electronics

Article

Encrypted Malicious Traffic Detection Based on Word2Vec

Andrey Ferriyan [1,*,†], Achmad Husni Thamrin [1,†], Keiji Takeda [2,†] and Jun Murai [3,†]

1 Graduate School of Media and Governance, Keio University, Kanagawa 252-0882, Japan; husni@sfc.keio.ac.jp
2 Faculty of Environment and Information Studies, Keio University, Kanagawa 252-0882, Japan; keiji@sfc.keio.ac.jp
3 Keio University, Tokyo 108-8345, Japan; jun@sfc.keio.ac.jp
* Correspondence: andrey@keio.jp
† These authors contributed equally to this work.

Abstract: Network-based intrusion detections become more difficult as Internet traffic is mostly encrypted. This paper introduces a method to detect encrypted malicious traffic based on the Transport Layer Security handshake and payload features without waiting for the traffic session to finish while preserving privacy. Our method, called TLS2Vec, creates words from the extracted features and uses Long Short-Term Memory (LSTM) for inference. We evaluated our method using traffic from three malicious applications and a benign application that we obtained from two publicly available datasets. Our results showed that TLS2Vec is promising as a tool to detect such malicious traffic.

Keywords: privacy preserving IDS; TLS; Network Intrusion Detection System; encrypted malicious traffic

check for updates

Citation: Ferriyan, A.; Thamrin, A.H.; Takeda, K.; Murai, J. Encrypted Malicious Traffic Detection Based on Word2Vec. *Electronics* **2022**, *11*, 679. https://doi.org/10.3390/electronics11050679

Academic Editors: Constantinos Kolias, Georgios Kambourakis and Weizhi Meng

Received: 16 January 2022
Accepted: 17 February 2022
Published: 23 February 2022

1. Introduction

Most Internet traffic is encrypted, which can be seen in the number of active certificates issued by Let's Encrypt [1], Firefox Telemetry [2], and Google transparency report [3]. Along with this change, malware has started to TLS to hide its malicious activities. They share their information by receiving, sending instructions, and retrieving sensitive data from the infected machines. In 2020, VMWare [4] had estimated that 70% of the attacks would leverage encryption and will be increased more in the next year.

The traffic encryption prevents a traditional Network Intrusion Detection System (NIDS) from inspecting the payload, which is crucial to determine whether the traffic is benign or malicious. On the other hand, the most well-known approaches to classification, such as port-based [5] and signature-based [6], do not work well. Port-based inspection can classify traffic according to the service name and port number registry assigned by Internet Assigned Number Authority (IANA) [7]. However, this approach does not work correctly if the application changes to a custom port. Signature-based IDS may solve the problem, but needs a predetermined set of rules and features, which is cannot reliably detect unknown malicious traffic. A lof of research used payload-independent statistical approaches from 1999 until recent years [8–11]. The use of flow attributes such as packet length, inter-arrival time, and flow duration makes it unnecessary to dissect the payload and avoid breaching the user's privacy [12].

For detecting malicious encrypted traffic, we propose an approach that uses Deep Learning techniques incorporated with Word2Vec, which we call TLS2Vec. In this research, our goal is to detect encrypted malicious traffic that utilizes SSL/TLS using packets from the TLS handshake and the payloads without waiting for the conversation to finish. Hence, with TLS2Vec we preserve privacy as the traffic is not decrypted and we can take an immediate action while the traffic is running. We assume that TLS2Vec is used as a part of an ensemble of NIDS, which the ensemble includes a host behavior analysis system and others. Our method has two important characteristics: first, we do not need to decrypt the traffic, which will impact in keeping privacy. Second, we can take immediate action as soon as we determine that the traffic is malicious. We use two datasets that use encryption protocol in their traffic for our evaluations: the CTU-Malware-Capture dataset [13] and Jason Stroschein's public Github malware samples [14]. The contribution from our paper is detecting malicious traffic using a TLS session before the conversation finishes between client and server.

We organize our paper as follows. We review the existing related works and provide the essential features extracted from their techniques in Section 2. The datasets are described in Section 3, while Section 4 covers our proposed method, TLS2Vec. Section 5 provides the evaluation results and analysis. Finally, the last section concludes our paper, including future work, and open questions.

2. Related Works

The detection of encrypted malicious traffic has gained attention in the network security fields. While many research topics are related to traffic classification with different techniques, only a few studies focus on encrypted malicious traffic. The signature-based approach proposed by [15] relies on Snort [16] to detect encrypted malicious traffic. While Snort is capable of Deep Packet Inspection (DPI), the pattern matching rules are inadequate to handle malicious traffic, especially encrypted. Papadogiannaki et al. [17] proposed a fast signature-based NIDS, detecting malicious traffic even if it is encrypted. While the detection in signature-based is swift for known attacks, its rules must be updated regularly to keep up with all available attack signatures.

Callegati et al. and Sen et al. proposed man-in-the-middle (MITM) and DPI [18,19]. MITM and DPI must decrypt traffic using a key for inspections, hence they are expensive and cannot work without the key, and potentially breach privacy. That is why traffic analysis is preferred [20], but analysis without decryption has challenges due to conversations over encrypted channels.

Wala and Cotton [21] studied traffic analysis with fingerprinting by aggregating several pieces of information from Zeek-IDS [22], such as operating system and the version, open ports on a host, and types of browsers. Prasse et al. [23] proposed how to detect malware traffic on client computers based on HTTPS traffic analysis based on statistical patterns in the timing and the sequence of the network flows from and to the client.

Anderson and McGrew [24] analyzed the network flow with a feature set involving domain experts in supervised learning. The analysis was based on millions of TLS encrypted sessions from a commercial malware sandbox for more than one year. Shekhawat et al. [25] proposed detecting malicious traffic by performing feature analysis on several logs generated from Zeek-IDS. This analysis determined the relative importance of these features from three of the logs. Zheng et al. [26] proposed two-layer detection to identify malicious TLS flow. The author's method uses two different layers. In the first layer, a set of TLS handshake features was employed and trained to distinguish between benign and malicious TLS. The second layer is then used to identify malware families using TLS handshake features and statistical features. The authors claimed 99.45% accuracy was reached using their method. Dai et al. [27] proposed malicious traffic detection based on multi-view features. The authors extract the features from multiple views such as flow statistics, SSL handshake field, and certificate key. While their methods are comprehensive, using flow statistics requires communication to finish between client and server. At the same time, it will affect the delay of the action because we need to wait for the session to finish between communicating parties.

Several works have been carried out in unsupervised learning to detect anomalies within network flow. The work in [28] proposed an unsupervised Network Intrusion Detection System (NIDS) to detect unknown network attacks in encrypted networks. Su et al. [29] implemented a hierarchical clustering algorithm to extract huge training data. At the same time, Li et al. [30] proposed a clustering algorithm with a combination of K-Nearest Neighbors (KNN) for detecting anomalies. The advantage of unsupervised learning is detecting unknown malicious traffic within an encrypted network. However, without adequate information to determine the ground truth, there is a possibility that a high false alarm rate may occur. Furthermore, the algorithm might take time to analyze any possibilities.

Based on Baroni et al. [31], the implementation of Word2Vec has proven to be successful in different areas. In non-network areas, Baek and Chung [32] used Word2Vec to build a multimedia recommendation system based on trust relationships by encoding the sentiment words in related comments into word vectors. At the same time, Chuan et al. [33] adopted Word2Vec to capture the relationship between harmonic and meaningful tonal in music. Ring et al. [34] proposed a similarity measure between IP Addresses using Word2Vec, where the idea is to extract available context information from the network connection. Goodman et al. [35] proposed translated packets into vectorized representations then formed a sequence of words. In contrast, Li et al. [36] used Word2Vec to deal with the semantic gap in the anomalous HTTP traffic.

3. Dataset Description

In this research, we used two public datasets that contain malicious encrypted traffic. We chose datasets with more encrypted malicious traffic than the other publicly available datasets. We did not include datasets, such as [37], that are not publicly available. CTU-Malware-Capture [13] is a dataset produced from Malware Capture Facility Project [38] responsible for long-term captures. Second, we use Jason Stroschein's public Github malware samples [14]. We use Zeus, benign, and Cobalt from TU-Malware-Capture and Trickbot from Jason Stroschein. The choice of Zeus, Cobalt, and Trickbot is based on their communication utilizing encryption, the variety of the activities [39], and how their activity opens the backdoor for further malicious applications [40].

All the datasets were raw PCAP files consisting of TLS packets and any other packets. We considered only the TLS packets. Table 1 shows the basic information of label, total packets, total TLS sessions, and the average number of application data packets per session (Avg. App Data/Session) from the total of the datasets. The Zeus, Benign, and Cobalt are from CTU-Malware-Capture. While Trickbot is from public Github of Jason Stroschein.

Table 1. Basic information of the datasets.

Label	Total Packets	Total TLS Sessions	Avg. App Data/Session
Zeus	2,201,308	10,581	1.932
Benign	1,601,294	1461	1.249
Cobalt	1,471,709	250	2.000
Trickbot	895,172	33	1.909

4. Methodology

The key of our research is to analyze the TLS session by identifying the features from the TLS handshake and the payload. Most of the encrypted network traffic has a specific format that differs from the others. Thus, using the knowledge of this format, it is possible to differentiate and identify the malicious traffic. TLS2Vec has three steps:

(1) Feature Extraction: extracts features from raw PCAP to build a corpus.
(2) Building Vocabulary and Token Parser: uses tokenization technique to extract words from the training dataset and then applies word embedding technique to represent words.
(3) Training Model: TLS2Vec trains the dataset using LSTM and BiLSTM.

Our study analyzes whether the TLS handshake and payload can be used to identify the malicious traffic from benign before the conversation finishes between client and server. We want to emphasize that the results from our methods can be used to take immediate action as soon as we can classify the traffic as malicious. Since opening and analyzing the encrypted content without a private key is almost impossible, this leaves us with the unencrypted content that resides in the header, the TLS handshake, and the statistical information of payload. We decided to analyze the TLS handshake and payload and remove the rest.

We do not include the analysis of the host's behavior in this research. The analysis of the host's behavior is used to determine the encrypted communication from remote access trojan (RAT). The method includes the study of the transmission multiple session-based using 5-tuple. Thus, our method cannot run independently but as an ensemble with the NIDS based on host behavior.

4.1. Feature Extraction

TLS improves the SSL version 3 protocol that provides transport-level security over TCP [41]. TLS consists of a Record Protocol that acts as an envelope for Application Data [42]. The possibility of extracting information from encrypted content is limited. However, it is possible to obtain information in two ways: the unencrypted and its properties and the packet length of the payload. We can get information from the unencrypted phase: the initial handshake and its properties, and the authentication information exchange identifier from the client and the server. During the initial TLS handshake, the client and server negotiate with both exchanging cipher suites information and which protocol version is used. Many cipher suites exist and are not all implemented by the client's application. Usually, the client's application specify the supported cipher suites and which cipher suites are recommended for the initial handshake. The important thing from this behavior is distinguishing malicious applications coming from the client. The next is the authentication exchange information, such as the extensions [43]. Our method considered extensions such as elliptic_curves and ec_point_formats for the features. The malicious applications tend to use weak encryption and offer fewer extensions. Such weak encryption is probably because of the lack of concern with a strong encryption mechanism. Several authors disregard the encryption mechanism. On the other hand, others are only concerned with strong encryption [44].

In payload-based inspection, most of the techniques use the information from the payload of the application layer [45]. Our method is different from [18,19], which requires decrypting the traffic using the key. The packet length from the TCP Application Data is the additional information feature extracted from encrypted traffic. Although some TLS server communication parameters can be changed over time, an application-specific profile usually reflects the Application Data's content size. We considered using the TCP packet length from Application Data. While we cannot see into the unencrypted content, the packet length is directly linked to the payload of the traffic and usually follows an application-specific profile [46]. Thus, we can use packet length to determine which traffic is benign or malicious based on the specific application profile.

The unencrypted phase communication between client and server follows TLS protocol, which can be treated as one sentence in a TLS session. Each sentence has a specific pattern that can distinguish between malicious and benign. Similar words with correlation are represented by similar vectors closely placed in a vector space. In the network, the sequence from the TLS sessions is textual data that should be encoded into a numeric representation. In this case, we argue that a numeric vector can represent the sequence of traffic with similar vectors in the embedding space. Similar traffic represented by words will have the same vector space, benign or malicious.

Figure 1 shows the steps from feature extraction to building a corpus. The single box is a TLS session with two payloads from a PCAP file with a right arrow representing a client's packet and a left arrow representing a server's packet with features. The first two arrows in the left box are handshakes while the rest represent TCP Application Data, which we call payload. We collected zero to a total of two or more payloads from each client and server. Multiple TLS sessions exist in the PCAP file with the symbol of TLS_t. The arrow in the middle represents the features that are being extracted to a corpus. The document symbol on the right represents a corpus containing a TLS handshake and two payloads from the features.

The TLS sessions consist of hundreds of features. Most of the contents are encrypted except the handshake containing handshake protocols like Client Hello, session ID, the version of the TLS, length, the cipher suites, compression method, and extensions. The PCAP contains multiple session conversations between the client and the server, from the three-way handshake to the TLS sessions. We extract the features from the raw PCAP from two datasets and preprocess the data based on the session. Consider a raw PCAP with multiple TLS sessions (TLS_t, $TLS_{t+1} \ldots TLS_{t+n}$). For each session, we extract the TLS handshake TLS/SSL version (version), Cipher Suites List from Client Hello or Server Hello (cipher), Extension Length from Client Hello or Server Hello (ext_len), Elliptic Curves from Client Hello (elliptic_curves), Elliptic Curve Point Formats from Client Hello (ec_point_formats), and the payload (len). The extracted features are then constructed into words. Each record in our dataset describes one TLS session, which can be viewed as a sentence.

Our study treated a sentence as a TLS language that follows TLS protocol. Because we extract TLS session raw PCAP files, the decimal representation from the extracted data will create a row of numbers representing a communication session between a client and a server. We then add a symbol on each feature based on their representation, as shown in Figure 1 in the document symbol. In the document symbol from Figure 1, C, ET, EC, ECP, CAPP, or SAPP represent cipher, ext_len, elliptic_curves, ec_point_formats, and len. The information in the document symbol is a sample of a single TLS session from the client and server taken from Trickbot malicious traffic. The collection of all these words is called a corpus.

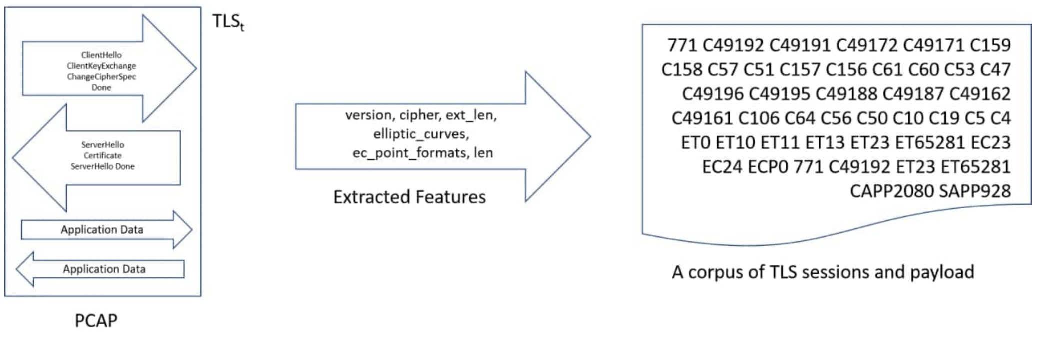
TLS$_t$
ClientHello
ClientKeyExchange
ChangeCipherSpec
Done
ServerHello
Certificate
ServerHello Done
Application Data
Application Data
PCAP
version, cipher, ext_len, elliptic_curves, ec_point_formats, len
Extracted Features
771 C49192 C49191 C49172 C49171 C159 C158 C57 C51 C157 C156 C61 C60 C53 C47 C49196 C49195 C49188 C49187 C49162 C49161 C106 C64 C56 C50 C10 C19 C5 C4 ET0 ET10 ET11 ET13 ET23 ET65281 EC23 EC24 ECP0 771 C49192 ET23 ET65281 CAPP2080 SAPP928
A corpus of TLS sessions and payload

Figure 1. Overview of TLS2Vec.

4.2. Building Vocabulary and Token Parser

In this phase, converting words from the corpus into vector space models uses Gensim [47] version 4.1.2. Word2vec technique trains each word in the corpus and converts each into a 300-dimensional vector. We chose a dimension of 300 because prior research has shown it to be a good choice [48–50]. In this phase, building vocabulary utilizes the tokenization technique to extract words from the training dataset and apply word embedding. A token parser then used the vocabulary to convert each TLS session in the dataset into a token sequence.

In general, the Word2Vec technique has two architectures, the Continuous Bag-of-Words (CBOW) and Skip-gram. The CBOW tries to predict the target word based on the surrounding words' context. On the other hand, Skip-gram tries to predict the context word from a word. In CBOW, the representation words are combined to predict the word in the middle. Both architectures might be used for different purposes in the NIDS incorporated with machine learning. Regarding training time between CBOW and Skip-gram, the former is slightly faster and has less computational cost, while Skip-gram might better predict a large dataset. Skip-gram is more suitable for anomaly detection, which does not make any assumption about the malicious traffic instead of on benign traffic. While in CBOW, the prediction of malicious traffic is based on the surrounding information and its similarity. To determine which one has a better prediction, we evaluated both CBOW and Skip-gram.

4.3. Training Model

Two types of Recurrent Neural Network (RNN) were used to evaluate the performance: Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM). LSTM and BiLSTM have a specialty of their effectiveness in memorizing particular patterns. At the same time, both algorithms are suitable for detecting malicious traffic with payloads [51,52].

5. Evaluations

This section evaluates the performance of the TLS2Vec by performing with binary and multiclass targets. We grouped three malicious traffic from the Zeus, Cobalt, and Trickbot into a single class Malicious for the binary target. The total number of TLS sessions of Malicious traffic was 10,163 after grouping. The considered evaluation used 5-fold cross-validation (CV-5). Note that an imbalance of data distribution exists in this dataset. We used both the categorical_crossentropy and binary_crossentropy for the loss function. The Adam optimizer was applied with a learning rate of 0.001. The rest of the hyper-parameters used their default values. The softmax was used for the activation function for categorical_crossentropy, while sigmoid was used for binary_crossentropy. Figure 2 illustrates the evaluation procedure. Table 2 provides the hyper-parameters regarding detection based on LSTM and BiLSTM.

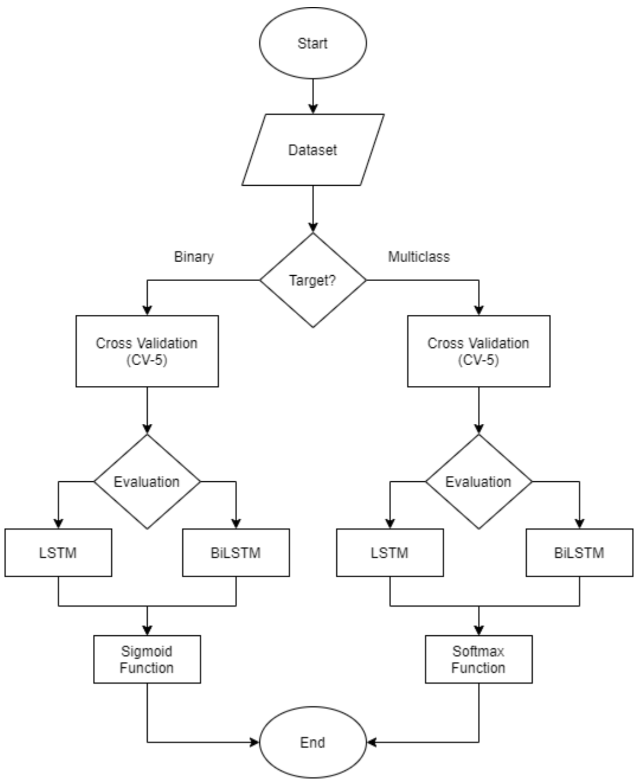

Figure 2. The evaluation procedure.

Table 2. LSTM and BiLSTM hyper-parameter for binary dan multiclass target.

Hyper-Parameter	Binary Target	Multiclass Target
Activation Function	sigmoid	softmax
Epoch	30	30
Optimizer	adam	adam
Learning Rate	0.001	0.001
Batch Size	32	32
Loss Function	binary_crossentropy	categorical_crossentropy

In general, the overall accuracy metric might be used to measure the results in the binary target. However, to rely on the performance only for accuracy is unfavorable, especially when involving a class imbalance dataset. We adopted F1-score to give more insight into the dataset in this research. Based on our analysis of the datasets, we found that most of the payloads from the malicious traffic from each session mainly were two payloads, with several sessions having only three and very few sessions having more than that. From this result, we proceed with our analysis using zero payload, one payload, and two or more payloads. We compared our analysis with a non-TLS2Vec method. In non-TLS2Vec we used the same features as in TLS2Vec except we did not use Word2Vec and removed the symbols that represent features such as cipher, ext_len, elliptic_curves, ec_point_formats, and len. We did this because as far as we know, we did not find any similar methods.

In the first evaluation, we started with a TLS handshake with zero, one, and two or more payloads with TLS2Vec with CBOW, TLS2Vec with Skip-gram, and a Non-TLS2Vec. As illustrated in Figure 3, both CBOW and Skip-gram with LSTM and BiLSTM performed similarly well in detection, with the former slightly better. The better performance was understandable with both can predict the next words in the input sentence from the TLS. In a non-TLS2Vec method, the performance significantly decreased along with the addition of one to two or more payloads. In CBOW-LSTM, the high F1 score started with zero then was relatively stable in one to two or more, meaning the LSTM can detect almost all the Malicious and Benign with the default hyper-parameter. Compared with the counterparts, CBOW-BiLSTM slightly decreased after zero payload and then increased again in two or more. The primary reason was that CBOW-BiLSTM needed to fetch more training data to reach equilibrium with one payload. On the other hand, Skip-gram BiLSTM was slightly better than Skip-gram LSTM. The figure showed us that starting from zero and then increasing after one payload indicates that having more payload is better for improving the performance. The result was consistent as BiLSTM has run inputs in two ways, and this implies that having more payloads might detect more in Benign and Malicious using default hyper-parameters. Based on this result, we continued to use TLS2Vec with CBOW and Non-TLS2Vec, and exclude TLS2Vec with Skip-gram In the subsequent evaluation, we used a different strategy for detection by using a multiclass target. The results showed in Figure 4.

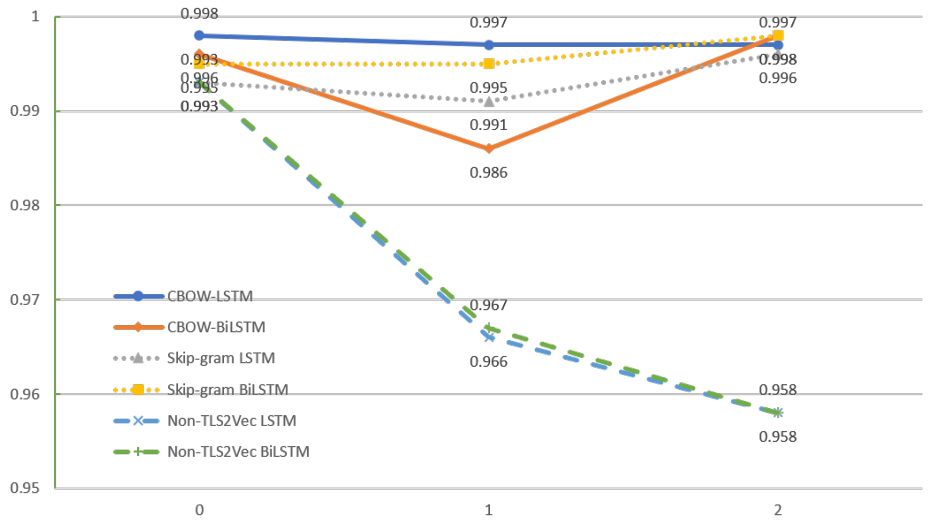

Figure 3. The evaluation between TLS2Vec with CBOW, TSL2Vec with Skip-gram, and Non-TLS2Vec. The x-axis is the payload from zero, one, and two or more payloads, while the y-axis is the F1 score.

The figure illustrated the trend in the increase of detection for both CBOW-LSTM, CBOW-BiLSTM, and non-TLS2Vec. In CBOW-LSTM, both Zeus and Cobalt were stable, starting from zero, one, and two or more. The result indicates LSTM can detect all the Zeus and Trickbot starting from the TLS handshake. While in the Benign class, the performance slightly decreases with one payload and then slightly increases with two or more payloads. The results showed that LSTM needed fetching more training data to reach the same performance with zero payload for one to two or more. As for the counterpart, Trickbot was increased along with payloads. Although the performance went up, the trend was stable. In a non-TLS2Vec, the performance was decreased along with the addition of the payloads for all classes.

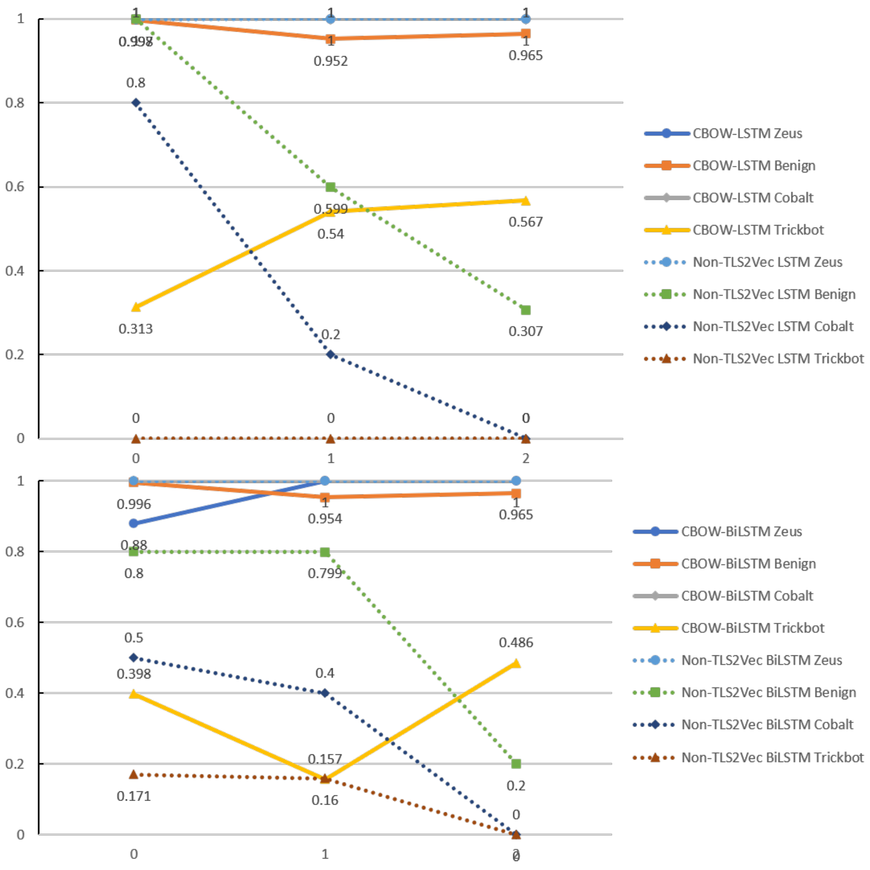

Figure 4. The multiclass target with LSTM (**upper**) and BiLSTM (**lower**). The x-axis is the payload from zero to two or more, while the y-axis is the F1 score.

In the following evaluation, we added more parameters for evaluation using discretization transformation by binning the payload. Discretization transformation reduced the traffic patterns and discriminated among the traffic by grouping those with similar payloads [53,54]. Furthermore, binning reduced the vocabulary. The payload from the datasets has 872 unique values.

The problem in the discretization was selecting the intervals or bins. We used two different methods for selecting the number of bins: bit-length, which we called Bit-Width, and the Sturges rule, used for constructing histogram and non-stationary data [55]. For the Bit-Width (BW), we used intervals from 0 to 63, 64 to 127, 128 to 255, 256 to 511, 512 to 1023, and 1024 or more. The payload in the intervals of six categories will be converted according to the bins. For the Sturges rule, we used two different combinations called Sturges-EqualWidth (SEW) and Sturges-EqualFreq (SEF). SEW ensured each bin has an equal range of the original value, while with SEF, each bin has the same number of samples. For SEW and SEF, selecting a number of bins followed the Sturges rule for only one payload. The bins range results from one payload then applied to the payload two or more.

Notice that during the binning process with SEW, we found that several bins had zero members (see Figure 5). The results were due to SEW's nature, which groups members by their equal range. From this point, if the bin's range was zero, no payload exists within the range. In the subsequent evaluation, we started with implementing BW, SEW, and SEF and then evaluated all four classes. Figures 6–8 show the results of the multiclass target using LSTM and BiLSTM with BW, SEW, and SEF, respectively.

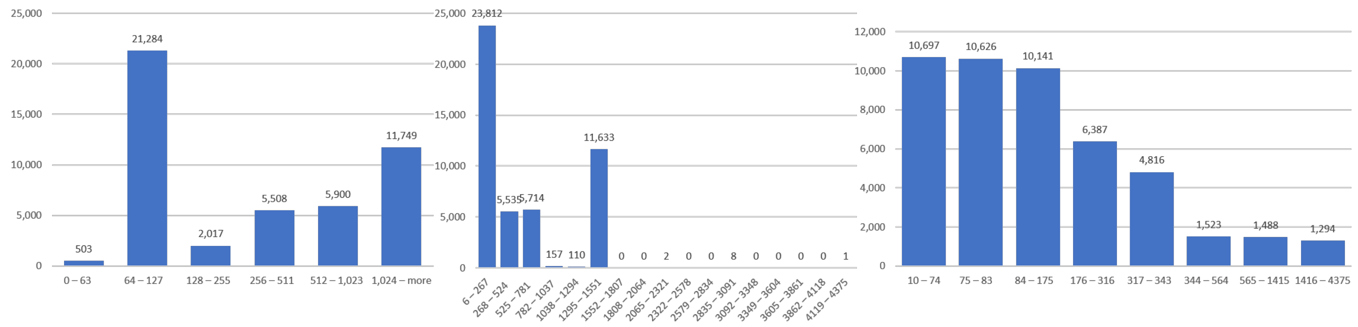

Figure 5. The histogram of one-payload length with BW (**left**), SEW (**center**), and SEF (**right**). The x-axis is the bin range, while y-axis is the member count.

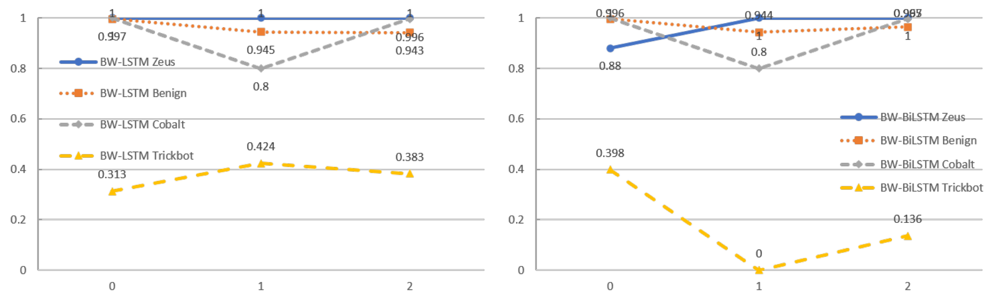

Figure 6. Multiclass target evaluation results using BW-LSTM and BW-BiLSTM. The x-axis is the payload from zero (with no binning), one, and two or more, while the y-axis is the F1 score.

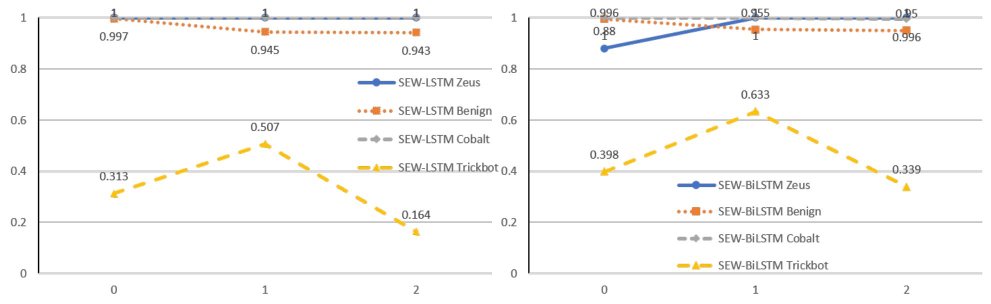

Figure 7. Multiclass target evaluation results using SEW-LSTM and SEW-BiLSTM. The x-axis is the payload from zero (with no binning), one, and two or more, while the y-axis is the F1 score.

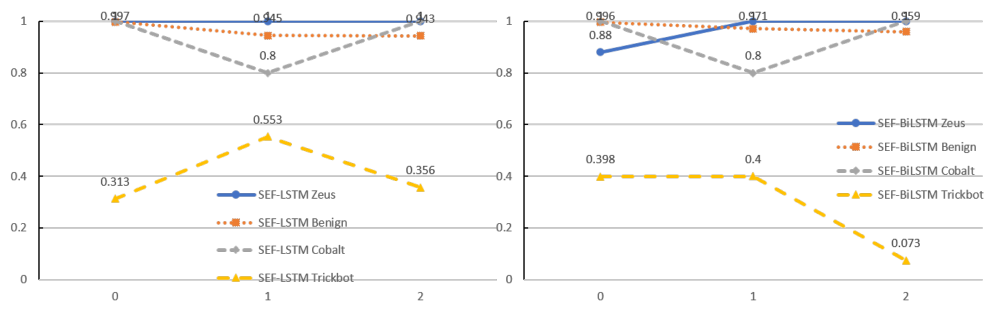

Figure 8. Multiclass target evaluation results using SEF-LSTM and SEF-BiLSTM. The x-axis is the payload from zero (with no binning), one, and two or more, while the y-axis is the F1 score.

BW, SEW, and SEF were relatively stable for Zeus and Benign. While for Cobalt, we found a decrease in BW and SEF. Similarly, Trickbot also decreased in SEW and SEF. The results showed that binning has no significant effect on detection.

We concluded that our methods can be used to detect encrypted malicious traffic based on our results. We can see in the evaluation using CBOW-LSTM, CBOW-BiLSTM, then using Skip-gram LSTM, and Skip-gram BiLSTM. We then continued the evaluation using three different kinds of binnings. First, we compared CBOW and Skip-gram starting with zero to two or more payloads, which increase performance. Our method can detect Benign and Malicious with an average of 0.999 using only TLS handshake information. Both of the models can learn to distinguish between Malicious and Benign. The TLS handshake consists of a cipher suite produced from the Malicious was different from the Benign, and the model can learn and detect to distinguish this information. The reason is that the TLS handshake from non-malicious traffic might be produced from the library of the OS. In contrast, the TLS handshake from the malicious traffic produced by the malicious application [56]. From this point, we can generalize our model using a TLS handshake.

Another advantage of using our method is that we can detect the binary target and the multiclass target using the payload from each traffic. The average detection rate is 0.82 for total from Zeus, Benign, Cobalt, and Trickbot combined. We then evaluated our method using three different binnings: BW, SEW, and SEF, to better characterize the traffic by reducing the vocabulary. However, implementation of binnings will reduce the vocabulary. In general, the implementation of binning has no significant effect on detecting malicious traffic. Furthermore, we found that using our method has disadvantages over the minority class when combined with the binnings. Our results found that using only a TLS handshake is enough to detect encrypted malicious traffic. The additional payloads from one and two or more can increase the performance. However, the effect depends on the TLS handshake.

6. Conclusions and Future Work

Today, most of the network traffic is changing from unencrypted to encrypted. While encryption preserves user's privacy, at the same time, many threats start increasing their malicious intent over the network. The contribution of this paper is a method for detecting malicious traffic using a TLS session before the conversation finishes between client and server. To achieve this, we extract meaningful information from the TLS handshake and some portion from the TCP length of Application Data, which we call payload, and then create a corpus of traffic. We then predicted malicious traffic using based on the TLS session and payload.

Based on the evaluation, we conclude that TLS2Vec is promising to handle encrypted malicious traffic with higher accuracy, especially as malicious applications use their own TLS library instead of available library in the host. We also find that TLS2Vec performs better than Non-TLS2Vec, which use neither symbols nor Word2Vec embedding. Binning the payload length in order to reduce the vocabulary does not improve the performance, but rather it gives worse performance for the minority class. We want to conclude our research that LSTM and BiLSTM have advantages in CBOW for detecting encrypted malicious traffic.

Furthermore, we want to highlight the points that make our solution different from others. Using our method, TLS2Vec can detect encrypted malicious traffic using TLS handshake and a portion of the payload. The impact is that we do not need to dissect the traffic to maintain the user's privacy.

There are at least three open questions based on this paper. In terms of traffic analysis and NIDS, shorter packets from the payload must be further analyzed. Another question is how NIDS may work given the traffic is encrypted. In terms of Deep Learning, future research should address whether we can use the corpus to detect the variants of malicious traffic, such as Zeus.

Author Contributions: Conceptualization, A.F.; data curation, A.F., funding acquisition, K.T., J.M.; investigation, A.F.; methodology, A.F.; formal analysis, A.F., A.H.T.; Resources, A.H.T., K.T., J.M.; Supervision, A.H.T., K.T., J.M.; Validation, A.H.T.; writing—original draft preparation, A.F.; writing—review and editing, A.H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: Our sincere appreciation for the Indonesia government, particularly LPDP (Lembaga Pengelola Dana Pendidikan Indonesia = Indonesia Endowment Fund for Education) that provides the scholarship to study at Keio University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lets Encrypt Status Report. 2021. Available online: https://letsencrypt.org/stats (accessed on 17 December 2021).
2. Firefox Telemetry. 2021. Available online: https://docs.telemetry.mozilla.org/datasets/other/ssl/reference.html (accessed on 17 December 2021).
3. Google Transparency Report. 2021. Available online: https://transparencyreport.google.com/https/overview?hl=en (accessed on 2 December 2021).
4. The Relevance of Network Security in an Encrypted World. 2021. Available online: https://blogs.vmware.com/networkvirtualization/2020/09/network-security-encrypted.html (accessed on 2 December 2021).
5. Sen, S.; Wang, J. Analyzing peer-to-peer traffic across large networks. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment, Marseille, France, 6–8 November 2002; pp. 137–150. [CrossRef]
6. Cao, Z.; Xiong, G.; Zhao, Y.; Li, Z.; Guo, L. A survey on encrypted traffic classification. In Proceedings of the International Conference on Applications and Techniques in Information Security, Melbourne, Australia, 26–28 November 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 73–81. [CrossRef]
7. Service Name and Transport Protocol Port Number Registry. 2021. Available online: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (accessed on 10 October 2021).
8. Marchette, D.J. A Statistical Method for Profiling Network Traffic. In Proceedings of the Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, USA, 9–12 April 1999; pp. 119–128.
9. Crotti, M.; Gringoli, F.; Pelosato, P.; Salgarelli, L. A statistical approach to IP-level classification of network traffic. In Proceedings of the 2006 IEEE International Conference on Communications, Istanbul, Turkey, 11–15 June 2006; Volume 1, pp. 170–176. [CrossRef]
10. Zhang, J.; Xiang, Y.; Zhou, W.; Wang, Y. Unsupervised traffic classification using flow statistical properties and IP packet payload. *J. Comput. Syst. Sci.* **2013**, *79*, 573–585. [CrossRef]
11. Amma, N.B.; Selvakumar, S. A statistical class center based triangle area vector method for detection of denial of service attacks. *Clust. Comput.* **2021**, *24*, 393–415. [CrossRef]
12. Sicker, D.C.; Ohm, P.; Grunwald, D. Legal issues surrounding monitoring during network research. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, CA, USA, 24–26 October 2007; pp. 141–148.
13. Stratosphere. Stratosphere Laboratory Datasets. 2015. Available online: https://www.stratosphereips.org/datasets-overview (accessed on 13 March 2020).
14. Jason Stroschein Public Github Malware Samples. 2021. Available online: https://github.com/jstrosch/malware-samples (accessed on 10 May 2021).
15. Etienne, L. Malicious Traffic Detection in Local Networks with Snort. 2009. Available online: https://infoscience.epfl.ch/record/141022?ln=en (accessed on 13 March 2021).
16. Snort IDS. 2021. Available online: https://snort.org/ (accessed on 10 May 2021).

17. Papadogiannaki, E.; Deyannis, D.; Ioannidis, S. Head(er)Hunter: Fast Intrusion Detection using Packet Metadata Signatures. In Proceedings of the 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Pisa, Italy, 14–16 September 2020; pp. 1–6. [CrossRef]
18. Callegati, F.; Cerroni, W.; Ramilli, M. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Secur. Priv.* **2009**, *7*, 78–81. [CrossRef]
19. Sen, S.; Spatscheck, O.; Wang, D. Accurate, Scalable in-Network Identification of P2p Traffic Using Application Signatures. In Proceedings of the 13th International Conference on World Wide Web (WWW'04), New York, NY, USA, 17–20 May 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 512–521. [CrossRef]
20. Anderson, B.; McGrew, D. Identifying Encrypted Malware Traffic with Contextual Flow Data. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (AISec'16), Vienna, Austria, 24–28 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 35–46. [CrossRef]
21. Wala, F.B.; Cotton, C. Unconstrained Endpoint Security System: UEPTSS. *Int. J. Netw. Secur. Its Appl. (IJNSA)* **2018**, *10*, 1–12. [CrossRef]
22. Zeek IDS. 2021. Available online: https://zeek.org (accessed on 10 May 2021).
23. Prasse, P.; Machlica, L.; Pevnỳ, T.; Havelka, J.; Scheffer, T. Malware detection by analysing encrypted network traffic with neural networks. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Skopje, Macedonia, 18–22 September 2017; Springer: Cham, Switzerland, 2017; pp. 73–88.
24. Anderson, B.; McGrew, D. Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17), Halifax, NS, Canada, 13–17 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1723–1732. [CrossRef]
25. Shekhawat, A.S.; Troia, F.D.; Stamp, M. Feature analysis of encrypted malicious traffic. *Expert Syst. Appl.* **2019**, *125*, 130–141. [CrossRef]
26. Zheng, R.; Liu, J.; Liu, L.; Liao, S.; Li, K.; Wei, J.; Li, L.; Tian, Z. Two-layer detection framework with a high accuracy and efficiency for a malware family over the TLS protocol. *PLoS ONE* **2020**, *15*, e0232696. [CrossRef] [PubMed]
27. Dai, R.; Gao, C.; Lang, B.; Yang, L.; Liu, H.; Chen, S. SSL Malicious Traffic Detection Based On Multi-View Features. In Proceedings of the 2019 the 9th International Conference on Communication and Network Security (ICCNS 2019), Chongqing, China, 15–17 November 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 40–46. [CrossRef]
28. Amoli, P.V.; Hämäläinen, T. A real time unsupervised NIDS for detecting unknown and encrypted network attacks in high speed network. In Proceedings of the 2013 IEEE International Workshop on Measurements & Networking (M&N), Naples, Italy, 7–8 October 2013; pp. 149–154.
29. Su, L.; Yao, Y.; Li, N.; Liu, J.; Lu, Z.; Liu, B. Hierarchical Clustering Based Network Traffic Data Reduction for Improving Suspicious Flow Detection. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security and Privacy in Computing Furthermore, Communications/12th IEEE International Conference on Big Data Science Furthermore, Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 744–753. [CrossRef]
30. Li, L.; Zhang, H.; Peng, H.; Yang, Y. Nearest neighbors based density peaks approach to intrusion detection. *Chaos Solitons Fractals* **2018**, *110*, 33–40. [CrossRef]
31. Baroni, M.; Dinu, G.; Kruszewski, G. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, MD, USA, 22–27 June 2014; Association for Computational Linguistics: Baltimore, MD, USA, 2014; pp. 238–247. [CrossRef]
32. Baek, J.W.; Chung, K.Y. Multimedia recommendation using Word2Vec-based social relationship mining. *Multimed. Tools Appl.* **2021**, *80*, 34499–34515. [CrossRef]
33. Chuan, C.H.; Agres, K.; Herremans, D. From context to concept: Exploring semantic relationships in music with word2vec. *Neural Comput. Appl.* **2020**, *32*, 1023–1036. [CrossRef]
34. Ring, M.; Dallmann, A.; Landes, D.; Hotho, A. IP2Vec: Learning Similarities Between IP Addresses. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18–21 November 2017; pp. 657–666. [CrossRef]
35. Goodman, E.L.; Zimmerman, C.; Hudson, C. Packet2Vec: Utilizing Word2Vec for feature extraction in packet data. *arXiv* **2020**, arXiv:2004.14477.
36. Li, J.; Zhang, H.; Wei, Z. The Weighted Word2vec Paragraph Vectors for Anomaly Detection Over HTTP Traffic. *IEEE Access* **2020**, *8*, 141787–141798. [CrossRef]
37. Lucia, M.J.D.; Cotton, C. Identifying and detecting applications within TLS traffic. In *Cyber Sensing 2018*; Ternovskiy, I.V., Chin, P., Eds.; International Society for Optics and Photonics, SPIE: Bellingham, WA, USA, 2018; Volume 10630, pp. 179–190. [CrossRef]
38. Malware Capture Facility Project. 2021. Available online: https://mcfp.felk.cvut.cz/publicDatasets/datasets.html (accessed on 10 May 2021).
39. Zeus Trojan Analysis. 2022. Available online: https://talosintelligence.com/zeus_trojan (accessed on 12 January 2022).
40. TrickBot: The Multi-Faceted Botnet. 2022. Available online: https://www.kaspersky.com/resource-center/threats/trickbot (accessed on 12 January 2022).

41. Allen, C.; Dierks, T. *The TLS Protocol Version 1.0*; RFC 2246; Internet Engineering Task Force: Wilmington, DE, USA, 1999. [CrossRef]
42. Dierks, T.; Allen, C. *Rfc5246: The Transport Layer Security (TLS) Protocol Version 1.2*; RFC 5246; RFC, Ed.; Internet Engineering Task Force: Wilmington, DE, USA, 2008.
43. Nir, Y.; Josefsson, S.; Pégourié-Gonnard, M. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier*; RFC 8422; Internet Engineering Task Force: Wilmington, DE, USA, 2018. [CrossRef]
44. Zeus Github. 2021. Available online: https://github.com/Visgean/Zeus/blob/c55a9fa8c8564ec196604a59111708fa8415f020/manual_en.html (accessed on 1 December 2021).
45. Khalife, J.; Hajjar, A.; Diaz-Verdejo, J. A Multilevel Taxonomy and Requirements for an Optimal Traffic-Classification Model. *Int. J. Netw. Manag.* **2014**, *24*, 101–120. [CrossRef]
46. Leroux, S.; Bohez, S.; Maenhaut, P.J.; Meheus, N.; Simoens, P.; Dhoedt, B. Fingerprinting encrypted network traffic types using machine learning. In Proceedings of the NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; pp. 1–5. [CrossRef]
47. Řehůřek, R.; Sojka, P. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 22 May 2010; ELRA: Valletta, Malta, 2010; pp. 45–50. Available online: http://is.muni.cz/publication/884893/en (accessed on 10 October 2021).
48. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
49. Rao, G.; Huang, W.; Feng, Z.; Cong, Q. LSTM with sentence representations for document-level sentiment classification. *Neurocomputing* **2018**, *308*, 49–57. [CrossRef]
50. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*; Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2013; Volume 26.
51. Rhode, M.; Burnap, P.; Jones, K. Early-stage malware prediction using recurrent neural networks. *Comput. Secur.* **2018**, *77*, 578–594. [CrossRef]
52. Xiao, X.; Zhang, S.; Mercaldo, F.; Hu, G.; Sangaiah, A.K. Android malware detection based on system call sequences and LSTM. *Multimed. Tools Appl.* **2019**, *78*, 3979–3999. [CrossRef]
53. Saia, R.; Carta, S.; Recupero, D.R.; Fenu, G.; Stanciu, M. A Discretized Extended Feature Space (DEFS) Model to Improve the Anomaly Detection Performance in Network Intrusion Detection Systems. In Proceedings of the KDIR, Vienna, Austria, 17–19 September 2019; pp. 322–329.
54. Saia, R.; Carta, S.; Recupero, D.R.; Fenu, G. A Feature Space Transformation to Intrusion Detection Systems. In Proceedings of the KDIR, Budapest, Hungary, 2–4 November 2020; pp. 137–144.
55. Tran, L.; Fan, L.; Shahabi, C. Outlier Detection in Non-Stationary Data Streams. In Proceedings of the 31st International Conference on Scientific and Statistical Database Management (SSDBM'19), Santa Cruz, CA, USA, 23–25 July 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 25–36. [CrossRef]
56. Gómez, G.; Kotzias, P.; Dell'Amico, M.; Bilge, L.; Caballero, J. Unsupervised Detection and Clustering of Malicious TLS Flows. *arXiv* **2021**, arXiv:2109.03878.