Detection of HTTPS Malware Traffic Without Decryption

Miranda Nyathi

Department of Computer Science
University of Cape Town

This dissertation is submitted to the Department of Computer Science at the University of Cape Town in fulfilment of the requirements for the degree of *Master of Science*

January 2022

Declaration

I know the meaning of plagiarism and I declare that all the work done in this dissertation, save for that which is properly acknowledged, is my own.

Miranda Nyathi
January 2022

Acknowledgements

First and foremost, Sombawa Thixo Mdali wezulu nomhlaba (God) thank you for carrying me through this journey.

I want to thank my supervisor, Dr. Andrew Hutchison, for leading and guiding me in this research. His experience from both industry and academia provided a well-rounded approach to his supervision. Studying and working was a challenge for me, and I think at some point, he believed in me more than I believed in myself, and that helped me give my ultimate best. Andrew introduced me to the world of cybersecurity in honours and my world has changed ever since. Today he sees me as a colleague and that mutual respect gives me more confidence in my career.

My outstanding employer Standard Bank Group thank you for funding my studies! SBG cares for the growth of its employees and strives to be an enabling environment. The bank has outstanding leadership and that goes for my manager and mentor; I am so lucky to have a leader that truly sees me and is my cheerleader. Thank you to my colleagues for allowing me to go on and on about my research and consistently being supportive.

I would have never finished my studies without my rock, my friends! Thank you for all the support, encouragement, and late-night study sessions. It truly takes a village and you guys are my home. I do everything I do for you to my lovely family and thank you for being understanding when I was not my best because of all the stress. A special shout out to the afrobeat artists; your music made 18 hours work days fun.

I have done all my tertiary education at the University of Cape Town because the university embodies excellence in every way! A token of gratitude to the Department of Computer Science for providing me with a world-class education. Now I stand tall in rooms I never thought I would ever be in because I feel prepared.

Thank you to the researchers at Cisco for pushing boundaries!

Abstract

Each year the world's dependency on the internet grows, especially its functionality relating to critical infrastructure and social connections. More than 80% of internet traffic is encrypted using Transport Layer Security (TLS) protocol, and it is predicted that this number will increase [8]. However, threat actors are also increasingly using the TLS protocol to hide malicious activities such as Command and Control, loading malware into a network, and exfiltration of sensitive data.

The use of TLS by threat actors poses a challenge to security professionals as traditional techniques used in the detection of HTTP malware cannot be applied in detecting Hypertext Transfer Protocol Secure (HTTPS) encrypted malware. To manage this, companies are using a traditional method called Transport Layer Security Inspection (TLSI), which involves decrypting packets to do full packet inspection. TLSI is expensive in computational performance and complexity, and over and above all, it violates the users' privacy.

Researchers from Cisco have proposed that it is possible to identify malicious encrypted traffic by techniques other than TLSI and that the unencrypted TLS handshake messages, certificates, and flow metadata of malicious traffic are distinct from benign. These differences can be effectively used in machine learning to classify malicious and benign encrypted traffic [35].

This dissertation aims to assess the feasibility and effectiveness of the proposed alternative to TLSI. We sourced thousands of malware and benign flows and then used the Cisco tool called Joy to extract the features from the unencrypted TLS handshake messages, certificates, and flow metadata. To understand the characteristic behaviour between malicious and benign flows, we did a data exploration, summarized the unique values of the features from our datasets, and compared them with the feature values from the Cisco datasets used in the research paper [35]. We then selected features that had the most differentiating power in our dataset. The selected features were inputs into the two supervised classifiers: logistic regression and random forest. The classifiers were trained and tested on the offline datasets

of benign and malware features, and we observed that the random forest performed better with an average accuracy of 98.92%. We concluded that it is viable and effective to use alternative techniques to detect HTTPS malware without TLSI.

Contents

List of Figures

List of Tables

Acronyms

0-RTT Zero Round-Trip Time.

AES Advanced Encryption Standard.

BD Bytes Distribution.

BIOS Basic Input/Output System.

C2 Command and Control.

CA Certificate Authority.

CIA Confidentiality Integrity Availability.

CPU Central Process Unit.

DH Diffie–Hellman.

DHE Diffie–Hellman Ephemeral.

DMZ Demilitarized Zone.

DNS Domain Name System.

DSA Digital Signature Algorithm.

ECDH Elliptic Curve Diffie–Hellman Key Exchange.

ECDHE Elliptic Curve Diffie–Hellman Key Exchange Ephemeral.

ECDSA Elliptic Curve Digital Signature Algorithm.

ECM Error-Correction Model.

FN False Negative.

FP False Positive.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IDT Interrupt Descriptor Table.

IP Information Protocol.

JSON JavaScript Object Notation.

K-NN K-Nearest Neighbors.

LMS Longest Meaningful Substring.

MAC Media Access Control.

MD Message-Digest.

NSA National Security Agency.

OS Operating System.

PSK Pre-Shared Key.

RSA Rivest–Shamir–Adleman.

SAN Subject Alternate Name.

SHA Secure Hash Algorithm.

SIDT Store Interrupt Descriptor Table.

SPLT Sequence of Packet Lengths and Times.

SSL Secure Sockets Layer.

TLS Transport Layer Security.

TLSI Transport Layer Security Inspection.

TN True Negative.

TP True Positive.

TPC Transmission Control Protocol.

TTL Time to Live.

UDP User Datagram Protocol.

USB Universal Serial Bus.

VM Virtual machine.

Chapter 1

Introduction

1.1 Context and Motivation

Malware had a benign beginning in the 1970s, with Creeper being the first experiment designed to test how a program might move between computers. In the 1980s, researchers became more interested in the behaviour of a 'virus' in terms of how it replicates itself, infecting computer systems, and how fast it spread through systems. However, over time malware has been used to destruct systems and critical infrastructure. On 02 November 1988, the Morris Worm spread via the Internet. It was the first malware to substantially impact infecting a network of computers connected to the Internet, and it brought down networks within 24 hours [55] [37].

All malware writers and cybercriminals aim to disseminate their malware across computer(s) to perform different functions such as stealing, encrypting, or deleting sensitive data, altering or hijacking core computing functions monitoring users' computer activities without their permission. To achieve this, they use phishing emails, social engineering, and malicious websites as entry vectors and Command and Control (C2) servers for communication between the infected system and the threat actors' machines.

In the 21st-century, malware has grown exponentially both in numbers and infections spread. Malware is becoming more sophisticated with the rising number of users' and critical infrastructures' dependence on the internet. The internet has evolved to the 21st-century battleground where malware is used to launch attacks on businesses' and countries' critical infrastructure. "Unlike traditional combat operations, cyberattacks do not require sophisticated weaponry to carry out their warfare. On the cyber battlefield, a single individual

with a laptop computer can wreak havoc on a business, the economy, and even our critical infrastructure" [28].

The Internet was initially built with no security in mind, “We didn’t focus on how you could wreck this system intentionally. You could argue with hindsight that we should have, but getting this thing to work at all was non-trivial.” [12] said Vinton G. Cerf, Google vice president who in the 1970s and 80s designed vital building blocks of the Internet. There was no privacy, authentication, or checking the traffic integrity, which made it natural for attackers to use the HTTP protocol for malicious use. This vulnerability demanded the existence of HTTPS. HTTPS does the same as HTTP: transferring and receiving information on the web, but securely by encrypting the data using one of the associated cryptographic protocols TLS or Secure Sockets Layer (SSL). These cryptographic protocols provide integrity and privacy. Unfortunately, privacy does not mean security; threat actors are also adopting the use of TLS to hide their malicious activities such as Command and Control, loading malware into a network, and exfiltration of sensitive data.

TLS uses the X.509 certificate for server authentication; X. 509 certificates contain a public key and a hostname, organization, or individual identity [25]. Certificates give users some assurance that the website or application they are using is legitimate. Some of the available certificates belong to malicious websites and should not be trusted. Security professionals have made efforts to identify certificates that belong to malicious websites, which has led to threat actors stealing legitimate certificates to evade detection. The benefit of having legitimate certificates for their malware is so great that there are criminal organizations dedicated solely to stealing certificates and selling them to other cybercriminals. These criminal organizations have now even created malware designed just to steal digital certificates.

In the February 2018 Zscaler SSL Threat Report, it is stated that since July 2017, the amount of SSL encrypted traffic on the Zscaler Cloud has increased by 10% to a total of 70% of all web traffic. The report also indicates that Zscaler cloud blocks an average of 8.4 million requests in SSL/TLS-based traffic daily, 600,000 of those containing advanced threats [11]. Moreover, the February 2019 Zscaler SSL Threat Report states that nearly 90% of the internet traffic moves over encrypted channels and that in the second half of 2018, the Zscaler cloud blocked 1.7 billion threats hidden in SSL traffic, which translates to an average of 9.1 million advanced threats blocked daily. The trend is clear, and we can expect the numbers to increase, especially given how cheap and easy it is to obtain valid TLS certificates [27].

A great deal of research has been done in detection of HTTP malware which involves inspecting the full packet. Packet inspection in the case of HTTPS means decrypting the

packet, inspecting the decrypted content for threats, and then re-encrypting the traffic before it enters or leaves the network. This technique is called TLSI and it is expensive in terms of performance and complexity. Besides these technical issues it violates the integrity of the packet and privacy of the users, who expect that with SSL/TLS they have a secure end-to-end connection between their browser and the destination server.

In November 2019, the United States of America National Security Agency (NSA) published an advisory that addresses the risks behind TLSI and provides mitigation measures for weakened security in organizations that use TLSI products [30]. The challenges in TLSI strengthen the need for research on malware HTTPS/encrypted traffic detection without decryption.

A promising result for an alternative to TLSI has been published by a team of researchers from Cisco. The researchers published the paper "Deciphering Malware's use of TLS (without Decryption)" proving that it is possible to detect malware while preserving privacy. They studied different malware families' use of TLS, observed the TLS handshake and extracted features from the flow metadata, certificates, and clienthello and serverhello messages [35]. They applied machine learning classifiers to each malware family, resulting in a multi-class classification problem.

This project aims to replicate the study done by the Cisco team and analyze the feasibility and effectiveness of their approach. The main difference with our study is that the classification will be independent of the malware families. The machine learning classifiers will be applied to all malware flows regardless of their respective malware families. We aim to have a binary classification, testing whether the flows are benign or malicious.

1.2 Overview of the Thesis

This dissertation is structured as follows: the second chapter outlines how data is transmitted over the internet using the TLS protocol, explains how malware uses TLS, the challenges in using TLSI to detect such malware, and presents current research on the alternative to TLSI. The third chapter details the lab environment and the implementation of the machine learning classifier, which includes data sourcing and flow filtering. The fourth chapter explores the data behavior and the selection of relevant features. The fifth chapter presents the flows' preprocessing and the machine learning models used for the classification and outlines; the testing of the flows, presents the results of the classification models, and analyzes their

performance. The sixth chapter discusses the results of the models, the project's limitations, and ways we could improve it.

Chapter 2

Background

The background chapter covers the technical context within which the dissertation will be done. This chapter presents the TPC/IP protocol details, cryptography algorithms used in the TLS encryption, digital certificate, and TLS handshake messages to understand the metadata and fields relevant to flag the malicious flows. In addition, the chapter also presents characteristics of malware using TLS, the challenges in the TLSI method used to detect encrypted malware traffic, current research on the alternative to TLSI, and machine learning classifiers used for anomaly detection.

2.1 Transmission Control Protocol/ Internet Protocol Model

The Transmission Control Protocol (TCP) enables clients and servers to communicate by sending packets successfully and intact over a network. The TCP/IP model defines how data is organised and transmitted between the client and server. The model is a combination of four layers: application, transport, internet, and network.

2.1.1 Application

The application layer is responsible for initiating requests at the application level and is concerned mainly with human interaction and the implementation of related protocols. It uses different protocols to complete the requests for different programs; some of the protocols used by the application layer include HTTP, HTTPS, and DNS. The application layer uses ports to pass on the data to the transport layer. The use of the ports makes it easier for the transport layer to understand the type of data from the application layer [49].

2.1.2 Transport

The transport layer performs host-to-host communications either the same or different hosts and on either the local network or remote networks separated by routers [48]. The transport layer uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP is preferred because of its reliability in ensuring data is transferred and reaches the host [44].

2.1.3 Internet

The internet layer handles packing, addressing, and routing packets over the network using the internet protocol (IP). The IP information is attached to each packet, and this information helps routers send packets to the right host [50].

2.1.4 Network

The network layer is responsible for the physical host-to-host delivery of packets through the network by encapsulating the IP packets into frames transmitted by the network and mapping IP addresses into physical addresses.

2.1.5 NetFlow

We can not talk about the IP and TCP protocol without talking about NetFlow. NetFlow represents all IP packets that pass in a network during a specified period, sharing the same IP protocol, source and destination IPs and ports. [40]. NetFlow answers questions regarding IP traffic: who, what, where, when, and how.

2.2 Cryptography Algorithms and Certificates

Before we look at the TLS/SSL protocol, we first need to understand cryptography fundamentals. Cryptography is the study of mathematical techniques to provide information security [54]. Cryptography uses encryption and decryption to ensure that the information or message cannot be read or understood by an eavesdropper.

2.2.1 Symmetric Encryption

Private key or symmetric encryption involves only one secret key to encrypt and decrypt information. The drawback with symmetric key encryption is that it requires both the sender

and receiver of the information to exchange the key used to encrypt the information before decryption.

2.2.2 Asymmetric Encryption

Public key or asymmetric encryption uses two keys, one to encrypt and the other to decrypt. The encryption is done with the recipient's public key, and decryption is done with the receiver's private key.

2.2.3 Cryptography Algorithms Key Strength

The security level of a cryptosystem is the number of operations required to defeat the objective of security. A brute force attack is how a cryptosystem gets breached, where an attach tries all possible keys to decrypt the encrypted information. The feasibility of a brute force attack depends on the length of the private key. In general, the longer a key is, the better security it provides, assuming it is genuinely random [6].

Key Size (bits)	**Number of Alternative Keys**	**Time required at 10^6 Decryption/μs**
32	$2^{32} = 4.3 \times 10^9$	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	10 hours
128	$2^{128} = 4.3 \times 10^{38}$	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	5.9×10^{30} years

Table 2.1 The average time required for exhaustive key search

However, key length on its own is not sufficient to ensure the security of the encrypted information. The absolute strength of a key also depends on the algorithm used for encryption. Some algorithms are inherently stronger than others for any given key length.

2.2.4 Hybrid Cryptosystem

A hybrid cryptosystem makes use of the advantages in both symmetric and asymmetric encryption, where asymmetric encryption is used for distributing the symmetric keys and symmetric encryption is used for encryption and decryption of the information. Figure 2.1 shows how a hybrid cryptosystem uses both symmetric and asymmetric encryption.

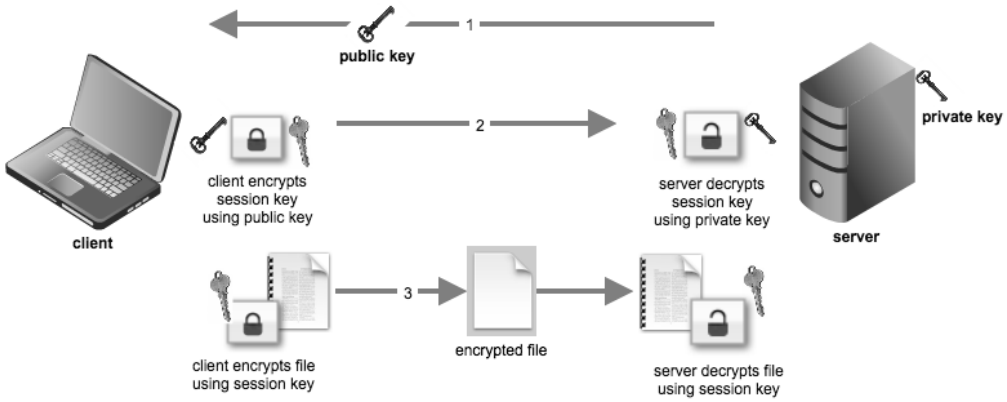

Figure 2.1 Hybrid Cryptosystem [57]

2.2.5 Hashing

Hashing requires no key or secret information; hashing is a cryptography method that converts any form of data into a unique irreversible fixed-size binary string. The unique irreversible fixed-size string is called the 'hash value', 'message digest', 'digital fingerprint', or 'checksum'. The primary use of hashing is to validate THE integrity of a piece of data. Two files can be assumed to be identical only if the checksums generated from each file, using the same cryptographic hash function, are identical [20]. The encrypted hash and other information like the hashing algorithm are known as a digital signature.

2.2.6 Certificates

A digital certificate is a cryptographic file that contains a digital signature. In the SSL/TLS context, a digital certificate is an X.509 certificate that asserts server identity and facilitates encrypted connections. The certificate binds the signing key to the server, and the digital signature informs the client that the server can be trusted. Certificates can be self-signed; however, certificates are more trusted when signed by a Certificate Authority (CA) [26].

Certificate Chain

The Chain of Trust refers to tracing the server's SSL certificate back to a Certificate Authority. There are three parts to the chain of trust: server, intermediate CA, and root CA. Figure 2.2 shows how to obtain a certificate and tracing it back to the root CA through the certificate chain.

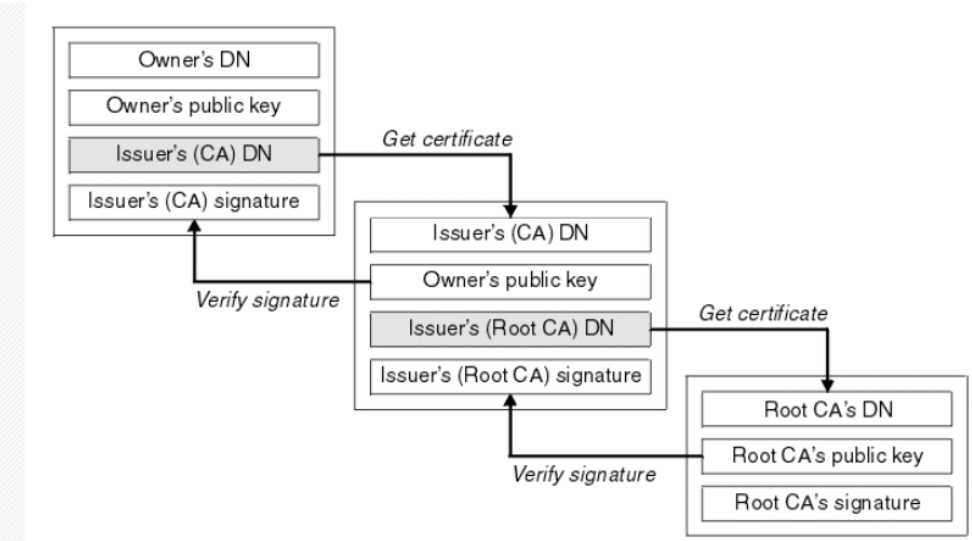

Figure 2.2 Certificate Chain [24]

2.3 Transport Layer Security

Confidentiality, Integrity, and Availability, known as the CIA triad, is a security model created to guide information security policies within an organization. These three elements of the CIA triad are considered the three most essential components of security. Below are the definitions from the ISO 27001 Standard [29]

- Confidentiality: "property that information is not made available or disclosed to unauthorized individuals, entities, or processes".
- Integrity: "property of accuracy and completeness".
- Availability: "property of being accessible and usable upon demand by an authorized entity".

The Internet was generally not built with security in mind, and this is why it is natural for the CIA triad to be compromised. This vulnerability demanded the existence of HTTPS. HTTPS does the same as HTTP, but securely by encrypting the data using cryptographic protocols and TLS/SSL. These protocols provide integrity and privacy

SSL and TLS are cryptographic protocols that use the cryptography algorithms explained in Section 2.2 to provide authentication and data encryption between servers, clients, and applications operating over a network. TLS evolved from SSL protocol, but SSL is no longer considered secure.

The TCP/IP model explained in Section 2.1 data is passed from the highest to the lowest layer, the TLS/SSL protocol is between the application and transport layer.

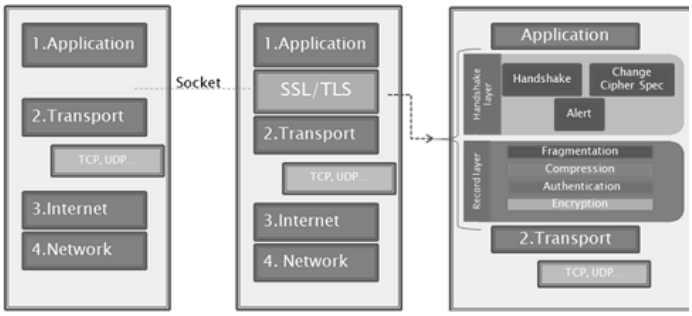

Figure 2.3 SSL/TLS Architecture [13]

Below is the whole history of SSL and TLS releases:

- SSL 1.0: security issues prevented its release.
- SSL 2.0: released in 1995. Deprecated in 2011 and has known security issues.
- SSL 3.0: released in 1996. Deprecated in 2015 and has known security issues.
- TLS 1.0: released in 1999 as an upgrade to SSL 3.0 and deprecated in 2020.
- TLS 1.1: released in 2006 and deprecated in 2020.
- TLS 1.2: released in 2008.
- TLS 1.3: released in 2018.

TLS 1.2 is currently the most widely implemented version of TLS and offers improvements over the older version, and it supports multiple key exchange algorithms and several cryptography algorithms. Even with the improvements, there are still concerns about the overall security, privacy, performance, and network overhead of TLS 1.2. These concerns are addressed in TLS 1.3 with a faster and simpler TLS handshake, more secure ciphersuites, and Zero Round-Trip Time (0-RTT) key exchanges to further streamline the TLS handshake [42].

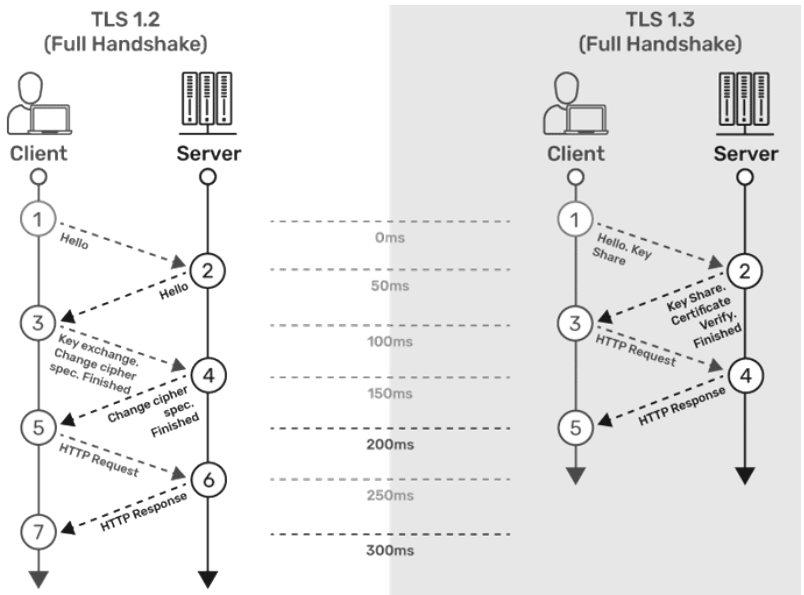

Figure 2.4 Comparison of TLS 1.3 and TLS 1.2 Handshake [42]

2.3.1 Handshake

The SSL/TLS handshake enables the SSL/TLS client and server to establish the secret keys with which they communicate. Below are the steps that enable the SSL/TLS client and server to communicate with each other:

- Agree on the version of the protocol to use.
- Select cryptographic algorithms.
- Authenticate each other by exchanging and validating digital certificates.
- Use asymmetric encryption techniques to generate a shared secret key. SSL/TLS then uses the shared key for the symmetric encryption of messages. This is the key distribution approach discussed in Section 2.2.4.

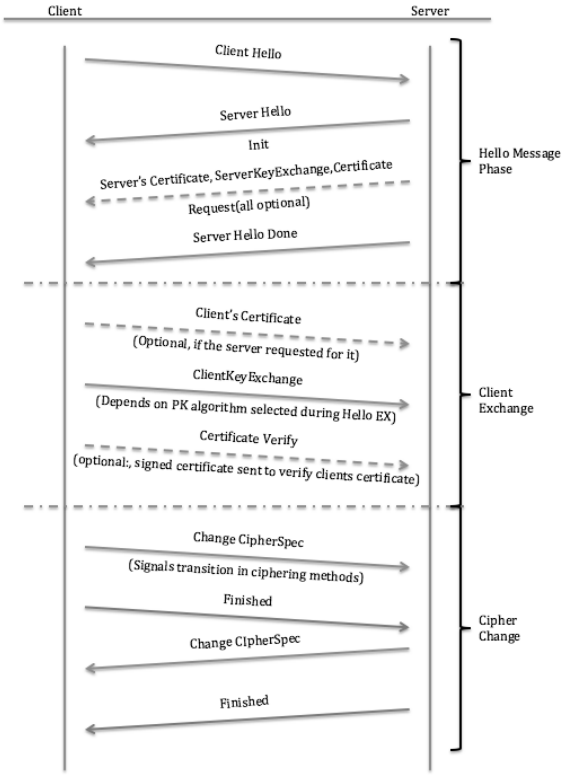

Figure 2.5 TLS Handshake [21]

Client Hello

The session begins with the client saying "Hello". The client provides the following:

- Version: TLS protocol version number that the client wants to use for communication with the server.
- Client random: 32-byte pseudorandom number used to calculate the Master secret and create the encryption key.
- Session identifier: unique number used by the client to identify a session.
- Ciphersuites: list of ciphersuites supported by the client ordered by the client's preference. The ciphersuite structure: TLS_Key Exchange Algorithms_Digital Signature Algorithm_Bulk Encryption Algorithms_Message Authentication Code Algorithms. An example $TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256$.
- Compression method: list of methods that will are used for compressing data before it gets encrypted.
- Extension: list of additional information provided to the server which is clear on what the server should do with the information in both cases where the server supports the information or not.

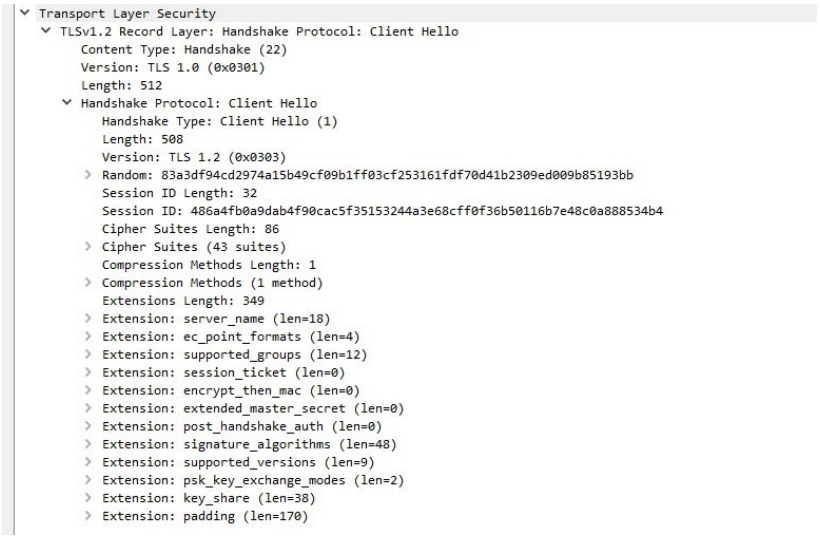

Figure 2.6 Client Hello

Server Hello

The server says "Hello" back. The server provides the following:

- Server version: highest TLS protocol version supported by the server and supported by the client.
- Server random: 32-byte pseudorandom number used to generate the Master Secret.
- Session identifier: unique number to identify the session for the corresponding connection with the client.
- Ciphersuite: single strongest ciphersuite that both the server and the client support. If there is no supporting cipher suite, then a handshake failure alert is created.
- Compression method: compression algorithm agreed by both the server and the client; this is optional.
- Extension: extensions agreed by both the server and the client.

Figure 2.7 Server Hello

Server Certificates

The server sends the client a chain of X.509 certificates to authenticate its public key.

Figure 2.8 Server Certificates

Key Exchange

Both the server and the client have public and private keys, both parties distribute their public key to each other. To calculate the encryption key, both parties use their private keys and each other's public keys. Figures 2.8 and 2.9 show the key exchange between the client and server.

Figure 2.9 Server Key Exchange

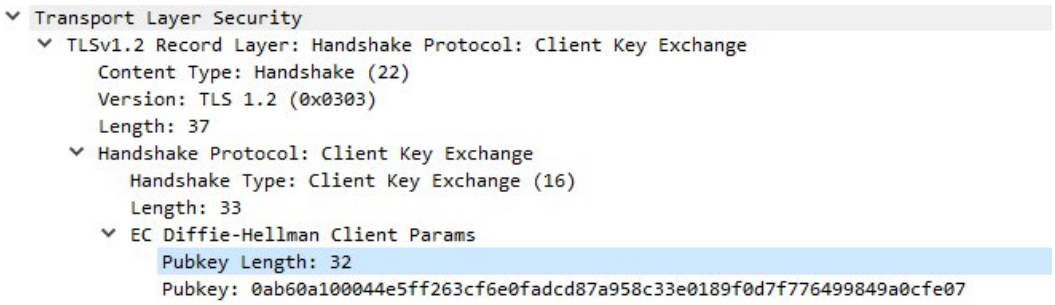

Figure 2.10 Client Key Exchange

Server Done

The server indicates it has finished the handshake with the client.

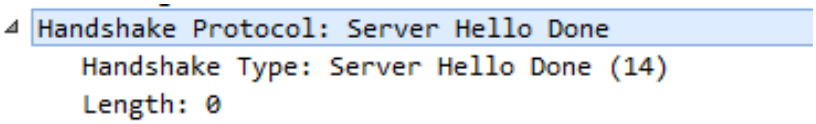

Figure 2.11 Server Done

Change Cipher Spec

Notifies both the client and server that the traffic will be encrypted under the negotiated CipherSpec and keys.

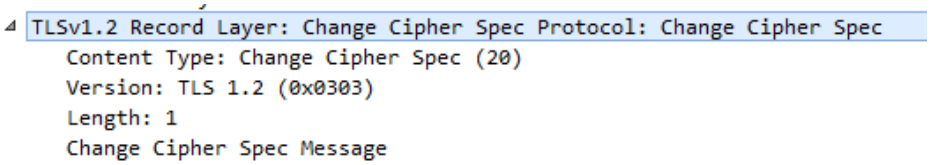

Figure 2.12 Cipher Change

2.4 TLS-based Malware

Several known malware use SSL/TLS to hide their malicious traffic, such as IcedID, Emotet, Trickbot, and many more. Below we highlight some of the TLS behaviours exhibited by the three malware. An extensive list of malware using TLS is in Appendix C. The tactics and techniques used by the malware discussed in this section are found in the MITRE ATT&CK database [17].

2.4.1 Emotet

Emotet first emerged in June 2014 and was primarily used to target the banking sector [17]. Emotet spreads through spam emails; the emails may contain familiar branding designed to look like a legitimate email to persuade users to click the malicious file(s) containing JavaScript code.

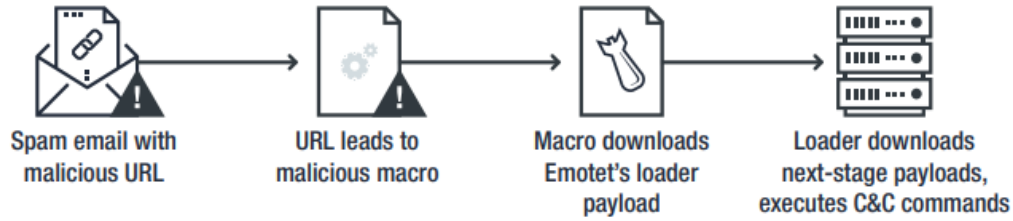

Figure 2.13 Emotet infection chain [16]

Emotet has become modular after evolving from malicious JavaScript files to macro-enabled documents to retrieve the virus payload from C2 servers. Emotet is also a downloader for other malware variants such as TrickBot, Qakbot IcedID [17], and Figure 2.14 and Figure 2.15 show this. Emotet mainly uses the HTTPS protocol for its communication, but it also uses HTTP over ports such as 20, 22, 7080, and 50000.

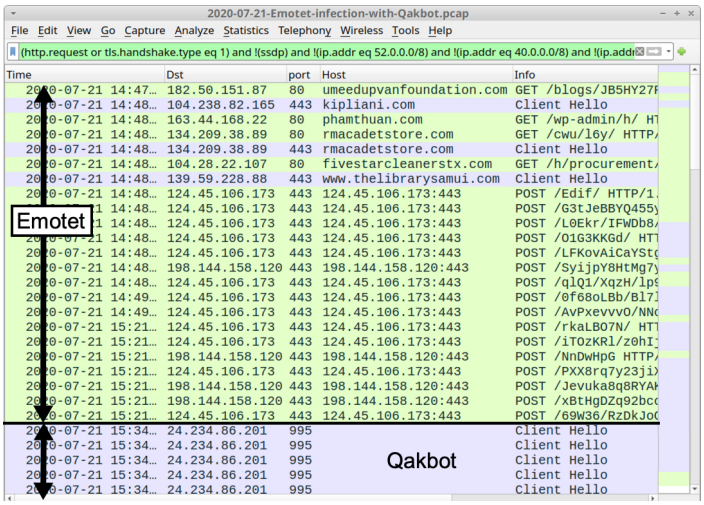

Figure 2.14 Emotet with Qakbot [16]

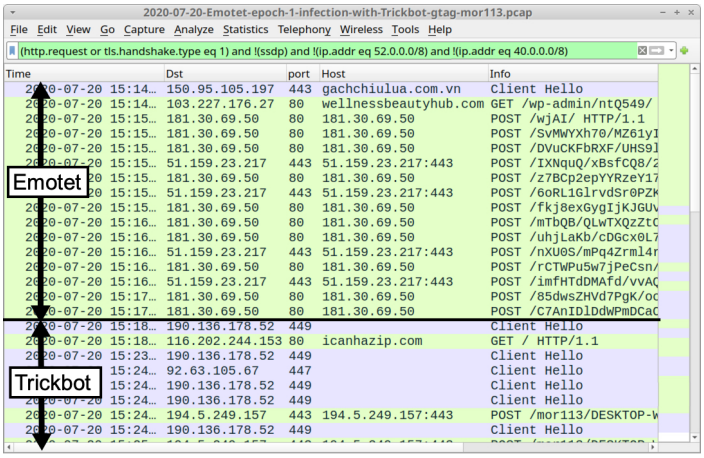

Figure 2.15 Emotet with Trickbot [16]

2.4.2 IcedID

IcedID is a modular banking malware designed to steal financial information that has been observed in the wild since 2017 [17]. IcedID spreads via malspam emails typically containing Office file attachments and used Emotet as a downloaded in multiple campaigns.

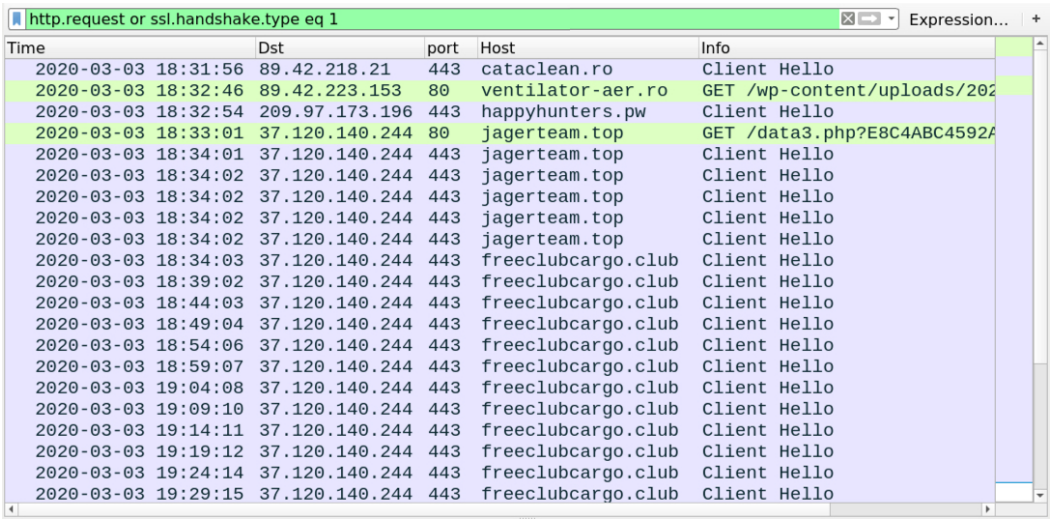

Figure 2.16 Traffic from the infection filtered in Wireshark [16]

IcedID uses TLS in all of its C2 communications and the SSL certificates are self-signed. Figure 2.17 shows that the issuer and the subject are the same.

Figure 2.17 IcedID self-signed certificate [16]

2.4.3 Trickbot

Trickbot is a module-based malware that emerged in late 2016 and was initially identified as a banking Trojan, but it has gradually extended its functionalities to collect credentials from its victims' email accounts, browsers and installed network apps [4]. Trickbot uses the HTTPS protocol to communicate with its C2 servers.

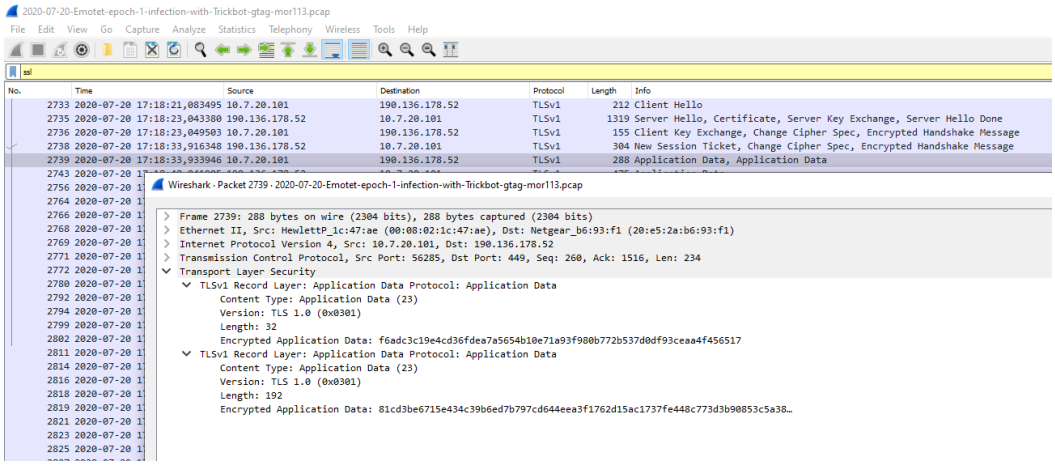

Figure 2.18 Trickbot from the infection filtered in Wireshark [16]

In Figure 2.16, we established that Trickbot communicates over HTTPS; another interesting thing about Figure 2.18 is its use of TLS 1.0 in 2020 as the vulnerabilities of this version are known.

2.5 Challenges in TLSI

Most enterprises use TLSI to inspect traffic contents before forwarding the packets inside and outside the network. However, this method has drawbacks. Below is a summary of the drawbacks discussed in the NSA report [30] , The Security Impact of HTTPS Interception [38] paper and The Risks of SSL Inspection [39] article.

- The NSA report highlighted risks around Forward Proxy, TLS session, and Certificate Authority. A forward proxy that forwards decrypted traffic to external inspection devices could misroute the traffic and expose sensitive traffic to unauthorized or weakly protected networks. Unexpected changes in TLS certificates received from external servers might indicate man-in-the-middle attacks against the proxy.
- In some cases, the TLSI software fails to validate the certificates of systems that it connects to, leaving the client not knowing if they are connected to a legitimate site.
- When detecting a certificate error, some of the TLSI software will first send the client's request to the server before notifying the client about the error. This communication allows the attacker to be still able to view or modify the client's sensitive data.
- Some of the TLSI software deploy TLS libraries with minimal customization, and the default settings for these libraries are vulnerable, making the TLSI itself vulnerable.
- TLSI is expensive in terms of performance and complexity, and this causes a drop in the network's performance.
- Decrypting packets for inspection is a violation of privacy.
- The products that perform the TLSI keep track of all the TLS certificates and keeps them needed for packet decryption, and managing this process is complex and costly.

2.6 Alternative to TLSI

Most of the network detection work has focused on techniques to detect HTTP malware. Therefore, there is limited research around the detection of HTTPS malware without decryption.

In Jakub Loko, Jan Kohout, Premysl Cech, Tomas Skopal1 and Tomas Pevny's paper on "k-NN Classification of Malware in HTTPS Traffic Using the Metric Space Approach" [46],

they presented the efficiency of metric indexing for approximate k-NN search over datasets of sparse high-dimensional descriptors of network traffic which reduce false-positive rates by an order of magnitude when compared to the ECM linear classifier while keeping the classification fast enough.

In František Strasák's paper "Detection of HTTPS Malware Traffic", a goal was set to detect HTTPS malware connections by extracting new features [56]. He obtained malware data from the Bro IDS program and he grouped; flows, SSL data, and X.509 certificates. For benign data, he used data from the Stratosphere project and created, by hand, his datasets. He used the algorithms: Neural Networks, XGBoost, and Random Forest to classify the HTTPS malware traffic. František Strasák's classifier had an accuracy of at least 96.64%. With the work of these key papers in mind, we now describe the problem statement for the research and an intended methodology to replicate, understand and assess these alternative approaches.

In Blake Anderson and David McGrew's work on 'Identifying Encrypted Malware Traffic with Contextual Flow Data' [35], they studied the differences in behaviour between benign and malware traffic using TLS handshake meta-data, DNS contextual flows linked to the encrypted flow, and the HTTP headers of HTTP contextual flows. They used the 10-fold cross-validation and l1-logistic regression machine learning supervised model with data features from DNS, TLS and HTTP flows from the same source IP address within a 5-minute window. The classifier they used had results of final accuracy is 99.993% and 0.00% false discovery rate. Their malicious dataset was collected from January to April 2016 from a commercial sandbox environment that receives user submissions and the benign dataset was collected during a 5-day period in April 2016 from a large enterprise network's DMZ.

In Blake Anderson, Subharthi Paul and David McGrew's paper on 'Deciphering Malware's use of TLS (without Decryption)' [33], their approach focused on using TLS within malware families to identify what characteristics of the specific family make it difficult to classify. They analyzed the differences between the benign and malware TLS parameters. They used a commercial sandbox environment to collect the first five minutes of a malware sample's network activity, and for benign traffic they collected TLS encrypted flows from an enterprise network. A logistic regression classifier with an l1 penalty was used as the supervised algorithm with the data features: flow metadata, sequence of packet lengths and times, byte distribution, and unencrypted TLS header information. The classifier depended on client-side TLS features and had an accuracy of 90.3% for the family attribution problem when restricted

to a single encrypted flow and an accuracy of 93.2% when they used all encrypted flows within a 5-minute window.

2.7 Anomaly Detection and Machine Learning

Machine learning is a combination of computational methods using experience to prove performance or make accurate predictions [47] by learning, meaning acquiring skills or knowledge by synthesizing useful concepts from historical data. Machine learning algorithms can do their learning either in a supervised or unsupervised manner, and the main difference between the algorithms is whether the data used for training is labelled or unlabelled.

- Supervised algorithms take data that is already labelled with ground truth and build a model that can predict the labels of unlabelled [53].
- Unsupervised algorithms take unlabelled data and learn patterns within, such that the new data can be mapped onto these patterns [53].

Our study is inspired by the research done by the Cisco team [35]; therefore, we also used supervised models. In Section 2.6, we discussed different researchers' approaches, and in all cases, the models were supervised classifiers. Moreover, supervised learning is more developed and understood than unsupervised learning [41], and it better suits our study as we have structure labeled data as input into the classifiers

In this study two classifiers will be considered:

- Logistic regression analyses a dataset of independent variables that determine an outcome and the outcome only has two possible outcomes (binary output).
- Random forest builds multiple decision trees the forest and merges them to get an averaged output to get a more accurate and stable prediction.

The Cisco team in [34] studied six common machine learning algorithms: linear regression, l1/l2-logistic regression, decision tree, random forest ensemble, support vector machine, and multi-layer perceptron, on how effective they are in classifying encrypted malware and random forest algorithm outperformed competing methods.

2.7.1 k-fold Cross-Validation

When evaluating predictive models, the datasets are partition into training and testing sets. Figuring out a split that will not affect the performance of the models cause noisy or biased estimates is a challenge. The k-fold cross-validation provides a way to improve the performance of machine learning models.

In k-fold cross-validation, the original dataset is randomly split into k equal size subsets and one subset is used for testing, and the remaining k-1 subsets are used for training. The classification model is then executed k times, each run a different set is selected as the testing set. The k results are then averaged to produce a single estimation. The advantage of this method is that all observations are used for both training and validation [1]. Below is an example of 5-fold cross-validation.

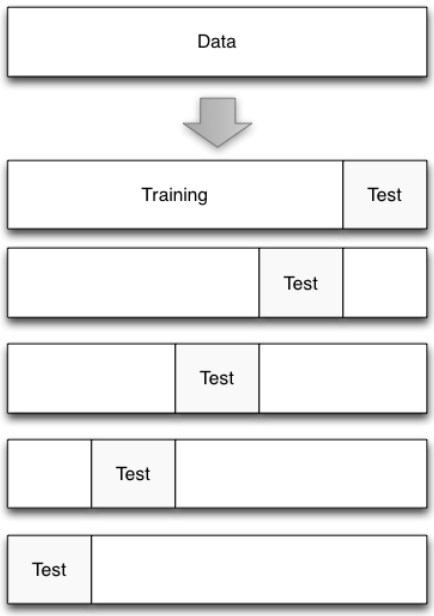

Figure 2.19 5-fold cross-validation

There is no science behind the selection of the k value. However, in the books Applied Predictive Modeling [43] and An Introduction to Statistical Learning [41] k=10 was found to provide a good trade-off of low computational cost and low bias in estimating model performance.

2.8 Summary

This chapter detailed; how data moves over the internet, malware uses the HTTPS protocol, and the different methods to detect malware over HTTPS.

The TCP protocol ensures that all data sent in a stream from client to server is in the correct order and is intact. The TCP protocol establishes the connection enabling the HTTP protocol to instruct the client and server how to read and process the data. To ensure that the data is transmitted securely, the client and server must agree 'handshake' on the cryptographic algorithms and keys for privacy and authentication. The SSL/TLS protocol does this handshake. The combination of the HTTP and TLS protocol is the HTTPS protocol.

Malware authors make use of the HTTPS protocol to hide the malware and evade detection. Through the analysis of IcedID, Emotet, and Trickbot, we noted that Malware distinctly uses TLS compared to benign use. This behavior is the basis of the machine learning classification model to detect malware without decryption.

Chapter 3

Design and Architecture

Data is the most important part of Machine Learning; we cannot train any classification model without data. Therefore, it is essential to understand the origins of our data for better interpretation and use. The first section details the process and techniques used to build an isolated lab environment separate from the local machine to capture malware traffic packets. The second section presents how and where the benign and malicious traffic packets were sourced. The third section shows the process of combining the traffic packets to respective TLS flows and filtering the final TLS flows used in the classification models. Figure 3.1 shows a high-level view of the experiment.

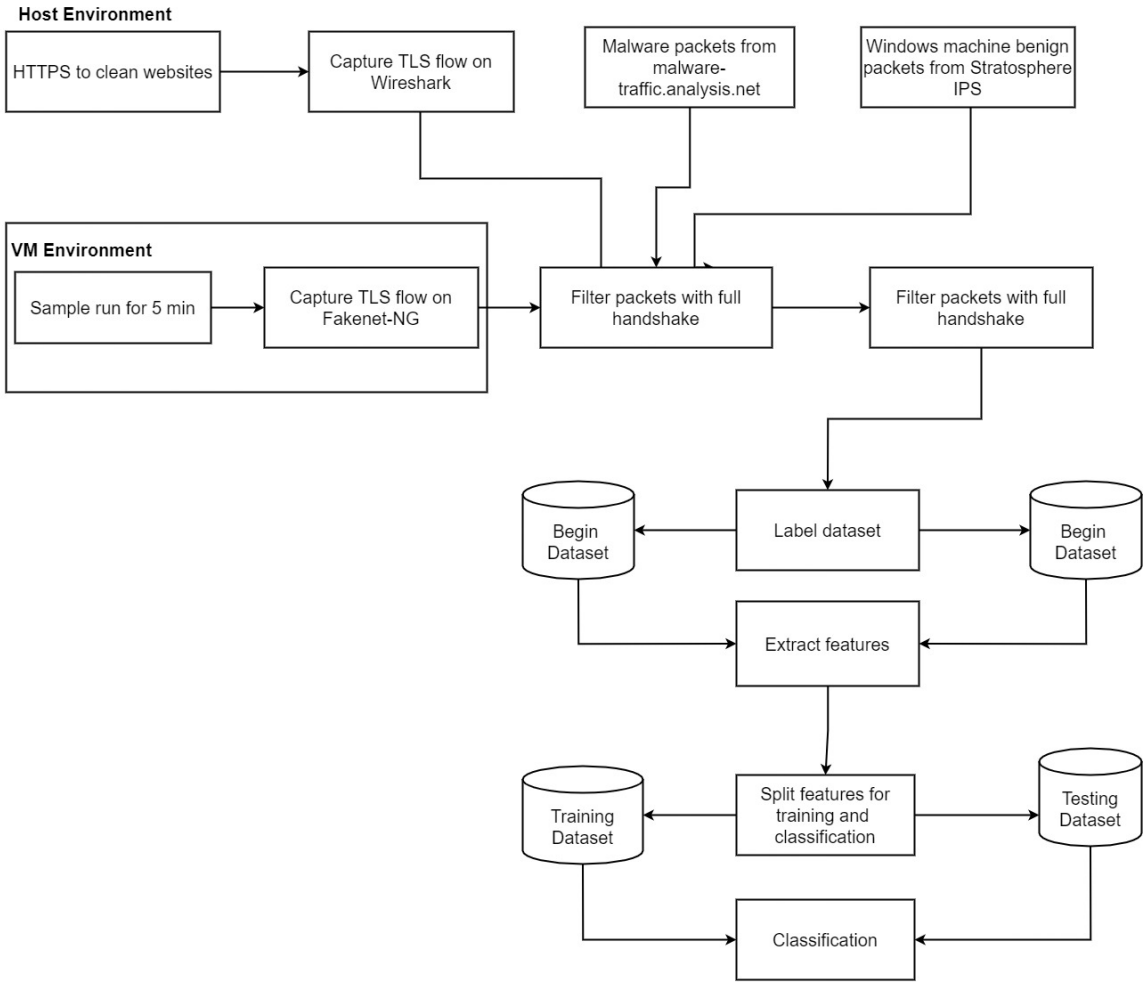

Figure 3.1 Experiment Design

3.1 Lab Environment

The lab is a Windows 7 Host running a VirtualBox Windows 10 guess Virtual machine (VM). A VM is an isolated environment that appears to be a whole computer but only has access to a portion of the computer resources. Virtual machine gives user the illusion of running directly on the physical machine [31].

A VM provides an isolated environment that allows researchers to trigger malware, intercept it and analyse its actions in a controlled environment. However, with the rising use of VMs for malware analysis, malware developers are actively trying to stop such analysis by detecting VMs.

Some of the common side-effects on VM detection by malware are that the malware:

- Does not connect with its C2 servers.
- Keeps its malicious code encrypted.
- Shuts down the VM

There are currently three categories of methods for locally detecting the presence of a VirtualBox VM, looking for VM artefacts in:

1. Hardware
2. Registry, File systems and Processes / Services
3. Memory

To counter malware detecting that it is in a VM, we make changes in some of the VM's artefacts mentioned above.

3.1.1 Hardware

Most of the modern user machines have the following hardware properties:

- 4G or more memory size
- More than 1 CPU core processor
- 80G or more drive size

Therefore, we changed our windows 10 VM default hardware allocate to 4G memory size, 2 CPU core processors, and an 80G drive.

A VM gives an abstract view of hardware components through the MAC address, BIOS, USB controllers, and other adapters. Malware queries the hardware component to detected the virtualized hardware. A MAC address is a unique identifier assigned to Network Interface Controllers on a machine. The prefix of a MAC address indicates the network adapter's vendor and the prefix for VirtualBox is 08:00:27, as seen in Figure 3.2. We then changed the VM's MAC address to the host's MAC address C85B77E52190 as seen in Figure 3.3

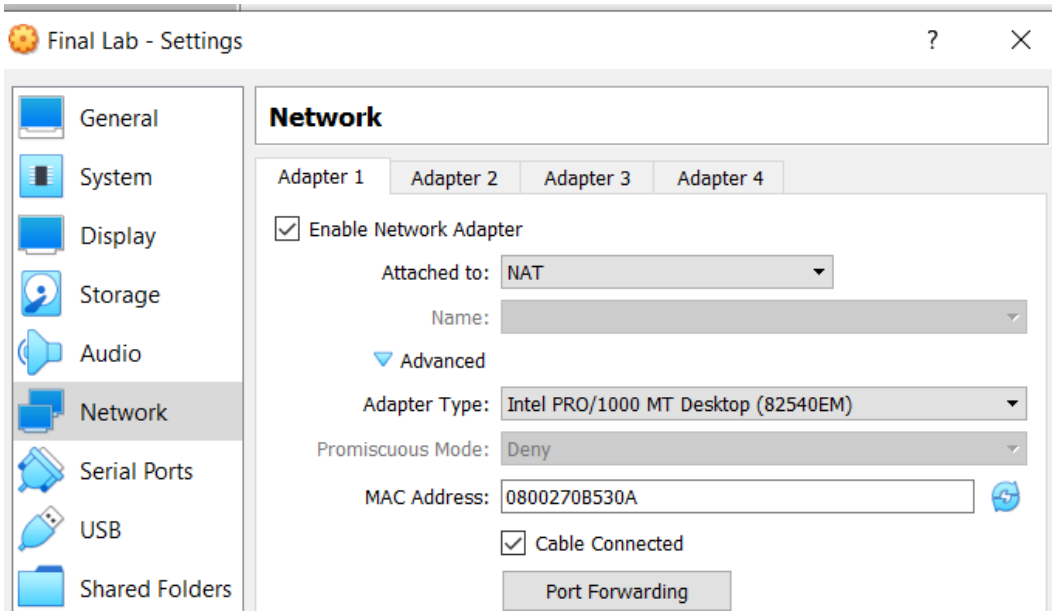

Figure 3.2 VM's Original MAC

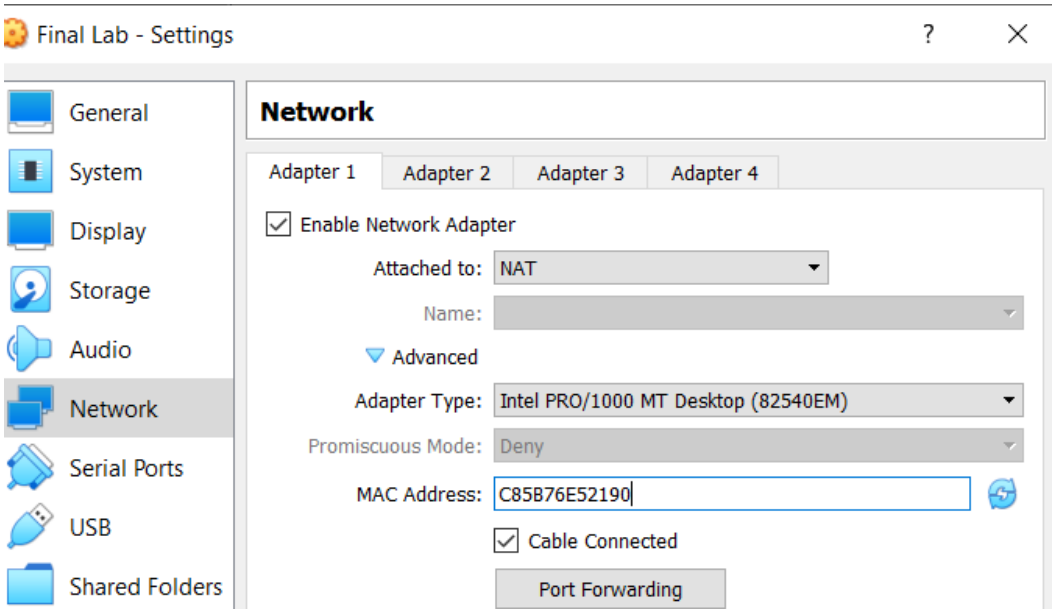

Figure 3.3 VM's New MAC

The Win32_BIOS and Win32_ComputerSystem base classes are used to retrieve and change a VM's computer make and model (manufacturer name, model number) and other hardware details. Figure 3.4 shows the VM's original hardware details and the SMBIOSBIOSVersion and Model values have the value "VirtualBox" which would immediately alert the malware that it is in a VM. We then used the Win32 base class script in Figure 3.5 to change these values to the host's values and the new VM's values are in Figure 3.6.

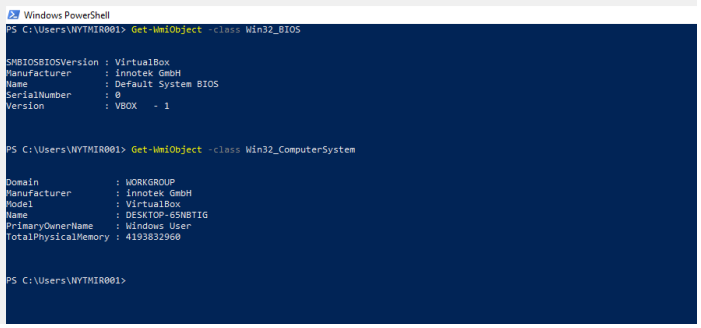

Figure 3.4 VM's Original Hardware Details

```
#pragma namespace ("\\\\.\\root\\cimv2")
class Win32_BIOS
{
    [key] string SMBIOSBIOSVersion;
    string Manufacturer;
    string SerialNumber;
    string Name;
    uint16 BiosCharacteristics[];
    string Version;
};

[DYNPROPS]
instance of Win32_BIOS
{
    SMBIOSBIOSVersion = "V2.90";
    Manufacturer = "TOSHIBA";
    SerialNumber= "z9131790W";
    Name ="Ver 1.00PARTTBL";
    BiosCharacteristics = {4,7,8,9};
    Version= "TOSQCI - 6040000";
};


class Win32_ComputerSystem
{
    [key] string Name;
    string Domain;
    string Manufacturer;
    string Model;
    string OEMStringArray[];
    string PrimaryOwnerName;
};

instance of Win32_ComputerSystem
{
    Name = "DESKTOP-65NBTIG";
    Domain = "WORKGROUP";
    Manufacturer = "TOSHIBA";
    Model = "Satellite";
    OEMStringArray = {"Welcomes to the Jungle"};
    PrimaryOwnerName = "Miranda";
};
```

Figure 3.5 Win32 Base Classes Script

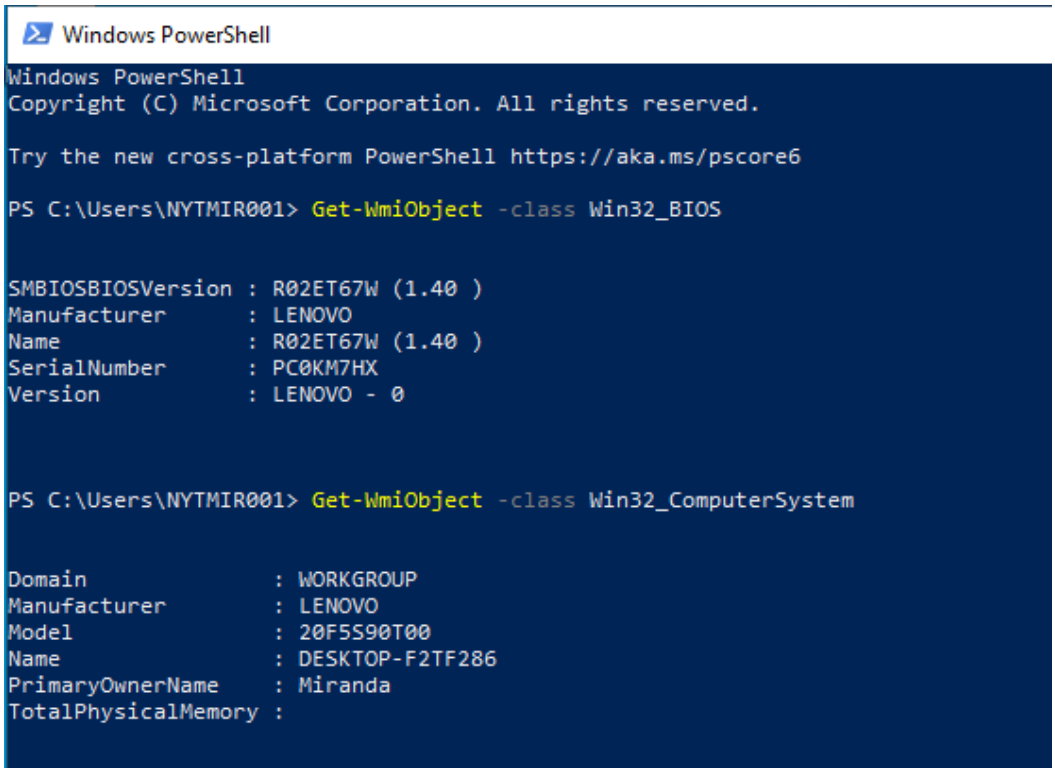

Figure 3.6 Win32 Base Classes Change

3.1.2 Registry, File systems and Processes / Services

The registry is a hierarchical database that contains critical data for the operation of Windows and the applications and services that run on Windows [23]. Virtualized Windows environments will often contain various registry entries not commonly found on physical machines, the presence of which helps the malware detect that it is running on a VM. Whenever a new VM is spawned, the guest operating system (OS) leaves registry keys related to it. Appendix A shows the lists of VirtualBox files. Most of these files below are generated when VirtualBox guest addition is installed; guest addition allows the transfer of data from USB to the VM machine. Ejecting guest addition will automatically remove the files. Figure 3.7 shows the registry of the VM after we changed its registry keys values to the host.

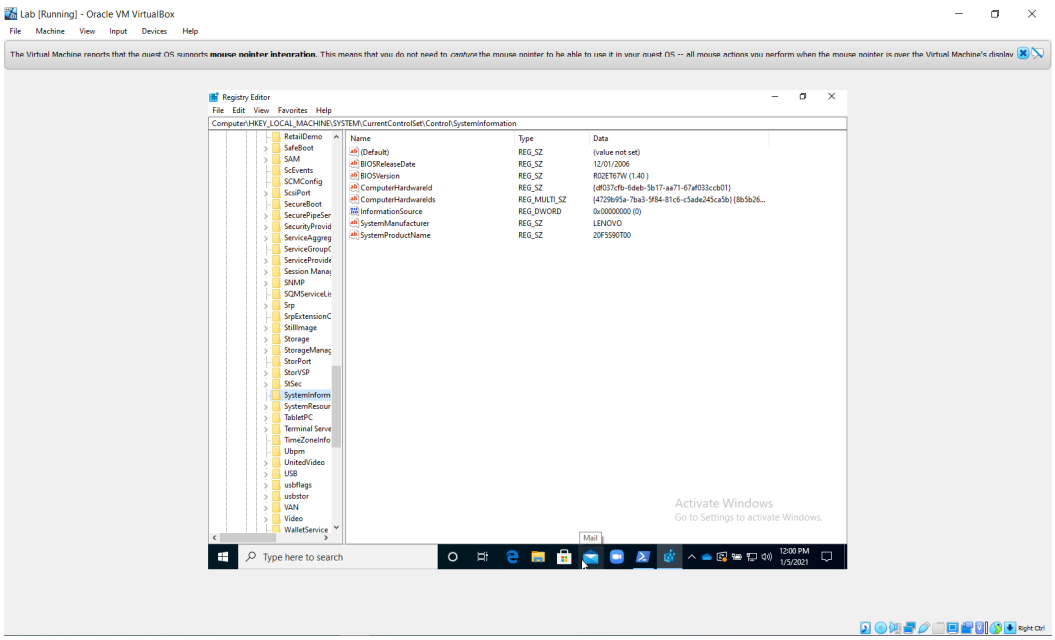

Figure 3.7 VM's Registry Change

3.1.3 Memory

The location of various memory structures, especially Interrupt Descriptor Table (IDT), varies in a VM compared to a physical machine. In November 2004, a researcher named Joanna Rutkowska introduced code called "The Red Pill". Her code runs a single machine language instruction called Store Interrupt Descriptor Table (SIDT); this instruction stores the contents of the IDT Register, and the IDT is typically located at 0xffXXXXXX, while on a VM it is located at 0xe8XXXXXX [45]. Unfortunately, there is no counter for the memory check. Any malware with the capability to do the memory check will immediately know it is in a VM. Emotet is one of the malware families that have this capability.

3.2 Data sources

To train and test our supervised classification algorithms, we need features extracted from both the malware and benign TLS flows. This chapter presents where the TLS flows were sources and how the features got extracted.

3.2.1 Benign Data

The benign data is the combination of requests to the United States of America's universities' websites and the domains of the websites in Alexa [3] top 500 websites visited in: South Africa, Nigeria, Kenya, Ghana, China, Canada, United Kingdom, South Korea, United States of America, Australia, Denmark, India, Netherlands, Switzerland, Hong Kong and Russia. After filtering out overlapping websites in Alexa top 500 websites for the countries and combining them with university websites, it came to a total of 6540 websites. To increase the number of packets, we downloaded more pcap files of traffic coming from Windows machines from the Stratosphere IPS 2017 database [15].

3.2.2 Malware Data

One of the challenges of analyzing malware using HTTPS is the lack of an excellent public dataset. For this study, we wanted to verify if the claim made by the Cisco team in [35] from 2016 still holds as malware evolves and adopts new ways of evading detection.We got 40000 malware samples from VirusTotal academic malware sample repository for the second half of 2019 submissions to accomplish this. The malware samples were not labelled whether they use HTTP or HTTPS connection. Therefore, we attempted to execute "all" the samples and only capture the flows of the malware samples that use the HTTPS protocol.

The 40000 malware samples from VirusTotal were all in a zipped file which we unzipped inside the VM as we wanted to automate the execution of the malware samples. Manually executing one malware sample at a time would have been impossible in terms of time consumption. Among the 40000 samples, the most advanced malware with the VM detection memory check capability discussed in Section 3.1.3 were in the VM. We managed to capture some packets; however, the VM repeatedly crashed, and we continuously rolled back to the base image of the VM, but in the end, it became difficult to capture packets for all the samples.

To increase the number of TLS packets, we sourced more data from malware-traffic-analysis.net [16], downloaded pcap files for the years: 2018 and 2019. The final pcap files are a combination of the ones captured in the lab and those from malware-traffic-analysis.net.

3.2.3 Benign TLS Packet Capture

A python script executed the automated HTTPS requests to the clean websites in the host environment, and Wireshark captured the packet. For each HTTPS request: redirects were not allowed, and only packets through port 443 were captured.

3.2.4 Malware TLS Packet Capture

The malware samples were transferred to the guest VM. Each malware sample was executed and allowed to run for 5 minutes, and the packets of the malicious traffic were captured with Fakenet-NG. FakeNet-NG is a next-generation dynamic network an analysis tool for malware analysis, and it allows analysts to intercept and redirect all or specific network traffic while simulating legitimate network services [10]. Packets captured with Fakenet-NG were saved as Wireshark pcap files. In our experiment, Fakenet-NG is set up to only listen to HTTPS connections made by the malware.

3.3 Flow Filtering and Feature Extraction

The respective malware and benign packets were combined into one big pcap file on Wireshark resulting in one malware pcap file and one benign pcap file. Figure 3.8 gives an overview of how we changed the pcap files to flow, filtered the flows, and extracted features from the flows.

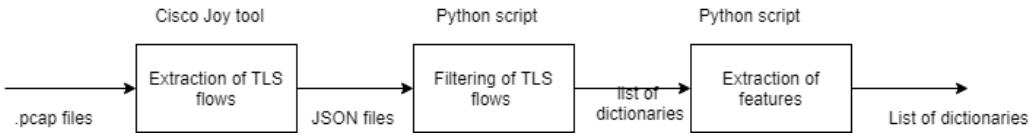

Figure 3.8 Feature extraction pipeline

The open-source Cisco Joy [5] tool was used to extract features from the pcap files and stores them in JSON files. Joy combines the packets into flows, and each flow has the following features: flow metadata, non-encrypted TLS data, certificate, the sequence of packet lengths and sequence of packet times (SPLT) of TLS records, and the empirical probability distribution of the bytes within the data portion of a flow.

Flows with an incomplete handshake and the TLS data from the clientHello, serverHello, certificate, and clientKeyExchange messages were discarded. The number of packets for

each malware file flow is independent, some malware traffic will have more packets than other. Malware flows with more packets can introduce bias in the model. Therefore, to ensure non-bias in the packet distribution only the first 300 packets in each flow were selected.

For the SPLT features to be effective, flows with less than three packets in each direction were removed. Some of the pcap files from malware-traffic-analysis.net [16] were captured beyond the 5 minute period we set to capture both benign and malicious packets. Therefore, we remove packets that occurred after the 5 minutes windows.

As a precaution to verify that the TLS flows we observed were indeed malware behavior, not a consequence of Windows 10 or 7's default TLS library, we removed any TLS sessions having the same ordered ciphersuite list as the default SChannel implementation. [35]. The default Windows 10 and 7 SChannel implementation list is in Appendix B. Table 3.1 shows the number of flows before and after filtering.

TLS Flows	**Before Filtering**	**After Filtering**
Benign	80321	65863
Malicious	10201	6735

Table 3.1 TLS flows filtering

3.4 Summary

The VM's hardware, registry, file systems, and services were changed to increase the chances of the malware executable not detecting that it is in an isolated environment. The malicious data sourced as executable files from VirusTotal and packets from malware-traffic-analysis.net; the executable files were ran in the lab, and their traffic packets capture with Fake-Net.

The benign data sourced as packets from the Stratosphere IPS 2017 database and HTTPS requests to the United States of America's universities' websites and the domains of the websites in Alexa top 500 websites visited in 21 countries. The requests' packets were captured in the lab with Wireshark.

Both malicious and benign packets were parsed into flows in JSON files using the Cisco tool Joy. The tool combined the packets into flows, and each flow has the following features: flow metadata, non-encrypted TLS data, certificate, the sequence of lengths and arrival times of TLS records, and the empirical probability distribution of the bytes within the data portion of a flow. Flows with less than three packets in each direction or captured beyond the 5 minute period or ciphersuite list that matched the default Windows 10 and 7 SChannel implementations were filtered out.The final flows are 65863 benign flows and 6735 malicious flows.

Chapter 4

Data Exploration and Features

Detecting HTTPS malware without decryption using machine learning supervised classifiers can only work if there are clear differences in malware and benign TLS flows behaviour. In Section 2.4, we discussed different malware traffic behaviours over TLS, and now we verify the existence of the distinct characteristics in our own benign and malware flows. Since our study is inspired by the Cisco team's research [35], we use their features as a baseline and compare the trends they found in their dataset with ones in our dataset. This chapter details the selection of the features for our study.

4.1 Cisco Dataset

The figures in this section are all taken from the Cisco paper [35], and in all the figures the red bars represent malware data and the blue represents benign data.

4.1.1 TLS Clients

The Cisco team reported that malware infected clients offer a unique set of ciphersuites compared to normal clients. The ciphersuites are usual weak or outdated and not recommended for use. In addition, malware clients hardly advertised diverse extensions compared to benign clients. This can be seen in Figure 4.1.

In Section 2.2.3, we established that the key strength of the depends on its length and the encryption algorithm. However, the Cisco team found that the client's public key length has discriminatory power; most of the benign traffic used a 512-bit (ECDHE_RSA) public key and malware almost exclusively used a 2048-bit (DHE_RSA) public key [35].

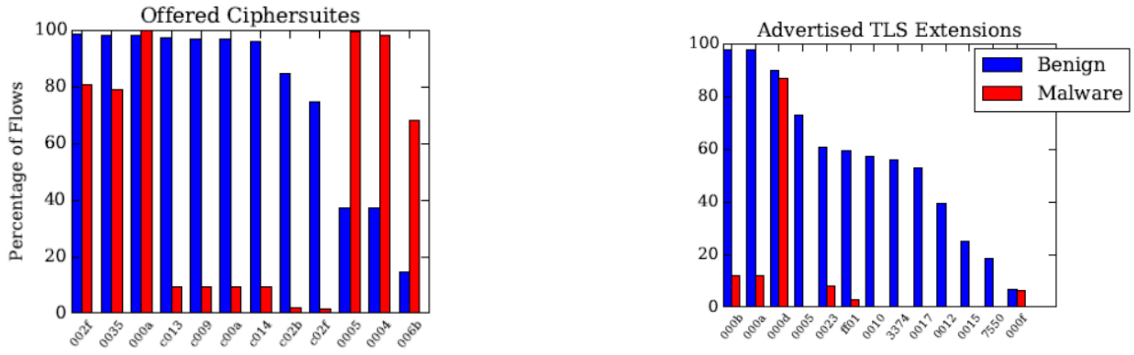

Figure 4.1 Offered ciphersuites and extensions [35]

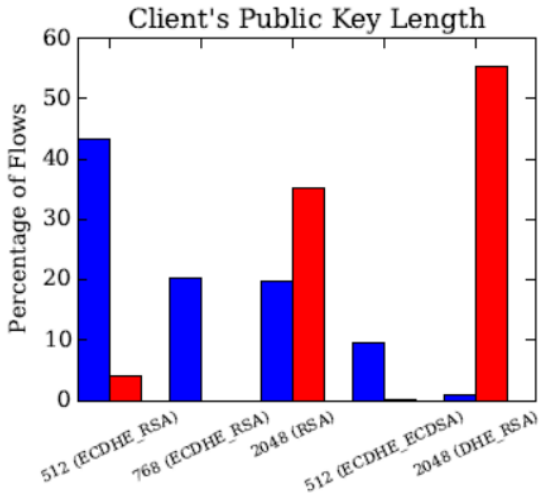

Figure 4.2 Client key length [35]

4.1.2 TLS Servers

Cisco reported that the C2 servers selected weaker ciphersuites compared to servers for benign communication and that lack of diversity in the extensions offered by malware clients resulted in the malware C2 servers rarely selecting extensions.

The presence of an SSL/TLS certification gives the impression that the traffic is not malicious, and this makes a certificate a valuable resource to threat actors as it can reduce the chance of early malware detection. Cisco reported that malicious servers send more self-signed certificates compared to normal servers and that there is no specific certificate period of validity that distinguishes malware and benign certificates. However, specific periods were used more often than others in the certificates' period of validity.

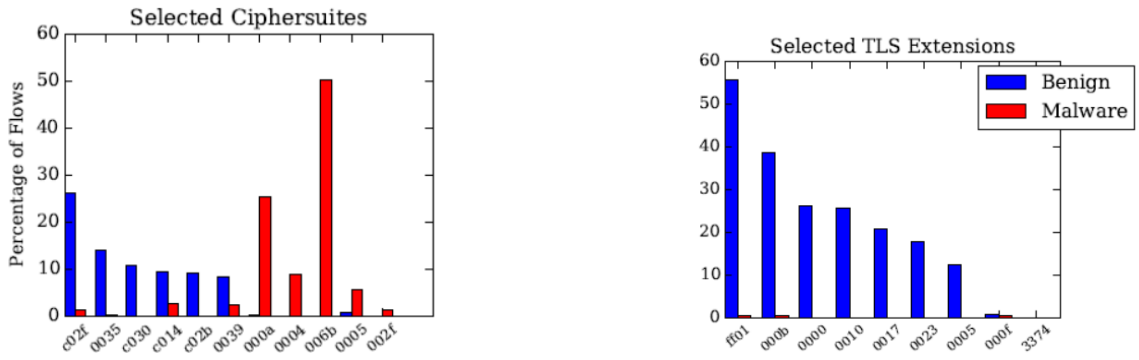

Figure 4.3 Selected ciphersuites and extensions [35]

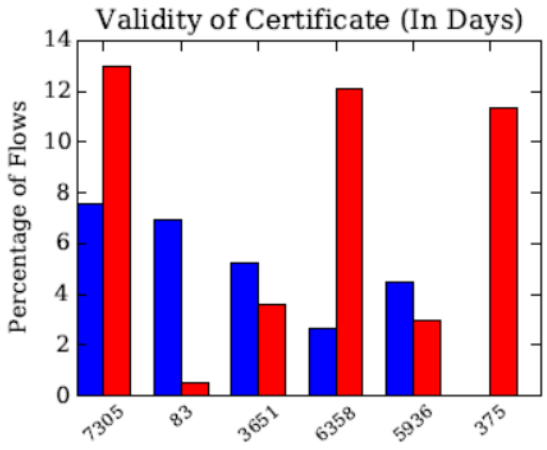

Figure 4.4 Server certificate [35]

4.1.3 Distributions

The size and Timing of the first few packets allow us to estimate the type of data inside the encrypted channel [14]. A sequence of packet lengths helps us understand how the malware communicates with its command control server and the time interval between packets helps understand how the malware writes to the disk. Figure 4.5 shows the clear difference between benign and malware sequencing of packet lengths and time.

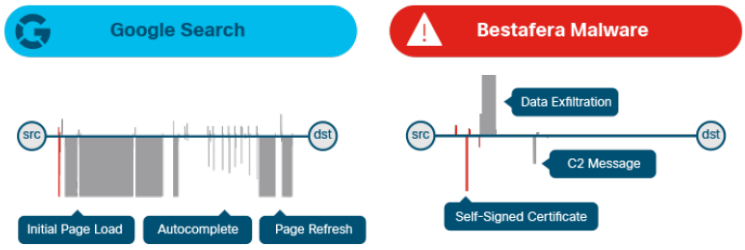

Figure 4.5 Packet lengths and inter-arrival time [7]

Byte Distribution

The byte distribution is a length-256 array representing probability that a specific byte value appears in the payload of a packet within a flow.

Sequence of Packet Lengths and Times (SPLT)

The Markov chain is used to model the sequence of packet lengths and times data [35]. The Markov chain property is that given the present state, the probability of the future state is independent of the past, meaning the future depends only on the present and not on the past. In relation to Figure 4.6, the Markov chain helps us at a point in time look at the packet length or inter-packet time and predict the next possible packet length or inter-packet time. Those predictions help us build a sequence of packet length or inter-packet time, which when analysed can help classify whether traffic is malicious or not.

The Markov Chain information can be captured in a matrix, the current states are listed vertically, and the subsequent potential states are listed horizontally. The values in the matrix are the number of transitions the current state can move to a potential state.

Implementation of the SPLT

- SPLT elements are collected for the first 50 packets of a flow and zero-length payloads are ignored [35].

- Length values are discretized into equally sized bins of 150-byte bins.
- Time values are discretized into equally sized bins of 50 milliseconds bins.
- A matrix T is then constructed where i is the row position of the current bin and j is the column position of the potential bin. Each entry $T[i\colon j]$ counts the number of transitions between bin i and j.
- Ten bins were used for both the length and times, resulting in a 10×10 matrices.
- The rows of T are then normalized to ensure a proper Markov chain [35].

4.2 Our Dataset

The list of all the ciphersuites and extensions observed in our dataset is in Appendix C.

4.2.1 TLS Clients

In Figure 4.7, we see that the malware-infected clients offered more ciphersuites than normal clients. This is interesting as the Cisco team observed that clients infected with malware offered a limited number of ciphersuites, leading to servers selecting weak ciphers. Figure 4.6 does not show all the ciphersuites offered for visual clarity. Nonetheless, we noted that 59% of the ciphersuites that malware-infected clients only offered are either legacy or ciphersuites that should be avoided. Therefore, there is a clear difference between the ciphersuites offered by malware-infected and benign clients.

In Figure 4.6, we see that the top ten ciphersuites offered by both malware and benign flows are almost the same. However, it is worth noting that $0 \times 000a$ was offed mostly by infected clients, which is aligned with the Cisco team's findings. Benign clients also offer this ciphersuite, just not with the same frequency as infected clients. The $0 \times 000a$ ciphersuite is weak and should not be in use.

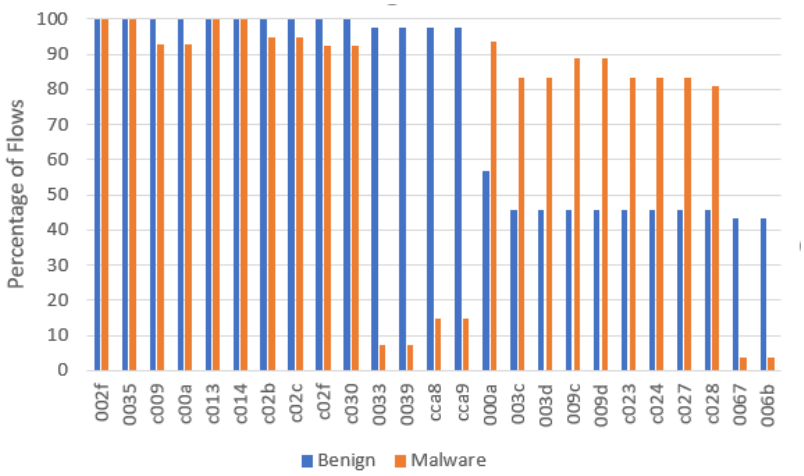

Figure 4.6 Offered ciphersuites

Please note that some values of the of the offered ciphersuites were omitted for clarity of presentation.

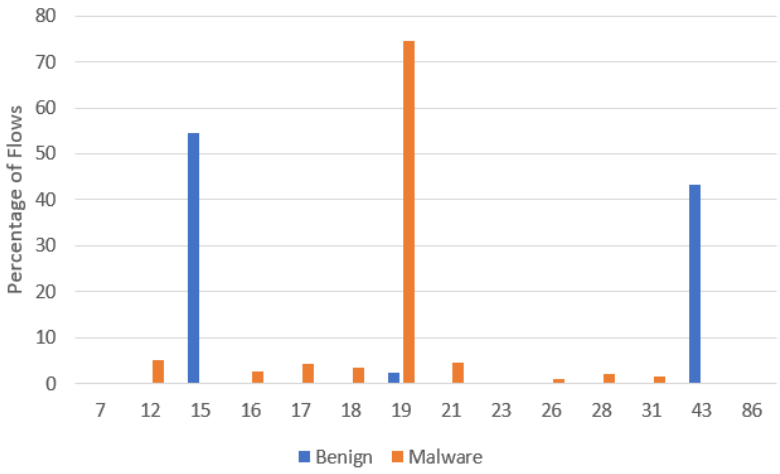

Figure 4.7 Number of offered ciphersuites

In Figure 4.8, we see both the benign and malicious clients offered diverse extensions. The benign clients mainly advertised the extensions 0×0000 (server name), $0 \times 000a$ (supported groups), and $0 \times 000f$ (heartbeat) and rarely offered 0×0018 (token binding) and $0 \times ff01$ (renegotiation info).

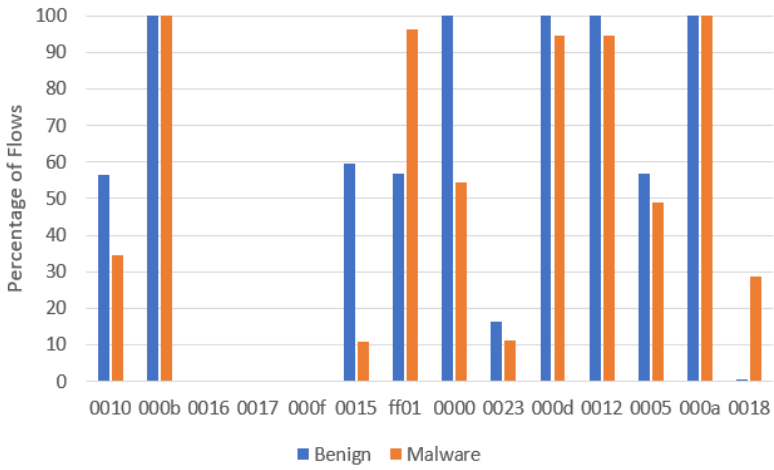

Figure 4.8 Offered extensions

In Figure 4.9, we see both benign and malicious clients mostly used the 266-bit and 528-bit key keys. The benign clients exclusively used the 1072-bit, 3088-bit, and 4112-bit keys. This exclusive selection of key length emphasizes that the client's public key has discriminatory power; there is a clear difference in the use of the key length.

Figure 4.9 Client key length

4.2.2 TLS Servers

In the previous section, we discussed the ciphersuite $0 \times 000a$, and now in Figure 4.10, we see benign servers only selected the ciphersuite. A small ratio of the benign flows selected the ciphersuite, but this is intriguing as malware-infected clients mostly offered it. Therefore, unexpected for benign servers to select it.

In Figure 4.10, we see that some ciphersuites were selected by benign servers more than malicious servers and vice-versa selected them. All the ciphersuites that were selected by both benign and malicious servers are still recommended for use.

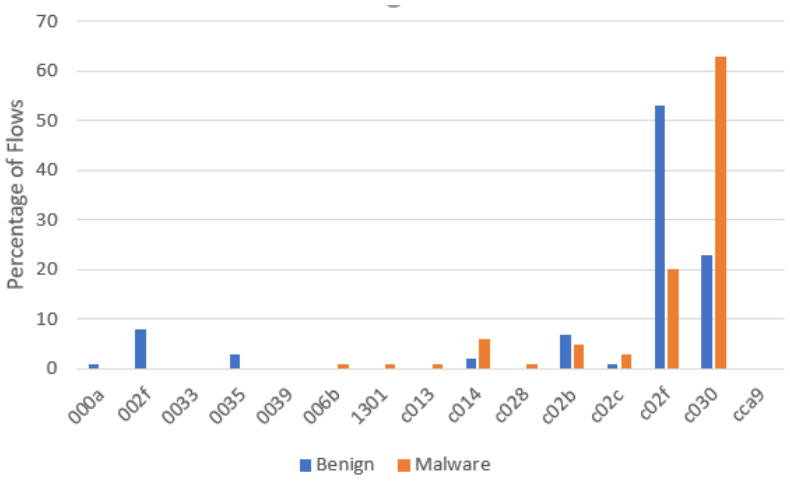

Figure 4.10 Selected ciphersuites

Both the benign and malicious servers were offered diverse extensions and they selected relatively the same extensions in different proportions with the exception of $0 \times 000a$ (supported groups) exclusively selected by benign servers.

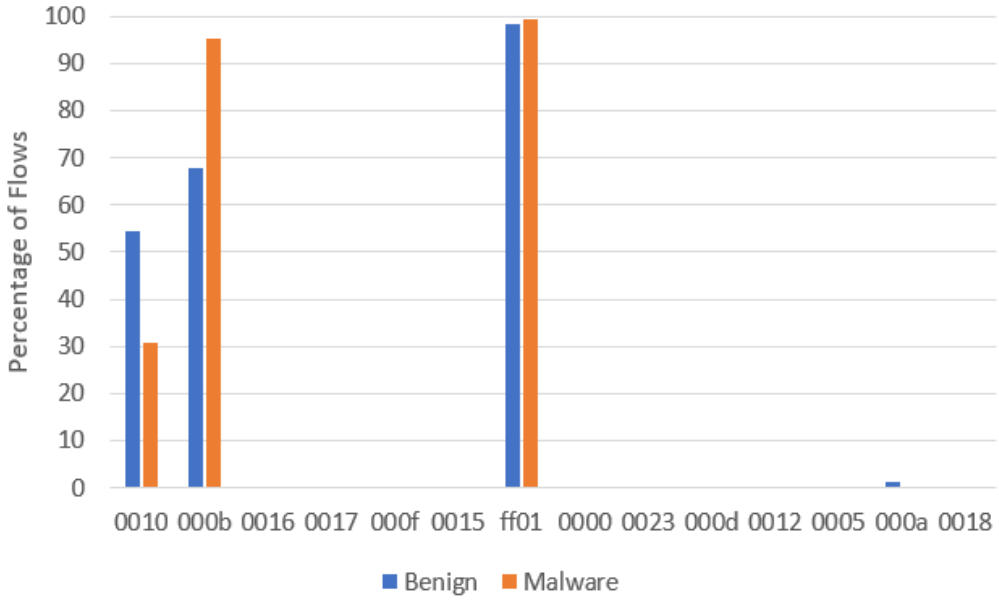

Figure 4.11 Selected extensions

4.3 Feature Selection

This section presents the selected features for our classification models and elaborates on the exclusion of some features based on the observed behaviour of our dataset.

Feature Set	Feature Name
SPLT	1. Sequence of packet lengths
	2. Sequence of packet inter-arrival times
Flow Metadata	3. Number of inbound bytes
	4. Number of outbound bytes
	5. Number of inbound packets
	6. Number of outbound packets
	7. Total duration of the flow in seconds.
Packet Payloads	8. Byte distribution
Unencrypted TLS Information	9. Offered ciphersuites
	10. Advertised TLS extensions
	11. Client's public key length
	12. TLS client
	13. Selected ciphersuites
	14. Selected TLS extensions
	15. Validity of certificate (In Days)
	16. Number of SAN entries

Table 4.1 Features from the Cisco paper [35]

Table 4.1 contains all the features in the Cisco paper [35], and our features will be a subset of these features.

There is a strong correlation between the selected (ciphersuites and extensions) and offered (ciphersuites and extensions) as the selected are the subset of the offered. In Section 4.2.2, we highlighted that all the selected ciphersuites are still recommended for use. Therefore, it is sufficient to only focus on offered ciphersuites and extensions.

The certificate features are also ignored as we only observed 2.2% certificate flows in the malware flows and 1.1% in the benign flows. On Wireshark the certificate data is not always in its packet, sometimes the data is in the server hello packet and we are assuming that Joy did not extract the certificate information from the server hello packets. We are excluding certificate features as we are concerned that they will introduce inconsistencies in

the classification. Table 4.2 contains the final selected features, the subset of the features in Table 1. These are the features that will be used on our supervised algorithms.

Feature Set	Feature Name	Type
SPLT	1. Sequence of packet lengths	Matrix
	2. Sequence of packet inter-arrival times	Matrix
Flow Metadata	3. Number of inbound bytes	Integer
	4. Number of outbound bytes	Integer
	5. Number of inbound packets	Integer
	6. Number of outbound packets	Integer
	7. Source port	Integer
	8. Destination port	Integer
	9. Total duration of the flow in seconds	Float
Packet Payloads	10. Byte distribution	Float vector
Unencrypted TLS Information	8. Offered ciphersuites	Binary vector
	11. Advertised TLS extensions	Binary vector
	12. Client's public key length	Integer

Table 4.2 Selected features

4.4 Summary

The features from the Cisco study [35] were used as a baseline for our data exploration to select the features for our machine learning models. The Cisco data showed that the malware uniquely uses HTTPS compared to benign, looking at ciphersuites, extensions, client key length, the validity of a certificate in days, byte distribution, and SPLT.

Our data also showed some similarities:

- Malware-infected clients offered more legacy or ciphersuites that should be avoided compared to benign clients.
- The ciphersuite $0 \times 000a$ was offered by both benign and malicious clients but primarily by malicious clients.
- The different client key lengths did not necessarily match the ones from the Cisco dataset, but there was a clear difference between the benign and malicious clients' key lengths.

Differences between our data and Cisco's:

- Only benign servers selected the $0 \times 000a$ ciphersuite, small ratio but worth mentioning.
- All the ciphersuites selected by both benign and malicious servers are still recommended for use.
- Malicious clients also offered diverse extensions even though it was not in the same proportion as the benign clients.

The selected features for our machine learning: sequence of packet lengths, sequence of packet inter-arrival times, number of inbound bytes, number of outbound bytes, number of inbound packets, number of outbound packets, source port, destination port, total duration of the flow in seconds, byte distribution, offered ciphersuites, advertised TLS extensions, and client's public key length.

Chapter 5

Testing and Results

This chapter outlines the implementation of classification of the flows from features pre-processing, model implementation, testing datasets, and testing metrics. This chapter also presents the results of both the supervised algorithms and their performances.

5.1 Preprocessing

The classifiers used in this project only take numeric attributes as inputs. Therefore, the raw data (text format) extracted from both the benign and malicious flows must be transformed to numeric values. The features are independent of each; therefore, their numeric values will be on different scales, and a massive difference in the scale can cause issues when the features are combined during modelling. To avoid this, we normalize the data by creating new values that maintain the general distribution and ratios in the source data while keeping values within a scale applied across all numeric columns used in the model [52].

5.1.1 Binary Vectors

The data labels: 1=benign and 0=malware. Figure 4.1 shows

```
create list of known ciphersuites
create a list of known extensions

for each flow:
    create a new empty array of zeros for ciphersuites
    create a new empty array of zeros for extensions
    for each offered ciphersuite:
        lookup the ciphersuites/extension in the respective array and get index of the ciphersuite/extension
        index ciphersuites/extensions zeros array and change the 0 to 1
```

Figure 5.1 Pseudo code to change datasets to binary values

5.1.2 Normalize Data

The normalization technique involves shifting and rescaling values to end up in the 0 to 1 range [22]. Features with high and wide ranges get more and consequently introduce bias in the classifier. The pseudocode in Figure 5.1 change datasets to binary values introduce bias in the classifier. Normalization aims to change the numeric values of features to a standard scale without distorting differences in the ranges of values.

5.2 Models

Two classification models, logistic regression and random forest were tested. A short description of these models is in Section 2.7. Python 3.8 [18] and the open-source machine-learning library scikit-learn[19] were used to implemented the classification models.

5.3 Testing Datasets

The 10-Fold cross-validation was performed to evaluate both the logistic regression and random forest model. Section 2.7.1 explains the k-Fold cross-validation process and why we selected $k = 10$. The data was shuffled before splitting it into the 10 folds, and this was done to ensure there is no bias introduced into the models during training.

5.4 Testing Metrics

To test the effectiveness of the models' classification, we have to test their accuracy. In this study, Negative (N) indicates benign flows, and Positive (P) indicates malware flows. The classifier can make a correct or an incorrect prediction. Each prediction can be one of the four outcomes in Figure 5.2, based on how it matches up to the actual value.

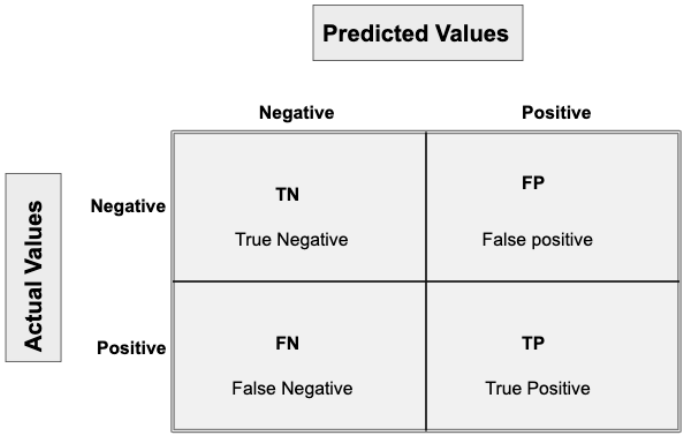

Figure 5.2 Confusion matrix with 2 class labels

- True Positive (TP): predicted it is malware and it is malware.
- True Negative (TN): predicted it is benign and it is benign.
- False Positive (FP): predicted it i malware but it is benign.
- False Negative (FN): predicted it is benign but it is malware.

The four metrics in Figure 5.3 are used to evaluate the models.

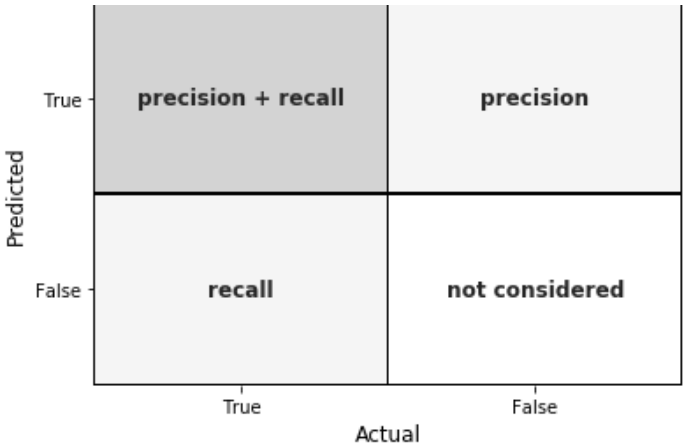

Figure 5.3 Confusion matrix with 2 class labels

- Accuracy: the number of correct predictions made as a ratio of all predictions made.
- Precision: the ratio how often is it correct $\frac{TP}{TP+FP}$
- Recall: the ratio of true positives to everything positive $\frac{TP}{TP+FN}$
- F_1 -score: is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. $F_1 = \frac{2}{\frac{1}{precision}+\frac{1}{recall}} = 2\frac{precision \times recall}{precision+recall}$

5.5 Results

As mentioned in Section 5.2, two different models were tested with features extracted from the benign and malware flows. We present the results of both the logistic regression and random forest models and compare the models' performance.

5.5.1 Model Parameters

Logistic Regression

The logistic regression classifier has three parameters: solver, penalty, and C. The solver is the algorithm used in the optimization problem, and in our case, we used the bilinear algorithm. During the model's training, it is important to avoid overfitting; avoid the model describing the random error in the data rather than the relationships between features. To counter overfitting, regularization is used by penalizing high-valued regression coefficients. The penalty is used to specify the norm used in the penalization, and we used L1. The parameter C is the inverse of regularization strength [51], and we used 1.

Random Forrest

In section 2.7, we explained that the random forest classifier uses a number (N) of decision trees, and selecting the right N for our experiment is a crucial step. Trees smoothen the average output of trees, which in turn increases the accuracy of the algorithm. However, a large number of trees is computationally expensive, so we must find the N that works best for our model.

Figure 5.5 plots the average accuracy of the random forest classifier 10-Fold cross-validation results for different values of N. Looking at the plot, 20 trees offer a good tradeoff between accuracy and computation time. Moreover, in the case the results are rounded up to the second decimal, all the values are 0.99.

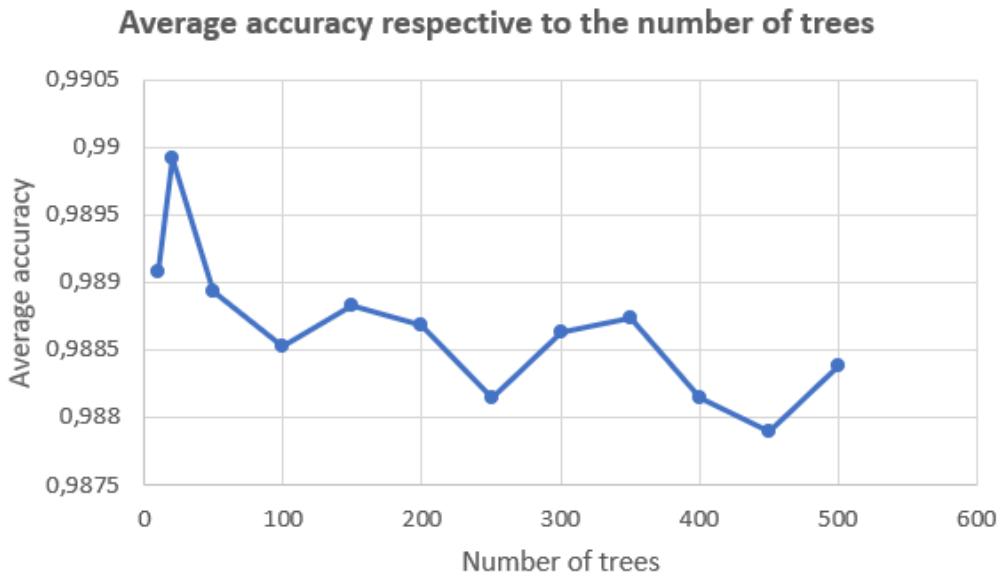

Figure 5.4 Average accuracy with respective number of trees(N)

5.6 Model Performance

5.6.1 Cross-Validation

The 10-fold cross-validation results for both models are shown in Table 5.1. The screenshots of all the fold results are in Appendix D.

Feature Combination	Logistic Regression	Random Forest
TLS	97.28%	97.54%
Meta + SPLT + BD	98.14%	98.52%
Meta + SPLT + BD + TLS	98.90%	98.92%

Table 5.1 10-Fold models Average Accuracy

Both the logistic regression and random forest were used in the Cisco research papers [33] [34] [35]. The results from these studies consistently show that the random forest performs better than the logistic regression. Looking at our results in table 5.1, the random forest classifier outperformed the logistic regression classifier as expected.

Feature Combination	Accuracy
TLS	98.2%
Meta + SPLT + BD	98.9%
Meta + SPLT + BD + TLS	99.6%

Table 5.2 Random forest Average Accuracy from Cisco [35]

Table 5.1 also shows that models' performances improve with more features and that the Meta + SPLT + BD + TLS feature combination yields the best results. Our results reconcile with the Cisco results in Table 5.2 concerning the feature combination.

5.6.2 Confusion Matrix

In Section 5.6.1, we established that the random forest is the better classifier for our classification problem. Therefore, we will only look at the effectiveness of only the random forest classifier using the testing metrics mentioned in Section 5.4. Table 5.3 shows the results of the 10-fold with the feature combination Meta + SPLT + BD + TLS.

Folds	No.Mal	No.Benign	TP	TN	FP	FN
1	639	1384	635	1367	17	4
2	643	1380	632	1368	12	11
3	689	1334	683	1327	7	6
4	639	1384	632	1367	17	7
5	677	1347	669	1336	11	8
6	662	1361	656	1350	11	6
7	649	1374	636	1349	25	13
8	616	1407	610	1389	18	6
9	648	1375	637	1366	9	11
10	648	1376	644	1361	15	4

Table 5.3 Testing Metrics

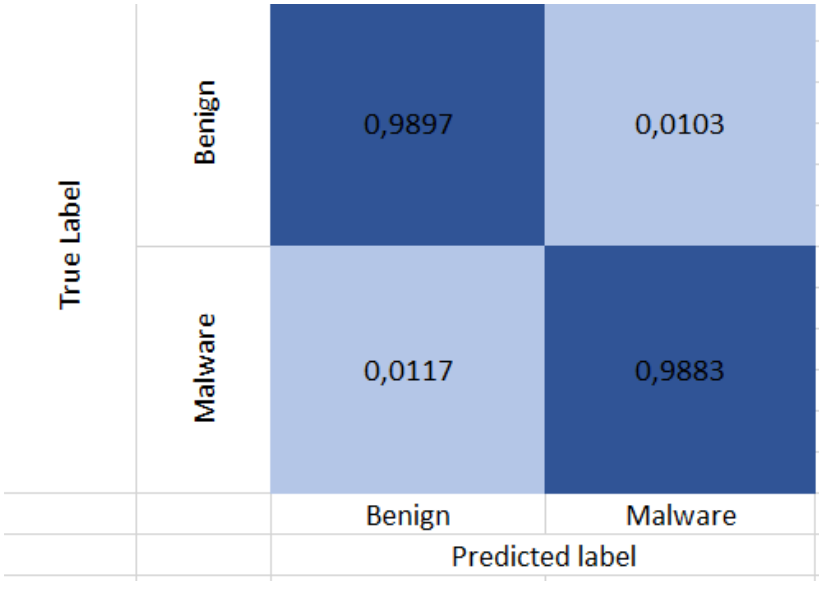

Figure 5.5 Random forest: confusion matrix with 2 class labels

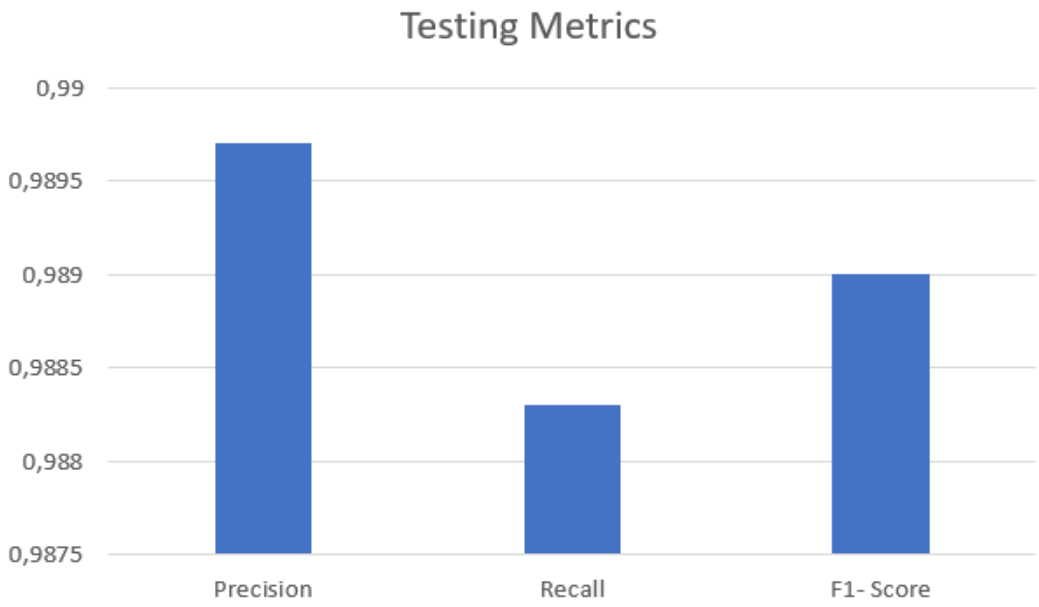

Figure 5.6 Random forest: Testing metrics

The results are impressive, with a near 100% accuracy, and almost all malware samples were found with great testing metrics. It is argued that these kinds of results are commonly inflated due to two pervasive sources of experimental bias: spatial bias caused by distributions of training and testing data that are not representative of a real-world deployment; and temporal bias caused by incorrect time splits of training and testing sets, leading to impossible configurations [32].

Malware evolves improving its capabilities and ability to evade detection, resulting in new variants. Users and systems generating benign traffic also change behaviours with progressive technology. Therefore, time has an impact on the models. The K-Fold cross-validation ignores the time-sensitivity and assumes that all flows are identically distributed in time. This assumption results in a positively biased classifier, as there is a high likelihood of correct classification than considering time-heterogeneous distributions.

5.7 Summary

Two classification models, logistic regression and random forest were tested and evaluated. The features for both models were extracted from flows, converted to numeric values, and normalized. The 10-Fold cross-validation was performed to evaluate both models.

Three feature combination used for testing; TLS, flow meta + TLS+ bytes Distribution and flow meta + TLS+ bytes distribution + SPLT. The results improved with the increased number of features for both models; therefore, the flow meta + TLS+ bytes distribution + SPLT combination yielded better results.The random forest model outperformed the logistic regression in all feature combinations.

Chapter 6

Discussion

6.1 Related Work

Detecting malicious encrypted traffic is a challenge for the security research community and the TLSI solution violates privacy, one of the security fundamentals. Several researchers have taken an interest in researching alternatives to TLSI. However, the most comprehensive study still comes from Cisco.

The set of features between our classifier and Cisco's differ as we excluded: total duration of the flow, TLS client, selected ciphersuites, selected extensions, the validity of the certificate, and the number of SAN entries in our experiment. Looking at Tables 5.1 and 5.2, Cisco got better results; even though the difference is in small decimals, it would matter in a network with millions of flows. It is not easy to point the exact contributor to the differences in experiments' results. It could be any of the factors: the size of the training dataset, the choice of features, the number of trees in the case of the random forest classifier, and classification based on malware family.

6.2 Limitations

The performance of a classifier depends on the relationship between the features and training and testing datasets. We wanted to use recent malware samples to see if there is a distinct change in the behaviour of TLS metadata, but we could not collect enough flows as our lab environment kept on crashing. Malware can be relevant for weeks or years depending on its activities and capabilities. Therefore, it is difficult to be confident about the period the current dataset will be useful for research. In an ideal experiment: every malware using TLS

is included, the dataset maintained with new variants, and regularly remove old malware so that the classifier can detect recent malicious flows.

Finding malware samples, setting up an undetectable sandbox environment, and capturing malicious flows are not easy and time-consuming. Benign flows should be easier to collect; however, it is challenging to simulate the dataset to represent different normal user behaviours. To add complexity, "normal" sites are sometimes hijacked and used for malicious activities.

The research done in detecting encrypted traffic is made public, which means attackers know the behaviour of the features we feed into the classifiers. The attackers could trick the classifier by changing the behaviour of the malware so that its communications would be almost indistinguishable from normal ones.

We used a combination of independent features to make the classification robust as there is no way of having a single parameter that is a sure sign of infection. However, we still cannot avoid false positives, which is a challenge for the analysts responsible for monitoring alerts as they have time constraints invalidating each false positive alert.

6.3 Possible Improvements

6.3.1 Data

- The malware samples we got from VirusTotal, and the captured files from malware-analysis.net did not have any tag whether or not the malware uses TLS for any of the communications. The security community needs to create a database of malware samples or captured files with TLS communication.
- To ensure that benign traffic is clean, we did a reputation lookup on VirusTotal and this step could be eliminated by getting the traffic from a few trusted and well-protected hosts.
- In our dataset, we did not see any malware samples that are use TLSv1.3 and we can not conclusively say malware samples are not using this version as this observation is subjective to our dataset. However, with the increase in adaptation of TLSv1.3, it would be interesting to see if it has any or no effect on the classification.

6.3.2 Classification

To improve the robustness of the classifier, we could include features from other types of protocols like DNS and HTTP headers. DNS is arguably one of the most critical systems as it facilitates the translation of the text-based URLs we type into search bars into the numerical IP addresses that computers use to communicate with each other. We could apply some of the features from EXPOSURE, a system that detects malicious domains by utilizing passive, large-scale DNS analysis techniques [36]. Figure 6.1 shows the 15 features used in EXPOSURE.

Feature Set	#	Feature Name
Time-Based Features	1	Short life
	2	Daily similarity
	3	Repeating patterns
	4	Access ratio
DNS Answer-Based Features	5	Number of distinct IP addresses
	6	Number of distinct countries
	7	Number of domains share the IP with
	8	Reverse DNS query results
TTL Value-Based Features	9	Average TTL
	10	Standard Deviation of TTL
	11	Number of distinct TTL values
	12	Number of TTL change
	13	Percentage usage of specific TTL ranges
Domain Name-Based Features	14	% of numerical characters
	15	% of the length of the LMS

Figure 6.1 Features from EXPOSURE [36]

In another Cisco paper, they used data features from TLS handshake metadata, DNS contextual flows linked to the encrypted flow, and the HTTP headers of HTTP contextual flows from the same source IP address within a 5 minute window [35]. Figure 6.2 shows the top 10 features from the paper.

Weight	Feature
3.38	DNS Suffix `org`
2.99	DNS TTL `3600`
2.62	TLS Ciphersuite `TLS_RSA_WITH_RC4_128_SHA`
2.28	HTTP Field `accept-encoding`
1.95	TLS Ciphersuite `SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA`
1.78	HTTP Field `location`
1.38	DNS `Alexa: None`
1.21	TLS Ciphersuite `TLS_RSA_WITH_RC4_128_MD5`
1.12	HTTP Server `nginx`
1.11	HTTP Code `404`

Figure 6.2 Features from multiple protocols [35]

Currently, we have a binary classification benign or malware; there is no attribution to malware families. It would also be helpful to have malware family classification as an output, which would improve tactics used by the security response teams.

Chapter 7

Conclusion

Our research aims to validate the feasibility and effectiveness of detecting malware HTTPS traffic without decryption as an alternative to TLSI and preserving privacy. The study was based on the assumption that malware authors configure their TLS servers and malware in a unique way, such that their HTTPS traffic has distinct features from normal traffic. The features were extracted from our datasets containing actual malware and benign traffic. The set of features consisted of TLS, flow metadata, SPLT, and byte distribution to describe the behaviour of HTTPS traffic. Two classifiers, logistic regression and random forest were trained and tested, and random forest performed better with an average accuracy of 98.92%.

The classifiers have limitations and could be reworked to reduce the number of false positives. However, even with the limitation, the approach discussed in this dissertation has an advantage over TLSI as it is not computationally expensive and does not violate users' privacy. We conclude that malware behavior over HTTPS is distinct from benign traffic and that these distinct characteristics can be used in creating meaningful features for machine learning models that achieve satisfactory classification.

References

[1] 10-Fold Cross-Validation. 2019. Available: https://www.openml.org/a/estimation-procedures/7. [Accessed online 5 January 2021].

[2] 339 Cipher Suites. 2020. Available: https://ciphersuite.info/cs/. [Accessed online 11 November 2020].

[3] Alexa: The Top 500 Sites On The Web. 2020. Available: http://www.alexa.com/topsites/. [Accessed online 13 May 2020].

[4] Analysis Of Trickbot Malware The Most Prolific Covid−19
Themed Malware. 2019. Available: https://lifars.com/2020/07/analysis-of-trickbot-malware-the-most-prolific-covid-19-themed-malware/.[Accessed online 10 October 2020].

[5] Cisco Joy Github. 2019. Available: https://github.com/cisco/joy. [Accessed online 17 October 2020].

[6] Cryptographic Key. 2019. Available: https://ldapwiki.com/wiki/Cryptographic%20Key. [Accessed online 20 April 2020].

[7] Detecting Encrypted Malware Traffic (Without Decryption). https://blogs.cisco.com/security/detecting-encrypted-malware-traffic-without-decryption?dtid=osscdc000283. [Accessed online 12 April 2020].

[8] Encrypted Traffic Analysis Will Be Mandatory Soon. 2020. Available: https://securityboulevard.com/2020/01/encrypted-traffic-analysis-will-be-mandatory-soon/. [Accessed online 12 April 2020].

[9] Evasions: Filesystem. 2018. Available:
https://evasions.checkpoint.com/techniques/filesystem.html.
[Accessed online 19 September 2020].

[10] Fakenet-Ng Github. 2020. https://github.com/fireeye/flare-fakenet-ng. [Accessed online 23 September 2020].

[11] February 2018 Zscaler Ssl Threat Report. 2018. Available:
https://www.zscaler.com/blogs/security-research/february-2018-zscaler-ssl-threat-report.
[Accessed online 12 March 2020].

[12] A Flaw In The Design: The Internet's Founders Saw Its Promise But Didn't Foresee Users Attacking One Another. 2020. Available:
https://www.washingtonpost.com/sf/business/2015/05/30/net-of-insecurity-part-1/?utm_term=.da4064f484ca. [Accessed online 12 March 2020].

[13] How To Use Ssl/Tls To Secure Your Communications: The Basics. 2020. Available: https://www.vircom.com/blog/how-to-use-ssltls-to-secure-your-communications-the-basics/. [Accessed online 12 March 2020].
[14] Malicious Encrypted Traffic Detection: Hitcon. 2018. Available: https://hitcon.org/2018/CMT/slide-files/d1_s1_r4.pdf. [Accessed online 15 March 2020].

[15] Malware capture facility project. 2019. Available: https://www.stratosphereips.org/datasets-normal. [Accessed online 13 February 2021].

[16] Malware Traffic Analysis. 2020. Available: https://www.malware-traffic-analysis.net. [Accessed online 17 January 2021].

[17] Mitre att&ck. 2020. Available: https://attack.mitre.org/. [Accessed online 13 April 2020].

[18] Python 3.8.0. 2021. Available: https://www.python.org/downloads/release/python-380/. [Accessed online 01 February 2021].

[19] Scikit-Learn Machine Learning In Python. 2020. Available: https://scikit-learn.org/stable/. [Accessed online 20 February 2021].

[20] Secure Digest Functions. 2020. Available: https://medium.com/@rathnaweeraatheesh72/secure-digest-functions-af852509c971. [Accessed online 15 August 2020].

[21] Ssl Introduction With Sample Transaction And Packet Exchange. 2019. Available: https:// www.cisco.com/c/en/us/support/docs/security-vpn/secure-socket-layer-ssl/116181-technote-product-00.html. [Accessed online 15 March 2020].

[22] Standardization In Machine Learning. 2021. Available: https://www.linkedin.com/pulse/standardization-machine-learning-sachin-vinay. [Accessed online 03 October 2020].

[23] Structure Of The Registry. 2021. Available: https://docs.microsoft.com/en-us/windows/win32/sysinfo/structure-of-the-registry. [Accessed online 13 June 2020].

[24] Web service security tutorial. Available: https://sites.google.com/site/ddmwsst/digital-certificates. [Accessed online 15 March 2020].

[25] What Is An Ssl/Tls X.509 Certificate? Available: https://www.venafi.com/blog/what-ssltls-x509-certificate. [Accessed online 11 August 2020].

[26] What Is Certificate Authority (Ca)? – Tips To Get Ssl Certificate From Certificate Authority. Available: https://aboutssl.org/certificate-authority. [Accessed online 18 February 2020]

[27] What's Hiding In Encrypted Traffic. 2020. https://www.zscaler.com/blogs/research/whats-hiding-encrypted-traffic-millions-advanced-threats. [Accessed online 12 March 2020].

[28] National Cybersecurity Protection Advancement Act of 2015; Congressional record. 2015. Available: https://www.congress.gov/congressional-record/2015/04/23/house-section/article/H2426-2. 161(60). [Accessed online 11 August 2020]

[29] ISO 27001: 2016 (EN) Information Technology—Security techniques—Information security management systems—Overview and vocabulary. 2016. Standard, International Organization for Standardization

[30] Managing risk from transport layer security inspection. 2019. Technical report, *National Security Agency*. 10

[31] I. Ali and N. Meghanathan. 2011. Virtual machines and networks-installation, performance study, advantages and virtualization options. *arXiv*. 1105.0061.

[32] K. Allix, T. Bissyande, J. Klein, and Y. Le Traon. 2015. Are your training datasets yet relevant?; 3.

[33] B. Anderson and D. McGrew. 2016. Identifying encrypted malware traffic with contextual flow data. *Proceedings of the 2016 ACM workshop on artificial intelligence and Security*. 1: 35–46.

[34] B. Anderson and D. McGrew. 2017. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*. 1: 1723–1732.

[35] B. Anderson, S. Paul, and D. McGrew. 2018. Deciphering malware's use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*. 14(3):195–211,

[36] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. 2011. Exposure: Finding malicious domains using passive dns analysis. *Ndss*. 1:1-17

[37] P. J. Denning. 1989. The science of computing: The internet worm. *American Scientist*, 77(2):126–128.

[38] W. Dormann. 2015 . The risks of ssl inspection. *Carnegie Mellon University CERT/CC Blog*. 1:3

[39] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxson. 2017. The security impact of https interception. *NDSS*. 2017.

[40] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. 2014. Flow monitoring explained: From packet capture to data analysis with netflow and

ipfix. *IEEE Communications Surveys & Tutorials*. 16(4):2037–2064.

[41] G. James, D. Witten, T. Hastie, and R. Tibshirani. 2013. An introduction to statistical Learning. *Springer*. 112.
[42] A. Khan. 2020. Key differences between tls 1.2 and tls 1.3. A10 Networks. [Accessed online 10 April 2020].

[43] M. Kuhn, K. Johnson, et al. 2013. Applied predictive modelling. *Springer*. 1: 26

[44] S. Kumar and S. Rai. 2012. Survey on transport layer protocols: Tcp & udp. *International Journal of Computer Applications*. 46(7):20–25.

[45] T. Liston. On the cutting edge: Thwarting virtual machine detection. 2006. Available: http://handlers.sans. org/tliston/ThwartingVMDetection_Liston_Skoudis. [Accessed online 20 August 2020]

[46] J. Lokoˇc, J. Kohout, P. ˇ Cech, T. Skopal, and T. Pevn`y. 2016. k-nn classification of malware in https traffic using the metric space approach. *Pacific-Asia Workshop on Intelligence and Security Informatics*. 1: 131–145.

[47] M. Mohri, A. Rostamizadeh, and A. Talwalkar. 2018. Foundations of machine learning. *MIT press.*

[48] P. B. Nath and M. M. Uddin. 2015. Tcp-ip model in data communication and networking. *American Journal of Engineering Research*. 4(10):102–107.

[49] J. Postel. 1980. Dod standard transmission control protocol. *ACM SIGCOMM Computer Communication Review*. 10(4):52–132.

[50] J. Postel and J. Reynolds. 1994. Internet official protocol standards. ISI, USC Information Sciences Institute.

[51] P. D. Prasad, H. N. Halahalli, J. P. John, and K. K. Majumdar. 2013. Single-trial eeg classification using logistic regression based on ensemble synchronization. *IEEE journal of biomedical and health informatics.* 18(3):1074–1080.

[52] M. H. Qasem and L. Nemer. 2018. Extreme learning machine for credit risk analysis. *Journal of Intelligent Systems*. 29(1):640–652.

[53] H. I. Rhys. 2020. Machine Learning with R, the tidyverse, and mlr. *Manning Publications*.

[54] B. Song. 2009. RFID authentication protocols using symmetric cryptography. PhD thesis. *Royal Holloway, University of London*.

[55] E. H. Spafford. 1989. The internet worm program: An analysis. SIGCOMM Comput. Commun. Rev., 19(1):17–57, Jan. 1989.

[56] F. Stˇrasak. 2017. Detekce malware v https komunikaci. B.S. thesis, ˇ Ceske vysoke uˇceni technicke v Praze. *Vypoˇcetni a informaˇcni centrum*.

[57] J. C. Villanueva. Symmetric vs asymmetric encryption. 2019. Available. https://www.jscape.com/blog/bid/84422/symmetric-vs-asymmetric-encryption. [Accessed online 10 April 2020]

Appendix A

VirtualBox Guest Addition Files

The list of VirtualBox guest addition files from [9]

C:\windows\System32\Drivers\VBoxMouse.sys
C:\windows\System32\Drivers\VBoxGuest.sys
C:\windows\System32\Drivers\VBoxSF.sys
C:\windows\System32\Drivers\VBoxVideo.sys
C:\windows\System32\vboxdisp.dll
C:\windows\System32\vboxhook.dll
C:\windows\System32\vboxmrxnp.dll
C:\windows\System32\vboxogl.dll
C:\windows\System32\vboxoglarrayspu.dll
C:\windows\System32\vboxoglcrutil.dll
C:\windows\System32\vboxoglerrorspu.dll
C:\windows\System32\vboxoglfeedbackspu.dll
C:\windows\System32\vboxoglpackspu.dll
C:\windows\System32\vboxoglpassthroughspu.dll
C:\windows\System32\vboxservice.exe
C:\windows\System32\vboxtray.exe
C:\windows\System32\VBoxControl.exe

Appendix B

Window TLS Ciphersuites

B.1 Windows 10

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_NULL_SHA256

TLS_RSA_WITH_NULL_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256;
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_DES_CBC_SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
TLS_RSA_WITH_NULL_MD5
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
TLS_PSK_WITH_AES_256_GCM_SHA384
TLS_PSK_WITH_AES_128_GCM_SHA256
TLS_PSK_WITH_AES_256_CBC_SHA384
TLS_PSK_WITH_AES_128_CBC_SHA256
TLS_PSK_WITH_NULL_SHA384
TLS_PSK_WITH_NULL_SHA256

B.2 Windows 7

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P384
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P384
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_NULL_SHA
SSL_CK_RC4_128_WITH_MD5
SSL_CK_DES_192_EDE3_CBC_WITH_MD5
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P521
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P521
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P521
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P521

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P521
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P521
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P521
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P521
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P521
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P521
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_WITH_NULL_MD5
TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
SSL_CK_DES_64_CBC_WITH_MD5
SSL_CK_RC4_128_EXPORT40_WITH_MD5

Appendix C

TLS Parameters Codes

C.1 Ciphersuites

Table C.1 lists all ciphersuites extracted from the training dataset, which represents 96 distinct ciphersuites. Their level of security evaluated on ciphersuite.info [2]

Table C.1 TLS Ciphersuites

Hex.	**Usage**	**Description**
0×c011	AVOID	TLS_ECDHE_RSA_WITH_RC4_128_SHA
0×002f	RECOMMENDED	TLS_RSA_WITH_AES_128_CBC_SHA
0×c014	RECOMMENDED	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
0×c00a	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
0×0035	RECOMMENDED	TLS_RSA_WITH_AES_256_CBC_SHA
0×c009	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
0×c013	RECOMMENDED	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
0×c02f	RECOMMENDED	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
0×c02c	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
0×c02b	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
0×c030	RECOMMENDED	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
0×0039	RECOMMENDED	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
0×0033	RECOMMENDED	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
0×000a	LEGACY	TLS_RSA_WITH_3DES_EDE_CBC_SHA
0×c024	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
0×c027	RECOMMENDED	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

0×c028	RECOMMENDED	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
0×009c	RECOMMENDED	TLS_RSA_WITH_AES_128_GCM_SHA256
0×003c	RECOMMENDED	TLS_RSA_WITH_AES_128_CBC_SHA256
0×003d	RECOMMENDED	TLS_RSA_WITH_AES_256_CBC_SHA256
0×009d	RECOMMENDED	TLS_RSA_WITH_AES_256_GCM_SHA384
0×c023	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
0×c0a1	RECOMMENDED	TLS_RSA_WITH_AES_256_CCM_8
0×c0ae	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
0×c09e	RECOMMENDED	TLS_DHE_RSA_WITH_AES_128_CCM
0×006b	RECOMMENDED	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
0×c0a3	RECOMMENDED	TLS_DHE_RSA_WITH_AES_256_CCM_8
0×c0a0	RECOMMENDED	TLS_RSA_WITH_AES_128_CCM_8
0×c09f	RECOMMENDED	TLS_DHE_RSA_WITH_AES_256_CCM
0×009f	RECOMMENDED	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
0×c0af	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
0×c0ad	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_256_CCM
0×009e	RECOMMENDED	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
0×c09d	RECOMMENDED	TLS_RSA_WITH_AES_256_CCM
0×c09c	RECOMMENDED	TLS_RSA_WITH_AES_128_CCM
0×c0ac	RECOMMENDED	TLS_ECDHE_ECDSA_WITH_AES_128_CCM
0×c0a2	RECOMMENDED	TLS_DHE_RSA_WITH_AES_128_CCM_8
0×0038	RECOMMENDED	TLS_DHE_DSS_WITH_AES_256_CBC_SHA
0×0032	RECOMMENDED	TLS_DHE_DSS_WITH_AES_128_CBC_SHA
4	AVOID	TLS_RSA_WITH_RC4_128_MD5
0×0013	LEGACY	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
5	AVOID	TLS_RSA_WITH_RC4_128_SHA
0×006a	RECOMMENDED	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
0×0040	RECOMMENDED	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
0×c012	LEGACY	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
0×c002	AVOID	TLS_ECDH_ECDSA_WITH_RC4_128_SHA
0×c00c	AVOID	TLS_ECDH_RSA_WITH_RC4_128_SHA
0×c007	AVOID	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
0×0010	LEGACY	TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA

85	LEGACY	TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA
0×c003	LEGACY	TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
0×c008	LEGACY	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
0×0069	LEGACY	TLS_DH_RSA_WITH_AES_256_CBC_SHA256
0×0016	LEGACY	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
0×0086	LEGACY	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA
0×c00d	LEGACY	TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
0×0031	LEGACY	TLS_DH_RSA_WITH_AES_128_CBC_SHA
0×00a5	LEGACY	TLS_DH_DSS_WITH_AES_256_GCM_SHA384
0×0043	LEGACY	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA
0×003f	LEGACY	TLS_DH_RSA_WITH_AES_128_CBC_SHA256
0×0042	LEGACY	TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA
0×0036	LEGACY	TLS_DH_DSS_WITH_AES_256_CBC_SHA
0×00a4	LEGACY	TLS_DH_DSS_WITH_AES_128_GCM_SHA256
0×000d	LEGACY	TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
0×003e	LEGACY	TLS_DH_DSS_WITH_AES_128_CBC_SHA256
0×0037	LEGACY	TLS_DH_RSA_WITH_AES_256_CBC_SHA
0×0007	LEGACY	TLS_RSA_WITH_IDEA_CBC_SHA
0×00a1	LEGACY	TLS_DH_RSA_WITH_AES_256_GCM_SHA384
0×0098	LEGACY	TLS_DH_RSA_WITH_SEED_CBC_SHA
0×0068	LEGACY	TLS_DH_DSS_WITH_AES_256_CBC_SHA256
0×00a0	LEGACY	TLS_DH_RSA_WITH_AES_128_GCM_SHA256
0×0030	LEGACY	TLS_DH_DSS_WITH_AES_128_CBC_SHA
0×00a3	RECOMMENDED	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
0×00a2	RECOMMENDED	TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
0×0099	RECOMMENDED	TLS_DHE_DSS_WITH_SEED_CBC_SHA
0×0087	RECOMMENDED	TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA
0×c00f	RECOMMENDED	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
0×0084	RECOMMENDED	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
0×c004	RECOMMENDED	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
0×c005	RECOMMENDED	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
0×c025	RECOMMENDED	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
0×c02e	RECOMMENDED	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384

88	RECOMMENDED	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
0×c032	RECOMMENDED	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
0×c00e	RECOMMENDED	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
0×c026	RECOMMENDED	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
0×c02a	RECOMMENDED	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
0×c031	RECOMMENDED	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
0×0041	RECOMMENDED	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
0×009a	RECOMMENDED	TLS_DHE_RSA_WITH_SEED_CBC_SHA
0×c029	RECOMMENDED	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
0×0045	RECOMMENDED	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
0×c02d	RECOMMENDED	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
0×0044	RECOMMENDED	TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA
0×0096	RECOMMENDED	TLS_RSA_WITH_SEED_CBC_SHA

C.2 Extensions

Table C.2 lists all extensions seen in the training dataset. It represents 14 out of 44. extensions.

Hex.	Dec.	Description
0×0000	0	server_name
0×0005	5	status_request
0×000a	10	supported_groups
0×000b	11	ec_point_formats
0×000d	13	signature_algorithms
0×000f	15	heartbeat
0×0010	16	application_layer_protocol_negotiation
0×0012	18	signed_certificate_timestamp
0×0015	21	padding
0×0016	22	encrypt_then_mac
0×0017	23	extended_master_secret
0×0018	24	token_binding
0×0023	35	session_ticket
0×ff01	65281	renegotiation_info

Table C.2 TLS extensions

Appendix D

Logistic Regression Results

```
Num Positive:   6510
Num Negative:   13722
[4]
Features Used:
        TLS Information         (198)
Total Features: 198
acc_cv
0.9757785467128027
correct_cv
1974
acc_cv
0.9728126544735541
correct_cv
1968
acc_cv
0.9663865546218487
correct_cv
1955
acc_cv
0.9767671774592189
correct_cv
1976
acc_cv
0.9654150197628458
correct_cv
1954
acc_cv
0.9708353929807217
correct_cv
1964
acc_cv
0.9742956005931784
correct_cv
1971
acc_cv
0.9802273850716757
correct_cv
1983
acc_cv
0.9713297083539298
correct_cv
1965
acc_cv
0.974802371541502
correct_cv
1973
```

Figure D.1 TLS Feature

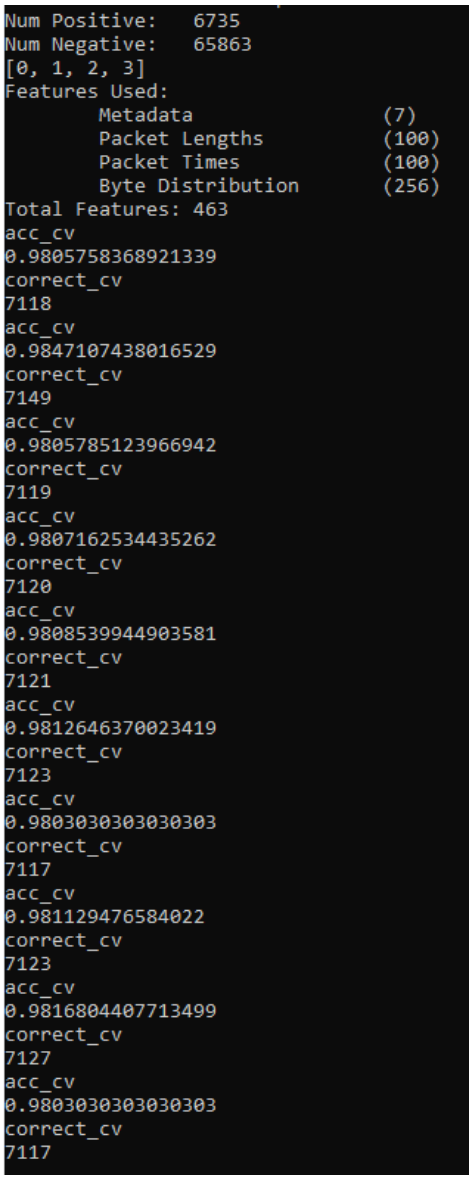

Figure D.2 Flow Meta, SPLT and Bytes Distribution Features

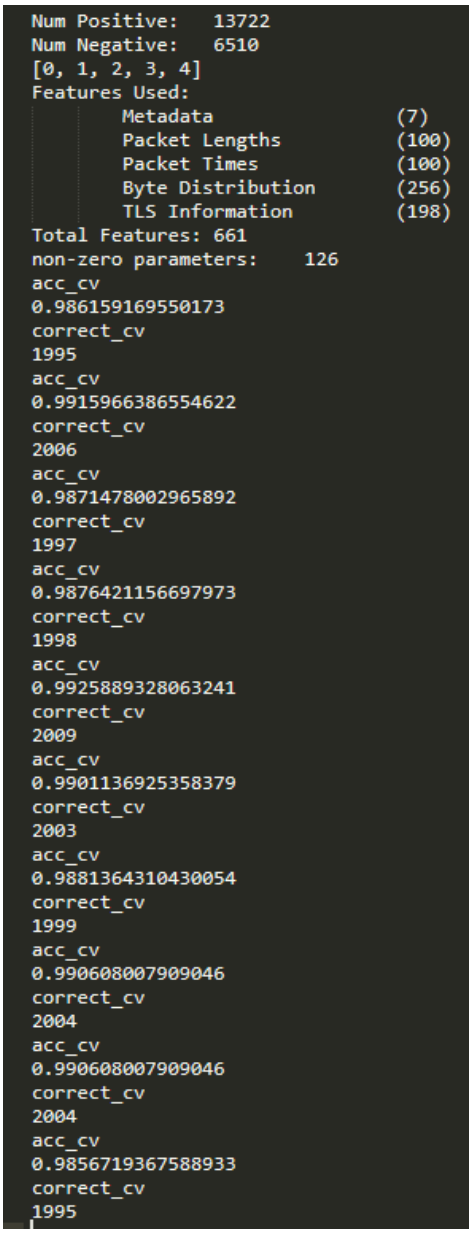

Figure D.3 TLS, Flow Meta, SPLT and Bytes Distribution Features

Appendix E

Random Forest Results

```
Num Positive:   6510
Num Negative:   13722
[4]
Features Used:
        TLS Information         (198)
Total Features: 198
acc_cv
0.972318339100346
correct_cv
1967
acc_cv
0.9698467622343054
correct_cv
1962
acc_cv
0.9772614928324271
correct_cv
1977
acc_cv
0.9772614928324271
correct_cv
1977
acc_cv
0.9688735177865613
correct_cv
1961
acc_cv
0.9703410776075136
correct_cv
1963
acc_cv
0.9747899159663865
correct_cv
1972
acc_cv
0.967869500741473
correct_cv
1958
acc_cv
0.9767671774592189
correct_cv
1976
acc_cv
0.974308300395257
correct_cv
1972
```

Figure E.1 TLS Feature

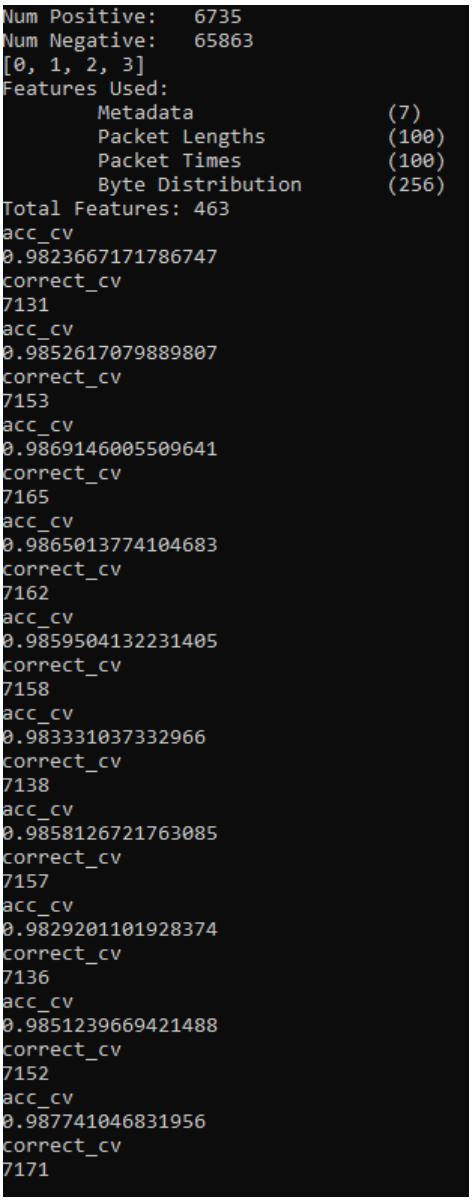

Figure E.2 Flow Meta, SPLT and Bytes Distribution Features

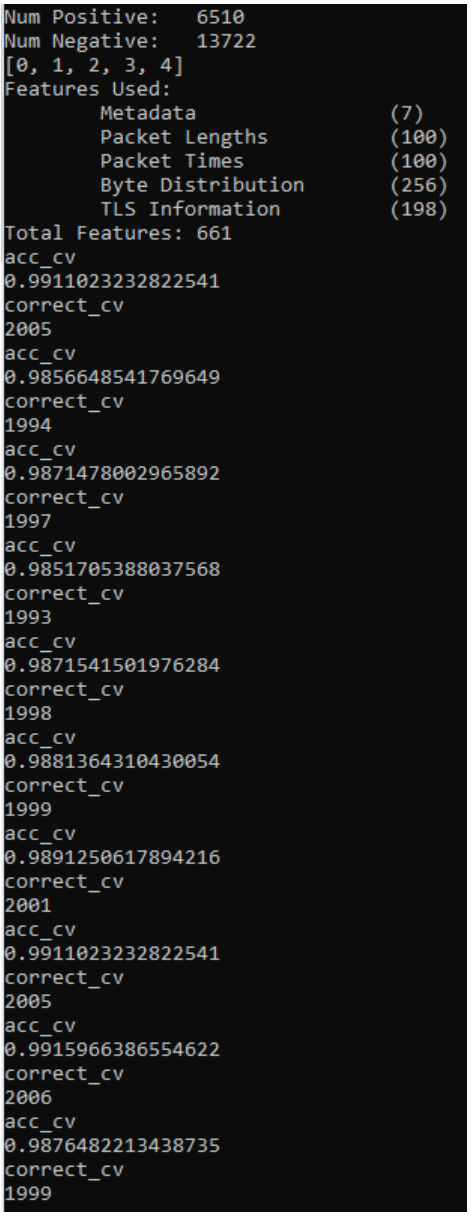

Figure E.3 TLS, Flow Meta, SPLT and Bytes Distribution Features