

Article

# Towards a Near-real-time Protocol Tunneling Detector based on **Machine Learning Techniques**

Filippo Sobrero <sup>1</sup>, Beatrice Clavarezza <sup>1</sup>, Daniele Ucci <sup>1</sup>, and Federica Bisio <sup>1</sup>

- aizoOn Technology Consulting; name.surname@aizoongroup.com
- This paper is an extended version of our paper published in IEEE Symposium Series on Computational Intelligence, 2021.

Abstract: In the very last years, cybersecurity attacks have increased at an unprecedented pace, becoming ever more sophisticated and costly. Their impact has involved both private/public companies and critical infrastructures. At the same time, due to the COVID-19 pandemic, the security perimeters of many organizations expanded, causing an increase of the attack surface exploitable by threat actors through malware and phishing attacks. Given these factors, it is of primary importance to monitor the security perimeter and the events occurring in the monitored network, according to a tested security strategy of detection and response. In this paper, we present a protocol tunneling detector prototype which inspects, in near real time, a company's network traffic using machine learning techniques. Indeed, tunneling attacks allow malicious actors to maximize the time in which their activity remains undetected. The detector monitors unencrypted network flows and extracts features to detect possible occurring attacks and anomalies, by combining machine learning and deep learning. The proposed module can be embedded in any network security monitoring platform able to provide network flow information along with its metadata. The detection capabilities of the implemented prototype have been tested both on benign and malicious datasets. Results show 97.1% overall accuracy and an F1-score equals to 95.6%.

Keywords: passive network analysis; dns tunneling; anomaly detection; machine learning; deep learning

## 1. Introduction

Cybersecurity attacks keep increasing year over year at an unprecedented pace, becoming ever more sophisticated and costly [1,2]. The growth between 2021 and 2022 has resulted in a rise of attacks' volume and impact on both private/public companies and critical infrastructures. Companies comprise digital service providers, public administrations and governments, and include businesses operating in finance and health sectors. In particular, service providers have experimented more than 15% raise in intrusions (infamous has been the case of Solarwinds [3]) compared to 2021 [1], a trend destined to grow in the next years [4]. At the same time, due to the COVID-19 pandemic, the security perimeters of many organizations expanded to cope with the new needs of remote working, causing an increase of the attack surface exploitable by attackers [4]. The European Union Agency for Cybersecurity estimates that more than 10 terabytes of data are stolen monthly from target assets that are made unavailable, until a ransom is payed [1], while IBM calculates that the average cost of these attacks is \$4.54M, arriving up to \$5.12M [2]. On the other hand, malware attacks are still on the rise after the pause recorded during the pandemic and phishing continues to be the common attack vector for initial access [1].

Given these factors, it is of primary importance to monitor the security perimeter and the events occurring in the network, according to a tested security strategy of detection and response. According to Gartner [4], newly proposed solutions should be automated as much as possible, since human errors continue to play a crucial role in most security

Citation: Sobrero, F.; Clavarezza, B.; Near-real-time Protocol Tunneling Detector based on Machine Learning Techniques. J. Cybersecur. Priv. 2023, 1,

Received

1-12. https://doi.org/

Revised:

Accepted:

Published:

Copyright: © 2023 by the authors. Submitted to J. Cybersecur. Priv. for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/).

breaches. In this paper, we present a protocol tunneling detector prototype which inspects – in near real time – a company's network traffic using machine learning. Tunneling techniques allow attackers to create a tunnel through a network by encapsulating traffic inside another protocol [5] and, hence, can be used to let infected machines to contact their corresponding command-and-control centers. Thus by abusing legitimate network traffic protocols, like DNS [6], the attacker maximizes the time in which the infection remains undetected. In this work, we rely on a commercial network security monitoring platform for detecting and investigating potentially malicious or anomalous activities [7-10], but the proposed solution can be easily integrated into any network security monitoring platform able to provide network flow information along with its metadata. The platform we employ is responsible for collecting, processing network flows and dispatching them to one, or more, advanced cybersecurity analytics (ACAs), which are able of recognizing the signals of possible occurring attacks and anomalies. In this scenario, the detector monitors only cleartext protocols, but it works jointly with an ACA responsible for analyzing encrypted traffic [10]. Indeed, while some cleartext protocols are extensively used (i.e., DNS), nowadays the vast majority of Internet traffic is encrypted [11–15]: this enabled threat actors to perform malware campaigns relying on HTTPS for delivering malware and contacting command-and-control centers [16]. Just in 2020, 67% of malware has been delivered via encrypted HTTPS connections [17]. Along with malware delivery, malicious secure communications are used to exfiltrate data and steal sensitive information from private and public companies [18–20]. While the analytics dealing with encrypted traffic has been extensively described in [10], we extend this previous work by backing up secure connection analysis to the monitoring of cleartext protocols. As mentioned before, the latter can be used to discover the abuse of and signal network packets' contents which are not usually observed in the monitored network. The module, presented in this paper, extracts a sequence of N bytes of each single network packet and computes features associated to the collected stream of bytes. Through the combination of deep learning and machine learning, each network packet is assigned to a specific network protocol and, if a connection exhibits anomalies (e.g., an interleaving of different protocols), a security analyst is notified about the discovered inconsistency. More specifically:

- we implement a protocol tunneling detector prototype which analyzes, in near real time, a byte sequence of the packets flowing in the monitored network
- the proposed prototype combines
  - an artificial neural network (ANN), based on [21], that accurately classifies cleartext protocols and identify possible anomalies in network connections
  - a support vector machine that is able to detect compressed/encrypted traffic within unencrypted connections
- we design and implement an input sanitization module, which automatically removes inconsistent data from models' training sets to significantly increase the models' performance

With respect to [21], we changed both the input byte sequences we provide to the ANN and their sizes in bytes (as detailed in Sections 4.1 and 5). The performance of the proposed approach has been evaluated on different datasets that either contain legitimate traffic or simulate DNS tunneling attacks, which are the most common [6]. The obtained overall accuracy of the proposed prototype is 97.1%, along with an F1-score equals to 95.6%.

The rest of the paper is organized as follows: Section 2 discusses related work, while Section 3 introduces basic notions that will be later used to detail the proposed approach (Section 4). The experimental evaluation is reported in Section 5 and, finally, Section 6 concludes the paper.

#### 2. Related Work

Tunneling attacks are a specific typology of network attacks in which an attacker creates a tunnel through a network by encapsulating traffic inside another protocol [5].

This allows the attacker to bypass traditional network security controls and potentially exfiltrate sensitive information. Therefore, as discussed in Section 1, using cleartext network protocols may pose a significant risk when these are abused by malicious actors.

In this context, DNS tunneling represents one of the most common techniques employed for covertly exfiltrating data from a network, by encoding the data in DNS queries and responses. Since this method is becoming increasingly prevalent, a growing body of research aims at detecting and mitigating DNS tunneling attacks. In [22], the authors review detection technologies from a perspective of rule-based and model-based methods with descriptions and analyses of DNS-based tools and their corresponding features, covering detection approaches developed from 2006 to 2020 by means of a comparative analysis.

Latest works in the area of DNS tunneling detection mainly cover three main categories, i.e., detection approaches via machine learning, real-time detection approaches, and detection of DNS tunneling variants (e.g., fast flux [8], and domain generation algorithms (DGAs) [7]).

Regarding the first group, researchers have recently proposed deep learning algorithms such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for detecting DNS tunneling traffic. In [23], the authors develop a novel DNS tunneling detection method employing a Convolutional Neural Network (CNN) to analyze DNS queries and responses and identify DNS tunneling activities. The proposed approach is evaluated using a dataset of real-world DNS traffic and show promising results in detecting DNS tunneling attacks with high accuracy. The work of [24] apply both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for detecting DNS tunneling traffic. The authors show that these algorithms can effectively spot and identify malicious patterns.

The second group of studies has focused on developing real-time detection systems for DNS tunneling. These systems use a combination of several detection techniques to quickly identify malicious DNS traffic [25]. In [26], the authors present an overview of principal countermeasures for DNS tunneling attacks.

Regarding the state of the art of approaches that analyze encrypted communications, it has already been presented in [10]. The approach we present and evaluate in the next sections passively extracts both sequential and statistical features from network flows to detect tunneling attacks in cleartext protocols. As sequential features, we refer to those characteristics obtained from raw flow sequences. Differently from the works previously described in this section, we directly examines, for each packet, a specific sequence of bytes for tunneling detection by using artificial neural network, which are simpler deep learning models.

## 3. Background

#### 3.1. DNS Tunneling

Protocol tunneling is an attack technique commonly used to maximize the time in which the infection remains undetected in a targeted network. In this context, the DNS protocol is usually abused in order to bypass security gateways and, then, to tunnel malware and other data through a client-server model [6]. Figure 1 depicts a typical DNS tunneling scenario: firstly, an attacker registers a malicious domain (e.g., attacker.com) on a C&C center managed by her; at that point, assuming that the attacker has already taken control over a machine inside the targeted network and violated its security perimeter, the infected computer sends a query to the malicious domain. Since DNS requests are typically allowed to move in and out of the network, the query through the DNS resolver reaches the attacker's C&C center, where the tunneling program is installed. This established tunnel can be used either to exfiltrate data and sensitive information or for other malicious purposes.

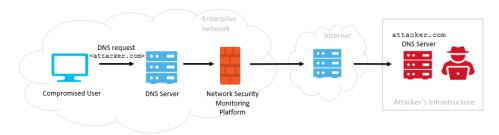


Figure 1. A DNS tunneling example

# 3.2. Support Vector Machines

The original formulation of Support Vector Machines [27] (SVMs) is related to the resolution of supervised tasks with the objective of finding a maximum margin hyperplane that separates two or more classes of observations. In the last years, also one-class SVMs have been shown to represent a suitable choice in the context of anomaly detection [28]. It is defined as a boundary-based anomaly detection method, which modifies the original SVM approach by extending it in order to deal with unlabeled data. Like traditional SVMs, one-class SVMs can also benefit of the so called kernel trick when extended to non-linearly transformed spaces, by defining an appropriate scalar product in the feature space.

## 3.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are deep learning models that have been successfully applied to a vast number of knowledge fields ranging from computing science to arts [29]. They are internally constituted by groups of multiple neurons, which can be thought of as mathematical functions that take one or more inputs. In ANNs, inputs are processed only forward and are multiplied by weights within each neuron and summed up to be then passed to an activation function and become the neuron's output. In general, artificial neural networks consist of three different layers: input, hidden and output; the first layer accepts inputs, while the hidden layers process them to learn the optimum weights. Finally, the output layer produces the result.

#### 4. Protocol tunneling detector

The proposed architecture splits the burden of processing the traffic of a monitored network into two different sub-modules: the first mainly deals with secure connections, while the second inspects unencrypted traffic. As previously discussed, the former analytics has been detailed in [10]. At a glance, it detects possible anomalies occurring during a SSL/TLS handshake between a client, located inside the network monitored by the software platform outlined in Section 1, and an external server. The SSL/TLS detection analytics examines information contained in X.509, SSL, and TLS exchanged protocol messages. Instead, the second module looks for anomalies in unencrypted traffic, regarding the abuse of specific protocols (i.e., tunneling attack techniques). To provide these detection capabilities, this prototype collects a sequence of bytes from each network packet and inspects its content. The content, along with its features, is fed to a testing module, which detects possible anomalies that are signaled to security analysts.

### 4.1. General approach

Figure 2 reports the general structure of the proposed anomaly detection methodology: for each packet observed in the network, the prototype collects a sequence of N bytes belonging to the highest network protocol used in the communication. As an example, in a secure connection which relies on HTTPS, the bytes returned by the extraction process are the ones related to HTTPS, and not to the other lower-layer protocols (e.g., TCP). From the obtained bitstream, we extract the following sequential features (i.e., those features obtained from raw flow sequences):

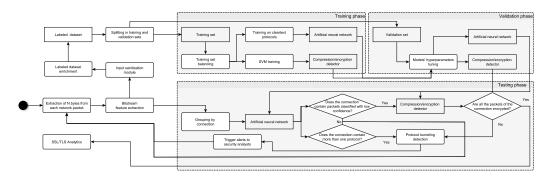


Figure 2. Protocol tunneling detector prototype overview.

- binary representation of collected bytes
- bitstream entropy and *p*-values obtained from statistical tests for random and pseudorandom number generators for cryptographic applications [30]
- statistical properties of the bitstream hexadecimal representation

and we keep the protocol label associated to the bitstream itself. While the binary representation of the *N* bytes is meant to label the protocol of each packet under analysis, the sequential features allow to understand if the packet content is either compressed or encrypted.

After feature extraction, the raw dataset constituted by streams of bits and their corresponding labels is properly sanitized. Indeed, it is easily possible to lightly label the network packets belonging to a connection by simply looking either at the ports or at the connection metadata. However, this labeling may be prone to errors since it either does not take into account potential custom configurations of services (e.g., SMB protocol operating on a port different from 445) or intentional misuse of specific protocols by attackers (as in the case of tunneling). Moreover, cleartext protocols may transfer packets containing compressed data, whose presence could compromise the correct identification of the correct network protocol. Hence, it is paramount to have a refined and clean dataset to let models perform at their best. During our experimental evaluations, we have found out that the accuracy of the trained models, after refining the raw dataset, has significantly increased: 7% for the ANN and 20% for compression/encryption detector. To achieve this performance boost, we have specially implemented an input sanitization module, shown in Figure 3. In this module, we combine unsupervised and supervised support vector machines (SVMs) to clean the raw dataset: first, for each network protocol, we train a one-class SVM both on cleartext and encrypted protocols, in order to filter out outliers from the raw dataset. As an example, in protocols like HTTP and SMB, requests and responses may contain either the content of (compressed) files or other types of information that are not strictly correlated with the specific protocol communication patterns. Thus, in order to exclude these outliers, we build one-class SVMs, one for each different protocol, whose hyperparameters are properly tuned on the raw labeled dataset. Trained models are then applied to identify outliers and remove them from the raw dataset. This refined dataset is then used to train a SVM by applying a one-vs-all classification for detecting packets which are either compressed or encrypted. This single classifier is applied to remove both compressed and encrypted packets from cleartext protocols. It is worth mentioning that, in proxied environments, encrypted packets may be present in connections labeled as HTTP:

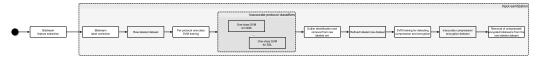


Figure 3. Input sanitization module.

indeed, in these scenarios, also secure communications pass through the proxy, even if these connections are erroneously labeled as HTTP.

This sanitized dataset is then split in training and validation sets to essentially build two different models: (i) an artificial neural network (ANN) able to classify cleartext protocols (e.g., DNS) and (ii) a SVM that is a compression/encryption detector for identifying, respectively, compressed and encrypted packets. As later shown in Section 5, after construction, the training set is considerably unbalanced towards secure protocols. For this reason, we apply SMOTE data augmentation technique [31] to increase the samples of those protocols belonging to minority classes. During the test phase, performed light labeling based on connection's destination port is not taken into account and the resulting bitstreams are grouped by connection. Each packet is given in input to a trained ANN (whose training process is detailed in Section 4.3) and the analytics both verifies if, in the connection, there are some packets that have been classified with low confidence and more than one protocol is present. While in this latter case, the co-presence of multiple protocols might signal a possible tunneling attack, when the ANN classifies packets with low confidence, then, the connection could contain either compressed/encrypted packets or packets whose byte sequences differ from the ones usually observed in the network. To distinguish between these two cases, a more in depth verification is carried out: if the connection is not entirely encrypted, meaning that it is a not a secure communication, the prototype checks if the packets signaled as anomalous (i.e., with low confidence) by the ANN are either encrypted or belongs to another protocol. If either encryption or compression is detected, the anomaly is notified to security analysts. On the other hand, if the entire connection is encrypted, it is collected and stored in a database, periodically accessed in order to retrieve data and metadata about X.509, SSL, and TLS exchanged protocol messages in order to be analyzed by the analytics described in [10].

### 4.2. Feature extraction

As discussed in Section 4.1, sequential features allow to understand if the content of a network packet is either compressed or encrypted. We rely on a statistical package developed by the Information Technology Laboratory at the National Institute of Standards and Technology, containing a set of 15 tests that measure the randomness of a binary sequence [30]. These tests have been designed to provide a first step towards the decision whether or not a generated binary sequence can be used in cryptographic applications, namely if the sequence appears to be randomly generated. In other words, each new bit of the sequence should be unpredictable. From a statistical point of view, each test verifies if the sequence being under analysis is random. This null hypothesis can be either rejected or accepted depending on the statistic value on the data exceeding or not a specific value – called critical value - that is typically far in the tails of a distribution of reference. Test reference distributions used in the NIST tests are the standard normal and the  $\chi^2$  distributions. Even if the statistical package contains 15 tests, we use only 5 of them, because the length Nof the binary sequence we test does not meet the corresponding input size recommendation in [30]. To each sequence we apply the following tests: frequency within a block, longestrun-of-ones in a block, serial test, approximate entropy and cumulative sums. In addition, in our experimental evaluations, we extract some statistical properties and compute the Shannon entropy metrics [32] that, combined with the previously mentioned tests, have shown to improve the overall accuracy of the classification. As statistical properties, the following features are extracted from the corresponding hexadecimal representation h of a bitstream of *N* bytes:

- number of different alphanumeric characters in h normalized over h length
- number of different letters in *h* normalized over *h* length
- longest consecutive sequence of the same character in *h* normalized over *h* length

### 4.3. Input sanitization

For accurately training machine learning models, the training set should be as much "clean" as possible. In Section 4.1, we have already discussed how labeling based on connection metadata could be error prone either due to potential custom configurations of services, intentional misuse of specific protocols by attackers, or network protocols encapsulating compressed data. In addition, during our experimental evaluations, we have observed that in some cases the employed traffic analyzer can assign an empty label or multiple labels to a single network packet. While in the first case, bitstreams with empty labels can be easily discarded for the training phase, in the presence of multi-labels is possible to assign a unique correct label if, among the labels, there exist a protocol that is monitored by the prototype itself. As an example, if the assigned labels are NTLM, GSSAPI, SMB, and DCE\_RPC the resulting label is SMB. For these reasons, the very first step of the sanitization module is to correct the multi-labels associated to bitstreams and discard the empty ones. Then, we train an ensamble of one-class SVMs, one for each protocol (see Figure 3): each different classifier is properly tuned to filter out outliers from the raw dataset. As stated in Section 4.1, HTTP and SMB requests or responses may contain either the content of (compressed) files or other types of information that are not strictly correlated with the specific protocol communication patterns. Trained models are then applied to identify these kind of network packets and remove them from the raw dataset. This preprocessed dataset is used to train a supervised support vector machine, called compression/encryption detector, by applying a one-vs-all classification for detecting packets which are either compressed or encrypted. It is worth noting that all these models are still inaccurate because they are trained on a "dirty" dataset. Hence, to further increase the quality of the labels and obtain the final training set, the compression/encryption detector is fed with cleartext bitstreams to remove possible compressed/encrypted packets from cleartext protocols, as in the case of proxied environments. The result of this sanization process is a dataset which allows to train and validate two accurate models: an artificial neural network for cleartext protocols and a SVM for compressed and encrypted traffic.

#### 4.4. Anomaly detection

During the test phase (see Figure 2), bitstreams are analyzed by the trained ANN. In turn, the ANN flags three different cases as potential tunneling attacks and alerts security analysts when these cases occur: (i) the high confidence detection of more than one protocol in the same connection, (ii) the low confidence detection of one protocol for all the packets in the same connection, and (iii) the labeling, both with high and low confidence, of one or more protocols for the packets belonging to the same connection (as in the case of secure protocols over DNS). As later specified in Section 5, in the ANN, the high/low confidence threshold c can be dinamically set. In any case, the detection of encrypted packets into a cleartext connection generates alert notifications enriched with the information about the presence of encrypted protocol messages. Possibly, notified alerts can be filtered whitelisting source and/or destination IPs to reduce the false positives caused by well-known machines.

Hence, if some packets of the connection are classified with low confidence, the corresponding bitstream's sequential features (refer to Section 4.2) are given in input to the compression/encryption detector. If all the packets contained in the connection are encrypted, then the connection and its corresponding metadata are given in input to the SSL/TLS analytics for further scrutiny [10]. On the contrary, if the connection contains some compressed/encrypted packets or none of them, depending on the protocol, the connection is considered anamalous. Indeed, it is worth noting that not in all cases the combination of two different protocols is a signal of an occurring attack: as already discussed, SMB and HTTP connections can contain protocol-specific messages along with compressed data; however, DNS messages interleaved with other protocols are highly suspicious.

**Figure 4.** Packet distribution for each network protocol, before and after balancing.

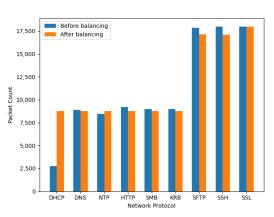


Table 1. Benign test set summary.

Statistics	Count [(%)]
DNS packets	30,669 (1.10%)
SMB packets	65,944 (2.35%)
HTTP packets	262 (0.01%)
NTP packets	46 (0.002%)
DHCP packets	20 (0.001%)
KRB packets	741 (0.03%)
SFTP packets	69, 158 (2.46%)
Not labeled packets	61,552 (2.20%)
SSL packets	2,571,608(91.84%)
Distinct connections	51,459
Distinct source machines	758
Distinct dest. machines	1,566

# 5. Experimental evaluation

The proposed prototype and the experimental evaluations have been, respectively, implemented and performed in Python. The size N we have chosen for the byte sequences, extracted from network packets, is 52 bytes. More in detail, we retrieve the first 64 bytes of the payload of each TCP/UDP packet, from which we remove the first 12B: indeed, a preliminary evaluation has shown that these first bytes had a very low variance in their binary representation among different packets of the same protocol. The specific selection of the byte sequence to extract has improved the accuracy of the trained neural network, increasing its anomaly detection capabilities.

For the experimental evaluation of the proposed prototype, we collected both benign and malicious datasets. The benign communication dataset contains a subset of legitimate traffic observed in a real corporate network during a period of about 2 days. From this initial dataset, we sample connections to start building the models' training sets and the dataset that will be used for testing. Figure 4 summarizes general statistics about the collected training set in terms of packets, before and after sanitization. Next to it, Table 1 reports how the test set of legitimate network traffic is characterized. The sanitization process makes the training set, which is obviously unbalanced towards encrypted protocols, balanced: indeed, after sanitization, the number of packets belonging to respectively cleartext and secure protocols is almost even. It is worth noting that the balanced training set for the ANN, containing DHCP, DNS, NTP, HTTP and SMB packets, comprises also data belonging to KRB network protocol (i.e., encrypted): our experimental evaluations have shown that during the test phase, the neural network performs better when it is trained also with encrypted byte sequences. As ANN, we use a Keras¹ sequential model with 3 hidden layers.

Table 2. Support vector machine hyperparameter settings.

Model	Kernel	γ	ν	t	С
DHCP one-class SVM	RBF	0.7	0.03	0.77	_
DNS one-class SVM	RBF	0.7	0.03	0.77	_
NTP one-class SVM	RBF	0.03	0.1	0.92	_
HTTP one-class SVM	RBF	0.08	0.07	0.91	_
SMB one-class SVM	RBF	0.06	0.08	0.77	_
KRB one-class SVM	RBF	0.04	0.05	0.97	_
SFTP one-class SVM	RBF	0.7	0.05	0.97	_
SSH one-class SVM	RBF	0.7	0.05	0.97	_
SSL one-class SVM	RBF	0.0001	0.0028	0.97	_
Compression/encryption detector	RBF	0.01	_	_	100

The input layer accepts 416 bits (i.e., 52B) and the output layer consists of 6 neurons, one for each cleartext protocol and KRB. Regarding SVMs, we rely on the open-source library scikit-learn<sup>2</sup>. For completeness, we report in Table 2 the hyperparameters we have used to train the different SVMs in the sanitization module and, in addition, the hyperparameters we obtained by tuning the compression/encryption detector in the validation phase. It is worth mentioning that the parameter t, in Table 2, is used for each protocol one-class SVM as a threshold to filter only those outliers which have a Shannon entropy greater than t. The intuition behind this filtering is that byte sequences having high entropy do not specifically belong to cleartext protocol communications and, thus, they have to be discarded from the training set.

On the other hand, malicious datasets are constituted by packet captures (PCAPs) shared by [33], [34], and [35]. The former dataset contains 3 different types of DNS tunnels generated in a controlled environment, whose size are approximately 750MB each. Tunneled data contains respectively SFTP, SSH, and Telnet malicious protocol messages. Each sample is made up of one single connection containing millions of DNS packets. It is reasonable to note that such connections would either easily stand out to security analysts or be simply detectable through well-known statistical approaches (e.g., outlier detection). Subsequently, as stated in Section 4.1, our approach groups data by connection and, therefore, a single malicious packet is enough to flag the entire connection as anomalous. For the above reasons, we have decided to split each sample in n different connections, composed by approximately 5,000 DNS packets each. The size of the split, reported in Table 3, has been chosen according to the size of the connections monitored in the controlled environment. The second malicious dataset, instead, was born by the collaboration between the Bell Canada company's Cyber Threat Intelligence group and the Canadian Institute for Cybersecurity. In this dataset, we take only into account DNS packets that, in their payloads, contain exfiltrations of various types of files and we discard legitimate traffic. Moreover, it is worth mentioning that all the packets contained in [34] have been truncated at capture time to 96B; this has required a slightly different approach to test these samples that will be discussed later in this Section. Finally, [35] is a single packet capture to test detection and alerting capabilities of Packetbeat<sup>3</sup>, Elastic's network packet analyzer. Malicious packet captures have been injected into the network security platform in order to be processed and analyzed as ordinary traffic. Table 3 reports a summary of the malicious assembled datasets: for each PCAP, we list the number of packets in the capture and which ones of these packets have been successfully processed by the platform's network analyzer (i.e., those packets whose size is greater or equal than 64B); in addition, Table 3 depicts the number of connections in the PCAP and how many of them have been identified as protocol tunneling attacks (i.e., true positives TP). Finally, the true positive rate TPR of the proposed detector is reported for each packet capture. Analogously, Table 4 reports the same information contained in Table 3, but with reference to the test set described in Table 1. Being legitimate traffic, the last two columns reports the connections mistakenly

**Table 3.** Malicious test set summary.

Tunnel type	No. of PCAP packets	No. of processed PCAP packets	No. of connections	TP	TPR(%)
Telnet over DNS tunnel [33]	2.4M	2.2M	457	457	100%
SFTP over DNS tunnel [33]	2M	1M	209	209	100%
SSH over DNS tunnel [33]	2.8M	2.7M	545	545	100%
Light file exfiltration [34]	187,500	102,000	7,617	7,361	96.6%
Heavy file exfiltration [34]	1.34M	765,000	43,964	42,441	96.5%
Data exfiltration over Iodine DNS tunnel [35]	438	247	1	1	100%

<sup>1</sup> Keras library: https://keras.io/

Scikit-learn library: https://scikit-learn.org/stable/index.html

Elastic Packetbeat: https://www.elastic.co/beats/packetbeat

Table 4. Benign test set summary.

Dataset	No. of PCAP packets	No. of processed PCAP packets	No. of connections	FP	FPR(%)
Legitimate traffic	5.4M	2.8M	51,459	2,966	5.8%

classified as tunnels (i.e., false positives FP) and the false positive rate FPR. The results of the evaluation, reported in Table 3 and 4, show a false positive rate and a true positive rate, respectively, equal to 5.8% and 96.6%. The overall accuracy of the proposed prototype is 97.1%, while the resulting F1-score is 95.6%.

We conclude this section by discussing how we slightly modified the proposed approach, used in the other datasets, to be compliant with [34]. Indeed, the DNS packets contained in this dataset have been truncated during traffic acquisition, resulting in byte sequences that do not have the same length. In order to solve this dataset generation problem, we reduced all the DNS packets to a common length of 44B, discarding the shorter byte sequences and trimming the longer ones. The result of the filtering operation is clearly shown in Table 3, where the number of processed PCAP packets is more than 54% less than the ones received in input by the traffic analyzer.

Being the bitstream lengths different from the datasets [33] and [35], we have retrained our ANN to be fed with 44B sequences. On the contrary, we have maintained for this evaluation the same hyperparameters for the different SVMs, reported in Table 2, and the same threshold c, used in the other experiments. In particular, for all our experimental evaluations, we set c to 0.999999 in order to maximize the algorithm sensitivity and to compensate for the lesser information provided by the processing of [34]. This explains why, in the experimental evaluations, we were not able to achieve a very low false positive rate, as shown in Table 4. However, in context where a high number of false positives could be detrimental, c can be tuned to obtain a 0.5% false positive rate or lesser without losing accuracy on protocol tunneling attacks.

#### 6. Conclusion

In this paper, we proposed a software prototype for detecting protocol tunneling attacks in a monitored network. Relying on a combination of machine learning and deep learning techniques, the proposed solution identifies anomalous connections that deviate from the ones usually established in the network. Since machine learning models are built based only on legitimate traffic, the proposed solution is therefore able to deal with zero-day attacks, because malicious traffic is not required for the learning phase. The prototype has been evaluated both on malicious and benign datasets: results show a very high accuracy in detecting malicious samples and a low false positive rate on legitimate traffic.

As future work, we plan to optimize the algorithm through a deeper analysis on how the choice of bytestream length affects the computational time, in order to find a value which guarantees the best trade-off between efficiency and accuracy. Indeed, in this work, we mainly focused on accuracy. Secondly, we envision that the engineered prototype will be integrated into a streaming architecture, where new data are analyzed by the proposed prototype as soon as they are collected to provide the fastest possible response. In parallel, the models of the protocol tunneling detector are periodically retrained to keep them up to data with possible deviations from the usual behaviour of the monitored network. Finally, in Section 4.4 we outlined the usage of an IP whitelisting filter. Once in production, the prototype can be easily extended with other security- analyst-defined whitelists as, for example, domain or autonomous system whitelists. This will allow the analysts to apply domain-specific knowledge of the monitored network to the protocol tunneling detector, further reducing potential false positives and improving overall performance.

# References

- ENISA Threat Landscape 2022. [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022. Accessed 06 Feb. 2023.
- 2. Cost of a data breach 2022. A million-dollar race to detect and respond. [Online]. Available: https://www.ibm.com/reports/data-breach. Accessed 06 Feb. 2023.
- 3. The SolarWinds Cyber-Attack: What You Need to Know. [Online]. Available: https://www.cisecurity.org/solarwinds. Accessed 06 Feb. 2023.
- 4. 7 Top Trends in Cybersecurity for 2022. [Online]. Available: https://www.gartner.com/en/articles/7-top-trends-in-cybersecurity-for-2022. Accessed 06 Feb. 2023.
- 5. Protocol Tunneling. [Online]. Available: https://attack.mitre.org/\techniques/T1572/. Accessed 06 Feb. 2023.
- 6. Encrypted Traffic Analysis. [Online]. Available: https://www.enisa.europa.eu/publications/encrypted-traffic-analysis. Accessed 06 Feb. 2023.
- 7. Bisio, F.; Saeli, S.; Lombardo, P.; Bernardi, D.; Perotti, A.; Massa, D. Real-time behavioral DGA detection through machine learning. In Proceedings of the International Carnahan Conference on Security Technology (ICCST). IEEE, 2017, pp. 1–6. https://doi.org/10.1109/ccst.2017.8167790.
- 8. Lombardo, P.; Saeli, S.; Bisio, F.; Bernardi, D.; Massa, D. Fast Flux Service Network Detection via Data Mining on Passive DNS Traffic. In Proceedings of the International Conference on Information Security. Springer, 2018, pp. 463–480. https://doi.org/10.1007/978-3-319-99136-8\_25.
- 9. Saeli, S.; Bisio, F.; Lombardo, P.; Massa, D. DNS Covert Channel Detection via Behavioral Analysis: a Machine Learning Approach. In Proceedings of the International Conference on Malicious and Unwanted Software (MALWARE), 2020, pp. 46–55.
- 10. Ucci, D.; Sobrero, F.; Bisio, F.; Zorzino, M. Near-real-time Anomaly Detection in Encrypted Traffic using Machine Learning Techniques. In Proceedings of the IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021. IEEE, 2021, pp. 1–8. https://doi.org/10.1109/SSCI50451.2021.9659955.
- 11. Felt, A.P.; Barnes, R.; King, A.; Palmer, C.; Bentzel, C.; Tabriz, P. Measuring HTTPS Adoption on the Web. In Proceedings of the Proceedings of the 26th USENIX Conference on Security Symposium, 2017, p. 1323–1338.
- 12. The Relevance of Network Security in an Encrypted World. [Online]. Available: https://blogs.vmware.com/networkvirtualization/2020/09/network-security-encrypted.html/. Accessed 06 Feb. 2023.
- 13. Encryption, Privacy in the Internet Trends Report. [Online]. Available: https://duo.com/decipher/encryption-privacy-in-the-internet-trends-report. Accessed 06 Feb. 2023.
- 14. Keeping Up With the Performance Demands of Encrypted Web Traffic. [Online]. Available: https://www.fortinet.com/blog/industry-trends/keeping-up-with-performance-demands-of-encrypted-web-traffic. Accessed 06 Feb. 2023.
- 15. Google Transparency Report: HTTPS encryption on the web. [Online]. Available: https://transparencyreport.google.com/https/overview?hl=en. Accessed 06 Feb. 2023.
- 16. Cisco Encrypted Traffic Analytics. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.pdf. Accessed 06 Feb. 2023.
- 17. ENISA Threat Landscape Malware. [Online]. Available: https://www.enisa.europa.eu/publications/malware/at\_download/fullReport. Accessed 06 Feb. 2023.
- 18. Korolov, M. Cyber Security Review. Treasury & Risk 2012.
- 19. Taylor, R.W.; Fritsch, E.J.; Liederbach, J. Digital crime and digital terrorism; Prentice Hall Press, 2014.
- 20. Yadav, T.; Mallari, R.A. Technical aspects of cyber kill chain. arXiv preprint arXiv:1606.03184 2016.
- 21. Applying Machine Learning to Network Anomalies. [Online]. Available: https://www.youtube.com/watch?v=qOfgNd-qijI. Accessed 06 Feb. 2023.
- 22. Wang, Y.; Zhou, A.; Liao, S.; Zheng, R.; Hu, R.; Zhang, L. A comprehensive survey on DNS tunnel detection. *Computer Networks* **2021**, *197*, 108322.
- 23. Palau, F.; Catania, C.; Guerra, J.; Garcia, S.; Rigaki, M. DNS tunneling: A deep learning based lexicographical detection approach. arXiv preprint arXiv:2006.06122 2020.
- 24. Zhang, J.; Yang, L.; Yu, S.; Ma, J. A DNS tunneling detection method based on deep learning models to prevent data exfiltration. In Proceedings of the Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13. Springer, 2019, pp. 520–535.
- 25. Ahmed, J.; Gharakheili, H.H.; Raza, Q.; Russell, C.; Sivaraman, V. Real-time detection of DNS exfiltration and tunneling from enterprise networks. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2019, pp. 649–653.
- Rajendran, B.; et al. DNS amplification & DNS tunneling attacks simulation, detection and mitigation approaches. In Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT). IEEE, 2020, pp. 230–236.
- 27. Vapnik, V. The nature of statistical learning theory; Springer science & business media, 2013.
- 28. Swersky, L.; Marques, H.O.; Sander, J.; Campello, R.J.; Zimek, A. On the evaluation of outlier detection and one-class classification methods. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2016, pp. 1–10. https://doi.org/10.1109/dsaa.2016.8.

- 29. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938.
- 30. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf. Accessed 06 Feb. 2023.
- 31. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority over-Sampling Technique. *J. Artif. Int. Res.* **2002**, *16*, 321–357.
- 32. Shannon, C.E. A Mathematical Theory of Communication. The Bell System Technical Journal 1948, 27, 379-423.
- 33. Berg, A.; Forsberg, D. Identifying DNS-tunneled traffic with predictive models. CoRR 2019, abs/1906.11246, [1906.11246].
- 34. Mahdavifar, S.; Hanafy Salem, A.; Victor, P.; Razavi, A.H.; Garzon, M.; Hellberg, N.; Lashkari, A.H. Lightweight Hybrid Detection of Data Exfiltration Using DNS Based on Machine Learning. In Proceedings of the 2021 the 11th International Conference on Communication and Network Security; Association for Computing Machinery: New York, NY, USA, 2022; ICCNS 2021, p. 80–86. https://doi.org/10.1145/3507509.3507520.
- 35. Iodine DNS Tunnel. [Online]. Available: https://github.com/elastic/examples/blob/master/Security%20Analytics/dns\_tunnel\_detection/dns-tunnel-iodine.pcap. Accessed 06 Feb. 2023.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.