Automatic Detection of Various Malicious Traffic Using Side Channel Features on TCP Packets

George Stergiopoulos, Alexander Talavari, Evangelos Bitsikas, and Dimitris Gritzalis(✉)

Information Security and Critical Infrastructure Protection (INFOSEC) Laboratory, Department of Informatics, Athens University of Economics and Business, Athens, Greece
{geostergiop, dgrit}@aueb.gr, alex.talavari@gmail.com

Abstract. Modern intrusion detection systems struggle to detect advanced, custom attacks against most vectors; from web application injections to malware reverse connections with encrypted traffic. Current solutions mostly utilize complex patterns or behavioral analytics on software, user actions and services historical data together with traffic analysis, in an effort to detect specific types of attacks. Still, false positives and negatives plague such systems. Behavioral-based security solutions provides good results but need large amounts of time and data to train (often spanning months or even years of surveillance) - especially when encryption comes into play. In this paper, we present a network-traffic monitoring system that implements a detection method using machine learning over side channel characteristics of TCP/IP packets and not deep packet inspection, user analytics or binary analysis. We were able to efficiently distinguish normal from malicious traffic over a wide range of attacks with a true positive detection rate of about 94%. Few similar efforts have been made for the classification of malicious traffic but existing methods rely on complex feature selection and deep packet analysis to achieve similar (or worse) detection rates. Most focus on encrypted malware traffic. We manage to distinguish malicious from normal traffic in a wide range of different types of attacks (e.g. unencrypted and encrypted malware traffic and/or shellcode connections, website defacing attacks, ransomware downloaded cryptolocker attacks, etc.) using only few side channel packet characteristics and we achieve similar or better overall detection rates from similar detection systems. We compare seven different machine learning algorithms on multiple traffic sets to produce the best possible results. We use less features than other proposed solutions and thus require less data and achieve short times during training and classification.

Keywords: Malware traffic · Malware detection · Machine learning
Defacement · SVR · Neural networks · CART · Botnet · Reverse shells
Trojan

J. Lopez et al. (Eds.): ESORICS 2018, LNCS 11098, pp. 346–362, 2018.
https://doi.org/10.1007/978-3-319-99073-6_17

1 Introduction

One of the most serious open issues in securing today's IT networks is the inability of current solutions (i.e. intrusion detection and prevention systems (IDPS), antivirus etc.) to detect advanced and often meticulously custom ongoing malicious attacks. Such attacks are often tailored to specific victims and sophisticated code is used that is not currently known by the security community. Security companies need to update their security solutions constantly, only to often fail to detect "0-day" malware and custom attacks against all vectors, from injecting commands to websites to detecting encrypted malware traffic. Also, some attacks such web application defacements utilize custom string and hex coding of malicious data that cannot be efficiently detected.

Current solutions utilize either complex pattern matching or behavioral analytics on software, users and services in an effort to classify ongoing network events as suspicious. Still, false positives and negatives plague signature-based security software. On the other hand, behavioral based models have better detection rates but require large periods of time to effectively monitor users and systems and/or big datasets describing multiple scenarios to be able to accurately detect malicious traffic [22]; often unrealistically large amounts of data and time. On top of these, modern malware uses encrypted traffic or inject themselves to whitelisted apps (e.g. browsers) to communicate with C&C servers and exfiltrate data, which makes behavioral analysis and pattern matching even less successful over network traffic.

1.1 Contribution

We present a network traffic monitoring system that implements machine learning over network captures to distinguish normal from multiple types of malicious TCP/IP traffic. A few similar efforts have been made for classification of some types of malicious traffic (e.g. encrypted malware traffic), yet existing methods rely mostly on complex feature selection and/or large datasets. Overall, the main contributions of this article are summarized as follows:

1. We manage to simultaneously detect multiple types of malicious traffic (unencrypted and encrypted malware traffic and/or shellcode connections, website defacing attacks, ransomware downloaded cryptolocker attack, etc.) using a few side channel characteristics of TCP packets and not complex features or deep packet inspection.
2. We achieve the same or better overall detection rates with similar detection systems while using less features (e.g. no TLS, certification features or deep packet inspection). Consequently, our system requires less training and classification.
3. We test and compare seven (7) different machine learning algorithms over millions of network captures spanning 8 GB of network. Experiments showed that decision tree classifiers have good detection rates with side channel features but may be prone false positives with packet crafting feature values and consequently trick our classifier. Using KNN seems to greatly reduce this. However, we should avoid neural networks as preliminary tests with neural networks show that they offer

worse detection rates while requiring way larger amounts of time and data for training. Also, in some instances, neural networks seem prone to biases.
4. Our system provides faster training and classification than other detection systems during offline training and testing due to its smaller feature set. The use of side channel features greatly reduces the size of traffic data that needs to be analyzed for training or detecting of various types of malicious traffic and not only encrypted malware, which seems to dominate current research projects.

Specifically, we first select an optimal set of features from raw features extracted from TCP traffic packets. We minimize the number of features used in similar previous research by only utilizing the ones that refer to side channel characteristics; i.e. packet size ratios and timing events between packets. We provide experiments on real-world malicious traffic data from three different datasets, namely FIRST 2015 [5], Milicenso [6] and CTU13-1 [7]) to demonstrate the effectiveness and efficiency of our approach over multiple types of malicious traffic, even with fewer, selected features.

Section 2 presents related work concerning malicious network traffic and similar classification approaches and argues about the differences with our presented system. Section 3 describes the datasets utilized in the current project and presents our data sanitization process. Section 4 presents the detection methodology implemented in the proposed system. Section 5 describes our experimental results, while Sect. 6 discusses further improvements and potential future work.

2 Related Work

Most mainstream approaches to detecting malicious traffic mostly rely on heuristic analysis of packets, payloads and session trends (like packets per min) along with botnet architecture [8, 9, 10]. Others rely on statistical analysis for classifying various types of traffic [20].

Our approach is similar to [13, 27]. In [13], researchers utilize some of the same features as we do to extract information from the physical aspects of the network traffic. They too utilize machine learning but focus on OSI layer 7 features to distinguish between malicious and normal encrypted traffic. Thus, significant differences exist. The main differences of our work with [13, 27] are the following: (a) We do not restrict our machine learning and detection system only to encrypted traffic but try to achieve similar (or even better) detection rates without distinguishing between different malicious traffic, (b) we provide full payload analysis per packet and in relation to previous packets sent, whereas researchers in [13] researchers analyze tuples that check payload sizes for entire originator-responder sessions, (c) we minimize selected features by only using the ones that refer to side channel characteristics, while achieving better results, and (d) we do not aim to only understand and distinguish malicious encrypted traffic from malware but extend this to multiple types of both encrypted and unencrypted malicious traffic, ranging from defacing attacks, reverse shells, encrypted connections etc.

Cisco published a white paper concerning new advancements in detecting malicious traffic using similar side channel features [27]. Cisco utilizes similar types of data

elements or telemetry that are independent of protocol details, such as the lengths and arrival times of messages within a flow. Their technology supports various Cisco routers and switches to perform detection of malicious traffic in network sessions that utilize the Transport Layer Security (TLS) protocol.

In summary, we manage similar performance and smaller datasets utilizing only five features on side channel characteristics, twenty two (22) features less than [13] and four (4) less than Cisco [27].

Authors in [21] also use malicious HTTPS traffic to train neural networks and sequence classification to build a system capable of detecting malware traffic over encrypted connections. Similarities with our work is that we use features to train a machine learning algorithm. The difference with our work is that: (a) we are able to detect multiple types of malicious traffic and not only encrypted malware traffic and (b) we utilize less data (and corresponding domain features) while achieving better and faster results, albeit not only on encrypted traffic but on a dataset consisting of 200 K traffic samples of different malicious traffic flows.

Using CART and KNN decision algorithms instead of neural networks, we can achieve faster classification once the system is trained and have a more interpretable model to detect hidden interconnections of traffic features. On the other hand, neural networks might be more accurate (although our preliminary results do not support this), provided there is enough training data, although they can be prone to over-fitting as well; this is why another reason why we tested other algorithms more suitable to unknown dataset characteristics.

Other approaches in analyzing encrypted HTTPS traffic are few [18, 19]. Most of them focus on identifying target malware/botnet servers [19] or web servers contacted [18], instead of understanding malicious traffic of various types.

The following publications are worth mentioning although they differentiate and either utilize different technologies to achieve similar goals, or aim to analyze different aspects of network traffic albeit with similar algorithms. Authors in [11, 17] utilize signal processing techniques (e.g., Principal Component Analysis (PCA)) to create aggregates of traffic and payload inspection data, in an effort to detect anomalous changes to network flows [14]. They utilize a distance metric to understand network-change patterns in traffic. Lakhina et al. [15] modelled network flows as combinations of eigen flows to distinguish between short-lived traffic bursts, trends, noise, or normal traffic. Terrell et al. [16] grouped network traces into time-series and selected features, such as the entropy of the packet and port numbers, to detect traffic anomalies. While these approaches are based on models of malware behavior (not unlike signature-based intrusion detection), our approach seeks to identify important features on the physical characteristics of malicious network sessions and utilize them to train machine learning algorithms. This way, we increase the detection rate by (a) not relying on instances of malware traffic to understand future malware and (b) by creating a trained model that predicts the value of a network TCP sessions based on network values of several input (or independent variables). Our approach is nonparametric, therefore it does not rely on data belonging to a particular type of distribution. Also, it can utilize variables multiple times in different decision analyses, thus uncovering complex interdependencies between sets of variables [12].

The selected machine learning algorithm and relevant network features enhance malicious traffic detection in both encrypted and unencrypted traffic, ranging from a series of different malicious types such as botnets, defacement attacks, reverse shells, Trojans, etc. To our knowledge, no other prototype is able to accomplish this.

3 Datasets

We utilized datasets with both malicious and normal traffic from various sources to build our database. Selecting useful and balanced datasets was vital in order to be certain that the achieved detection rates correspond to real-world capabilities. Datasets are public and contain traffic of real malware, defacing attacks, reverse shells and software exploitation attacks along with normal traffic.

To guarantee the malicious traffic data quality and validate our detection rates, we opted to use malicious traffic from datasets built from major companies and institutions. The datasets used both for training and testing our system are the following:

- FIRST 2015 [5]: Dataset created for the needs of a hands-on lab for Network Forensics. It is a collection of 4.4 GB pcap files containing normal as well malicious traffic. Traffic is composed from Reverse Shell shellcode connections, website defacing attacks, ransomware downloaded attack cryptolocker and a command and conquer exploit attack (C2) over SSL that takes over the victim machine.
- Milicenso [6]: Dataset containing normal and malware traffic for the Ponmocup Malware. It contains malicious traffic from a malware/trojan that connects the victim PC on a botnet.
- CTU13-1 [7]: Dataset containing Botnet Traffic of the Neris Botnet. All traffic is mostly encrypted botnet traffic, because the normal traffic that was captured at the same time is not public.

Dataset traffic was included in pcap and pcapng files containing captured packets. Packets from the FIRST 2015 were pcap files captured using Snort [24], whereas Milicenso and CTU13-1 datasets were raw tcpdumps of monitored connections. Traffic flows are captured using methods like WireShark [23], Snort [24], or raw TCP dumps.

3.1 Threat Model

Aforementioned datasets contain malicious traffic that covers a range of different attack scenarios.

First 2015 dataset

- **Website defacement attack** (FrogSquad defacement, First 2015). Attackers uploaded a FrogSquad image to: www.pwned.se/skyblue/fr.jpg.
- **Webshell (PHP backdoor)** on infected web server. FrogSquad sent multiple commands using cm0 backdoor. FrogSquad traffic from later come back, from the same class CIP network.
- **Spear Phising email attack.** APT4711 spear phishing email to Krusty (192.168.0.54). From First2015 [5]: “Krusty uses SSL encrypted IMAP (TCP 993)

towards imap.google.com, so we cannot inspect the contents of his email. However, we do know that Krusty opened the attachment at 10.35.36 UTC, which caused a Command-and-control (C2) software do be downloaded".

- **Malware traffic (reverse shell).**

CTU-13-1 dataset

As mentioned on the CTU-13 manual [7], "The CTU-13 is a dataset of botnet traffic that was captured in the CTU University, Czech Republic, in 2011. The goal of the dataset was to have a large capture of real botnet traffic mixed with normal traffic and background traffic". Traffic selected for our experiments contain several botnet scenarios with more than 160 different malware samples. Scenarios include:

Click Fraud attacks. The bot sent spam, connected to an HTTP CC, and use HTTP to do some ClickFraud.
IRC communication for spam and clickjacking. Neris botnet that run for 6.15 h in a University network. The botnet used an HTTP based C&C channel and not an IRC C&C channel as it was erroneously reported before. Send SPAM and perform click-fraud using some advertisement services.
Malware traffic. The machine was successfully infected with POST requests. Malware connect to command & control (CnC) server using a raw TCP connection.
Encrypted malware traffic. HTTPS and SSH traffic.
UDP and ICMP DDoS.
Trickbot banking Trojan. Trickbot (Trojan.Trickybot) C2 over HTTPS. • Uses Scheduled Tasks to re-run the main binary every few minutes and connect using SSL port. Most – but not all –communication with C&C is encrypted.

The dataset contains Background, Botnet, C&C Channels and Normal botnent traffic flows.

Milicenso dataset

This dataset contains traffic from live use of the Ponmocup malware/Trojan infection and communication traffic. Traffic contains:

Redirect domains, kritikaa.ilanes.com 178.211.33.205
Malware download, ml.buymeaslut.com 82.211.45.82
C2 /phone home, intohave.com 64.179.44.188 (DNS request only).

3.2 Data Validation

Since the dataset is mainly comprised of malicious traffic captures, the first step was to balance the amount of normal and malicious traffic. To ensure the quality of the dataset used, we opted for two things:

- Provide as much 'normal' traffic as malicious one per session analyzed. Normal traffic originates from different types of services and network communications. For each setup, referenced datasets provide more information [5, 6, 7].
- Increase the amount of encrypted malware traffic to approach data sizes of other attacks. FIRST [5] dataset encrypted malicious traffic was noticeably less than other

forms of malicious traffic. To achieve this, we utilized the Trickbot network to obtain captures from CTU-13 extended dataset pcaps [7].

Since our task is not to distinguish between specific types of malware but rather a high-level detection of any type of malicious traffic, the notion of a biased dataset in terms of having the same amount of malicious traffic for each type of attack is not as relevant as in [13]. Also, our dataset is comprised of real traffic data from multiple types of attacks, either from captured malware, capture-the-flag hacking events or similar environments. Thus, a potential imbalance of malicious traffic sub-classes within each attack) is a real world representation and needs not to be tampered with (e.g. the no of packets corresponding to malware reverse connection in comparison with the no of packets corresponding to service exploitation during the same attack. Also, the amount of data for all types of attacks is big enough to exclude unrealistic biases in data. Thus, there is no need for rebalancing the classes of malicious traffic, the only exception being the addition of extra encrypted malicious traffic from different case studies, due to the small size of network captures in comparison with the rest.

We opt to report detection results using accuracy, precision, recall and f1 score in both mixed (shuffled) and ordered dataset samples. These are popular metrics and indicators of the overall performance of the prototype [3] and are used in multiple similar research projects [13, 18].

4 Detection Methodology

4.1 Problem Definition

Given a TCP/IP network traffic flow, our system aims to sample and classify each connection as malicious (i.e. produced by malicious events such as a web attack or malware) or normal. Essentially, the system is comprised of two parts: traffic flow side channel feature selection and network traffic classification.

Side channel feature selection: The first task is to choose correct, descriptive features of TCP traffic that do not refer to the content of a packet, but rather to the physical characteristics, such as time ratio between packet sending, size of payload etc.

Traffic classification: The second task is to use the selected features to classify new traffic streams as malicious or normal. We do not aim to distinguish between types of malicious traffic. It is our belief that human interaction and digital forensics will always provide better solutions in dissecting security events. Instead, our system aims to warn against any potential malicious traffic for response teams to take action.

4.2 Feature Selection

In this subsection we discuss the features we selected to feed into our Machine-Learning algorithms and the rationale of the proposed system. We use features based on side channel characteristics of TCP traffic to analyze packet-to-packet sequences inside network sessions.

It is known that for any set of features, “there will be a fundamental limit to the kind of determinations a NIDS can develop from them” [31]. Choosing a correct set of

features must always take into account the diversity of normal as well as malicious traffic. A good approach is to examine the invariance of features in diverse malicious traffic scenarios [31]. To this end, we opted to base our feature selection on previous publications [13, 26, 27] that utilized similar side-channel packet features for similar purposes. Authors in [13] and Cisco [27] made extensive tests and concluded in similar albeit quite larger feature sets than us. Authors in [26] had previously used a subset of features also found in [13, 27], albeit for different purposes (i.e. to leak sensitive information from web application content).

Our intuition was that, the intersection of these features sets could minimize the features needed for the detection of malicious traffic, while at the same time achieve the same results. Also we believed that the same feature set could expand potential malicious traffic detection beyond encrypted malicious traffic; which was the focus of [13, 27]. Thankfully, we found that these types of features are enough to identify malicious traffic. Since these features do not require complex aggregation of information, the runtime footprint is small and the system can be easily adapted to analyze traffic in real-time. Overall, we opt for five features on side channel characteristics, twenty two (22) features less than [13] and four (4) less than Cisco [27].

Packet Size (Ps): Every connection is defined by the packets exchanged between a sender and a receiver. Packet size is known to be good both for predicting the type of connection and protocols used [25]. For that reason this is a basic feature of our project.

Payload Size (PAs): It is a feature that defines a packet. The payload is the heart of any malicious traffic. In TCP, the payload is enclosed in the TCP Data Segment. Research has shown that side channel analysis of payload sizes can be used as a feature for information leakage [26].

Payload Ratio (Pr): It refers to the ratio of the payload size to the total packet size. Malicious traffic can exhibit similar patterns concerning content ratios, so we opted to include this as a basic feature. The formula is shown below, where PAs refers to the payload size and Ps refers to the packet size

$$Pr = \frac{PAs}{Ps} \tag{1}$$

Ratio to Previous Packet (Rpp): We noticed that, when malicious traffic flows inside the network, the packets are sequential and often exhibit specific trends in size. This can be used for fingerprinting malicious traffic. By comparing two packets in a row that belong in the same session, we can get the ratio to the previous packet. The value defaults to 0 for the first packet of the session. The formula is shown below.

$$Rpp = \frac{Pp}{PPs} \tag{2}$$

where Pp refers to the current packet size and PPs to the previous packet size in the same session.

Time Difference (Td): The time difference between a packet and the previous packet of a session can be used to fingerprint malicious traffic. The value defaults to 0 for the first packet of a session. The formula is:

$$Td = Pt - PPt \tag{3}$$

Pt refers to Packet time and PPt to Previous Packet time. Both times refer to how long it took for a packet to be delivered.

Flag: A simple label that classifies the packet as either 0 or 1, where 0 stands for normal traffic and 1 for malware traffic.

Fig. 1. Workflow of the system from network capture to classification (training & validation)

4.3 Traffic Classification

Our proposed system utilizes offline training to train a machine learning algorithm. The offline analysis aims to extract traffic patterns and train the classifiers with labelled traffic flows from real-world datasets. These real-world traffic flows from different apps, malware types and attacks provide the data to train and verify our system's classifier. Each traffic flow contains a sequence of packets and corresponding sessions along with packet receiving time, packet length, and packet protocol type. We opted to use the Scikit-learn library over Python to train and implement our classification system. The workflow of the entire system is depicted in Fig. 1. The above mentioned features and labelled traffic flows are used for training multiple algorithms. Deciding on a machine-learning algorithm was no trivial task. For experiments, we opted to compare results between seven algorithms (see Table 1). Algorithms were selected as follows: We gathered all machine learning categories used in similar research [16, 26, 27] and detected their predictive model (e.g. decision tree, neural networks etc.). Then we opted to use the most efficient algorithms from each model area.

The data mining module was also implemented using Python. It utilizes the Scapy [1] Python library for packet captures and feature extraction to the SQL database for easier manipulation of samples for machine-learning modules.

Using CART and KNN decision algorithms instead of neural networks, we can achi-eve faster classification once the system is trained and have a more interpretable model to detect hidden interconnections of traffic features. On the other hand, neural networks might be more accurate, provided there is enough training data, although they can be prone to over-fitting as well; another reason why we tested other algorithms mo—re suitable to unknown dataset characteristics. Preliminary tests showed that neural networks take a lot of time without having clear advantages over others neither in classification nor optimization (see Sect. 5.1).

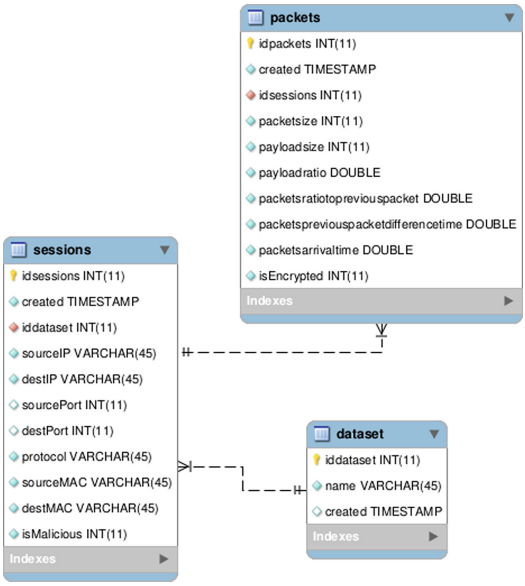

Fig. 2. Structure of the MariaDB database for traffic dataset

Table 1. Machine learning algorithms tested for malicious traffic discrimination

Logistic Regression	Linear Discriminant (LDA)	K-Neighbours (KNN)
Decision Tree (CART)	Gaussian Naïve Bayes	Support Vector (SVC)
Neural Network (Multilayer Perception)		

The machine-learning module uses Pandas [2] and Scikit-learn [3] Python libraries. Input files are CSV records exported from the database. The module performs machine-learning on the dataset using the aforementioned machine learning algorithms.

The database used is MariaDB [4]. MariaDB is a fork of the MySQL database after the acquisition of the later from Oracle. It was chosen due to its performance gains over MySQL when exporting CSV files. The database role is to reduce the footprint on the system disk while allowing us to create specialized subsets from available data for testing our Machine Learning module. Database structure is presented in Fig. 2.

5 Experimental Results

The proposed system was tested on a Dell Inspiron 15-3537 (Intel Core i7-4500U, 8 GB RAM). Parsing PCAPs to build the SQL database for later training and building the system took approximately 10 h. Classification and training took about 15 min for experiment 1 and 5 min for experiment 2 on average, for all models. All tests utilized Python and the above mentioned libraries. A sample SQL query for selecting random side channel data samples from the database is depicted at Table 2.

Table 2. Sample SQL query for exporting random malware packet characteristics from dataset

```
SELECT
p.packetsize,p.payloadsize,p.payloadratio,p.packetsratiot
opreviouspacket,p.packetspreviouspacketdifferencetime,s.i
sMalicious
INTO OUTFILE "/tmp/outmalware.csv"
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY "\n" FROM packets AS p INNER JOIN
sessions s ON (p.idsessions = s.idsessions) WHERE
s.isMalicious = 1 ORDER BY RAND()  LIMIT 10000000 ;
```

5.1 Experiment 1: Entire Datasets with Randomly Mixed Traffic - All Types of Malicious Traffic

In the first experiment we utilized all traffic from all datasets. Classification used random samples from the entire traffic database; malicious and normal traffic. The number of packets used for testing classification can be found at Table 3. The random selection of various types of malicious traffic was performed uniformly using MariaDB at the time of CSV export. This aims to remove any bias in data selection. The dataset was split 70/30 and all ∼8 GB of dataset traffic was utilized. Side channel features were extracted from each packet and imported to MariaDB. This includes the FIRST 2015, Milisenco, and CTU-13 datasets, along with all packets from each network session. This is done to test non homogenous network traffic behavior with our feature extraction. All types of malicious traffic were used in this experiment.

Table 3. Traffic packets from sessions in dataset

Total malicious traffic	6669881
Total non-malicious traffic	7968518
Non malicious non encrypted	6337244
Non malicious encrypted	1631274
Malicious non encrypted	6214670
Malicious encrypted	455211
Total	**14638399**

Machine learning and classification results are depicted at Tables 4 and 5. By viewing the hit map for true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), it is obvious that CART and KNN algorithms have a clear advantage over others in detecting malicious traffic using side channel characteristics of packets. CART gets a 94.5% detection rate with 4.4% FP and 6.8% FN, while KNN achieves 94% detection with about 5% FP and 7.7% FN, on the 200 K network traffic sessions sample. We opted not include SVC because the algorithm does

not seem to scale as well as the rest of the machine learning models. Overall, CART and KNN are the best performing models and they will the ones used on the following experiment.

Table 4. Detection comparison of algorithms – Experiment 1

AI	Accuracy	True Pos	False Pos	False Neg	True Neg
LR	0.61625	137214	25753	89370	47663
LDA	0.62428	145331	17636	95078	41955
KNN	**0.92987**	152077	10890	9541	127492
CART	**0.94506**	152560	10407	7256	129777
NB	0.52005	154225	8742	135241	1792
SVC	0.77211	148421	13017	82517	56045

Table 5. Performance comparison of algorithms – Experiment 1

AI	Precision		Recall		F1-score		Support	
	0	1	0	1	0	1	0	1
LR	0.61	0.65	0.84	0.35	0.70	0.45	162967	137033
LDA	0.60	0.70	0.89	0.31	0.72	0.43	162967	137033
KNN	0.95	0.91	0.92	0.94	**0.94**	**0.93**	162967	137033
CART	0.96	0.92	0.94	0.95	**0.95**	**0.94**	162967	137033
NB	0.53	0.17	0.95	0.02	0.68	0.03	162967	137033
SVC	0.78	0.75	0.74	0.79	0.76	0.77	162967	137033

An interesting finding was that, decreasing the overall size of the random network traffic sample under classification seems to increase the detection rate (i.e. detection of potential malicious traffic). To support this and remove potential biases in smaller sets of captured traffic, we tried various combinations of malicious and normal traffic, as we will show in Experiment 2.

Preliminary tests with neural networks show that these classification algorithms provide worse results that decision tree (such as CART, LDA) and instance-based algorithms like KNN. Notice here that preliminary tests with neural networks show that they offer worse detection rates to the aforementioned algorithms while requiring way larger amounts of time and data for training (see Tables 6 and 7). Also, in some instances, neural networks seem prone to biases.

Table 6. Detection comparison for neural networks (Multilayer Perception)

AI	Accuracy	True Pos	False Pos	False Neg	True Neg
NN (Multilayer Perceptron)	0.85031	152616	10351	7062	129971

Table 7. Performance of basic neural networks (Multilayer Perception)

AI	Precision		Recall		F1-score		Support	
	0	1	0	1	0	1	0	1
NN (Multilayer Perceptron)	0.86	0.84	0.86	0.83	0.86	0.83	162967	137033

5.2 Experiment 2: 20 K Limited Packet Sample for Feature Testing - All Types of Malicious Traffic

As mentioned previously, we detected that utilizing smaller network flow data over a trained classifier to detect malicious traffic seems to increase the True Positive detection rate. Thus, in this second experiment we purposely only use 20 K malicious packets (and consequently the same amount of clean, normal traffic) from the FIRST 2015 Dataset to test our classifier. This experiment provided insight of the performance of each algorithm with limited data.

Again, the random selection was performed uniformly by MariaDB at the time of CSV export to remove any bias. The number of packets is deliberately small since we want to confirm our assumptions at the feature selection stage of the project that the selected side channel features are pretty good for classifying malicious traffic even when data is scarce. Again, the sample along with equal sized normal traffic was split (70–30) for updating the classifier and testing for malicious traffic detection. All types of malicious traffic were used in this experiment.

After running the ml.py module on our dataset we get the following table of results. By viewing the hit map for true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), it is obvious that some algorithms have a clear advantage over others. Specifically, CART and KNN show good potential, with $\sim$89% detection rate for any given mixed malicious traffic sample, with low false negatives and false positives (around 10%). Execution times for offline training only took a couple of minutes and validation took <2 min. This proves that, even with random session, limited amount of data to train a classifier, the selected features provide very good results given the situation in very small timeframes (see Tables 8 and 9).

An interesting find is that SVC performance increases noticeably when smaller datasets are used for training and classification. This shows that SVC is prone to biases since, as we increase the training sample, its detection rate falls the fastest. KNN and CART still hold the best result percentages, while their drop in detection rates is expected; albeit very small considering the difference in data.

Table 8. Detection comparison of algorithms – Experiment 2

AI	Accuracy	True Pos	False Pos	False Neg	True Neg
LR	0.539125	1276	801	1062	861
LDA	0.548375	1321	756	1070	853
KNN	**0.888312**	1878	199	264	1659
CART	**0.888625**	1904	173	283	1640
NB	0.542625	187	1890	21	1902
SVC	0.873062	2014	63	428	1495

Table 9. Performance comparison of algorithms – Experiment 2

AI	Precision		Recall		F1-score		Support	
	0	1	0	1	0	1	0	1
LR	0.55	0.52	0.61	0.45	0.58	0.48	2077	1923
LDA	0.55	0.53	0.64	0.44	0.59	0.48	2077	1923
KNN	0.88	0.89	0.90	0.86	**0.89**	**0.88**	2077	1923
CART	0.87	0.90	0.92	0.85	**0.89**	**0.88**	2077	1923
NB	0.90	0.50	0.90	0.99	0.16	0.67	2077	1923
SVC	0.82	0.96	0.97	0.78	0.89	0.86	2077	1923

5.3 Experiment 3: Detection of Encrypted Malware Traffic

Many companies (e.g. CISCO [27]) are publishing technical reports about new intrusion detection systems (IDSes) that utilize similar features, yet only detect encrypted malicious traffic. To our knowledge, no tool is able to generalize this ability to multiple types of malicious traffic, from defacement SQLi attacks to encrypted traffic, botnets and injections like ours. Still, for arguments sake, we opt to show that malicious encrypted malware traffic can be distinguished using less features than [27] while still maintaining a high detection rate. During the third and last experiment, we focus only on the selected side channel features (Sect. 4.2) and show that we are still able to adequately detect encrypted malicious traffic.

For this experiment, our trainer program selected samples from all different encrypted malicious traffic sessions from all datasets; whether botnets, reverse shells, malware data transfer etc. To remove biases, the experiment was executed three times using (i) uniformly random samples from all datasets and types of encrypted traffic (e.g. see Table 2 above), (ii) biased (more botnet traffic in terms of 80%–20%), and (iii) per dataset. The dataset was split 70–30 for training and classification.

The hit maps for true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are depicted at Tables 10 and 11 (averages from three executions).

This experiment yielded the best results. From observations, we can conclude that these types of side channel features are effective for discriminating encrypted malware traffic; especially if we do not care to understand the type of malicious encrypted traffic or the content being transmitted. We noticed a low percentage of False Positives and False negatives ($\sim$8% of the total positive malicious sample).

Table 10. Detection comparison for encrypted malicious traffic – Experiment 3

AI	Accuracy	True Pos	False Pos	False Neg	True Neg
KNN	0.996334	488685	1495	648	135118
CART	**0.99852**	488849	**440**	**484**	136173

Table 11. Performance comparison for encrypted malicious traffic – Experiment 3

AI	Precision		Recall		F1-score		Support	
	0	1	0	1	0	1	0	1
KNN	1.0	1.0	1.0	0.99	**1.0**	0.99	489333	136613
CART	1.0	1.0	1.0	1.0	**1.0**	**1.0**	489333	136613

6 Conclusions, Findings and Future Work

In this paper, we presented seven (7) machine-learning algorithms and their performance for detecting multiple types of malicious traffic, both encrypted and unencrypted, based on selected side channel features. The project currently works retroactively on already captured data. The presented experiments adequately prove that side channel characteristics of TCP packets can be effectively used together with machine learning to detect most types of malicious traffic, even if wide differences exist on the types of ongoing attacks and to their corresponding traffic.

Some of our most important conclusions are the following:

- The best detection rate achieved was about 94.2% on CART and 93.4% using KNN algorithms, on full-scale mixed types of malicious data for various datasets totaling about ~8 GB in size.
- Preliminary results show that machine learning algorithms that utilize Decision Tree classifiers may be prone to packet crafting, if an attacker has access to the prediction model, parameters and the entire sample. Although this is generally not feasible, we should state here that the possibility exists. To this end, preliminary tests may support that Instance based algorithms like KNN along with the selected side channel features greatly reduce such attacks.
- We detected specific, descriptive features describing side channel characteristics of TCP packets (such as packet size, delivery time ratios etc.) and built lightweight classification modules (less than a few megabytes) that are able to run on real time traffic and detect ongoing malicious attacks to enhance network security. We showed that we can achieve very good malicious traffic detection percentages without utilizing full-scale TLS and connection certification features, but instead focus only on typical side channel packet characteristics.
- The use of side channel features *significantly reduces the amount of analysis* and network traffic that needs to be saved for detection. Thus, the system can be used to supplement network security analysts to gain a better understanding of their network traffic and get robust alerts on security incidents without relying on error-prone IDPS pattern matching or heavy behavioral analytics. We plan to combine our system with well-known traffic monitoring systems, like Bro [28], Snort [29], or Suricata [30].

Our experiments demonstrated the applicability of the proposed system for detecting multiple types of malicious traffic *without discriminating among types of malicious attacks*.

Our future work aims to build a working prototype for large-scale enterprise networks and work along well-known network traffic sniffers and monitoring systems (Bro, Snort, Suricata). We also aim to extend the system to incorporate more features like "connection type" and "TTL" feature to further enhance the detection mechanisms against DDoS attacks and spoofed packets.

References

1. Biondi, P.: Scapy (2011)
2. McKinney, W.: PyData development team. Pandas: Powerful Python Data Analy. Toolkit 1625 (2015)
3. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011). http://scikit-learn.org/
4. MariaDB database server. https://mariadb.com/ Accessed 1 Jan 2018
5. First.org, Hands-on Network Forensics - Training PCAP dataset from FIRST 2015. www.first.org/_assets/conf2015/networkforensics_virtualbox.zip
6. Milicenso, Ponmocup Malware dataset (Update 2012-10-07, http://security-research.dyndns.org/pub/botnet/ponmocup/analysis_2012-10-05/analysis.txt Accessed 1 Jan 2018)
7. CTU-13 dataset, CTU University, Czech Republic, 2011, https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-1/
8. Livadas, C., Walsh, B., Lapsley, D., Strayer, T.: Using machine learning techniques to identify botnet traffic. In: Proceedings of the IEEE LCN Workshop on Network Security (2006)
9. Cooke, E., Jahanian, F., McPherson, D.: The zombie roundup: understanding, detecting, and disrupting botnets. In: Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet (2005)
10. Binkley, J., Singh, S.: An algorithm for anomaly-based Botnet detection. In: Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet (2006)
11. Gu, G., Porras, P., Yegneswaran, V., Fong, M.W., Lee, W.: BotHunter: detecting malware. Infection through IDS-Driven Dialog Correlation. In: Proceedings of the USENIX Security Symposium (2007)
12. Timofeev, R.: Classification and regression trees (cart) theory and applications. Humboldt University, Berlin (2004)
13. Střasák, F.: Detection of HTTPS malware Traffic (Detekce Malware v HTTPS komunikaci). BSC thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum (2017)
14. Taylor, C., Alves-Foss, J.: NATE - network analysis of anomalous traffic events, a low-cost approach. In: Proceedings of the New Security Paradigms Workshop (2001)
15. Lakhina, A., Papagiannaki, K., Crovella, M.: Structural analysis of network traffic flows. In: Proceedings of ACM SIGMETRICS/Performance (2004)
16. Terrell, J., et al.: Multivariate SVD analyses for network anomaly detection. In: (Poster) Proceeding of ACM SIGCOMM (2005)
17. Yen, T.-F., Reiter, M.K.: Traffic aggregation for malware detection. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 207–227. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70542-0_11
18. Kohout, J., Pevny, T.: Automatic discovery of web servers hosting similar applications. In: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, IEEE, pp. 1310–1315 (2015)

19. Lokoč, J., Kohout, J., Čech, P., Skopal, T., Pevný, T.: k-NN classification of Malware in HTTPS traffic using the metric space approach. In: Chau, M., Wang, G.A., Chen, H. (eds.) PAISI 2016. LNCS, vol. 9650, pp. 131–145. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31863-9_10
20. Crotti, M., et al.: Traffic classification through simple statistical fingerprinting. ACM SIGCOMM Comput. Commun. Rev. **37**(1), 5–16 (2007)
21. Prasse, P., et al.: Malware Detection by HTTPS Traffic Analysis (2017)
22. Chari, S., et al.: A platform and analytics for usage and entitlement analytics. IBM J. Res. Dev. **60**(4), 7-1 (2016)
23. Combs, G.: "Wireshark." (2007). http://www.wireshark.org/lastmodified Accessed 12 Feb
24. Roesch, M.: Snort: lightweight intrusion detection for networks. In: Lisa, Vol. 99, no. 1 (1999)
25. Liu, J., et al.: Effective and real-time in-app activity analysis in encrypted internet traffic streams. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2017)
26. Chen, S., et al.: Side-channel leaks in web applications: a reality today, a challenge tomorrow. In: IEEE Symposium on 2010 Security and Privacy, IEEE (2010)
27. Encrypted Traffic Analytics, Cisco public, White paper (2017). www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.pdf Accessed Mar 2018
28. Bro, I.: http://www.bro-ids.org (2008)
29. Beale, J., Baker, A., Esler, J.: Snort: IDS and IPS toolkit. Syngress (2007)
30. Suricata, I.D.S.: open-source IDS. IPS/NSM engine (2014). (http://suricata-ids.org/)
31. Sommer, R., Paxson, V.: Outside the closed world: on using machine learning for network intrusion detection. In: IEEE Symposium on 2010 Security and Privacy (SP), IEEE (2010)