10 Feb 2021] – Malware Analysis*, 2021)

2.2.3 Malware Analysis Techniques

2.2.3(a) Static Analysis

Static analysis of malware is a process focused on inspecting without running a malware program (Gibert et al., 2020). Instead, the analysis works by disassembling the malicious binary file using software such as IDA Pro and examining the program logic (Gibert et al., 2020). This strategy does not require much resources and time compared to other analysis types, such as dynamic analysis (Singh & Singh, 2021). However, most malware types currently try to avoid this type of analysis by depending on values that are hard to determine statically, such as current system date, indirect jump instructions, and many other ways (Gibert et al., 2020).

2.2.3(b) Dynamic Analysis

It is the method of running a given malware sample in a managed environment and tracking its behaviour to evaluate the malicious activity (Gibert et al., 2020). In this type of analysis, system behaviour and network traffic are monitored for any unusual changes (Jenseg, 2019). In addition to solving the static analysis issues, an advantage of this analysis is that it can analyze large datasets and be automated (Gibert et al., 2020). However, this approach may damage the system environment if it is not well protected and is inefficient for analysis when the malware suddenly changes its behaviour during detection (Gibert et al., 2020).

(i) Collecting Network Traffic

To analyze malware behaviour on the network, the first required phase is to put network sensors that collect traffic (Gibert et al., 2020). According to where they are deployed, there are two types of network sensors, active and passive (Kohout et al., 2018). The active type, known as Intrusion Prevention System (IPS), intercepts the traffic, analyzes it, and then responds by either permitting or denying it; In contrast, the passive, known as Intrusion Detection System (IDS), collects the traffic by making a copy, analyzes it, and take appropriate actions (*IDS vs. IPS: Definitions, Comparisons & Why You Need Both | Okta*, 2021). In this thesis, the focus will be on the passive mode.

In passive mode, there are three approaches for traffic collection:

1) **Port mirroring**: is a technique that can be used by transmitting a replica of the traffic shown to another port on enterprise network switches; A downside to this approach is that the switch allows a programmatic duplicate of the traffic, and packets can be lost or sent out of order as the port is over-subscribed (Jenseg, 2019).
2) **Network TAP**: is a method where a network interface is mounted between two network interfaces, and an exact copy of the traffic detected is made. Because this technique takes a precise copy of traffic, the port mirroring drawback is solved (Jenseg, 2019).
3) **Host Capturing**: is a technique where the traffic passing the host is captured (Jenseg, 2019).

(ii) Types of Collected Data

Dynamically malware analysis is achieved by locating network sensors that collect traffic data (Jenseg, 2019). There are two types of data: numerical and string. Numerical type is any data that contains numbers, including statistical values like the duration of the flow, the start time, end time, port number, the number of bytes sent and received, and others (Jenseg, 2019). String type is human-readable data, such as Server Name Indication (SNI), certificate subject country, and organization in TLS (Jenseg, 2019). In Hypertext Transfer Protocol (HTTP), string data is like User-agent, Content-Type, and Uniform Resource Locator (URL) fields (Anderson et al., 2018).

2.2.4 TLS Protocol Overview

The TLS Protocol is a cryptographic protocol whose primary objective is to "provide privacy and data integrity between two applications that communicate" (T. Dierks, Certicom, C. Allen, 2020). In January 1999, the first version of TLS was released, replacing the now-deprecated Secure Sockets Layer (SSL) protocol (Roques et al., 2019). After that, in Aug 2008, TLSv1.1 was replaced by TLSv1.2 with several major security enhancements (*RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2*, 2021), including the replacement of Message Digest Algorithm 5 (MD5) and Secure Hash Algorithm 1/256 (SHA1) algorithms. Although TLS v1.3 has been launched recently for further security improvements, the most widespread version of TLS is TLSv1.2 (Warburton, 2020).

The TLS protocol lies below the application and above the transport layer, mainly Transmission Control Protocol (TCP) (Figure 2.5) (LAKE, 2021). TLS over User Datagram Protocol (UDP), DTLS, are independently standardized. Nowadays, TLS encrypts most HTTP traffic forming Hypertext Transfer Protocol Secure (HTTPS), according to Google (*HTTPS Encryption on the Web – Google Transparency Report*, 2020). However, the use of TLS is not restricted to HTTP alone, and potentially, any application layer protocol will make use of TLS, such as Simple Mail Transfer Protocol (SMTP) protocol forming Simple Mail Transfer Protocol Secure (SMTPS) for email encryption (Roques et al., 2019).

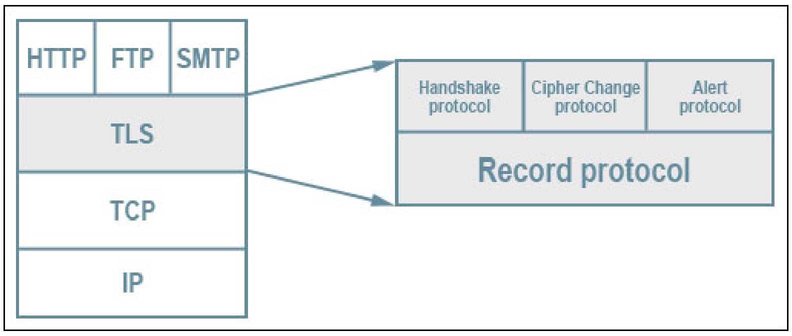
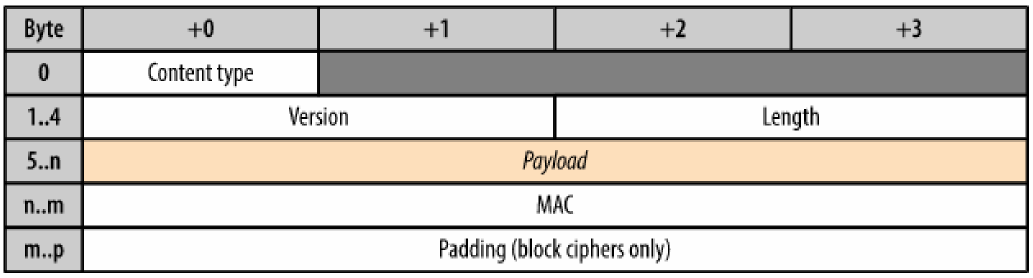


Figure 2.5 TLS protocol stack (LAKE, 2021)

All data shared within a TLS session is framed using a well-defined protocol, similar to the Internet Protocol (IP) or TCP layers below it, called TLS Record Protocol

(Figure 2.6) (*Networking 101: Transport Layer Security (TLS) - High Performance Browser Networking (O'Reilly)*, 2021). The TLS Record Protocol is responsible for defining, securing, and verifying various types of messages: handshake, alert, change cipher spec, and data (via the "Content Type" field) (*Networking 101: Transport Layer Security (TLS) - High Performance Browser Networking (O'Reilly)*, 2021).

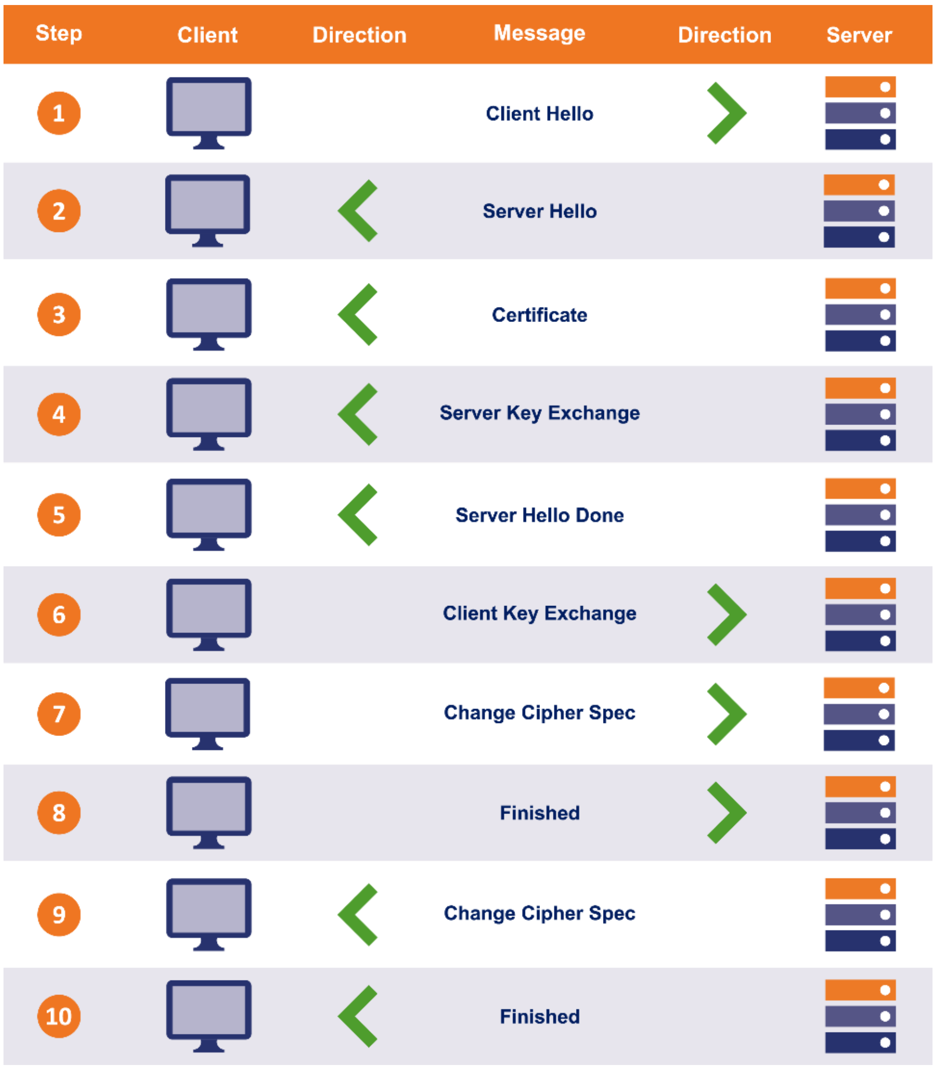


Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

Figure 2.6 TLS record structure (*Networking 101: Transport Layer Security (TLS) - High Performance Browser Networking (O'Reilly)*, 2021)

(i) Handshake Protocol Process Overview

The TLS Handshake protocol, which runs on top of the TLS record layer, generates the session state's cryptographic parameters (*RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2*, 2021). Two round trips of clear-text messages between a client and a server to agree on many parameters and create a TLS session, as in Figure 2.7 (Nohe Patrick, 2019).



Step	Client	Direction	Message	Direction	Server
1	🖥		Client Hello	❯	🗄
2	🖥	❮	Server Hello		🗄
3	🖥	❮	Certificate		🗄
4	🖥	❮	Server Key Exchange		🗄
5	🖥	❮	Server Hello Done		🗄
6	🖥		Client Key Exchange	❯	🗄
7	🖥		Change Cipher Spec	❯	🗄
8	🖥		Finished	❯	🗄
9	🖥	❮	Change Cipher Spec		🗄
10	🖥	❮	Finished		🗄

Figure 2.7 TLSv1.2 handshake messages (Nohe Patrick, 2019)

The client begins with a **Client Hello** message containing:

1) **Ciphersuites** are the client cryptographic algorithms that are supported.
2) **Compression methods**.
3) **Server Name Indication (SNI)** is an extension to specify the server to be connected to.
4) **Client version** is the TLS version the client selected.

5) **Extensions** have many types that determine the information the client needs from the server. Some of the extensions are Elliptic Curve (EC) Point Formats, EC Supported Groups, and Signature Algorithms.

The server returns several messages:

1) **Server Hello** has the server selected Ciphersuites and compression methods.
2) **Certificate** determines the chain of TLS certificates the server sends to the client. It also checks the authenticity, non-repudiation, and integrity using a digital signature. Each certificate has fields for verifying the legality of the issuer and receiver, determining the validity duration, and categorizing the validity levels.
3) **ServerKeyExchange** is needed when certain key exchange methods (such as Diffie-Hellman) are used and when the server does not have a certificate.

The client replies with several messages:

1) **ClientKeyExchange** enables the server to create the final symmetric session key. For example, at Rivest, Shamir, & Adleman (RSA), the client must create a random string of bytes called a pre-master password, then encrypt and transmit it with the server's public key (*The TLS Handshake: Taking a Closer Look - Hashed Out by The SSL Store^{TM}*, 2020).
2) **ChangeCipherSpec** informs the server that all subsequent communications must be encrypted with the session key.
3) **Finished** lets the server know that the client has completed the handshake.

Finally, the server ends the handshake with:

1) **ChangeCipherSpec** informs the client that the session key must encrypt all subsequent messages.
2) **Finished** lets the client know that the server has completed the handshake.

Following these steps, the handshake of TLS v1.2 is complete, and both parties will have a session key and start communicating with an authenticated, encrypted connection (Nohe Patrick, 2019).

The server certificate is one of the significant pieces of info transferred during the TLS handshake. The certificate carries several metadata fields that can be extracted and used in detection (Roques et al., 2019). Some of the most notable server certificate fields are clarified in Table 2.3. Typically, TLS certificates meet the X.509 format (*RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, 2020).

Table 2.3 Some server certificate fields (Roques et al., 2019)

Server Certificate Field	**Description**
Issuer	The entity that verified the server's legitimacy and issued the certificate (in most cases, a CA).
Validity	Includes two sub-fields from the date the certificate is valid to the day the certificate expires.
Subject	The certificate recipient
Subject Public Key Info	Includes two subfields that show the public key algorithm on the server and the public key itself
Extensions (optional)	Includes several fields indicating how to use the certificate and additional certificate information
Certificate Signature Algorithm and Certificate Signature Value	The signature algorithm and the certificate body signature from the issuer