JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY
http://e-journal.uum.edu.my/index.php/jict

How to cite this article:
Keshkeh, K., Jantan, A., & Alieyan, K. (2022). A machine learning classification approach to detect TLS-based malware using entropy-based flow set features. *Journal of Information and Communication Technology, 21*(3), 279-313. https://doi.org/10.32890/jict2022.21.3.1

A Machine Learning Classification Approach to Detect TLS-based Malware using Entropy-based Flow Set Features

*[1]Kinan Keshkeh, [2]Aman Jantan & [3]Kamal Alieyan
[1&2]School of Computer Science, Universiti Sains Malaysia, Malaysia
[3]Faculty of Computer Sciences and Informatics,
Amman Arab University, Jordan

*kinan.keshkeh@student.usm.my,
aman@usm.my,
k.alieyan@aau.edu.jo
*Corresponding author

Received: 30/1/2022 Revised: 13/3/2022 Accepted: 20/3/2022 Published: 17/7/2022

ABSTRACT

Transport Layer Security (TLS) based malware is one of the most hazardous malware types, as it relies on encryption to conceal connections. Due to the complexity of TLS traffic decryption, several anomaly-based detection studies have been conducted to detect TLS-based malware using different features and machine learning (ML) algorithms. However, most of these studies utilized flow features with no feature transformation or relied on inefficient flow feature transformations like frequency-based periodicity analysis and outlier percentage. This paper introduces TLSMalDetect, a TLS-based malware detection approach that integrates periodicity-independent

entropy-based flow set (EFS) features generated by a flow feature transformation technique to solve flow feature utilization issues in related research. The effectiveness of EFS features was evaluated in two ways: (1) by comparing them to the corresponding outlier percentage and flow features using four feature importance methods, and (2) by analyzing classification performance with and without EFS features. Moreover, new Transmission Control Protocol features not explored in the literature were incorporated into TLSMalDetect, and their contribution was assessed. This study's results proved that EFS features of the number of packets sent and received were superior to the related outlier percentage and flow features and could remarkably increase the performance up to ~42 percent in the case of Support Vector Machine accuracy. Furthermore, using the basic features, TLSMalDetect achieved the highest accuracy of 93.69 percent by Naïve Bayes (NB) among the ML algorithms applied. From a comparison view, TLSMalDetect's Random Forest precision of 98.99 percent and NB recall of 92.91 percent exceeded the best relevant findings of previous studies. These comparative results demonstrated TLSMalDetect's ability to detect more malware flows out of total malicious flows than existing works. It could also generate more actual alerts from overall alerts than earlier research.

Keywords: Malware detection, machine learning, TLS, entropy, flow features.

INTRODUCTION

With the growing usage of encryption to ensure users' privacy, malware authors opt to embrace encryption protocols such as Transport Layer Security (TLS) and communication over Hypertext Transfer Protocol Secure (HTTPS) connections to hide malicious connections. According to Sophos Labs, roughly 44 percent (almost half) of information-stealing malware and 23 percent of all malware categories employ TLS while transmitting or receiving orders from Command and Control (C&C) servers, installing harmful payloads, or accessing data provided by that payload (Nagy, 2020).

An Intrusion Detection System (IDS) is a network security system that scans for unusual behavior and warns users of malware. It is

installed either on the host side (Host Intrusion Detection System, HIDS) or the network (Network Intrusion Detection System, NIDS). An IDS may also be classified into two types: signature-based, which utilizes databases with known attack patterns, and anomaly-based, which identifies unexpected abnormal behavior by comparing it to the network's regular activity. However, signature-based NIDS cannot detect TLS-based malware due to the unavailability of clear-text metadata and the complexity of traffic decryption. As a result, NIDS research has been shifted to anomaly-based detection that does not require decryption.

In the research area of TLS-based malware anomaly detection in a network, several feature types have been used, such as flow (e.g., the number of packets or bytes, duration), packet, TLS handshake, and contextual Hypertext Transfer Protocol (HTTP) and Domain Name System (DNS). The features are extracted from traffic, pre-processed, and passed into detection techniques such as machine learning (ML). Nevertheless, involving certain flow features, such as the number of packets sent/received or duration, could be inefficient in malware anomaly behavior classification (Anderson et al., 2018; Jenseg, 2019; Liu et al., 2019; Maroušek, 2017; Roques et al., 2019). The explanation for this is that the range of feature values in the malicious traffic could vary over time (in the same malware capture or different malware captures), making it impossible to predict or classify. For instance, supposing the number of packets sent of several flows are: {1, 1, 7, 7, 10, 10, 10, 1, 4, 11, 11, 20, 20}, the abnormal behavior (regularity) is seen; however, the value ranges vary, making classification prediction unfeasible. With this issue, the classification model learns less important or uninformative (irrelevant) flow features and takes longer to train.

Feature engineering addresses this challenge with domain knowledge. One of the feature engineering processes is feature transformation, by which new features are evolved from existing ones through arithmetic and aggregation operations. Using feature transformation on flow features in sets (groups) can provide additional features capable of exposing anomalous behavior. Nevertheless, only a few feature transformation methods have been used in related works to tackle the range change of flow feature value with various drawbacks.

One method is to apply frequency-based analysis like Fourier (converting from a time domain to a frequency domain) to discover

the periodicity of a flow set feature (Fehrman et al., 2020). The higher the frequency, the more likely the malware exists. Although this method often works, malware can show a non-periodically continuous anomaly pattern, rendering ML detection ineffective based on frequency.

Another method uses statistics to distinguish outliers in feature values, avoiding the limitations of periodicity assumption. By calculating standard deviation and mean, the percentage of all values out of range of *mean+ standard deviation* and *mean – standard deviation* (outliers) can be discovered (Dai et al., 2019; Strasák, 2017). The differences in percentages can distinguish the traffic, whether it is malicious or benign. Nevertheless, this methodology has a disadvantage because malware can have the same outlier percentage as normal traffic but with different anomaly value distributions. Furthermore, outlier values substantially impact the mean and standard deviation (very small/large values can reduce/increase the standard deviation), and outliers are uncommon in small samples.

This paper aims to address the above-highlighted related works' weaknesses and boost the detection performance by proposing TLSMalDetect, a TLS-based malware detection approach based on ML classification. TLSMalDetect uses an entropy-based flow set feature transformation technique to generate periodicity-independent features named entropy-based flow set (EFS) features. EFS features' efficiency is assessed by examining their effect on classification performance and comparing them to the related works' corresponding outlier-based and flow features using different feature importance methods. The study also evaluates new Transmission Control Protocol (TCP) features not explored in the literature based on feature importance. Moreover, it tests TLSMalDetect's detection performance using seven ML classification algorithms and selects the best accuracy achieving algorithm. Finally, TLSMalDetect performance is compared to two similar studies based on the best precision and recall.

The rest of the paper is organized as follows. The second section presents a review of the related works. The proposed TLSMalDetect approach is described in the third section. The fourth section presents the results and discussion, followed by the final section that concludes the research and offers future work recommendations.

RELATED WORKS

Since TLS-based malware detection based on signature violates users' privacy and requires expensive computation, NIDS researchers have focused on anomaly-based detection using ML. As a result, various features such as TLS, TCP, contextual DNS and HTTP, flow, and packet features have been extracted from the traffic to create reliable detection systems.

TLS is a cryptographic protocol that offers privacy and integrity between a client and a server and is mainly used with HTTP. Before a TLS connection is formed between two parties, handshake messages are conveyed carrying clear-text metadata from which features can be retrieved. Client hello, a client-side TLS handshake message, was used in the early detection because it is the first message sent (Liu et al., 2019). It also contains numerous metadata fields like ciphersuites, extensions, compression techniques, and client version. Most of the literature have utilized ciphersuites as they are a good differentiator between malware and benign traffic (Anderson et al., 2018; Anderson & McGrew, 2016; 2017; Calderon et al., 2018; Liu et al., 2019; Roques et al., 2019; Senecal et al., 2019). Client hello extensions are also essential, like Server Name Indication (SNI) extension. Using the CTU-13 Stratosphere dataset, Bazuhair and Lee (2020) examined the SNI value if it is an Internet Protocol (IP) address and included that as a feature in their approach based on Perlin noise with Convolutional Neural Network (CNN). They also employed Naïve Bayes (NB) in the performance result comparison. Nevertheless, other studies focused on TLS handshake server messages, especially certificate features (Kato et al., 2019; Torroledo et al., 2018). Kato et al. (2019) conducted a study on android TLS malware detection, achieving Random Forest (RF) accuracy of 93.90 percent by merging the TLS certificate features of the new scheme with the old scheme and simple DP-based scheme (SDPBS). As demonstrated, TLS features are essential in the detection; therefore, they are utilized with other features in this paper's approach, TLSMalDetect.

Besides TLS, TLS-based malware also uses firewall port-opened application protocols such as HTTP and DNS to assist in the attacks, like Qbot malware (*Malware-Traffic-Analysis.Net - Qbot (Qakbot) Infection*, 2020). Different contextual HTTP and DNS features are

used in two types: statistical-based like the number of IPs in a DNS request, or string-based such as HTTP content-type (Anderson & McGrew, 2016; Calderon et al., 2018; Senecal et al., 2019).

In addition, researchers have used other Transmission Control Protocol/Internet Protocol (TCP/IP) stack protocols that function under the TLS protocol layer, such as TCP and IP. For example, Anderson et al. (2019) integrated TCP packet length and TCP Push flag features in a framework for HTTPS/TLS malware detection that applies two ML classifiers. Strasák (2017) divided 13 TCP connection states into two groups: established and not established, and then calculated the "Ratio of Established States of Connection" feature. Furthermore, Anderson and McGrew (2017) used packet features in two feature sets: Standard, which only has packet features, and Enhanced, which has packet features and TLS features. Among all other RF accuracies in the literature, they attained the highest accuracy rate of 99.99 percent using the Enhanced set. Packet features were further processed using the Markov chain (binning) to provide additional vector features known as the sequence of packet lengths/times (SPL/T) (Jenseg, 2019; Zheng et al., 2020).

Moreover, a majority of the existing works have included network flow features in their approach feature sets. Table 1 shows several flow features used and ways for extracting them. Some researchers used flow features without grouping flows into sets. For example, Anderson et al. (2018) extracted flow features (metadata) using the software they wrote. Then, by applying l1-logistic regression, the researchers compared the findings of three feature type groups, including flow features, and obtained an excellent overall accuracy of 99.6 percent. In contrast, other researchers grouped flows into sets and utilized flow set features. For instance, Dai et al. (2019) and Strasák (2017) used the flow set (or connection, as they called it) features, such as the total of the duration, number of packets, and bytes.

However, using flow or flow set features is not always effective in ML classification since the range of feature values might vary over time and become unclassified. As a result, feature transformation of flow features within a set is required to reveal the set's malicious behavior. Table 1 also summarizes the flow feature transformation methods employed in the research works and the consequent weaknesses.

Table 1

A Comparison of Flow Feature Usage in Related Works

Study	Number of flow features	Duration	In/Out bytes	In/Out packets number	Inter-Arrival Time (IAT)	Src/Dst ports	Starting time of flow	Includes features of the whole flow set?	Flow feature transformation applied	Weaknesses
Bazuhair and Lee (2020)	5	✓	✓	✓	✓	-	-	Yes	Nil	Flow feature values may not be classifiable
Dai et al. (2019)	6	✓	✓	✓	✓	-	-	Yes	Outlier percentage	Normal and malware outlier percentages can be the same
Strasák (2017)	6	✓	✓	✓	✓	-	-	Yes	Outlier percentage	Normal and malware outlier percentages can be the same
Anderson et al. (2018)	4	✓	✓	✓	-	✓	-	No	Nil	Flow feature values may not be classifiable

(continued)

Study	Number of flow features	Duration	In/Out bytes	In/Out packets number	Inter-Arrival Time (IAT)	Src/Dst ports	Starting time of flow	Includes features of the whole flow set?	Flow feature transformation applied	Weaknesses
Liu et al. (2019)	3	✓	✓	✓	-	-	-	No	Nil	Flow feature values may not be classifiable
Anderson et al. (2019)	2	-	✓	✓	-	-	-	No	Nil	Flow feature values may not be classifiable
Fehrman et al. (2020)	3	-	-	-	✓	-	✓	No	Fourier Analysis (periodicity)	It does not work if the malware flow feature is not periodic
Roques et al. (2019)	5	✓	✓	✓	-	✓	-	No	Nil	Flow feature values may not be classifiable
Jenseg (2019)	3	✓	✓	✓	-	-	-	No	Nil	Flow feature values may not be classifiable
Maroušek (2017)	3	✓	✓	-	✓	-	-	No	Nil	Flow feature values may not be classifiable

Some existing research works applied two flow feature transformation methods: Fourier frequency analysis and outlier percentage. Fehrman et al. (2020) applied the first method in their system, which did not rely on ML. They detected TLS-based malware beacons by getting the frequency using Fourier transformation on flow set features and calculating the number of server TLS certificates. The higher the frequency, the more likely the malware exists, and the fewer servers, the less chance of malicious connection existence. Nevertheless, the research is limited since TLS-based malware may not exhibit periodic anomaly behavior. Dai et al. (2019) proposed a detection method based on multi-view features, including flow statistics, and compared it with Strasák's (2017) work based on feature engineering. As in Strasák's (2017) study, Dai et al. (2019) calculated the percentage of all values out of the Mean+/-StandardDeviation; the normal traffic usually has a higher percentage than the malware one. The percentage is then used as a new feature in ML. Unfortunately, this approach is limited as the malware distribution of feature values may imitate the benign one, and outlier values substantially impact the mean and standard deviation.

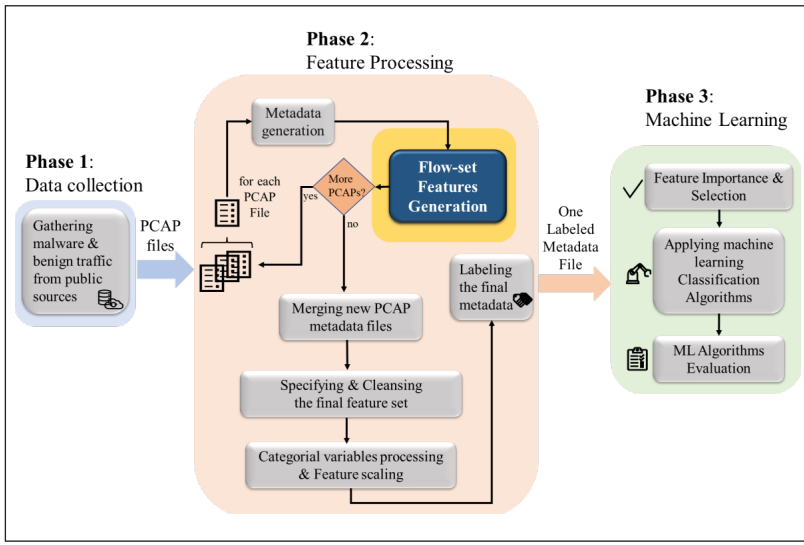
In summary, different features were utilized in the related works of TLS-based malware anomaly-based detection. These features were extracted from TLS, contextual DNS and HTTP, TCP, packet, and flow. However, as seen in Table 1, most related works had drawbacks in using flow features with no feature transformation or using inefficient transformation methods such as outlier-based and frequency analysis features. Besides, only one related study by Bazuhair and Lee (2020) applied NB for ML results comparison using features taken from the CTU-13 dataset. Therefore, this paper proposes a TLSMalDetect approach to address the previous weaknesses by using periodicity-independent EFS features created from a flow feature transformation technique. Furthermore, the approach includes NB as one of several algorithms to test its performance.

THE PROPOSED TLSMALDETECT APPROACH

This section explores the proposed TLSMalDetect approach, which efficiently detects TLS-based malware by adding EFS features. TLSMalDetect has three phases: 1) data collection; 2) feature processing; and 3) machine learning. Figure 1 depicts the overall TLSMalDetect structure.

Figure 1

The Proposed TLSMalDetect Approach



Phase 1: Data Collection

Traffic captures from public sources were collected in packet capture (PCAP) format. Both malicious and benign traffic types were needed to train the ML model in Phase 3.

Phase 2: Feature Processing

In this phase, the PCAP files were processed and prepared for the input of Phase 3. The sub-phases are explained as follows:

1) Metadata Generation

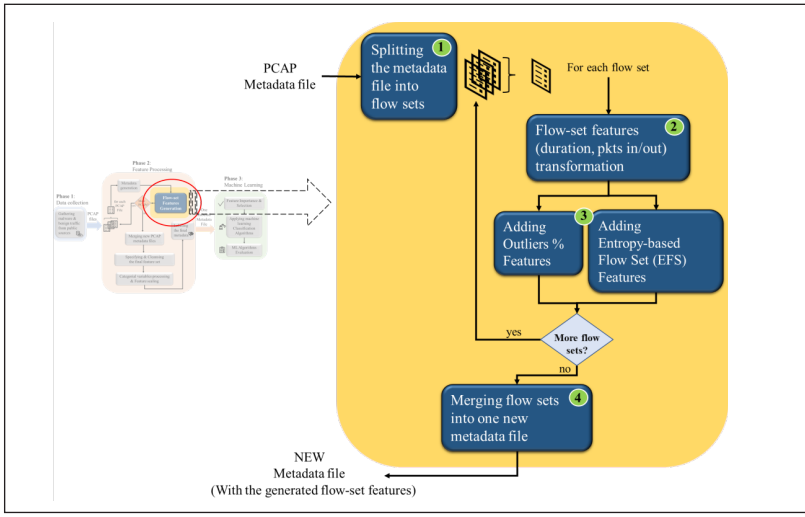
PCAP files contain raw traffic data that must be processed further to obtain useful features. Therefore, each PCAP file was parsed into a metadata file using the Tranalyzer open-source software (*Tranalyzer - About*, n.d.). The metadata file comprised many unidirectional flows; each had several features, like packets and bytes, TCP stream, TLS handshake, etc.

2) Flow set Feature Generation

Each metadata file was read in this sub-phase, and a new one with additional flow set features was created through four steps, as illustrated in Figure 2. First, each metadata file was split into multiple sets. According to Albright (2021) from Databox, the average user session duration is 2–3 minutes. After examining many traffic captures, the study found that 300 flows usually form at least a 3-minute session. Therefore, a flow set was chosen to have 300 unidirectional flows. If a PCAP had less than 300 flows, all flows were taken. It is worth emphasizing that flows were divided based on count rather than destination IP since malware may connect with several server IPs, causing anomalous flow behavior through all of them.

Figure 2

Flow Set Features Generation Steps



Second, the duration and the number of packets sent and received for each flow set were transformed, generating new additional features. The feature transformation methods applied are:

(i) Entropy

As malware is an automated program, it usually exhibits regularity for some features, while benign software shows randomness. This behavior difference was captured using the entropy in Equation 1 given.

$$Entropy\ (C\ ) = -\sum_{i=1}^{n} p(c_i) \log\ (\ p(c_i)\ ) \qquad (1)$$

Where C = { $c_1, c_2, \ldots c_i$} and c_i represents the duration or number of packets sent/received for each flow in a flow set, n is the number of the unique values, and $p(c_i)$ indicates the probability of each value c_i. For the duration, the numbers were rounded to have two digits after the decimal point.

Applying entropy has several advantages: (1) It is periodicity independent; as the equation indicates, it just needs a probability of each value. Therefore, it can differentiate the anomaly behavior (regularity) of malware values even if they are nonperiodic. (2) It is independent of the distribution of values, meaning that it can discover malicious regularity even though both malware and benign have the same value distribution (e.g., Gaussian) with the same number of outliers (outside Mean+/-Standard Deviation).

(ii) Outliers

This existing work method was used to be compared with entropy features. It works by calculating the proportion of values outside the range (Mean+/-Standard Deviation).

Third, the resulting entropy values (EFS) and the outlier percentages were added as six extra flow set features to each flow to indicate the behavior of the contained flow set. Finally, Steps 2 and 3 were repeated for each flow set. When there were no more flow sets to take, the flow sets were merged in one new metadata file, including the extra EFS and outlier percentage features, and then the file was stored.

3) Merging New PCAP Metadata Files

The previous Sub-phases (1) and (2) were repeated for each PCAP file. Then, all PCAP metadata files produced were merged in sequence in one master metadata file.

4) Specifying the Feature Set and Cleansing Data

The master metadata file contained many irrelevant features outside the scope of this study. These features were removed in this sub-

phase, and other features were kept, which are listed in Table 2. The literature's most common TLS, TCP, and flow features were selected besides the created EFS and outlier percentage features. Additionally, TCP features that have not been utilized in related studies were included.

After that, flows with missing feature values were removed to improve classification accuracy while minimizing the cost of dealing with a large metadata file. Following malware and benign flow filtration, the malware flows reduced to 328 flows. The remaining 328 flows were added from another malware capture of Trickbot to balance the two classes and reduce the ML prediction bias. The final number of flows for each class (malware and benign) was 139,560.

Table 2

The Selected Feature Sets

Feature Type	Data Type	Feature
Flow	Numeric	The number of input packets and output packets
Flow	Numeric	The flow duration (seconds)
Flow set	Numeric	The entropy of the number of input packets and output packets (EFS)
Flow set	Numeric	The entropy of flow duration (EFS)
Flow set	Numeric	The outlier percentage of the number of input packets and output packets
Flow set	Numeric	The outlier percentage of flow duration
Flow	Numeric	The number of input bytes and output bytes
Flow	Numeric	The min, max, average, and standard deviation of layer 3 packet size
Flow	Numeric	The min, max, average, and standard deviation of layer 3 packet IAT
Flow	Numeric	Sent packets/bytes per second
TCP	Numeric	TCP packet sequence count and ACK count
TCP	Numeric	TCP sent sequence bytes
TCP	Numeric	The fault number of TCP sequence and ACK
TCP	Numeric	TCP flawless ACK received bytes (*new*)

(continued)

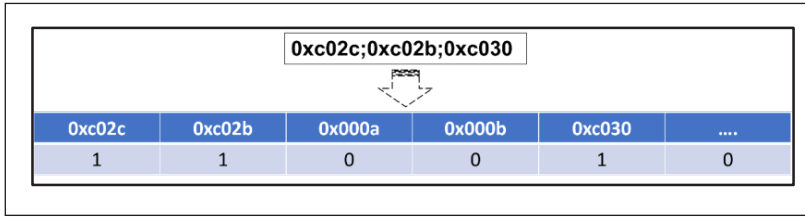
Feature Type	Data Type	Feature
TCP	Numeric	TCP effective window size change down count and up count (*new*)
TCP	Numeric	TCP effective window size direction change count (*new*)
TCP	Numeric	TCP packet count ratio below window size WINMIN threshold (*new*)
TCP	Value List	TCP flags (FIN/SYN/PSH/ACK/URG/ECN-Echo/CWR)
TCP	Value List	TCP Anomaly states (*new*) (FIN-ACK/ SYN-ACK/ RST-ACK/ SYN-FIN/SYN-FIN-RST/ FIN-RST/Null/XMas flags, L4 option field corrupt, SYN retransmission, Sequence Number retry/out of order/jump forward, ACK number out of order, Duplicate ACK)
TCP	Numeric	The number of TCP option packets in a flow (*new*)
TCP	Numeric	The count of TCP options a flow has (*new*)
TLS	Numeric	SSL (TLS) version
TLS	Numeric	The number of client extensions, EC points, and EC point formats
TLS	Value List	The list of client extensions and client EC points
TLS	Numeric	The number of protocols
TLS	Numeric	The number of supported client ciphers
TLS	Value List	The list of supported client cipher
TLS	Value List	The number of change_cipher, alert, handshake, application data, and heartbeat records

5) Categorical Variables Processing and Feature Scaling

Features that had a list of values, like the client extensions, were split by the semicolon ";" or "_", and then were binary encoded in new columns (features), as shown in Figure 3. Furthermore, SSL version feature values were encoded by the one-hot-encoding method. The total number of features was 272. Finally, all feature values were scaled (normalized) into a fixed range between 0 and 1.

Figure 3

Binary Encoding of Value List Features



6) Labeling the Final Data

Flows were tagged or labeled with either malware (number 1) or normal flow (number 0). This procedure was essential for the ML phase.

Phase 3: Machine Learning

In this phase, the total number of features (272) of the labeled master metadata file was reduced to 136 due to the experiment's resource limitations. The 136 basic features were the top features obtained using RF feature selection with 10-fold cross-validation (CV) as the RF algorithm has shown promising results in the literature. Moreover, for a proper result comparison with Jenseg's (2019) study, the same number of features used in that study, the top 23, were identified using the coefficient-based method, i.e., LR regularization L1.

After that, each feature of the 136 basic features was assigned importance values using four methods: 1) Mutual Information (MI) - Filter method; 2) Logistic Regression (LR) regularization with penalty L1 - Embedded method; 3) RF - Embedded method; and 4) XGBoost - Embedded method. Then, based on the feature importance values, EFS, outlier percentage, and flow features were compared to know the most superior features. The newly presented TCP features were also assessed based on feature importance. In the case of RF and XGboost methods, 10-fold CV was used, and the average of feature importance values for all folds was calculated to avoid bias.

Furthermore, different ML algorithms were applied to assess the TLSMalDetect detection performance and examine the effect of including and excluding EFS features on that performance. The

algorithms were LR, NB, RF, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT), and XGBoost. The best performing ML algorithm for TLSMalDetect was also determined based on accuracy metric as the two dataset classes (malware and benign) were balanced. Moreover, for accurate results without bias, a 5-fold CV was applied to obtain the following metrics: F1-score, accuracy, recall, precision, and AUC. The reason for choosing 5-fold is that Anderson et al. (2018) obtained an excellent LR accuracy of 99.6 percent using a total of ~500K flows and 10-fold CV. This indicated ~50K flows per fold. For this paper's experiment, out of 279,120 total flows, a 5-fold CV would provide ~50K flows per fold.

EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the results of the EFS features and TLSMalDetect assessments. It also statistically explores the malware families in the dataset and the contrasts between the malware and benign feature values to understand their ground truth.

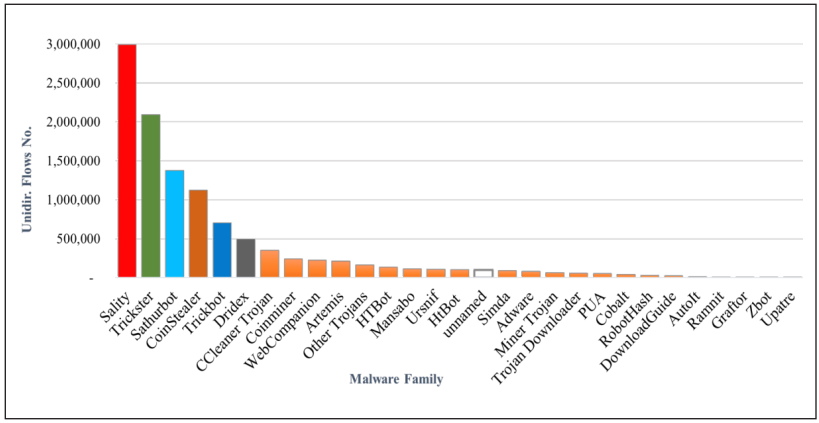
Datasets

Two public sources were used to collect the experiment dataset: CIC-IDS2017 and Stratosphere IPS - Malware Capture Facility Project (MCFP) (from years: 2013, 2016–2018, 2020, 2021) (*IDS 2017 | Canadian Institute for Cybersecurity | UNB*, n.d.; *MCFP Dataset (Malware Capture Facility Project - CTU University)*, n.d.). The total benign and malicious unidirectional flows collected were 1,552,221 and 10,919,379, accordingly. It was assumed that the malware flows contained a few portions of benign traffic generated by operating system requests. However, these portions do not simulate human behavior.

The malware flows contained 28 families illustrated in Figure 4, according to MCFP Stratosphere IPS project categorization. Two malware executables, which formed 94,701 flows, were from unknown families, and they were found to access the malicious websites http://node.viaxmr.com/ and http://5.8.88.175/. As shown in Figure 4, the most common malware family was Sality. This type of malware infects Windows operating systems and can communicate through encrypted peer-to-peer networks, creating a botnet.

Figure 4

Malware Families



Contrast between Malware and Benign Feature Values

The selection of features must be inspired by evident feature value differences between malware and benign flows. Consequently, certain features in the labeled master metadata file from Phase 2 were displayed in figures.

1) EFS and Outliers Percent

Figures 5 and 6 display EFS and outlier percentage of the number of packets received (numPktsRcvd) and sent (numPktsSnt). As seen in these figures, the malware entropy values were often low as compared to the benign values, indicating the regularity of malware. Furthermore, most of the benign (green) portion in EFS figures was more considerable and purer than the benign (green) portion in outlier percentage figures. Therefore, numPktsRcvd and numPktsSnt EFS features seemed better than the matching outlier percentage at differentiating malware.

Figure 5

Differences in Outlier Percentage and EFS of the Number of Packets Sent

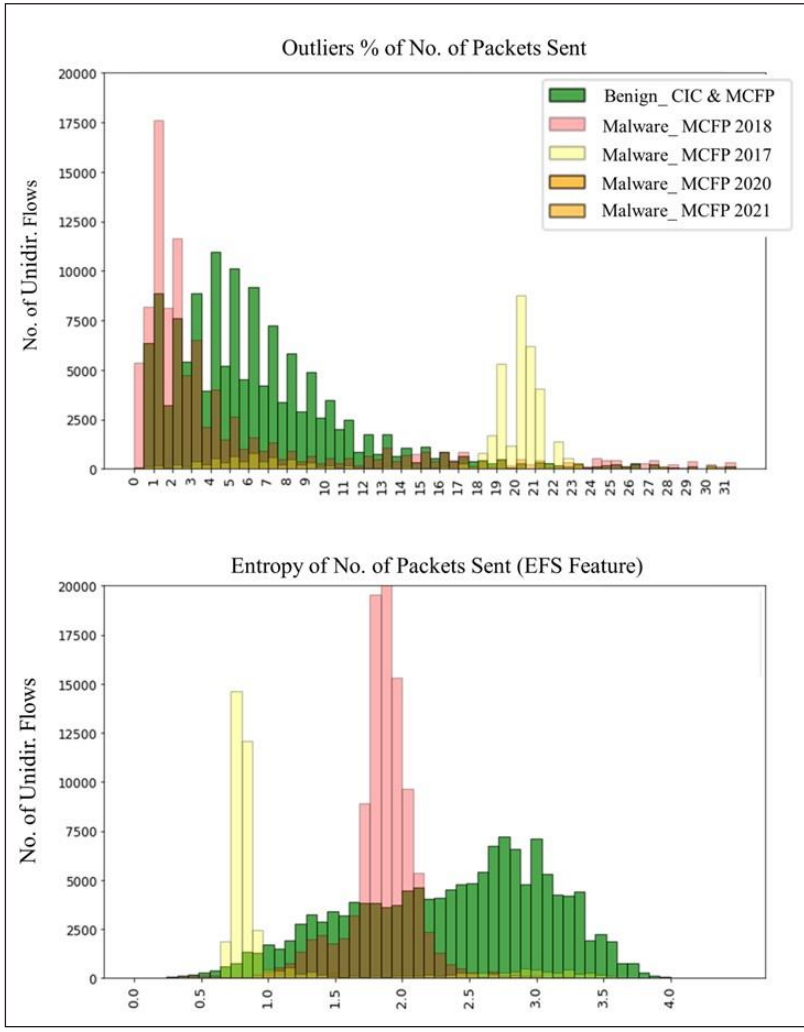
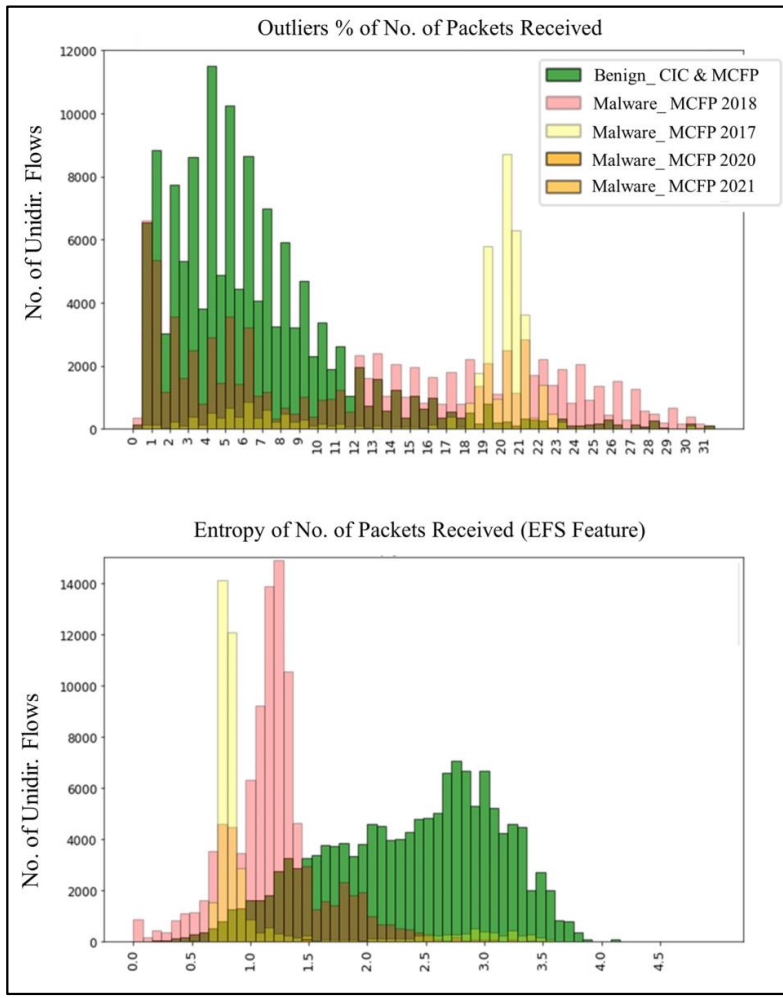


Figure 6

Differences in Outlier Percentage and EFS of the Number of Packets Received



2) TLS

Figures 7 and 8 illustrate the distinctions between malware and benign ciphersuites and extensions of the TLS client hello message (some values were excluded for graph clarity). As seen in Figure 7, nearly all malware flows offered a few extensions (2 or 3), while almost

all benign flows offered more (9 or 10). Additionally, more than 77 percent of malware flows had 12 ciphersuites offered, and more than 60 percent of benign flows had 15 ciphersuites offered. These results were consistent with those of Jenseg (2019) but differed in the number of flows.

Figure 7

Differences in the Number of Client-Offered Ciphersuites and Extensions

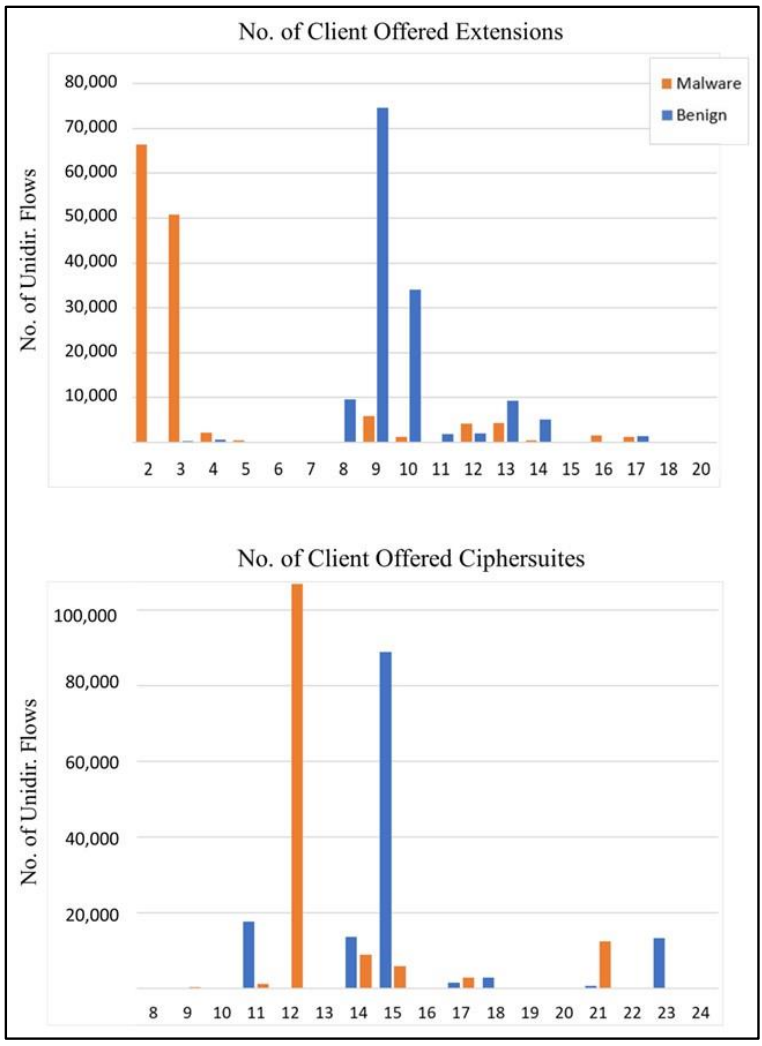
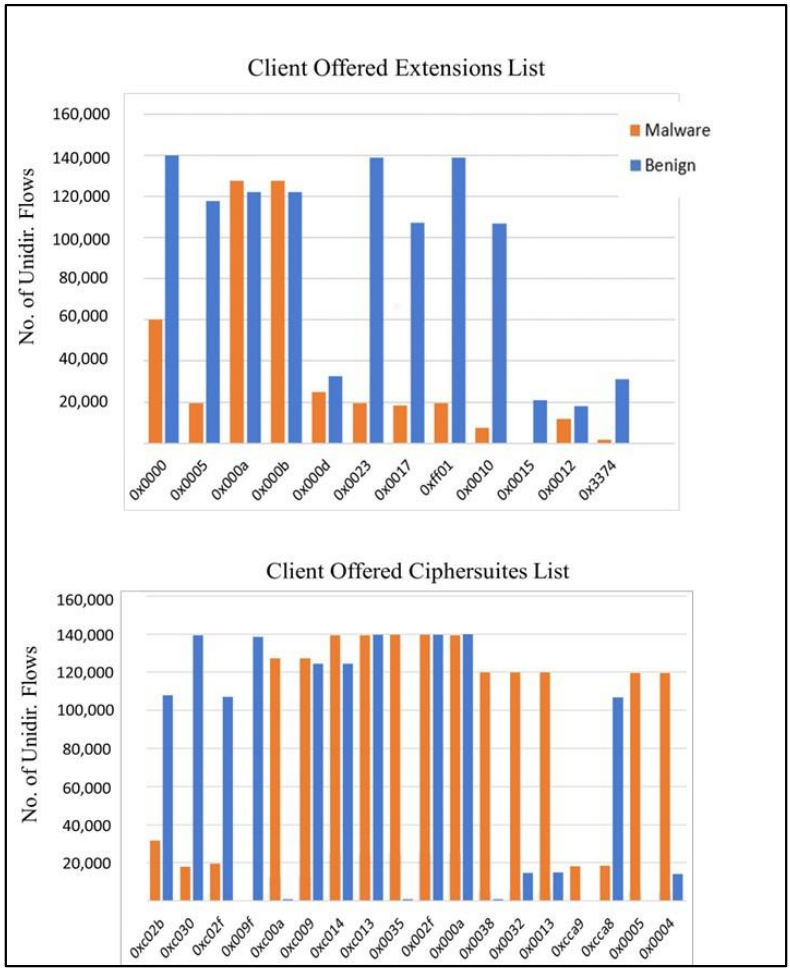


Figure 8

Differences in Client-Offered Ciphersuites and Extensions



As shown in Figure 8, ~90 percent of malware flows provided the extensions 0x000a (supported groups - elliptic curves) and 0x000b (EC point formats). Nevertheless, these extensions were not good distinguishers of malware because the normal flows offered them at almost the same frequency. On the other hand, around 100 percent of benign flows advertised 0xff01 (renegotiation_info), 0x0023 (session ticket), and 0x0000 (server name) extensions. Consistent with the studies by Anderson et al. (2018) and Roques et al. (2019),

some extensions were seen in 70 percent or more of the benign flows and rarely seen in the malware flows. These extensions were 0x0005 (Status request), 0x0010 (ALPN), 0x0017 (Extended master secret type), 0xff01(renegotiation_info).

When investigating the advertised ciphersuites, the distinction between malware and benign flows became more apparent. As seen in the ciphersuites part of Figure 8, four ciphersuites perfectly distinguished flows, three for malware and one for benign flows. Interestingly, these four ciphersuites were all considered weak, according to Rudolph and Grundmann (n.d.). The four ciphersuites were as follows:

- 0x0035 (advertised by ~100% of malware flows and ≤ 3% of benign flows).
- 0x0038 (advertised by ~86% of malware flows and ≤ 3% of benign flows).
- 0xc00a (advertised by ~92% of malware flows and ≤ 3% of benign flows).
- 0x009f (advertised by ~100% of benign flows and ≤ 3% of malware flows).

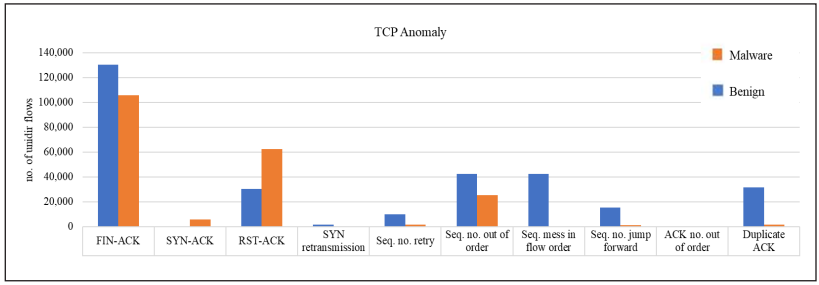
The description of the hex code of ciphersuites can be found on the IANA website (*Transport Layer Security (TLS) Parameters*, n.d.).

3) TCP

Figure 9 presents the variances between malware and benign traffic relevant to TCP anomaly states. As shown in the figure, the malware solely sent SYN-ACK flag (the benign flows had zero SYN-ACK flags) and sent more of RST-ACK flag than the benign applications. One possible explanation for the malware's only usage of the SYN-ACK flag is that the malware operated as a TCP server receiving connection (with SYN flag) from the attackers or, in the case of botnets, from other malware peers. Another explanation is that the malware sent SYN-ACK packets in response to modified SYNs sent by the C&C server to exhaust another victim device (Seaman, 2019). The other observation of the increase in RST-ACK flag could be attributed to RST attacks such as RST hijacking to reset the victim's session or RST flood/DDoS.

Figure 9

Differences in TCP Anomaly Features

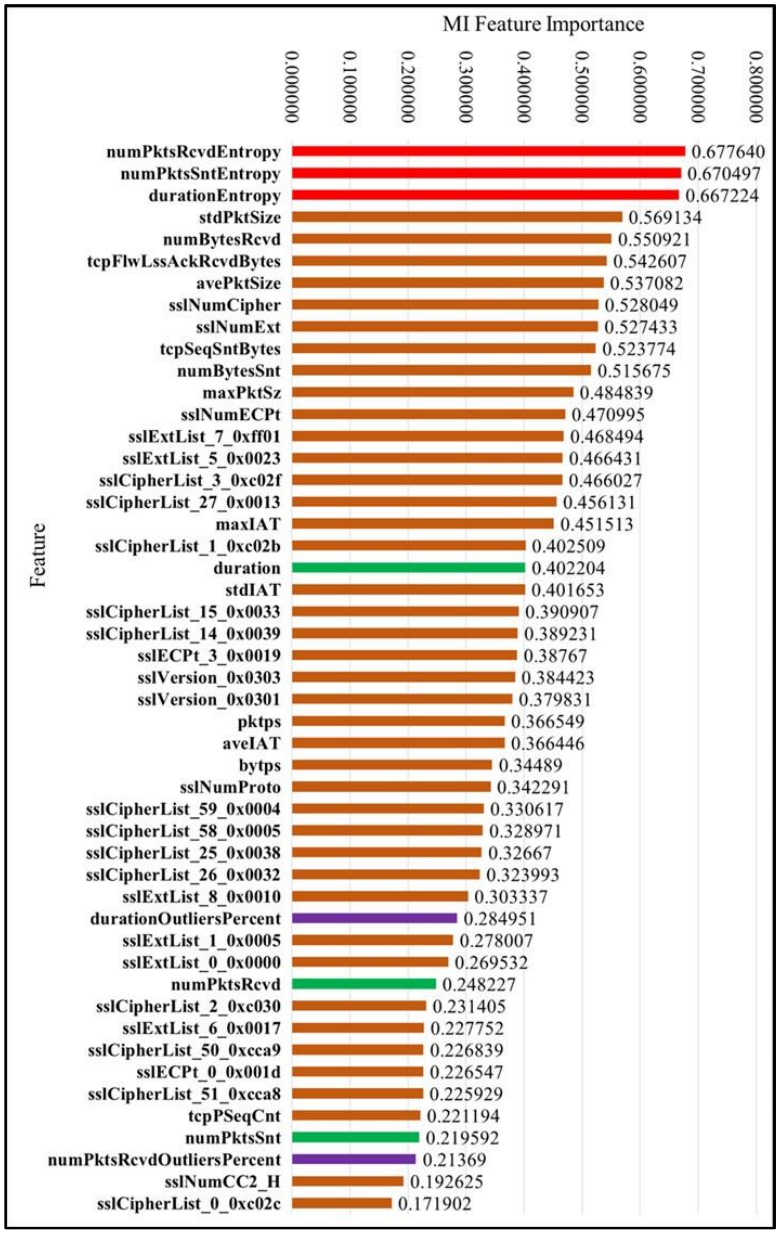


Feature Importance and EFS Features Superiority

The feature importance values for the top 50 features using MI are depicted in Figure 10. The EFS features (in red) were the top three, exceeding the corresponding outlier percentage (in purple) and flow features (in green). The numPktsSntOutliersPercent feature was not shown in the top 50 features figure as it was less significant.

Figure 10

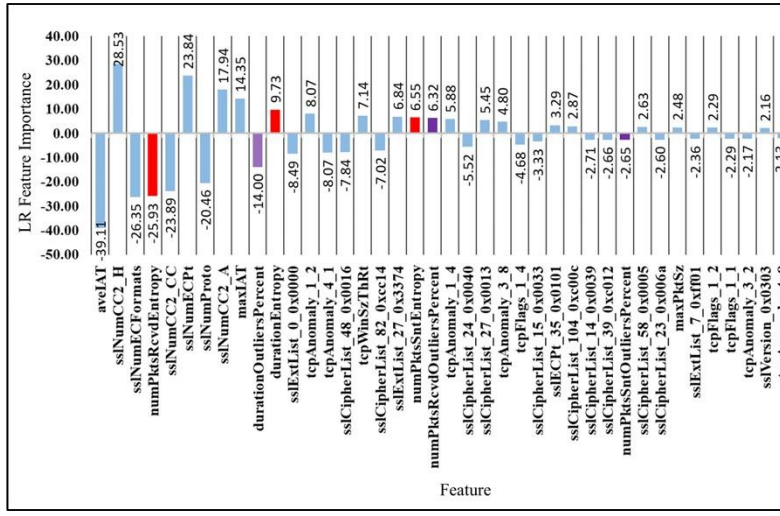
Top 50 Features using Mutual Information – Filter Method



When the LR regularization feature importance method was used, the superiority of EFS over the matching outlier percentage and flow features was further proven. Figure 11 illustrates the top 40 features obtained by applying that method. The feature importance values were the coefficients of the LR model; the closer to 0 the value is, the less important it is. As shown in the figure, all EFS features (in red) were superior to the matching outlier percentage (in purple) and flow features (not even in the top 40), excluding one EFS feature, durationEntropy. However, durationEntropy immediately followed durationOutliersPercent in the absolute value order of the coefficients, and it outranked the duration flow feature.

Figure 11

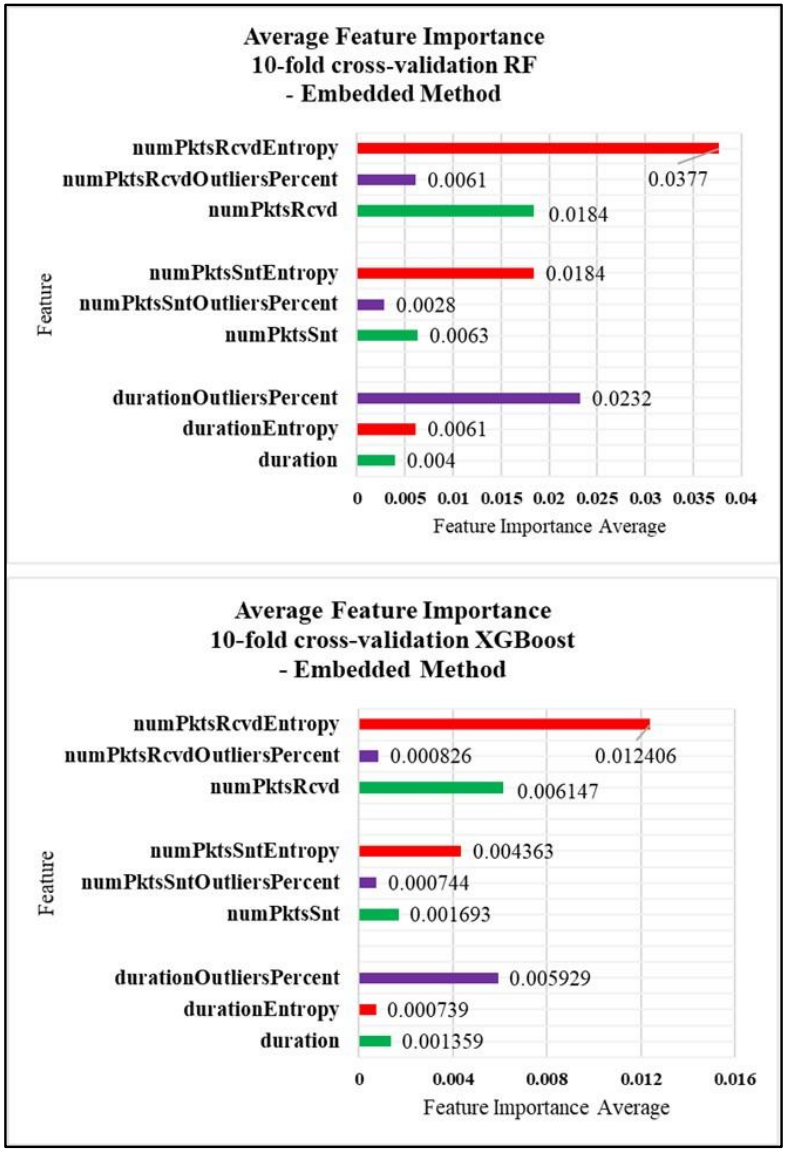
Top 40 Features using LR Regularization with Penalty L1 – Embedded Method



For more certainty, two additional tree-based embedded feature importance techniques were used with 10-fold CV to prevent bias: RF and XGboost. Figure 12 displays the sorted list of the average of the 10-fold feature importance values for EFS, outlier percentage, and flow features. Once more, the EFS features (in red) outperformed the matching outlier percentage (in purple) and flow features (in green) apart from durationEntropy, which was even lower than the duration flow feature using XGboost.

Figure 12

The Average Feature Importance using RF and XGBoost with 10-fold CV



Based on the above findings, it can be inferred that the EFS of flow duration is not a very powerful feature as compared to outlier

percentage and flow features. It could be because the duration values were taken precisely (two digits after the decimal point); therefore, the entropy did not distinguish well. However, the two EFS features of numPktsRcvd and numPktsSnt contributed more to TLS-based malware detection than the corresponding outlier percentage and flow features. They are superior to the matching outlier percentage because they can expose regularity in malware flow features without depending on value distribution and are more efficient than flow features as they can discover behavior regularity.

Interestingly, further analysis of Figures 10 and 11 revealed that the top 40 features encompassed TLS, TCP, and flow features. As a result, it may be advised that all these feature types should be utilized to identify TLS-based malware and should not be ignored. It is also worth noting that some of the newly presented TCP features confirmed their excellent capability to distinguish malware, as seen by their presence in the top features. TCP flawless ACK received bytes (tcpFlwLssAckRcvdBytes, ranked 6 by MI) and TCP Anomaly state SYN-ACK flag (tcpAnomaly_1_2, ranked 13 by LR) are two examples of these features.

Furthermore, the findings of the MI and LR feature importance methods could provide more information about the TLS features that have been infrequently employed in the literature, as found in the review study by Keshkeh et al. (2021). For example, the Elliptic Curve Group Number (sslNumECPt) and Ciphersuites Number (sslNumCipher) were shown to be essential for detection, with the former ranking 5 (LR) and the last ranking 8 (MI).

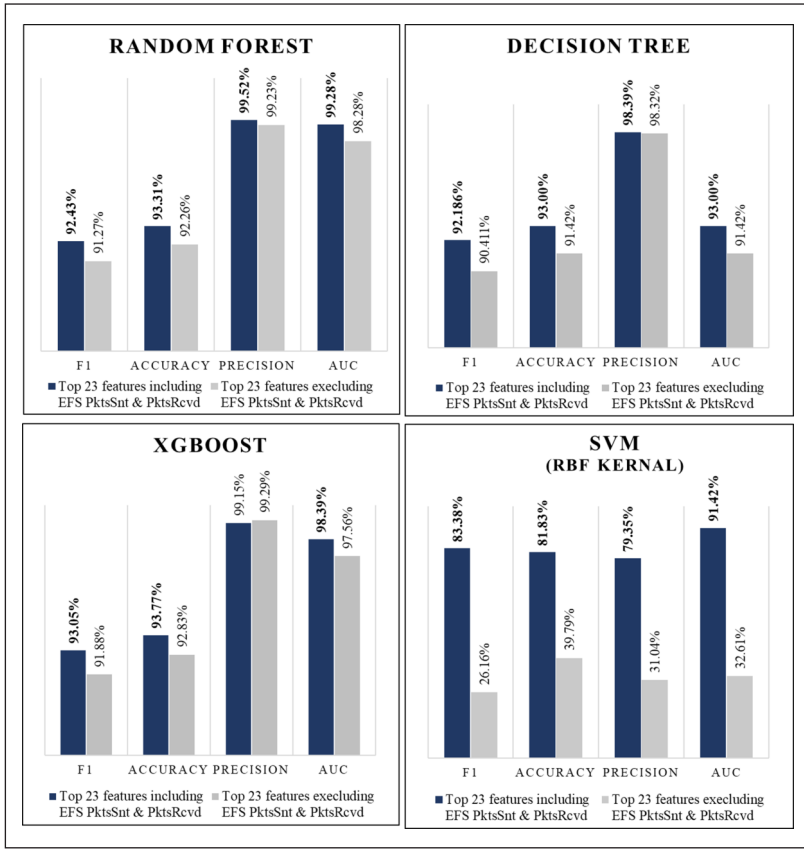
Effect of EFS Features on Performance

The two EFS features of numPktsRcvd and numPktsSnt were demonstrated to be the most significant. Therefore, they were selected to analyze the EFS features' impact on classification performance. One experiment was to apply classification algorithms using the top 23 features in two scenarios: including and excluding the two EFS features. Figure 13 compares F1, accuracy, precision, and AUC results for each algorithm and scenario. It was shown that adding the two EFS features improved all selected metrics, except for the minimal decrease in XGboost precision by ~0.14 percent. Furthermore, the

performance improvement was more apparent in SVM (~42% higher accuracy) and DT (~1.5% higher accuracy).

Figure 13

Classification Performance of the Top 23 Features with and without EFS of numPktsSnt and Rcvd

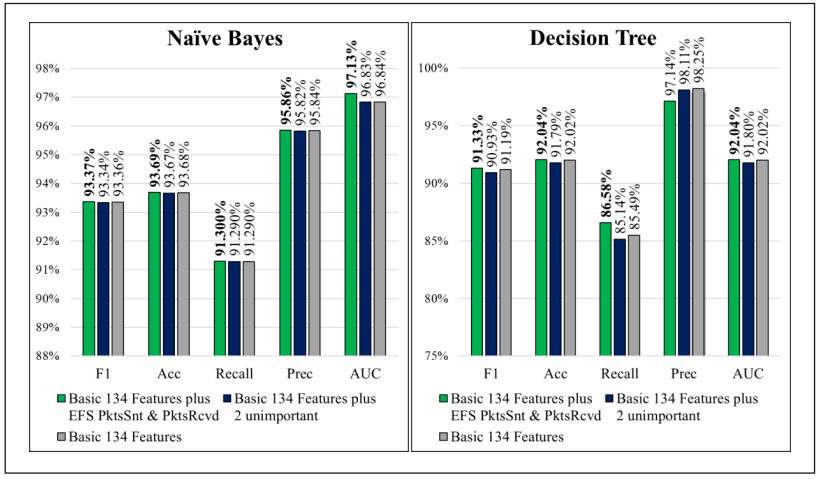


Another experiment used the 136 basic features to analyze the effect of the two EFS features. Three feature lists were used: 134 basic features were obtained by removing the two EFS features, 134 basic features plus two randomly chosen features that were not in 136 basic features, and 134 basic features plus the two EFS features. Then, two fast classification algorithms, NB and DT, were selected to gain all the metric results (around 42 and 129 seconds, accordingly). As shown in Figure 14, the best results were recorded when the two EFS features

were incorporated. Although DT precision decreased by ~1 percent, the F1 score and AUC were still greater when adding the two EFS features.

Figure 14

Classification Performance of the 136 Basic Features with and without EFS of numPktsSnt and Rcvd



Evaluating TLSMalDetect Detection Performance

The overall TLSMalDetect detection performance was evaluated with different ML classification algorithms implemented using 136 basic features and the top 23 feature sets. The seven classification models' metric performance results are listed in Table 3. Comparing the algorithm's metric results showed that NB had the best F1, accuracy, and recall, while RF had the greatest AUC and precision. The best values of each metric are emphasized in bold colors. Compared to the related works that used the same datasets, NB results were the highest. The NB model was also selected as the final model for TLSMalDetect because it scored the best accuracy utilizing the class-balanced dataset.

Furthermore, when the results of the two feature sets, the 136 basic and top 23, were analyzed, almost all second set values were higher than the first set values. This observation indicated the significance of dimensionality reduction in boosting performance.

Table 3

TLSMalDetect Classification Performance Evaluation

Algo.	F1		Accuracy		Recall		Precision		AUC	
	Basic features (136)	Top 23 features	Basic features (136)	Top 23 features	Basic features (136)	Top 23 features	Basic features (136)	Top 23 features	Basic features (136)	Top 23 features
XGBoost	92.90%	93.05%	93.58%	93.77%	88.59%	88.26%	98.26%	99.15%	98.57%	98.39%
NB	**93.37%**	**94.48%**	**93.69%**	**94.88%**	**91.30%**	**92.91%**	95.86%	97.48%	97.13%	98.49%
RF	91.76%	92.43%	92.63%	93.31%	86.06%	87.02%	**98.99%**	**99.52%**	**99.01%**	**99.28%**
LR	91.99%	91.97%	92.80%	92.84%	87.37%	86.96%	98.10%	98.58%	96.62%	97.58%
KNN	91.62%	91.73%	92.30%	92.73%	87.04%	86.04%	97.11%	99.32%	92.97%	94.36%
DT	91.33%	92.19%	92.04%	93.00%	86.58%	87.40%	97.14%	98.39%	92.04%	93.00%
SVM	60.90%	83.38%	71.78%	81.83%	61.17%	91.09%	67.73%	79.35%	84.69%	91.42%

Comparison with Related Studies

Detecting TLS-based malware using the MCFP dataset as a primary source and ML classification was comparable to what Strasák (2017) and Jenseg (2019) did, despite the feature differences. Therefore, it was insightful to compare TLSMalDetect performance with these two studies depending on the best precision and recall. Precision expresses how many alerts raised are for actual malware flows, and recall reflects how many malicious flows are triggered out of all malicious flows. Table 4 illustrates the best precision and recall of TLSMalDetect and the two related studies. The table showed that TLSMalDetect's best precision exceeded Strasák's (2017) best precision by more than 1 percent through utilizing the basic feature set. Additionally, using the top 23 features, TLSMalDetect's best recall slightly surpassed the best recall of Jenseg (2019) by ~0.3 percent. Moreover, it is worth noting that XGboost's TLSMalDetect precision of 98.26 percent also surpassed the best precision achieved in Strasák's (2017) study.

Table 4

Comparison with Other Two Related Studies

Feature Set	Evaluation Metric	Related Work	Best Result		Best Result Algo.	
			Related Work	TLSMalDetect	Related Work	TLSMalDetect
Basic feature set	Precision	Strasák (2017)	97.78%	**98.99%**	XGBoost	**RF**
Top 23 features	Recall	Jenseg (2019)	92.88%	**92.91%**	RF	**NB**

Based on the findings in Table 4, it can be concluded that compared to the total alerts, TLSMalDetect can raise true malware alerts more than Strasák's (2017) detector since it showed a higher precision rate. More importantly, TLSMalDetect can also detect more actual malware flows than Jenseg's (2019) approach out of all malicious flows due to its greater recall rate.

Although the recall rate increase is slight, its impact on detection in real-life implementation is substantial. Detecting a few more malware

flows could safeguard the whole network from severe security breaches.

Furthermore, the variation of ML algorithms used to obtain the best results in Table 4 demonstrated that no perfect ML algorithm works for all cases studied. Therefore, many algorithms should be tested to select the best one eventually.

CONCLUSION

TLS-based malware is one of the severest malware forms due to its reliance on encryption to conceal communication. Various anomaly-based detection research works have been introduced to leverage different features and ML algorithms in detecting TLS-based malware. However, most of these works ignored flow feature transformation or depended on inefficient flow feature transformation techniques such as obtaining the frequency (periodicity indicator) and outlier percentage. Therefore, a TLS-based malware detection approach named TLSMalDetect was proposed to overcome these drawbacks by integrating periodicity-independent EFS features. As seen in the findings, the EFS of numPktsSnt and numPktsRcvd were more effective than the corresponding outlier percentage and flow features using four feature importance methods: MI, LR regularization-L1, RF, and XGboost. The results also indicated that EFS features increased the detection performance, especially the accuracy by ~42 percent and ~1.5 percent in the cases of SVM and DT, accordingly. Apart from EFS, some newly integrated TCP features showed promising importance and were highly ranked using MI and LR regularization-L1 methods. Moreover, utilizing the basic features, TLSMalDetect had the best F1 (93.37%), accuracy (93.69%), and recall (91.30%) by NB, and the highest precision (98.99%) and AUC (99.01%) by RF. The NB results were also the first in the related literature using the same dataset. Finally, TLSMalDetect's performance outperformed two similar studies based on the best precision and recall.

Future research should explore the relationship between each malware family and the classification performance and study the impact of using more feature types such as contextual DNS and HTTP. It is also worth to investigate the performance of neural networks, such as Convolutional Neural Network (CNN), which have shown encouraging results in the related literature.

ACKNOWLEDGMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Albright, D. (2021). *Benchmarking average session duration: What it means and how to improve it*. https://databox.com/average-session-duration-benchmark#definition

Anderson, B. H., & Mcgrew D. (2019). *U.S. Patent No. 10,805,341: Leveraging point inferences on HTTP transactions for HTTPS malware detection*. Google Patents. https://patents.google.com/patent/US10805341B2/en

Anderson, B., & McGrew, D. (2016). Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, Co-Located with CCS 2016* (pp. 35–46). https://doi.org/10.1145/2996758.2996768

Anderson, B., & McGrew, D. (2017). Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F1296* (pp. 1723–1732). https://doi.org/10.1145/3097983.3098163

Anderson, B., Paul, S., & McGrew, D. (2018). Deciphering malware's use of TLS (without decryption). *Journal of Computer Virology and Hacking Techniques*, *14*(3), 195–211. https://doi.org/10.1007/s11416-017-0306-6

Bazuhair, W., & Lee, W. (2020). Detecting malign encrypted network traffic using perlin noise and convolutional neural network. In *10th Annual Computing and Communication Workshop and Conference, CCWC 2020* (pp. 200–206). https://doi.org/10.1109/CCWC47524.2020.9031116

Calderon, P., Hasegawa, H., Yamaguchi, Y., & Shimada, H. (2018). Malware detection based on HTTPS characteristic via machine learning. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy* (ICISSP) (pp. 410–417). https://doi.org/10.5220/0006654604100417

Dai, R., Gao, C., Lang, B., Yang, L., Liu, H., & Chen, S. (2019). SSL malicious traffic detection based on multi-view features. *ACM*

International Conference Proceeding Series, 40–46. https://doi.org/10.1145/3371676.3371697

Fehrman, B., Woody, E., & Lillo, J. (2020). *Detection of SSL/TLS malware beacons*. Google Patents.

IDS 2017 | Canadian Institute for Cybersecurity | UNB. (n.d.). https://www.unb.ca/cic/datasets/ids-2017.html

Jenseg, O. (2019). A machine learning approach to detecting malware in TLS traffic using resilient network features (Master's Thesis, NTNU). https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2617735

Kato, H., Haruta, S., & Sasase, I. (2019). Android malware detection scheme based on level of SSL server certificate. In *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*, *2* (pp. 379–389). https://doi.org/10.1109/GLOBECOM38437.2019.9013483

Keshkeh, K., Jantan, A., Alieyan, K., & Gana, U. M. (2021). A review on TLS encryption malware detection: TLS features, machine learning usage, and future directions. Abdullah N., Manickam S., Anbar M. (Eds)*, Advances in Cyber Security. ACeS 2021. Communications in Computer and Information Science*, *1487*, 213–229. https://doi.org/10.1007/978-981-16-8059-5_13

Liu, J., Zeng, Y., Shi, J., Yang, Y., Wang, R., & He, L. (2019). Maldetect: A structure of encrypted malware traffic detection. *Computers, Materials and Continua*, *60*(2), 721–739. https://doi.org/10.32604/cmc.2019.05610

Malware-Traffic-Analysis.net - Qbot (Qakbot) infection. (2020). https://www.malware-traffic-analysis.net/2020/01/29/index.html

Maroušek, J. (2017). *Efficient kNN classification of malware from HTTPS data*. https://dspace.cuni.cz/bitstream handle/20.500.11956/90345/BPTX_2016_1_11320_0_443594_0_184381.pdf?sequence=1

MCFP Dataset (Malware Capture facility project - CTU University). (n.d.). https://mcfp.weebly.com/mcfp-dataset.html

Nagy, L. (2020). *Nearly a quarter of malware now communicates using TLS – Sophos News*. https://news.sophos.com/en-us/2020/02/18/nearly-a-quarter-of-malware-now-communicates-using-tls/

Roques, O., Maffeis, S., & Cova, M. (2019). Detecting malware in TLS traffic (PhD dissertation, Imperial College London).

Rudolph, H. C., & Grundmann, N. (n.d.). *Cipher Suite Info*. https://ciphersuite.info/cs/

Seaman, C. (2019, July 2). *Akamai Blog Anatomy of a SYN-ACK Attack*. https://www.akamai.com/blog/security/anatomy-of-a-syn-ack-attack

Senecal, D., Kahn, A., Segal, O., … E. S.-U. P. A. 15, & 2019, U. (2019). Bot detection in an edge network using Transport Layer Security (TLS) fingerprint. *Google Patents*, *1*.

Strasák, F. (2017). Detection of HTTPS malware traffic. *Czech Technical University in Prague, Computing and Information Centre*, *May*, 1–49.

Torroledo, I., Camacho, L. D., & Bahnsen, A. C. (2018). Hunting malicious TLS certificates with deep neural networks. In *Proceedings of the ACM Conference on Computer and Communications Security* (64–73). https://doi.org/10.1145/3270101.3270105

Tranalyzer - About. (n.d.). from https://tranalyzer.com/

Transport Layer Security (TLS) Parameters. (n.d.). https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml

Zheng, R., Liu, J., Liu, L., Liao, S., Li, K., Wei, J., Li, L., & Tian, Z. (2020). Two-layer detection framework with a high accuracy and efficiency for a malware family over the TLS protocol. *PloS One*, *15*(5), e0232696. https://doi.org/10.1371/journal.pone.0232696