ARL-TR-9653 ● FEB 2023

DEVCOM ARMY RESEARCH LABORATORY

An Empirical Investigation of Packet Header-Only Network Traffic Anomaly Detection and Classification

by Michael J De Lucia, Daniel E Krych, Stephen Raio, and Jason E Ellis

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9653 ● FEB 2023

An Empirical Investigation of Packet Header-Only Network Traffic Anomaly Detection and Classification

Michael J De Lucia, Daniel E Krych, Stephen Raio, and Jason E Ellis
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
February 2023	Technical Report	January 2022	September 2022

4. TITLE AND SUBTITLE
An Empirical Investigation of Packet Header-Only Network Traffic Anomaly Detection and Classification

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Michael J De Lucia, Daniel E Krych, Stephen Raio, and Jason E Ellis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-ND Aberdeen Proving Ground, MD 21005	ARL-TR-9653

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Network traffic attack detection at the packet level is complicated by subtle factors surrounding dataset collection and utilization as well as evaluation methods that lead to results that are unreliable for real-world applications. We empirically investigate various machine learning techniques for identifying anomalous/malicious network traffic using only individual packet-level transport layer and network layer header fields (i.e., Transmission Control Protocol/Internet Protocol [TCP/IP]). While much of the work in network traffic anomaly detection uses flow-level metadata as features or deep packet inspection to leverage packet payload contents as features, our investigation solely used individual packet headers as features. We characterize and compare the anomaly detection performance of both supervised and unsupervised learning models using these header features and explore the use of different scoring methods. Our research sheds light on the complexities in using available datasets and common evaluation techniques: publicly available datasets with packet captures that do not match provided flow statistics, intricacies in attributing malice at the packet level, and the absence of real-world considerations for how anomaly detection methods are used in practice.

15. SUBJECT TERMS
Network security, cyber, machine learning, anomaly detection, network intrusion detection, network traffic, Network, Cyber and Computational Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	56

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Michael J De Lucia	(410) 278-6508

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Cyber-attacks are occurring at a rapid pace, overwhelming human analysts, and necessitating the use of faster and more efficient detection techniques. Machine learning algorithms help improve detection speed and identify patterns human analysts may have missed. Traditionally, a subject matter expert is required to identify the most influential features of a cyber-attack to embed these into a detection mechanism, such as a machine learning model. However, network traffic attack detection at the packet level is complicated by subtle factors surrounding dataset collection and utilization as well as evaluation methods that lead to results that are unreliable for real-world applications. Our prior work in malicious network traffic detection[1] sought to automate this process, leveraging the raw bytes of a packet, inclusive of the transport layer header and payload, and deep learning (i.e., one-dimensional Convolutional Neural Network [1D CNN] and Feed Forward Neural Network), and achieved an F1 score of 98.99% while reducing the need for subject matter expert feature engineering.

Furthermore, in our previous research[1] using supervised learning methods, we briefly noted that we achieved a comparable F1 score when using only the transport header field bytes as features versus the packet byte features combined with the transport header field bytes as features. We also note that 1) an increased prevalence of network packet encryption (e.g., Transport Layer Security [TLS]) precludes the inspection of the transport layer payload and 2) when models use flow-level or inter-packet features, decisions will be delayed by the waiting for connections to close or reach a timeout along with the calculation/processing time of these historical features. Therefore, we sought to investigate using solely header field bytes as features for anomaly detection at the packet level. A limitation of our prior method is that deep learning models bring several complexities, are resource intensive, and provide less explainability in comparison to traditional machine learning models. Observations of our prior work indicated a reduction in performance of the classifier when using only the payload of a packet in comparison to using the payload and header bytes. Our observations motivated further investigation of the utility of using only the header fields. We sought to determine the efficacy of using unsupervised anomaly detection methods to limit the necessity of labeled datasets containing both benign and malicious samples.

We focus our research on minimizing training and prediction time within resource-constrained environments, such as the tactical edge. Many researchers convert network traffic detection problems to features in another domain, such as image analysis, to enable the leveraging of domain-specific, state-of-the-art deep learning models such as in Wang et al.[2,3] and Zeng et al.[4] In Wang et al.,[2,3] the authors select

784 bytes from either the Transmission Control Protocol (TCP) session payload or across all layers and convert the bytes to a 28 × 28 gray scale image and use this feature set to train a LeNet-5 (i.e., image) inspired classifier. Zeng et al. use 900 bytes of the packet represented as 30 × 30 gray scale images as their feature set to train their classifier, consisting of three parallel architectures—a stacked Autoencoder (AE), a two-layer 1D-CNN, and a long short-term memory (LSTM) classifier. However, an additional step of conversion to certain domains can add overhead in terms of computation time and can potentially cause information loss. We acknowledge that domain conversion of the data can potentially lead to better-performing models due to the availability of state-of-the-art algorithms in other domains. These state-of-the-art algorithms may be able to identify different patterns in the same data, using different perspectives. However, data conversion may add computational overhead to the classification/detection decision, may possibly cause further information loss from misinterpretation or misrepresentation of the raw data, and may add limitations to the size of the feature set or feature types used. Therefore, to ensure we do not introduce any unnecessary computational overhead or limitations, and to adopt an approach that will be practical in a real-world setting where encrypted payloads are the norm, we constrain our features for our network anomaly detection and binary classification models to only the raw-form packet header fields (TCP/Internet Protocol [IP]), and do not use historical information such as network flow statistics or inter-packet features. We use these network header fields without domain conversion with the expectation that it will be more resource efficient and will prevent any potential misinterpretation or misrepresentation of the raw data during domain conversion.

Several prior studies leverage the raw bytes of a network packet as the features coupled with machine learning algorithms. The DeepPacket framework in Lotfollahi et al.[5] consisted of a stacked AE and a CNN to discern encrypted from non-encrypted traffic using the first 1480 bytes of IP payload as the input features. Yet another example is the DeepMAL architecture consisting of a combination of a 1D CNN and LSTM using the first 1024 bytes of the payload as input.[6] Another example of using the raw bytes of the TCP/User Datagram Protocol (UDP) header and payload is nprint, which focuses on network traffic analysis and classification.[7] While many of these works leverage the raw bytes of a packet, they do not focus on specific fields within the TCP and IP header for anomaly detection. Also, these works focus only on the supervised learning approach and require a sizeable amount of labeled data to perform training.

We propose to leverage unsupervised learning to perform anomaly detection. We also evaluate the differences in performance between supervised and unsupervised methods for anomaly detection. We acknowledge that anomaly detection is not the

same as attack detection. Namely, there exist anomalies that are considered benign and conversely some attacks would not be considered as anomalous at the individual packet header level. Specifically, the focus of our work is anomaly-based techniques to detect network attacks.

Our contributions include the following:

- An empirical investigation and analysis of the efficacy of solely using TCP/IP packet headers for anomaly detection and classification of network attacks
- An in-depth analysis/discussion on the intricacies and caveats introduced by the following:
    - Labeling malice when it resides at one or more levels in network traffic
    - Using anomaly detection techniques for attributing malice
    - Using evaluation techniques for network attack detection models that fail to consider and weigh how these models would be used in real-world scenarios/applications and what defines a successful detection (i.e., at what point does an alert or a group of alerts provide actionable information?)
- An analysis and comparison of using supervised learning as a baseline for detection performance and discussion on results that are biased to the network environment of which the dataset was created
- An empirical investigation and analysis of the performance of unsupervised learning (Principal Component Analysis [PCA] and AE) versus supervised learning (Random Forest, Logistic Regression, Stochastic Gradient Descent [SGD] Logistic Regression, and SGD Linear Support Vector Machine [SVM]) algorithms for network attack detection

2. Methodology

2.1 Detection Approach

When measuring the effectiveness of packet-level detection techniques, it is important to select an appropriate definition of successful detection. Detections can be made at the individual packet level, at the flow level, relative to a time frame, or relative to an attack campaign. Each might have merit in different scenarios, but ultimately a network analyst is concerned with detecting attack campaigns and

having a minimal number of false positive alerts. How you choose to evaluate your detection mechanisms can significantly overstate or understate the efficacy of your network attack detection approach.

If a detection mechanism can flag even a single packet in an attack campaign, that could be enough information for an analyst to uncover the rest of the malicious activity. In cases where the entire campaign cannot easily be distinguished from a single detection event, it could be useful to evaluate a detection mechanism by its ability to detect at least one packet in each malicious flow. Packet-level detection verification has the most caveats. In most cases, it is not reasonable to expect a packet-level detection technique to flag each packet that is part of a larger malicious activity as malicious. This is because there is a good chance that many of the packets comprising a malicious activity are indistinguishable from benign traffic activity. Consider a simple TCP handshake series of packets. Should the classification of these packets change if the handshake is initiated by a known-benign user versus a malicious actor, and how could they be told apart without blacklists/whitelists? Furthermore, it is not generally reasonable to assign the property of malice to all packets within an attack campaign or even individual packets full stop. Even though there are cases where every packet could be considered malicious—a port scan, for example—there are far more instances of only a few packets in an attack campaign being malicious, such as the push request of a Structured Query Language (SQL) injection attack, leaving the rest of the packets in that flow to be truly benign in nature.

Flow-level detection mechanisms can resolve the case where no individual packet in isolation is considered malicious, but a group of them is malicious. Consider a distributed denial of service (DDoS) against a web server where each individual request is indistinguishable from benign requests but grouped together; they can be considered malicious. Examples comparing where malice can reside in different levels of network traffic can be seen in Fig. 1 and note that malice may reside in a combination of these levels.

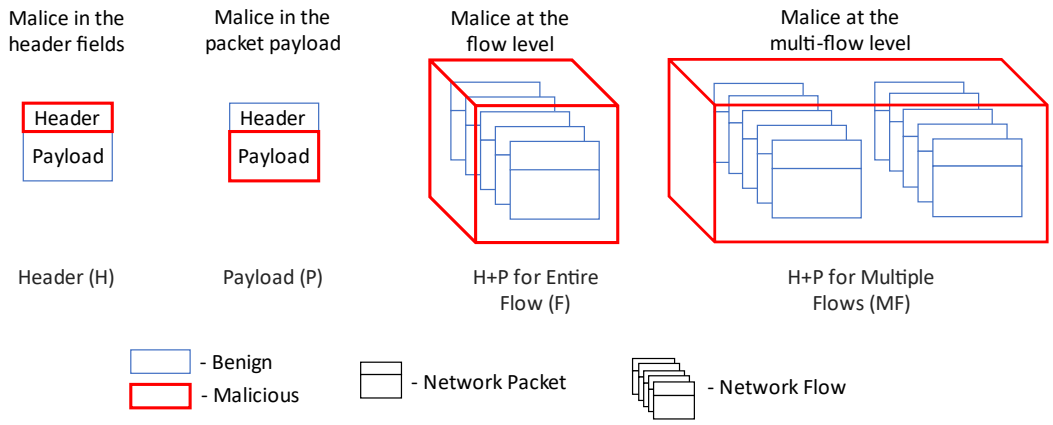


Fig. 1 Diagram describing where malice can reside, at the header, packet payload, flow, or multi-flow levels, along with combinations of these levels

Adding to the intricacies of identifying and labeling malice that may exist at one or more levels of network traffic, further considerations must be made when using anomaly detection techniques to detect malice. Figure 2 demonstrates the possible classifications and misclassifications of network packets by a flow-level anomaly detection model (our unsupervised learning models) when viewed in the context of this research, determining malice based on anomalousness. All packets can be classified under either the category of benign or malicious based on the ground truth flow-level label. Furthermore, packets can also be assigned a second category if they are considered anomalous based on the rarity of their feature values in the dataset. So, the four possible categorizations are benign and not anomalous (true negative with respect to malice), benign and anomalous (false positive with respect to malice), malicious and anomalous (true positive with respect to malice), and malicious and not anomalous (false negative with respect to malice). Respective examples of these categories are normal web traffic, abnormal but benign traffic such as a user trying to connect to the wrong port for a service, attack traffic such as a port scan of uncommon ports, and attack traffic on a legitimate secure shell server. Using anomaly detection algorithms to detect malice flags packets categorized as anomalous and assumes they are malicious. Therefore, even when rolling up all the packet detections in a flow for a flow-level malice detection, anomaly detection cannot be expected to avoid false negatives in environments where all malicious flows do not overlap the set of malicious and anomalous packets. Similarly, such techniques cannot be expected to avoid false positives in environments where the set of benign packets overlaps with the set of anomalous.

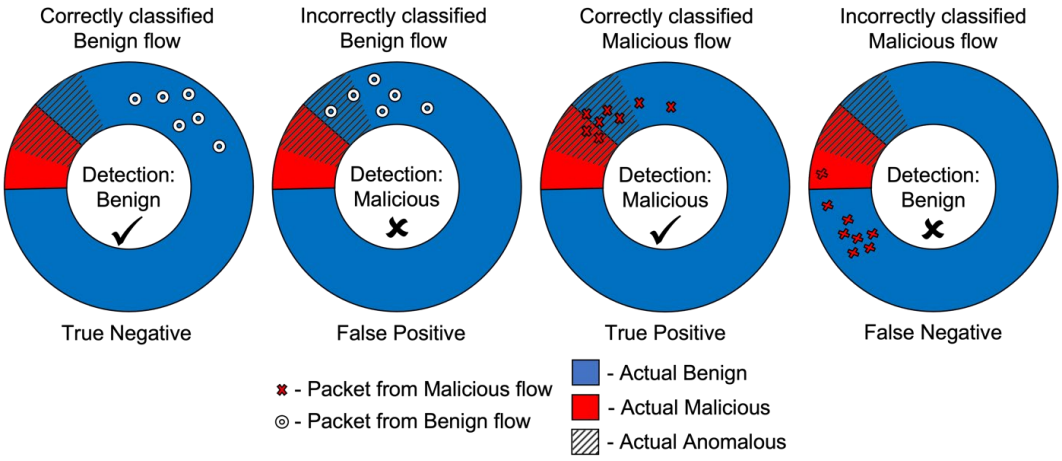


Fig. 2 Diagram comparing actual benign, malicious, and anomalous real-world network traffic, the malicious and benign packets within each flow and their individual packets at times overlapping several of these actual zones, and how these flows would be classified or misclassified by anomaly detection (unsupervised learning) models

Time frame–based evaluation of detection, while not something that would need to be done in practice, may be necessary during technique development when using unlabeled data. It could, however, suffer from conflation of multiple attacks if they happened in the same period.

Your choice of defining success may also be limited by the dataset and labeling available. Existing datasets may only be labeled at the flow level (e.g., CICIDS2017)[8] or based on time frames when malicious activities occurred (e.g., DARPA99).[9] When attempting to migrate these labels to a different granularity, such as migrating flow labels to the packet level, additional complexities arise. In our experience it is not straightforward to migrate flow labels to packets. For example, in the CICIDS2017 dataset, there are more packets than accounted for in flows, flows were constructed with flaws due to the flow creation software used,[10] and discrepancies in time stamp resolution hindered flow label migration.

When choosing how to evaluate a detection methodology, it comes down to what is important to the user. What is important from a network cybersecurity analyst perspective initially is discovery of an attack campaign and possibly an attack flow, not an individual packet detection, while maintaining a manageable number of false positive alerts.

2.2 Dataset Overview

For this work we focus on TCP/IP packets from the CICIDS2017 dataset from Sharafaldin et al.[8] with the corrections made by Engelen et al.[10] Namely, Engelen et al. corrected flow construction errors including pre-terminated flows, flow

direction reversal and not correctly terminating flows, as well as labeling issues, including imprecise timing and relabeling of failed attack attempts. Since our methods use packet-level features and perform anomaly detection based solely on packet headers, the flow-level labeling of the dataset was required to be transferred to the respective packets. Our labeling process adjusts flow time stamps for time zone offsets with the packet capture files and performs a best-effort match of packets to their respective flows based on flow start time and duration, propagating the matched flow label down to each packet. While not as precise as fully recreating the flow creation logic, this procedure worked well enough to attain enough data with confidence that it was properly matched to the flows. For the CICIDS2017 dataset, we keep only packets and flows where the flow had been fully matched to its packets (i.e., matches for the correct number of forward and backward packets in the flow were identified within the appropriate time window).

Our research focuses on per-packet anomaly detection or binary classification based solely on network (IP version 4) and transport (TCP) layer headers. We do not consider any transport layer payload or multi-packet metrics such as flow or inter-packet information. There are 14 different attack types (labels) in the CICIDS2017 dataset. Based on domain expertise, we hypothesize that only two, *Port Scan* and *Infiltration*, will be distinguishable from benign traffic. Additionally, we hypothesize that two of the denial of service (DoS) attacks, *DoS Slow Loris* and *DoS Slow HTTP Test*, will potentially be distinguishable depending on packet size of keep alive messages or window size relative to benign packet sizes, respectively. We hypothesize that all other attack types are less likely to be or not at all distinguishable from normal, benign traffic using only header features from an individual packet. Our expectations, alternative hypothesized methods for detection of each attack, and analysis of where malice resides for each type of attack are summarized in Table 1, utilizing terminology described in Figure 1. The results and our overall observations for each attack are shown in Section 3.

Table 1 Attack detection expectations and potential alternative methods. Individual packet header features, IP and TCP only

Traffic label	Detection expectation from machine learning model using header-only features, single packet per sample	Potential alternative detection methods	Where malice resides [header (H), payload (P), header + payload for entire flow (F), header + payload for multiple flows (MF)]
Bot (Botnet)	Likely not distinguishable from normal web traffic	Likely not distinguishable from normal web traffic without deep packet inspection	P
DDoS	Likely not distinguishable from normal web traffic	Tracking the number of requests/messages and number of sources	MF
DoS GoldenEye	Nothing distinguishable from normal web traffic requests, Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS) [HTTP(S)]	Tracking number of HTTP(S) requests from a source over time	F, MF
DoS Hulk	Nothing distinguishable from normal HTTP requests	Tracking number of HTTP requests from a source over time	F, MF
DoS Slow HTTP Test	This is some variation on the HTTP Slow Read Attack. Possibly distinguishable from the abnormally small window size in header field of message	Tracking for abnormally small TCP window sizes from the client-side of the connection.	H*, F, MF
DoS Slow Loris	Possibly distinguishable from the “keep alive” messages being sent. It would depend on if messages of that size are typical in the environment or not, and/or if an untypical port is used.	Tracking “keep alive” messages across a number of connections from the same source and flow duration could detect Slow Loris. Flow message direction would also be either all source to destination, or in the case of acknowledgements, 50/50 split in traffic based on direction.	H*, F, MF
FTP Patator	Nothing distinguishable from normal File Transfer Protocol (FTP) command traffic	Tracking proportion of FTP control traffic to data transfer traffic. Brute force attempts would be very control traffic heavy. Assuming FTP is set up to disconnect after X unsuccessful login attempts, tracking FTP connection startups over time could detect FTP brute forcing.	F, MF
HeartBleed	Nothing distinguishable from normal TLS traffic	Deep packet inspection	P
Infiltration	Distinguishable for certain types of port scans based on flags and destination ports used. The callback to the attacker may also be distinguishable if it uses an odd port.	Tracking number of destination ports used by a host over time, connections over time, or incomplete connections over time	H, MF
Port Scan	Distinguishable for certain types of port scans based on flags and destination ports used	Tracking number of destination ports used by a host over time, connections over time, incomplete connections over time	H, MF
SSH Patator	Nothing distinguishable from normal Secure Shell (SSH) traffic	Assuming SSH is set up to disconnect after X unsuccessful login attempts, tracking SSH connection startups over time could detect SSH brute forcing.	F, MF
Web Attack: Brute Force	Likely not distinguishable from normal web traffic	Tracking number of HTTP requests from a source over time	P, F, MF
Web Attack: SQL Injection	Likely not distinguishable from normal web traffic	Deep packet inspection	P
Web Attack: XSS	Likely not distinguishable from normal web traffic	Likely not distinguishable from normal web traffic without deep packet inspection	P

Our experiments use many different subsets of the CICIDS2017 dataset to form the training, test, and baseline datasets. Each experiment downsamples the number of packets (supervised learning scoring approach) or flows (unsupervised learning scoring approach) to the minimum of either the benign or specific malicious label packet/flow count as applicable. This scoring methodology is detailed in Section 3, Scoring. Table 2 summarizes the number of samples present in each train, test, baseline, and validation set for the unsupervised and supervised learning model experiments. For details of the breakdown of each dataset per label, see the Appendix (Datasets Detailed).

Table 2 Breakdown of all datasets' sample counts. Note: During scoring, subsets of samples are used as we downsample to a 50/50 benign/malicious split per malicious label.

Set	Benign samples	Malicious samples
unsupervised-train	~8M	0
unsupervised-test	~36.9M	~4.4M
unsupervised-baseline	~2M	0
unsupervised-validation	10K	0
supervised-train1	800K	400K
supervised-train2[a]	400K	400K
supervised-train3[a]	300K	300K
supervised-train4[a]	200K	200K
supervised-train5[a]	100K	100K
supervised-test1	~10M	~3.9M
supervised-test2[b]	~3.9M	~3.9M
supervised-baseline	100K	100K

[a] Subset of supervised-train1
[b] Subset of supervised-test1

2.3 Dataset Processing

The dataset was split into training, test, baseline, and validation sets without overlap, based on the days of the week in the CICIDS2017 dataset. We created separate sets for the supervised and unsupervised approaches. For each of the overall unsupervised and supervised datasets, we use or randomly sample from all of the cleaned CICIDS2017 dataset. Additionally, as seen in Table 2, the supervised test sets have approximately 3.9 million samples compared to the unsupervised test set having approximately 4.4 million samples. The difference in size for the supervised dataset is attributed to the addition of malicious samples in the training datasets.

For unsupervised model datasets, all packet samples used in train, baseline, and validation sets are randomly sampled from the network traffic flows, but all samples in the test set are part of complete network flows available in the original dataset. We train on approximately 80% of Monday's (i.e., benign) network packets. The test set is balanced between benign and malicious packets across Tuesday–Friday

network packet captures. A baseline set is crafted using approximately 20% of Monday's benign network packets. A validation set is crafted using 10,000 packets from Monday's benign network packets.

For supervised model datasets, all packet samples were randomly sampled from the network traffic flows; all benign training and baseline samples are from Monday, while all malicious data for all sets is sampled from Tuesday to Friday, and no samples overlap between train, baseline, and test sets, but each training set is a subset of train1, and test2 is a subset of test1. We train on a subset of the benign and malicious packets, respectively, from Monday and Tuesday–Friday network packet captures. We create a test dataset with a subset of the remaining benign and malicious network packets from Tuesday–Friday network packet captures. A baseline set is crafted using a subset of the benign and malicious packets, respectively, from Monday and Tuesday–Friday network packet captures.

For a detailed breakdown of the different datasets used in our experimentation and the number of packets/flows used per label, see the Appendix.

2.4 Feature Sets

We initially leverage the raw bytes of the TCP and IPv4 headers as features, thus some header fields were split across bytes. However, due to the predefined structures of the headers and poor initial results using raw bytes, we shifted our methodology to use the header fields as features. For our unsupervised learning models each feature was normalized between 0 and 1 and the min/max scaler was fit on the min/max values of the respective header field according to the request for comment (RFC). For supervised learning models, each feature (i.e., header field) was standardized using a standard scaler, also fit on the same min/max RFC constrained values.

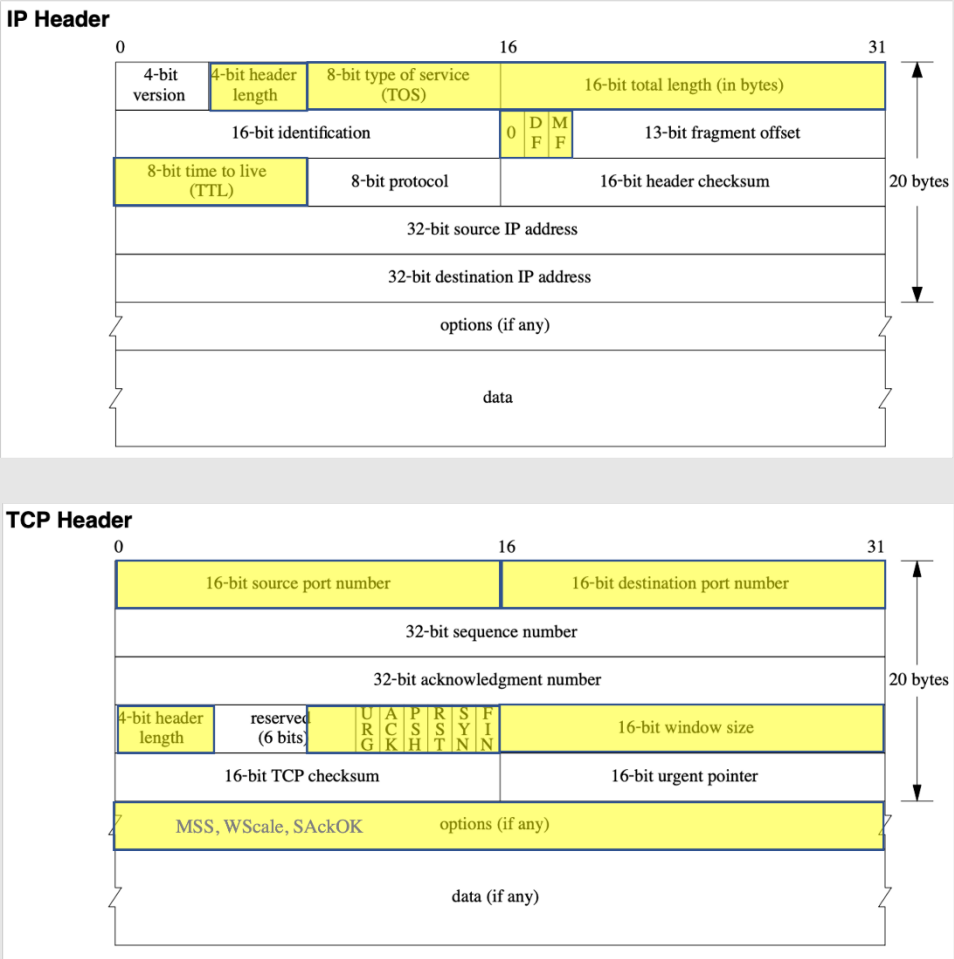
Through dataset analysis and subject matter expertise, we trim several TCP/IP header fields from our feature set. We observe that no packets across the CICIDS2017 contain IP options, and several TCP/IP header fields are unchanging (e.g., IP protocol when only using TCP) or randomly generated numbers (e.g., sequence/acknowledgement numbers). We also exclude features such as IP addresses and checksums. Our final feature set can be seen in Table 3 and highlighted in Figure 3.

The relationships between our numerical features and malice are not linear and may be negatively or positively correlated in nature. For example, an increase of 1 in value for port number or time to live (TTL) number does not linearly relate to an increase/decrease of malice level of the packet. Similarly, the magnitude of the

feature value does not linearly relate to maliciousness; a high value is not necessarily more malicious than a low value and vice versa. Therefore, the relationship between the features and malice is more complex than the relationship between the features and anomalousness.

Table 3 Ipv4 and TCP header features used for our feature set and the min/max values used for scaling based on real-world RFC constraints

Feature	Min	Max
ip_ihl	0	15
ip_tos	0	63
ip_flags	0	7
ip_totallen	0	65535
ip_ttl	0	255
tcp_sport	0	65535
tcp_dport	0	65535
tcp_do	0	15
tcp_flags	0	273
tcp_winsize	0	65535
tcp_opt_mss	0	65535
tcp_opt_wscale	0	255
tcp_opt_sackok	0	1



Adapted from W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley Publishing Company, Inc.© 1994.

Fig. 3 IPv4 and TCP headers highlighting the features used for our feature set

2.5 Unsupervised and Supervised Machine Learning Models Developed

2.5.1 Principal Component Analysis

We leverage the PCA algorithm for unsupervised learning. PCA is normally used for dimensionality reduction via Eigen decomposition, reorienting the dataset along new axes in lower dimensions while maintaining as much variance in the underlying data as possible. However, it also can be used for anomaly detection. PCA is fitted using the benign network packets (training set from Monday), and when applied on the test set, we identify anomalies using the reconstruction error. The reconstruction error is defined as the difference between the original input sample to PCA and the reconstructed sample, calculated using the mean absolute

error (MAE). A single PCA component is composed of a linear combination of the original features according to the variance. Thus, PCA provides to a limited extent the correlation among features, and the contribution of features by inspecting the resulting unit eigenvectors and squaring these to obtain the percentage of each feature present in each component. A high reconstruction error indicates a sample outside of the benign distribution that PCA was fitted on. We experiment using 1–12 components, creating 12 models, one for each principal component that encompasses itself and all principal components prior (e.g., "pc 3" model contains itself and principal component 1 and 2, totaling 3 dimensions). We investigate the explained variance of these models and explore insights through analyzing the variance of the features in both our benign and malicious datasets, discussed in Appendix A.1, Further Exploratory Dataset Analysis via PCA.

2.5.2 Autoencoder Model

The AE model is a neural network technique for dimension reduction and determining feature importance. An AE model contains an encoder and decoder component. The encoder reduces the number of neurons and dimensions, and the output is often referred to as the bottleneck layer or latent code. The decoder takes the latent code as input and reconstructs the sample back into the original dimensions. Again, as done with PCA, we leverage the reconstruction error to identify network attack packets. Our AE is composed of an input layer, a hidden latent code layer, and output layer. The MAE was used as the loss function. The encoder and decoder, respectively, use a Rectified Linear Unit (ReLU) and Sigmoid activation function. The optimizer for the AE is Adam. Additionally, a batch size of 128 and 2 training epochs are performed. A validation set, consisting of 10,000 benign samples from Monday is used for the AE experiments.

2.5.3 Supervised Machine Learning Models

We also compare our unsupervised learning methods with supervised learning methods. We leverage Random Forest, Logistic Regression, and SGD Linear SVM for detecting network attacks. We explored using SGD for Logistic Regression as well as default parameters of scikit-learn's Logistic Regression. The Random Forest model uses 200 decision trees and does not constrain the levels within a tree. The Logistic Regression uses the default lbfgs algorithm for optimization, log for a loss function, regularization via the L2 penalty function, and was limited to 1000 iterations. The SGD classifier models use SGD for optimization, and either log as a loss function or hinge (Linear SVM), regularization via the L2 penalty function, and was limited to 1000 iterations.

3. Results and Discussion

3.1 Exploratory Dataset Analysis

As part of our exploratory dataset analysis, we aimed to determine the correlation of our features in our different datasets. In Fig. 4 we see on the left the feature correlation heatmap for benign-only data, the unsupervised learning training set that consisted of 80% of our benign Monday data minus 10,000 packets used for the validation set. Furthermore, on the right side of Fig. 4 we see the feature correlation heatmap for the malicious-only data, an example malicious training set developed only to better understand the malicious data features and their relation to the benign data features, which consisted of 10% of our malicious Tuesday–Friday data (43,399 flows, 436,611 packets) randomly sampled. These heatmaps show all nonzero positive and negative correlations between our 13 TCP/IP header features, with –1 and 1 being the strongest negative and positive correlation values possible, respectively. These scores translate to shades of color in the heatmap, with darker shades of color indicating stronger correlation. All features that resulted in zero correlation with all other features were removed from the heatmap (e.g., ip_tos for the malicious heatmap).

Overall, we observe that the features mostly correlated are the TCP options (mss, wscale, and sackok) and the TCP data offset. Accordingly, the data offset value will be larger as the header is larger whenever TCP option fields are present. Additionally, a strong correlation is seen between TCP source ports and destination ports, which is logical as packets in a bidirectional network flow have alternating source and destination port number pairs.

Overall, the feature correlations are similar between the benign and malicious data. Of note, tcp_flags, tcp_winsize, and ip_ttl have significantly more correlation with other features for the malicious data compared to the benign data. Similarly, the TCP option features have slightly more correlation with other features for the malicious data compared to the benign data. The tcp_sport and tcp_dport features have slightly higher correlation with other features for the benign data compared to the malicious data.

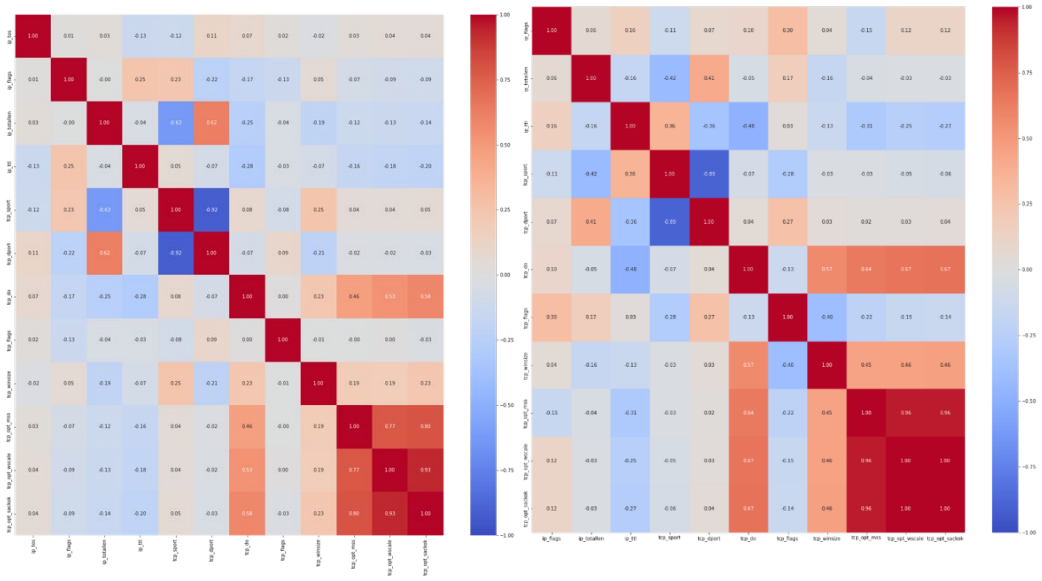


Fig. 4 (Left) Benign: Feature correlation heatmap of the unsupervised learning benign training set. (Right) Malicious: Feature correlation heatmap of 10% of malicious Tuesday–Friday data.

In Fig. 5 we observe the correlation between all features and the labels (0 for benign, 1 for malicious representing all malicious labels) combining benign and malicious samples, for both the unsupervised and supervised datasets. On the left of Fig. 5, the benign plus malicious unsupervised learning datasets feature the correlation heatmap for all training data (consisting of 80% of benign Monday data minus 10,000 packets [142,636 flows, 8,005,097 packets]) and 10% of the malicious packets from Tuesday–Friday data (43,399 flows, 436,611 packets) randomly sampled. On the right of Fig. 5, the benign plus malicious supervised learning datasets feature the correlation heatmap for train1.csv consisting of 800,000 benign packets from Monday data and 400,000 malicious packets from Tuesday–Friday data.

For the supervised learning we show the train1 set, of which all other supervised learning train sets are subsets and which was the set associated with the highest F1 scores throughout our experimentation. Again, all features that resulted in zero correlation with all other features were removed from the heatmap. Overall, the supervised dataset that contains significantly less samples has slightly higher correlation between the independent variables (features) and the dependent/target variable (label) as compared to the unsupervised dataset, with the ip_ttl, tcp_do, and the three tcp_opt features (tcp_opt_mss, tcp_opt_wscale, tcp_opt_sackok) somewhat standing out. Otherwise, we find similar correlations and conclusions to those observed in the heatmaps of Fig. 4.

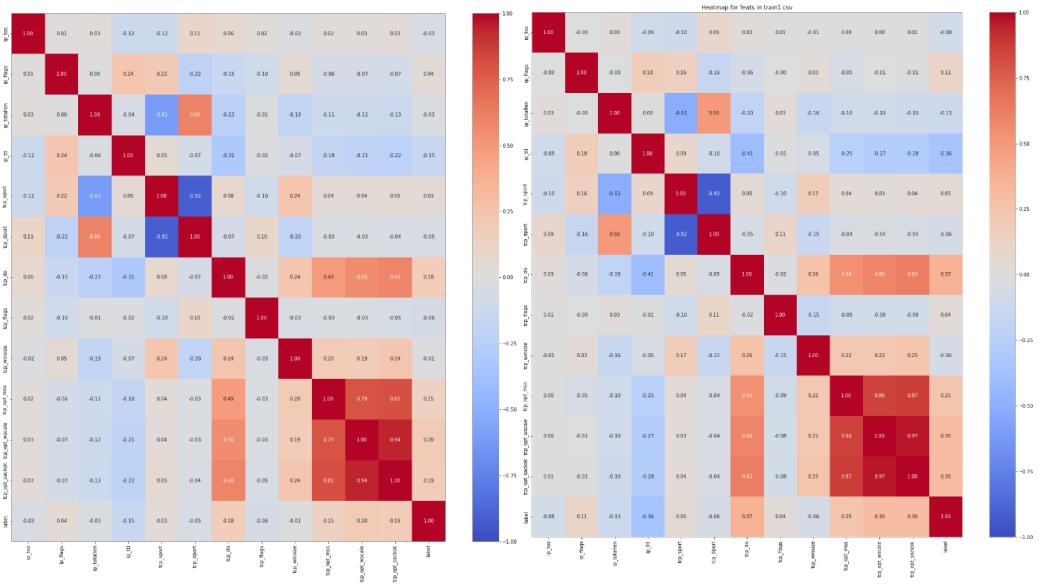


Fig. 5 (Left) Benign plus malicious: unsupervised learning datasets feature plus label correlation heatmap for all training data and 10% of the malicious packets from Tuesday–Friday data. (Right) Benign plus malicious: supervised learning datasets feature plus label correlation heatmap for train1.csv.

For additional analysis of the datasets, see Appendix A.1, Further Exploratory Dataset Analysis via PCA.

3.2 Metrics

A true positive (TP) is a classifier's prediction of a network attack sample as a class of network attack, while a false positive (FP) is a classifier's prediction of a benign sample as a class of network attack. A true negative (TN) is a classifier's prediction of a benign sample as a class of benign, while a false negative (FN) is a classifier's prediction of a network attack sample as a class of benign (i.e., missed attack). A high false positive rate is detrimental to a classifier, as it consumes analysis time and can lead to mistrust and eventual non-use of a tool. We compare the false positive rate of each classifier. The false positive rate (FPR) is defined as

$$FPR = \frac{FP}{FP + TN}$$

We also compare the F1 score, which is the harmonic mean of the precision (P) and recall (R). These are defined as

$$P = \frac{TP}{TP+FP}, \quad R = \frac{TP}{TP+FN}, \quad F1 = \frac{2PR}{P+R}$$

3.3 Scoring

For unsupervised model experiments, scoring was calculated at the network flow level, meaning only the packet/sample that individually scored the highest per flow is used in our calculations. This was performed to follow our previously discussed logic that in practice the high alert score of even one packet in a flow should be sufficient to trigger detection and in turn enable an analyst to detect the malicious flow.

$$Alert\ Score = \frac{X_i - P_{90}}{\sigma}$$

$where$:
$X_i = Reconstruction\ error\ of\ sample$
$P_{90} = 90th\ Percentile\ of\ baseline\ reconstruction\ error$
$\sigma = Standard\ deviation\ of\ all\ baseline\ reconstruction\ errors$

Additionally, when scoring each model per label, we downsample the number of flows to balance benign and malicious. The minimum number of either the benign or specific malicious label flow count from each day is used. For example, since *SSH Patator* occurred on Tuesday, it was scored against benign flows on Tuesday. For *DoS Hulk* and *Port Scan*, there were more malicious flows than benign, but for all other malicious labels, there were more benign flows than malicious on the respective day they occurred. Review of several PCA model plots of scoring benign baseline data (such as seen in Fig. 6) showed no optimal decision boundary, but most plots seemed to have a large portion of scores below or near 0. Therefore, we use an alert score of 0, the 90th percentile of the baseline data, as our benign and malicious decision boundary. Figure 6 shows the unsupervised baseline data scored using the PCA model with seven principal components. The PCA model with seven principal components performed best for three of the malicious labels, and although its average F1 score is lower than other PCA models, its average FPR score is better, but still too high for practical use. This can be seen in the respective Appendix Tables A-3 and A-4. Furthermore, the seven principal component PCA model explains approximately 99% of the variance in the original training dataset samples.

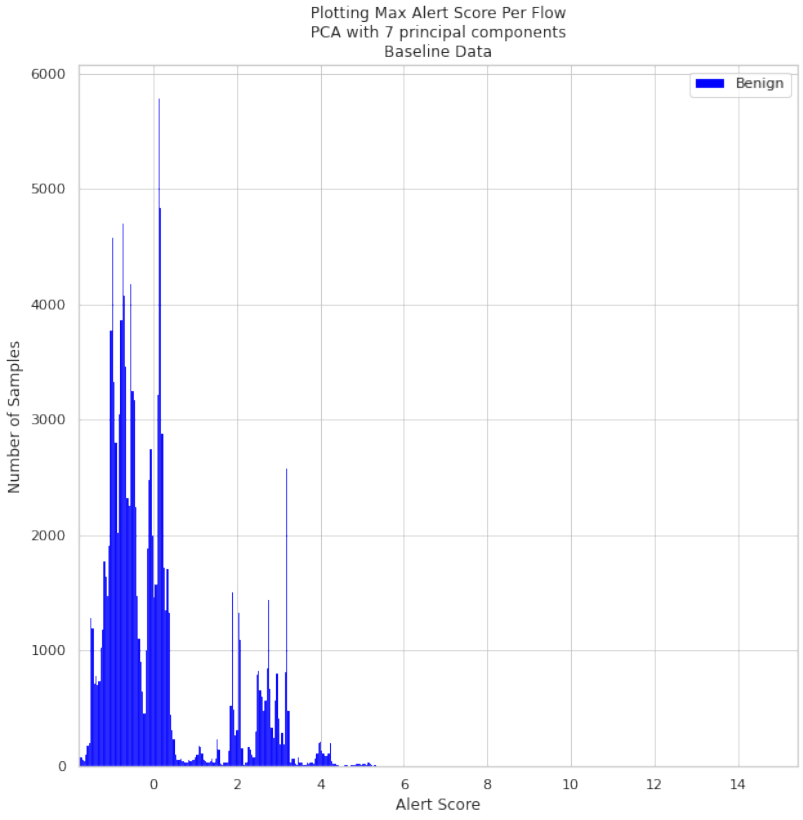


Fig. 6 Plot of unsupervised dataset baseline set (benign-only) tested with PCA model with seven principal components

For supervised model experiments, scoring was calculated at the packet level. Thus, each individual packet/sample's score is used in our calculations. When scoring each model per label, we downsample the number of packets to balance benign and malicious to the minimum number of either the benign or specific malicious label count. For supervised datasets, there were always more benign packets than malicious, thus we downsampled to the number of malicious packets per label.

Due to the differences in scoring at the flow level for unsupervised models and packet level for supervised models, these scores are not directly comparable.

We round our results to either two or three decimal places, except when this resulted in rounding to 1, in which case we limit this to 0.99.

3.4 Results and Analysis

Tables 4 and 5 contain a summary of the overall best F1 scores and the associated FPR for all unsupervised and supervised learning algorithms, per label, trained and tested on their respective datasets. We consider F1 scores greater than or equal to

0.75 and FPR scores less than or equal to 0.14 to be adequate and represented in bold, with anything outside of these bounds to be inadequate.

Table 4 Network attack detection observed results (scored at flow level) for unsupervised learning models trained/tested using unsupervised dataset. Individual packet header, IP and TCP only.

Traffic label	Principal Component Analysis		Autoencoder	
	Best F1	FPR	Best F1	FPR
Bot	0.70	0.86	**0.77**	0.36
DDoS	0.70	0.86	0.33	0.37
DoS GoldenEye	**0.77**	0.51	0.17	0.31
DoS Hulk	**0.75**	0.52	0.22	0.31
DoS Slow HTTP Test	0.68	0.54	0.01	0.31
DoS Slow Loris	0.67	0.97	0	0.3
FTP Patator	**0.79**	0.53	0.01	0.33
HeartBleed	**0.85**	0.36	**1**	**0**
Infiltration	0.67	1.00	0.05	0.19
Port Scan	0.69	0.90	0.3	0.37
SSH Patator	**0.79**	0.52	0.02	0.33
Web Attack: Brute Force	**0.81**	0.40	0.08	0.19
Web Attack: SQL Injection	**0.83**	0.42	0	0.25
Web Attack: XSS	**0.76**	0.63	0.11	0.22

Note: We consider F1 scores greater than or equal to 0.75 and FPR scores less than or equal to 0.14 to be adequate and represented in bold, with anything else outside of these bounds to be inadequate.

Table 5 Network attack detection observed results (scored at packet level) for supervised learning models trained/tested using supervised dataset. Individual packet header, IP and TCP only.

Traffic label	Random Forest		Logistic Regression		SGD Logistic Regression		SGD Linear SVM	
	Best F1	FPR	Best F1	FPR	Best F1	FPR	Best F1	FPR
Bot	**0.99**	**0.01**	0.62	0.26	0.62	0.27	0.58	0.26
DDoS	**0.99**	**0.01**	0.42	0.26	0.43	0.27	0.47	0.26
DoS GoldenEye	**0.99**	**0.01**	**0.89**	0.15	**0.88**	0.26	**0.92**	0.16
DoS Hulk	**0.99**	**0.01**	**0.90**	0.15	**0.88**	0.26	**0.88**	0.26
DoS Slow HTTP Test	**0.99**	**0.01**	**0.93**	**0.14**	**0.89**	0.26	**0.91**	0.16
DoS Slow Loris	**0.99**	**0.01**	**0.91**	**0.14**	**0.89**	0.26	**0.90**	0.16
FTP Patator	**0.99**	**0.01**	**0.90**	**0.14**	**0.89**	0.26	**0.89**	0.16
HeartBleed	**0.99**	**0.01**	**0.87**	0.26	**0.88**	0.26	**0.93**	0.16
Infiltration	**0.99**	**0.01**	0.57	0.26	0.57	0.26	0.57	0.26
Port Scan	**0.99**	**0.01**	**0.75**	0.26	**0.75**	0.26	**0.75**	0.26
SSH Patator	**0.99**	**0.01**	**0.93**	**0.14**	**0.93**	**0.13**	**0.93**	0.16
Web Attack: Brute Force	**0.99**	**0.01**	**0.93**	**0.14**	**0.94**	**0.13**	**0.93**	0.15
Web Attack: SQL Injection	**1.00**	**0.00**	**0.89**	0.25	**0.89**	0.25	**0.92**	0.17
Web Attack: XSS	**0.99**	**0.01**	**0.93**	0.15	**0.93**	**0.14**	**0.92**	0.16

Note: We consider F1 scores greater than or equal to 0.75 and FPR scores less than or equal to 0.14 to be adequate and represented in bold, with anything else outside of these bounds to be inadequate.

Top Performing Models

Both unsupervised and supervised models are compared. Among the unsupervised models, PCA performed the best in detection of network attacks. Specifically, PCA is best able to detect the *Heartbleed* and *Web Attack: SQL Injection* attacks using five components as shown in the Appendix, Table A-4.

We observe that within the first five components, as shown in the Appendix Table A-16 and visually in Fig. A-5, the features of TCP window size, IP TTL, and TCP option sackok compose approximately 99%, 96%, and 96%, respectively, of these first five components. Despite the PCA model with five components performing the best on these two attacks, overall it performed poorly when considering all attacks. The model that performed the best overall was PCA with nine principal components, closely followed by PCA with eight principal components, but both have such high FPRs that they would perform poorly in practice. The PCA model with seven principal components performed best for three of the malicious labels, and although its average F1 score is lower than other PCA models, its average FPR score is better but still too high for practical use. Overall, the PCA models performed better than the AE model. The AE model correctly classified the *Heartbleed* flows as malicious and performed decently when classifying the *Bot* flows, but otherwise it performed poorly.

Among the supervised models, the Random Forest performed the best, as shown in the Appendix Tables A-7 and A-8 with near perfect performance scores. The rest of the supervised learning algorithms—Logistic Regression, SGD Logistic Regression, and SGD Linear SVM—all performed similarly with high F1 scores but substantially high FPRs. The nonlinearity between the features and maliciousness, and the complexity of this relationship versus the features and maliciousness, may have attributed to the linear methods performing worse. For Random Forest, the most important feature was the IP TTL and TCP window size. Again, this aligns with our unsupervised PCA model analysis and observations made during our exploratory dataset analysis. Recall that in the Fig. 4 heatmaps we observed that the TCP window size and IP TTL features have more correlation with other features for the malicious-only samples compared to benign-only samples.

Further analysis of these two key features and their values in our dataset revealed that our models were learning artifacts specific to the dataset collection to draw their decision boundaries between benign and malicious samples. Features should be used that differentiate benign samples from malicious network attacks based on domain knowledge for maximum model generalization, not based on artifacts specific to the dataset or dataset collection process.

The IP TTL feature value stands out for the malicious samples since most of the attacks are initiated from a single Kali Linux machine in the external network, and all victim hosts were within the internal network. This differs from many of the benign network packets, which were between machines within the internal network, consisting of Linux/Windows machines and one Mac. Furthermore, the default values for both IP TTL and TCP window size differ by operating system and are adjustable, and thus susceptible to modifications by a malicious actor. The default TTL values for modern Linux/Mac is 64, and 128 for Windows.[11] The malicious samples involved attacks mostly from a single Linux machine that has a lower default TTL value, and as they came from the external network (unlike the packets seen in benign samples), more hops occurred that decremented this value further. Therefore, we note these patterns that are artifacts of the dataset generation topology. From a cyber domain perspective, these attack packets should not be distinguishable from normal traffic, both of which can occur internally or externally with either operating system. Thus, these feature values and the relationships modeled between them and other features in this dataset are not consistent, and are not generalizable (i.e., are not transferrable to other datasets/networks).

The feature value separations for IP TTL and TCP window size can also be seen in the mutual information score in Table 6 and empirical distribution plots in Fig. 7. We also observe a higher mutual informational score for the IP total length feature. This also aligns with Fig. 4 of our exploratory dataset analysis, which shows that the IP total length feature's correlations vary between benign-only and malicious-only samples, with the benign-only samples having more significant correlations for this feature. Additionally, TCP sport and dport values can be inconsistent, varying widely between different datasets and real-world networks. Malicious actors may use uncommon ports or common ports dependent on many factors (type of attack, motive, what is allowed or considered normal on the network, etc.), and thus these features and the relationships modeled between them and others in this dataset are also not generalizable. In Table 6, the mutual information score is calculated on the unsupervised learning dataset training data (consisting of 80% of benign Monday data minus 10,000 packets [142,636 flows, 8,005,097 packets]) combined with 10% of the malicious packets from Tuesday–Friday data (43,399 flows, 436,611 packets) randomly sampled. We find similarities between Table 6 and the Random Forest feature importance in Table A-17.

Table 6 Mutual information score per feature for the unsupervised train set and 10% of the malicious samples

Feature	Mutual information score
ip_ttl	0.283
ip_totallen	0.143
tcp_winsize	0.141
tcp_dport	0.124
tcp_sport	0.110
tcp_opt_wscale	0.059
tcp_do	0.051
tcp_opt_sackok	0.040
tcp_opt_mss	0.032
ip_tos	0.006
ip_ihl	0.000
ip_flags	0.000
tcp_flags	0.000

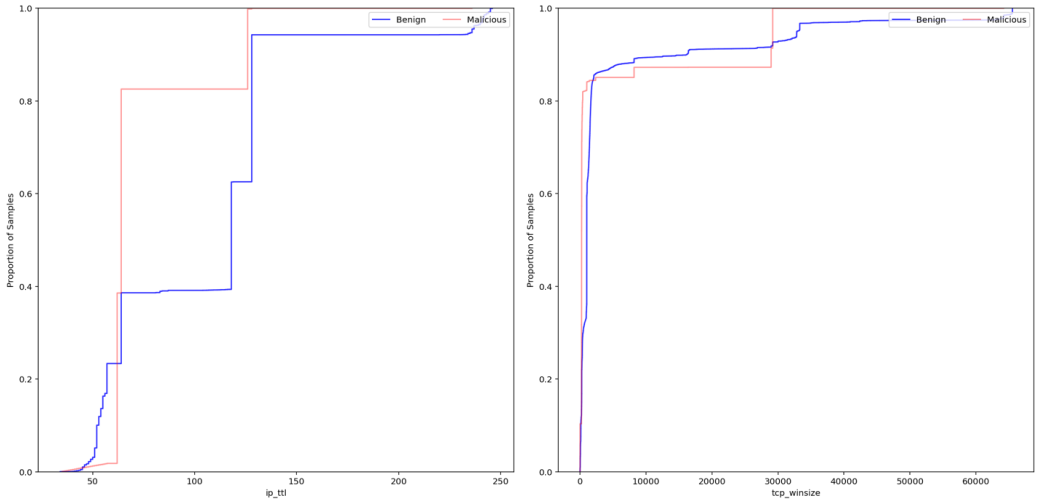


Fig. 7 Empirical distribution plots for (left) ip_ttl and (right) tcp_winsize features

Although the supervised model performed well and the unsupervised model performed adequately, the domain knowledge indicates these features do not separate the network attack from benign samples. Additionally, the empirical cumulative distribution for many of the features show a low contribution toward separability between benign and network attack samples. We believe that for these reasons, discrepancies exist between our expectations in Table 1 and observed results in Tables 4 and 5. Overall, we do not recommend the general use of network header fields as features to distinguish between benign and network attack samples.

For detailed results for each model, see Appendix A.2, Detailed Experimental Results per Algorithm/Model. Additionally, Appendix A.3, Model Explainability

Analysis per Algorithm/Model, includes tables and figures detailing the feature importance and related metrics toward understanding the decision-making process of each model.

4. Conclusion

We use select fields of the IP and TCP header as features and evaluate the use of unsupervised and supervised machine learning network attack detection methods. Specifically, we evaluate the PCA and Auto Encoder methods for unsupervised network attack detection via anomaly detection, and Random Forest, Logistic Regression, and SGD Linear SVM for supervised methods of network attack detection via classification. We observe PCA as the best model in the unsupervised case, and the Random Forest model in the supervised case is superior to all the models evaluated. However, these models learn patterns in the features that are not indicative of a network attack, but artifacts of the network used for dataset collection, which are not generalizable. These models rely heavily on features that have differing default values based on operating system and are easily modifiable by a malicious actor. For maximum model generalization, features should be used that differentiate benign samples from malicious network attack samples based on domain knowledge and that strongly connect to malice, and are not modifiable by a malicious actor. Additionally, we note the drawbacks of using an anomaly detection-based approach for benign-malicious classification when there exist anomalies that are considered benign and conversely attacks that would not be considered anomalous. We also note the intricacies and caveats introduced in labeling when malice can reside at one or more levels of network traffic (e.g., header, packet payload, entire flow with headers and payloads, multiple entire flows). We discuss how the relationship between these header features and malice is nonlinear and more complex than the relationship between the features and anomalousness, and how with anomaly detection it is only possible to learn the latter. Thus, we do not recommend the use of network header fields for the general detection of network attacks. Rather we advise the use of raw bytes of the payload in unencrypted network traffic for detection, based on prior studies.[1]

5. References

1. De Lucia M, Maxwell PE, Bastian ND, Swami A, Jalaian B, Leslie N. Machine learning raw network traffic detection. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III; 2021 Apr. p. 24. doi: 10.1117/12.2586114.

2. Wang W, Zhu M, Zeng X, Ye X, Sheng Y. Malware traffic classification using convolutional neural network for representation learning. 2017 International Conference on Information Networking (ICOIN); 2017; Da Nang, Vietnam. p. 712–717. doi: 10.1109/ICOIN.2017.7899588.

3. Wang W, Zhu M, Wang J, Zeng X, Yang Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), 2017 Jul; Beijing, China. p. 43–48. Doi: 10.1109/ISI.2017.8004872.

4. Zeng Y, Gu H, Wei W, Guo Y. Deep-full-range : a deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access. 2019;7:45182–45190. doi: 10.1109/ACCESS.2019.2908225.

5. Lotfollahi M, Siavoshani MJ, Shirali Hossein Zade R, Saberian M. Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Comput. 2020;24:1999–2012. doi:10.1007/s00500-019-04030-2.

6. Marín G, Caasas P, Capdehourat G. Deepmal-deep learning models for malware traffic detection and classification. In: Haber P, Lampoltshammer T, Mayr M, Plankensteiner K, editors. Data science–analytics and applications. Springer Vieweg, Wiesbaden; 2021. p. 105–112, doi:10.1007/978-3-658-32182-6_16.

7. Holland J, Schmitt P, Feamster N, Mittal P. New directions in automated traffic analysis. Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security; 2021 Nov; Virtual Event Republic of Korea. p. 3366–3383. doi: 10.1145/3460120.3484758.

8. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. 4th International Conference on Information Systems Security and Privacy (ICISSP); 2018 Jan; Portugal.

9. Lippmann R, Haines JW, Fried DJ, Korba J, Das K. The 1999 DARPA off-line intrusion detection evaluation. Comput Netw. 2000 Oct;34(4):579–595.

10. Engelen G, Rimmer V, Joosen W. Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. 2021 IEEE Security and Privacy Workshops (SPW); 2021. p. 7–12. doi: 10.1109/SPW53761.2021.00009.

11. Sk. How to identify operating system using TTL value and Ping command. OSTechNix; 2021 Sep 20 [accessed 2022 Nov 1]. https://ostechnix.com/identify-operating-system-ttl-ping/.

Appendix. Datasets, Results, Models, and Analysis Further Detailed

This appendix further details the datasets used in this research, our analysis, and results. In Tables A-1 and A-2 we provide a breakdown of the datasets used for unsupervised and supervised model experiments, including the packet and/or flow sample counts per label/set. Figure A-1 shows a visual breakdown of the supervised learning baseline dataset. Section A.1 contains additional exploratory analysis of the datasets using Principal Component Analysis (PCA). Section A.2 contains detailed experimental results of each algorithm/model. Section A.3 contains model explainability analysis data and includes tables and figures detailing the feature importance and related metrics toward understanding the decision-making process of each model.

Table A-1 Breakdown of unsupervised dataset packet/sample and flow counts per label, used for unsupervised learning model experiments (Principal Component Analysis [PCA], Autoencoder [AE]), consisting of baseline, validation, train, and test sets

Dataset	Label	pkt_sample_count	complete_flow_sample_count
unsupervised-baseline	BENIGN (Monday)	2003774	134770
unsupervised-validation	BENIGN (Monday)	10000	4676
unsupervised-training	BENIGN (Monday)	8005096	142626
unsupervised-testing	BENIGN (Tuesday)	10435402	114869
unsupervised-testing	BENIGN (Wednesday)	10454369	115926
unsupervised-test	BENIGN (Thursday)	8436954	175962
unsupervised-test	BENIGN (Friday)	7583846	102249
unsupervised-test	Bot (Friday)	9913	738
unsupervised-test	DDoS (Friday)	1273114	95123
unsupervised-test	DoSGoldenEye (Wednesday)	105874	7567
unsupervised-test	DoSHulk (Wednesday)	2243336	115926
unsupervised-test	DoSSlowhttptest (Wednesday)	17766	1742
unsupervised-test	DoSslowloris (Wednesday)	42351	4001
unsupervised-test	FTP-Patator (Tuesday)	111035	3973
unsupervised-test	Heartbleed (Wednesday)	49296	11
unsupervised-test	Infiltration (Thursday)	59722	32
unsupervised-test	PortScan (Friday)	321566	102249
unsupervised-test	SSH-Patator (Tuesday)	163304	2980
unsupervised-test	WebAttack-BruteForce (Thursday)	22937	151
unsupervised-test	WebAttack-SqlInjection (Thursday)	126	12
unsupervised-test	WebAttack-XSS (Thursday)	5710	27

Table A-2 Breakdown of supervised dataset sample counts per label, used for supervised learning model experiments (Random Forest, Logistic Regression, Stochastic Gradient Descent [SGD] Logistic Regression, and SGD Linear Support Vector Machine [SVM]), consisting of baseline, train, and test sets

Dataset	Label	pkt_sample_count	Dataset	Label	pkt_sample_count
supervised-baseline	BENIGN	100000	supervised-test1	FTP-Patator	60477
supervised-train1	BENIGN	800000	supervised-test2	FTP-Patator	60477
supervised-train2	BENIGN	400000	supervised-baseline	Heartbleed	459
supervised-train3	BENIGN	300000	supervised-train1	Heartbleed	3021
supervised-train4	BENIGN	200000	supervised-train2	Heartbleed	3021
supervised-train5	BENIGN	100000	supervised-train3	Heartbleed	2016
supervised-test1	BENIGN	10000000	supervised-train4	Heartbleed	1022
supervised-test2	BENIGN	3926052	supervised-train5	Heartbleed	529
supervised-baseline	Bot	167	supervised-test1	Heartbleed	45816
supervised-train1	Bot	605	supervised-test2	Heartbleed	45816
supervised-train2	Bot	605	supervised-baseline	Infiltration	16827
supervised-train3	Bot	459	supervised-train1	Infiltration	33759
supervised-train4	Bot	326	supervised-train2	Infiltration	33759
supervised-train5	Bot	163	supervised-train3	Infiltration	33759
supervised-test1	Bot	9141	supervised-train4	Infiltration	33759
supervised-test2	Bot	9141	supervised-train5	Infiltration	16875
supervised-baseline	DDoS	19800	supervised-test1	Infiltration	9136
supervised-train1	DDoS	79373	supervised-test2	Infiltration	9136
supervised-train2	DDoS	79373	supervised-baseline	PortScan	5033
supervised-train3	DDoS	59480	supervised-train1	PortScan	20022
supervised-train4	DDoS	39657	supervised-train2	PortScan	20022
supervised-train5	DDoS	19813	supervised-train3	PortScan	15061
supervised-test1	DDoS	1173941	supervised-train4	PortScan	10017
supervised-test2	DDoS	1173941	supervised-train5	PortScan	5024
supervised-baseline	DoSGoldenEye	1056	supervised-test1	PortScan	296511
supervised-train1	DoSGoldenEye	6380	supervised-test2	PortScan	296511
supervised-train2	DoSGoldenEye	6380	supervised-baseline	SSH-Patator	14907
supervised-train3	DoSGoldenEye	4260	supervised-train1	SSH-Patator	59535
supervised-train4	DoSGoldenEye	2124	supervised-train2	SSH-Patator	59535
supervised-train5	DoSGoldenEye	1037	supervised-train3	SSH-Patator	44631
supervised-test1	DoSGoldenEye	98438	supervised-train4	SSH-Patator	29896
supervised-test2	DoSGoldenEye	98438	supervised-train5	SSH-Patator	14991
supervised-baseline	DoSHulk	22869	supervised-test1	SSH-Patator	88862
supervised-train1	DoSHulk	136890	supervised-test2	SSH-Patator	88862
supervised-train2	DoSHulk	136890	supervised-baseline	WebAttack-BruteForce	6544
supervised-train3	DoSHulk	91290	supervised-train1	WebAttack-BruteForce	12909
supervised-train4	DoSHulk	45647	supervised-train2	WebAttack-BruteForce	12909
supervised-train5	DoSHulk	22824	supervised-train3	WebAttack-BruteForce	12909
supervised-test1	DoSHulk	2083577	supervised-train4	WebAttack-BruteForce	12909
supervised-test2	DoSHulk	2083577	supervised-train5	WebAttack-BruteForce	6444
supervised-baseline	DoSSlowhttptest	172	supervised-test1	WebAttack-BruteForce	3484
supervised-train1	DoSSlowhttptest	1111	supervised-test2	WebAttack-BruteForce	3484
supervised-train2	DoSSlowhttptest	1111	supervised-baseline	WebAttack-SqlInjection	40
supervised-train3	DoSSlowhttptest	737	supervised-train1	WebAttack-SqlInjection	74
supervised-train4	DoSSlowhttptest	370	supervised-train2	WebAttack-SqlInjection	74
supervised-train5	DoSSlowhttptest	181	supervised-train3	WebAttack-SqlInjection	74
supervised-test1	DoSSlowhttptest	16483	supervised-train4	WebAttack-SqlInjection	74
supervised-test2	DoSSlowhttptest	16483	supervised-train5	WebAttack-SqlInjection	35
supervised-baseline	DoSslowloris	444	supervised-test1	WebAttack-SqlInjection	12
supervised-train1	DoSslowloris	2598	supervised-test2	WebAttack-SqlInjection	12
supervised-train2	DoSslowloris	2598	supervised-baseline	WebAttack-XSS	1589
supervised-train3	DoSslowloris	1697	supervised-train1	WebAttack-XSS	3258
supervised-train4	DoSslowloris	837	supervised-train2	WebAttack-XSS	3258
supervised-train5	DoSslowloris	429	supervised-train3	WebAttack-XSS	3258
supervised-test1	DoSslowloris	39309	supervised-train4	WebAttack-XSS	3258
supervised-test2	DoSslowloris	39309	supervised-train5	WebAttack-XSS	1646
supervised-baseline	FTP-Patator	10093	supervised-test1	WebAttack-XSS	863
supervised-train1	FTP-Patator	40465	supervised-test2	WebAttack-XSS	863
supervised-train2	FTP-Patator	40465			
supervised-train3	FTP-Patator	30369			
supervised-train4	FTP-Patator	20104			
supervised-train5	FTP-Patator	10009			

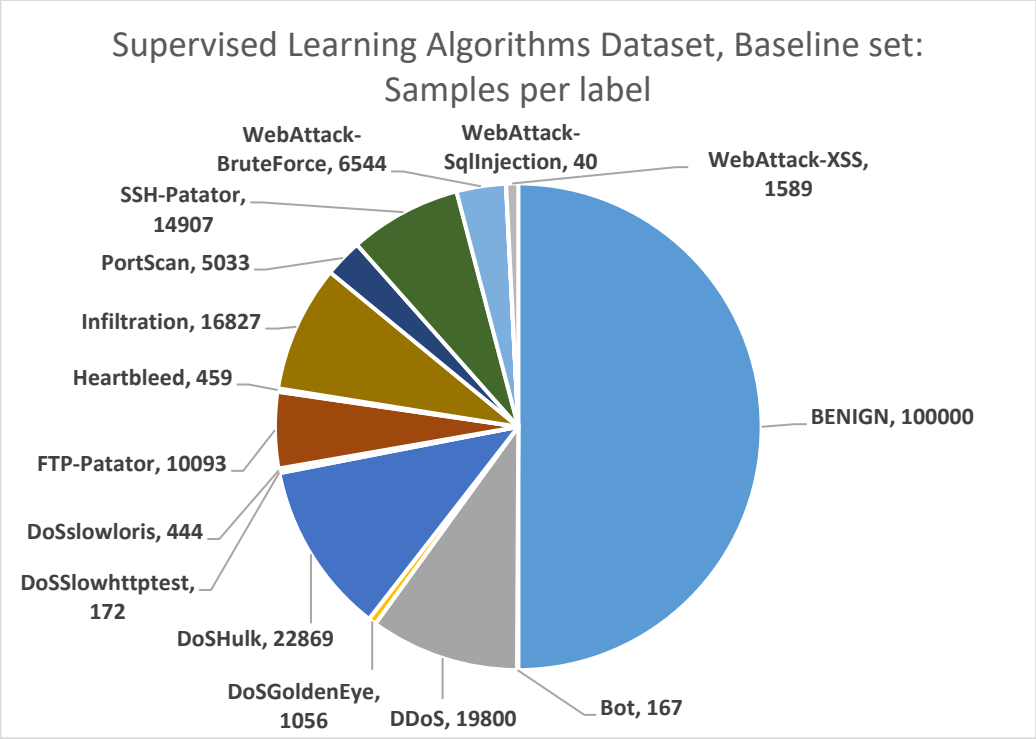


Fig. A-1 Supervised learning baseline dataset breakdown of samples per label

A.1 Further Exploratory Dataset Analysis via PCA

In Figs. A-2 and A-3 we further analyze the difference between the unsupervised benign training set and the example malicious training set by looking at how PCA reduces the dimensionality of the features into principal components, which features make up each principal component, and thus which features best describe the variance of the data. The example malicious training set was developed only to better understand the malicious data features and their relation to the benign data features, and consisted of 10% of our malicious Tuesday–Friday data (43,399 flows, 436,611 packets) randomly sampled. In Fig. A-2 we see that for both the benign training set and example malicious training set, PCA determined that the Transmission Control Protocol (TCP) source and destination ports best described the variance in the data and oriented the first principal component to these. From here the features that the principal components orient themselves to diverge when comparing the benign and malicious training set data. The second principal component for the benign data is exclusively oriented to the TCP option sackok while its counterpart for the malicious data is exclusively aligned to the TCP window size feature. Figure A-3 shows that significantly more of the variance in the malicious data can be explained with only one principal component, compared to the benign data.

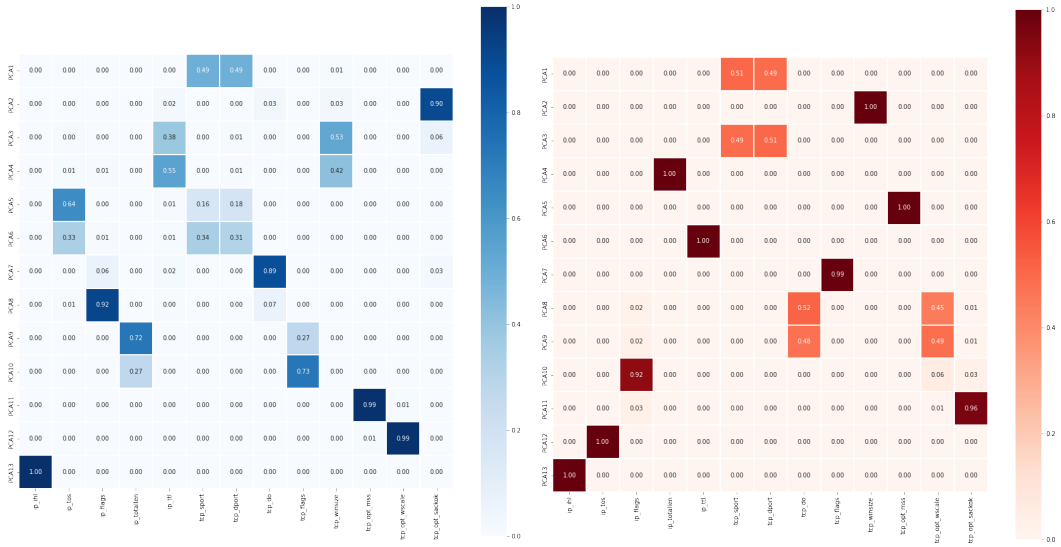


Fig. A-2 Visual heatmap depiction of the amount that each feature is used by each PCA principal component, obtained by squaring the unit eigenvectors, for (left) model trained on the unsupervised benign training set and (right) example malicious training set

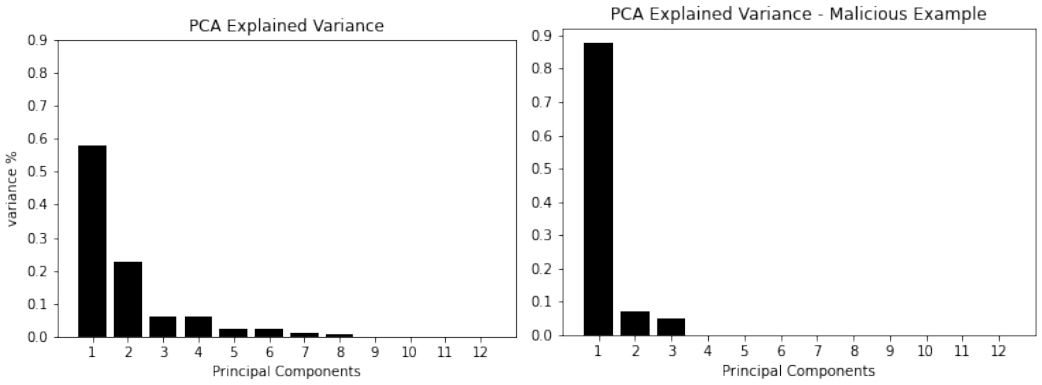


Fig. A-3 PCA models' explained variance graphs for (left) model trained on the unsupervised benign training set and (right) example malicious training set

A.2 Detailed Experimental Results per Algorithm/Model

A.2.1 Principal Component Analysis

Table A-3 PCA models' results: average F1 scores per model

Best model (average F1 score)	Average F1 Score	Average FPR
PCA_9_pcs	0.67	0.97
PCA_8_pcs	0.66	0.97
PCA_1_pcs	0.63	0.80
PCA_10_pcs	0.63	0.94
PCA_11_pcs	0.62	0.84
PCA_7_pcs	0.59	0.57
PCA_12_pcs	0.58	0.68
PCA_6_pcs	0.44	0.52
PCA_5_pcs	0.38	0.43
PCA_4_pcs	0.27	0.42
PCA_3_pcs	0.10	0.54
PCA_2_pcs	0.09	0.56

Note: FPR, false positive rate

Table A-4 PCA models' results: best scoring models (F1 score) per label

Best model per label	Label	PCA file	Benign flows	Malicious flows	Threshold	Precision	Recall	F1	Specificity	FPR	False positives	False negatives
PCA_10_pcs	All Malicious TUE-FRI	PCA_10_pcs_output_custom_P90_MAE_MAX-SCORE-PER-FLOW_TUE-FRI_downsampled.csv	334532	334532	0	0.52	0.99	0.69	0.09	0.91	303058	2139
PCA_1_pcs	Bot	PCA_1_pcs_output_custom_MAE_Friday.csv	738	738	0	0.54	1.00	0.70	0.14	0.86	633	0
PCA_1_pcs	DDoS	PCA_1_pcs_output_custom_MAE_Friday.csv	95123	95123	0	0.54	1.00	0.70	0.14	0.86	81456	0
PCA_7_pcs	DoS GoldenEye	PCA_7_pcs_output_custom_MAE_Wednesday.csv	7567	7567	0	0.65	0.95	0.77	0.49	0.51	3871	351
PCA_7_pcs	DoS Hulk	PCA_7_pcs_output_custom_MAE_Wednesday.csv	115926	115926	0	0.64	0.92	0.75	0.48	0.52	59986	9349
PCA_7_pcs	DoS Slowhttptest	PCA_7_pcs_output_custom_MAE_Wednesday.csv	1742	1742	0	0.60	0.80	0.68	0.46	0.54	935	348
PCA_9_pcs	DoS slowloris	PCA_9_pcs_output_custom_MAE_Wednesday.csv	4001	4001	0	0.51	1.00	0.67	0.03	0.97	3861	0
PCA_6_pcs	FTP-Patator	PCA_6_pcs_output_custom_MAE_Tuesday.csv	3973	3973	0	0.66	0.99	0.79	0.47	0.53	2089	5
PCA_5_pcs	Heartbleed	PCA_5_pcs_output_custom_MAE_Wednesday.csv	11	11	0	0.73	1.00	0.85	0.64	0.36	4	0
PCA_8_pcs	Infiltration	PCA_8_pcs_output_custom_MAE_Thursday.csv	32	32	0	0.50	1.00	0.67	0.00	1.00	32	0
PCA_10_pcs	PortScan	PCA_10_pcs_output_custom_MAE_Friday.csv	102249	102249	0	0.53	0.99	0.69	0.10	0.90	92335	79
PCA_6_pcs	SSH-Patator	PCA_6_pcs_output_custom_MAE_Tuesday.csv	2980	2980	0	0.65	0.99	0.79	0.48	0.52	1554	34
PCA_12_pcs	Web Attack - Brute Force	PCA_12_pcs_output_custom_MAE_Thursday.csv	151	151	0	0.70	0.95	0.81	0.60	0.40	61	8
PCA_5_pcs	Web Attack - Sql Injection	PCA_5_pcs_output_custom_MAE_Thursday.csv	12	12	0	0.71	1.00	0.83	0.58	0.42	5	0
PCA_1_pcs	Web Attack - XSS	PCA_1_pcs_output_custom_MAE_Thursday.csv	27	27	0	0.61	1.00	0.76	0.37	0.63	17	0
PCA_11_pcs	Web Attack - XSS	PCA_11_pcs_output_custom_MAE_Thursday.csv	27	27	0	0.61	1.00	0.76	0.37	0.63	17	0

A.2.2 Autoencoder

Table A-5 AE model's results average F1 scores per model (only one)

Best model (average F1 score)	Average F1 score	Average FPR
AE_output_MAE	0.22	0.27

Table A-6 AE model results: best scoring models (F1 score) per label

Best model per label	Label	AE test set file	Benign flows	Malicious flows	Threshold	Precision	Recall (sensitivity)	F1	Specificity	FPR	False positives	False negatives
AE_output_MAE	Bot	AE_output_MAE_Friday.csv	738	738	0	0.70	0.84	0.77	0.64	0.36	264	115
AE_output_MAE	DDoS	AE_output_MAE_Friday.csv	95123	95123	0	0.43	0.28	0.33	0.63	0.37	35365	68899
AE_output_MAE	DoS GoldenEye	AE_output_MAE_Wednesday.csv	7567	7567	0	0.29	0.12	0.17	0.69	0.31	2331	6632
AE_output_MAE	DoS Hulk	AE_output_MAE_Wednesday.csv	115926	115926	0	0.35	0.16	0.22	0.69	0.31	35536	96819
AE_output_MAE	DoS Slowhttptest	AE_output_MAE_Wednesday.csv	1742	1742	0	0.01	0.00	0.01	0.69	0.31	541	1736
AE_output_MAE	DoS slowloris	AE_output_MAE_Wednesday.csv	4001	4001	0	0.01	0.00	0.00	0.70	0.30	1218	3992
AE_output_MAE	FTP-Patator	AE_output_MAE_Tuesday.csv	3973	3973	0	0.01	0.00	0.01	0.67	0.33	1293	3955
AE_output_MAE	Heartbleed	AE_output_MAE_Wednesday.csv	11	11	0	1.00	1.00	1.00	1.00	0.00	0	0
AE_output_MAE	Infiltration	AE_output_MAE_Thursday.csv	32	32	0	0.14	0.03	0.05	0.81	0.19	6	31
AE_output_MAE	PortScan	AE_output_MAE_Friday.csv	102249	102249	0	0.40	0.25	0.30	0.63	0.37	38082	77175
AE_output_MAE	SSH-Patator	AE_output_MAE_Tuesday.csv	2980	2980	0	0.04	0.01	0.02	0.67	0.33	981	2939
AE_output_MAE	Web Attack - Brute Force	AE_output_MAE_Thursday.csv	151	151	0	0.20	0.05	0.08	0.81	0.19	28	144
AE_output_MAE	Web Attack - Sql Injection	AE_output_MAE_Thursday.csv	12	12	0	0.00	0.00	0.00	0.75	0.25	3	12
AE_output_MAE	Web Attack - XSS	AE_output_MAE_Thursday.csv	27	27	0	0.25	0.07	0.11	0.78	0.22	6	25

A.2.3 Random Forest

Table A-7 Random Forest models' results: average F1 scores per model for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset (average F1 score)	Average F1 score	Average FPR
train1-test1	0.99	0.01
train1-test2	0.99	0.01
train2-test1	0.99	0.01
train2-test2	0.99	0.01
train3-test1	0.99	0.01
train3-test2	0.99	0.01
train4-test1	0.99	0.01
train4-test2	0.99	0.01
train5-test1	0.99	0.01
train5-test2	0.99	0.01

Table A-8 Random Forest models' results: best scoring models (F1 score) per label for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset per label	Label	Training set malicious samples	Training set benign samples	Test set malicious samples	Test set benign samples	Precision	Recall	F1	Specificity	FPR	False positives	False negatives
all models equivalent [a]	Bot	400000	400000	9141	9141	0.99	0.99	0.99	0.99	0.01	73 - 76	25 - 38
all models equivalent	DDoS	400000	800000	1173941	1173941	0.99	0.99	0.99	0.99	0.01	7771 - 7997	8 - 23
all models equivalent	DoS GoldenEye	400000	800000	98438	98438	0.99	1.00	0.99	0.99	0.01	652 - 723	0
all models equivalent	DoS Hulk	400000	800000	2083577	2083577	0.99	1.00	0.99	0.99	0.01	13697 - 14123	0
all models equivalent	DoS Slowhttptest	400000	800000	16483	16483	0.99	1.00	0.99	0.99	0.01	123 - 128	0
all models equivalent	DoS slowloris	400000	800000	39309	39309	0.99	1.00	0.99	0.99	0.01	264 - 299	0
all models equivalent	FTP-Patator	100000	100000	60477	60477	0.99	0.99	0.99	0.99	0.01	413 - 430	89 - 159
all models equivalent	Heartbleed	400000	800000	45816	45816	0.99	0.99-1.00	0.99	0.99	0.01	323 - 336	0 - 15
all models equivalent	Infiltration	400000	800000	9136	9136	0.99	0.99-1.00	0.99	0.99	0.01	73 - 76	0 - 2
all models equivalent	PortScan	400000	400000	296511	296511	0.99	0.99	0.99	0.99	0.01	1909 - 2117	166 - 221
all models equivalent	SSH-Patator	300000	300000	88862	88862	0.99	0.99	0.99	0.99	0.01	595 - 641	47 - 131
all models equivalent	Web Attack - Brute Force	400000	800000	3484	3484	0.99	1.00	0.99	0.99	0.01	21 - 29	0
all models equivalent	Web Attack - Sql Injection	400000	800000	12	12	1.00	1.00	1.00	1.00	0.00	0	0
all models equivalent	Web Attack - XSS	400000	800000	863	863	0.99	1.00	0.99	0.99	0.01	6 - 7	0

[a] (train5-test1 and train5-test2 had slight differences: 0.98 F1, 0.97 Recall, 73-74 False Positives, 284 False Negatives)
all models = [train1-test1, train1-test2, train2-test1, train2-test2, train3-test1, train3-test2, train4-test1, train4-test2, train5-test1, train5-test2]

A.2.4 Logistic Regression

Table A-9 Logistic Regression models' results: average F1 scores per model for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset (average F1 score)	Average F1 score	Average FPR
train3-test1	0.80	0.26
train3-test2	0.80	0.26
train2-test1	0.80	0.26
train2-test2	0.79	0.26
train4-test1	0.78	0.26
train5-test1	0.78	0.26
train4-test2	0.78	0.26
train5-test2	0.78	0.26
train1-test1	0.67	0.15
train1-test2	0.67	0.15

Table A-10 Logistic Regression models' results: best scoring models (F1 score) per label for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset per label	Label	Training set malicious samples	Training set benign samples	Test set malicious samples	Test set benign samples	Precision	Recall	F1	Specificity	FPR	False positives	False negatives
train2-test1	Bot	400000	400000	9141	9141	0.68	0.56	0.62	0.74	0.26	2405	4001
train2-test2	Bot	400000	400000	9141	9141	0.68	0.56	0.62	0.74	0.26	2388	4001
train3-test1	Bot	300000	300000	9141	9141	0.68	0.56	0.62	0.74	0.26	2407	4001
train3-test2	Bot	300000	300000	9141	9141	0.68	0.56	0.62	0.74	0.26	2383	4001
train3-test1	DDoS	300000	300000	1173941	1173941	0.57	0.34	0.42	0.74	0.26	306164	775670
train3-test2	DDoS	300000	300000	1173941	1173941	0.57	0.34	0.42	0.74	0.26	305670	775670
train1-test1	DoS GoldenEye	400000	800000	98438	98438	0.86	0.93	0.89	0.85	0.15	14409	7310
train1-test2	DoS GoldenEye	400000	800000	98438	98438	0.86	0.93	0.89	0.85	0.15	14292	7310
train1-test1	DoS Hulk	400000	800000	2083577	2083577	0.86	0.95	0.90	0.85	0.15	308513	111950
train1-test2	DoS Hulk	400000	800000	2083577	2083577	0.87	0.95	0.90	0.85	0.15	306729	111950
train1-test2	DoS Slowhttptest	400000	800000	16483	16483	0.87	0.99	0.93	0.86	0.14	2378	213
train1-test2	DoS slowloris	400000	800000	39309	39309	0.87	0.95	0.91	0.86	0.14	5593	2050
train1-test2	FTP-Patator	400000	800000	60477	60477	0.87	0.93	0.90	0.86	0.14	8747	4401
train2-test1	Heartbleed	400000	400000	45816	45816	0.79	0.97	0.87	0.74	0.26	11945	1219
train2-test2	Heartbleed	400000	400000	45816	45816	0.79	0.97	0.87	0.74	0.26	11892	1219
train3-test1	Heartbleed	300000	300000	45816	45816	0.79	0.96	0.87	0.74	0.26	11917	1809
train3-test2	Heartbleed	300000	300000	45816	45816	0.79	0.96	0.87	0.74	0.26	11863	1809
train2-test1	Infiltration	400000	400000	9136	9136	0.65	0.50	0.57	0.74	0.26	2404	4582
train2-test2	Infiltration	400000	400000	9136	9136	0.66	0.50	0.57	0.74	0.26	2387	4582
train3-test1	Infiltration	300000	300000	9136	9136	0.65	0.50	0.57	0.74	0.26	2406	4582
train3-test2	Infiltration	300000	300000	9136	9136	0.66	0.50	0.57	0.74	0.26	2382	4582
train4-test1	Infiltration	200000	200000	9136	9136	0.66	0.50	0.57	0.74	0.26	2362	4582
train4-test2	Infiltration	200000	200000	9136	9136	0.66	0.50	0.57	0.74	0.26	2335	4582
train5-test1	Infiltration	100000	100000	9136	9136	0.66	0.50	0.57	0.74	0.26	2362	4582
train5-test2	Infiltration	100000	100000	9136	9136	0.66	0.50	0.57	0.74	0.26	2332	4582
train2-test1	PortScan	400000	400000	296511	296511	0.74	0.75	0.75	0.74	0.26	77354	72875
train2-test2	PortScan	400000	400000	296511	296511	0.74	0.75	0.75	0.74	0.26	77355	72875
train3-test1	PortScan	300000	300000	296511	296511	0.74	0.75	0.75	0.74	0.26	77133	72878
train3-test2	PortScan	300000	300000	296511	296511	0.74	0.75	0.75	0.74	0.26	77152	72878
train4-test1	PortScan	200000	200000	296511	296511	0.75	0.75	0.75	0.74	0.26	75719	72907
train4-test2	PortScan	200000	200000	296511	296511	0.75	0.75	0.75	0.74	0.26	75780	72907
train5-test1	PortScan	100000	100000	296511	296511	0.75	0.75	0.75	0.74	0.26	75712	72960
train5-test2	PortScan	100000	100000	296511	296511	0.75	0.75	0.75	0.74	0.26	75753	72960
train1-test2	SSH-Patator	400000	800000	88862	88862	0.87	0.99	0.93	0.86	0.14	12839	22
train1-test2	Web Attack - Brute Force	400000	800000	3484	3484	0.88	0.99	0.93	0.86	0.14	489	2
train2-test1	Web Attack - Sql Injection	400000	400000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train3-test1	Web Attack - Sql Injection	300000	300000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train4-test1	Web Attack - Sql Injection	200000	200000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train5-test1	Web Attack - Sql Injection	100000	100000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train1-test1	Web Attack - XSS	400000	800000	863	863	0.87	1.00	0.93	0.85	0.15	131	0
train1-test2	Web Attack - XSS	400000	800000	863	863	0.87	1.00	0.93	0.85	0.15	131	0

A.2.5 SGD Logistic Regression

Table A-11 SGD Logistic Regression models' results: average F1 scores per model for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset (average F1 score)	Average F1 score	Average FPR
train5-test1	0.79	0.27
train5-test2	0.79	0.27
train2-test1	0.79	0.26
train2-test2	0.79	0.27
train3-test1	0.79	0.26
train3-test2	0.78	0.26
train4-test1	0.78	0.26
train4-test2	0.78	0.26
train1-test1	0.65	0.14
train1-test2	0.65	0.14

Table A-12 SGD Logistic Regression models' results: best scoring models (F1 score) per label for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset per label	Label	Training set malicious samples	Training set benign samples	Test Set malicious samples	Test Set benign samples	Precision	Recall	F1	Specificity	FPR	False positives	False negatives
train2-test2	Bot	400000	400000	9141	9141	0.68	0.56	0.62	0.73	0.27	2425	4001
train5-test1	DDoS	100000	100000	1173941	1173941	0.56	0.34	0.43	0.73	0.27	312857	769111
train5-test2	DDoS	100000	100000	1173941	1173941	0.56	0.34	0.43	0.73	0.27	312168	769111
train2-test2	DoS GoldenEye	400000	400000	98438	98438	0.79	0.99	0.88	0.74	0.26	26004	679
train3-test1	DoS GoldenEye	300000	300000	98438	98438	0.79	0.99	0.88	0.74	0.26	25614	791
train3-test2	DoS GoldenEye	300000	300000	98438	98438	0.79	0.99	0.88	0.74	0.26	25540	791
train3-test1	DoS Hulk	300000	300000	2083577	2083577	0.79	0.99	0.88	0.74	0.26	543258	6794
train3-test2	DoS Hulk	300000	300000	2083577	2083577	0.79	0.99	0.88	0.74	0.26	542002	6794
train4-test1	DoS Hulk	200000	200000	2083577	2083577	0.79	0.98	0.88	0.74	0.26	535848	35692
train4-test2	DoS Hulk	200000	200000	2083577	2083577	0.79	0.98	0.88	0.74	0.26	534544	35692
train4-test1	DoS Slowhttptest	200000	200000	16483	16483	0.79	1.00	0.89	0.74	0.26	4261	0
train4-test2	DoS Slowhttptest	200000	200000	16483	16483	0.79	1.00	0.89	0.74	0.26	4269	0
train3-test1	DoS slowloris	300000	300000	39309	39309	0.79	1.00	0.89	0.74	0.26	10182	0
train3-test2	DoS slowloris	300000	300000	39309	39309	0.79	1.00	0.89	0.74	0.26	10190	0
train4-test1	DoS slowloris	200000	200000	39309	39309	0.80	1.00	0.89	0.74	0.26	10041	0
train4-test2	DoS slowloris	200000	200000	39309	39309	0.80	1.00	0.89	0.74	0.26	10063	0
train3-test2	FTP-Patator	300000	300000	60477	60477	0.79	0.99	0.89	0.74	0.26	15680	4
train4-test1	FTP-Patator	200000	200000	60477	60477	0.80	0.99	0.89	0.74	0.26	15563	4
train4-test2	FTP-Patator	200000	200000	60477	60477	0.80	0.99	0.89	0.74	0.26	15484	4
train2-test1	Heartbleed	400000	400000	45816	45816	0.79	0.99	0.88	0.74	0.26	12132	457
train2-test2	Heartbleed	400000	400000	45816	45816	0.79	0.99	0.88	0.74	0.26	12067	457
train3-test1	Heartbleed	300000	300000	45816	45816	0.79	0.98	0.88	0.74	0.26	11909	825
train3-test2	Heartbleed	300000	300000	45816	45816	0.79	0.98	0.88	0.74	0.26	11872	825
train3-test1	Infiltration	300000	300000	9136	9136	0.65	0.50	0.57	0.74	0.26	2400	4583
train3-test2	Infiltration	300000	300000	9136	9136	0.66	0.50	0.57	0.74	0.26	2385	4583
train4-test1	Infiltration	200000	200000	9136	9136	0.66	0.50	0.57	0.74	0.26	2366	4583
train4-test2	Infiltration	200000	200000	9136	9136	0.66	0.50	0.57	0.74	0.26	2348	4583
train3-test1	PortScan	300000	300000	296511	296511	0.74	0.75	0.75	0.74	0.26	77098	72995
train3-test2	PortScan	300000	300000	296511	296511	0.74	0.75	0.75	0.74	0.26	77092	72995
train4-test1	PortScan	200000	200000	296511	296511	0.75	0.75	0.75	0.74	0.26	76019	73057
train4-test2	PortScan	200000	200000	296511	296511	0.75	0.75	0.75	0.74	0.26	76076	73057
train1-test1	SSH-Patator	400000	800000	88862	88862	0.88	0.98	0.93	0.87	0.13	11940	1683
train1-test2	SSH-Patator	400000	800000	88862	88862	0.88	0.98	0.93	0.87	0.13	11797	1683
train1-test2	Web Attack - Brute Force	400000	800000	3484	3484	0.89	0.99	0.94	0.87	0.13	442	27
train2-test1	Web Attack - Sql Injection	400000	400000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train3-test1	Web Attack - Sql Injection	300000	300000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train4-test1	Web Attack - Sql Injection	200000	200000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train5-test1	Web Attack - Sql Injection	100000	100000	12	12	0.80	1.00	0.89	0.75	0.25	3	0
train1-test1	Web Attack - XSS	400000	800000	863	863	0.87	0.99	0.93	0.86	0.14	124	6
train1-test2	Web Attack - XSS	400000	800000	863	863	0.88	0.99	0.93	0.86	0.14	120	6

A.2.6 SGD Linear SVM

Table A-13 SGD Linear SVM models' results: average F1 scores per model for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset (average F1 score)	Average F1 score	Average FPR
train2-test1	0.80	0.27
train2-test2	0.80	0.27
train3-test1	0.80	0.27
train3-test2	0.80	0.27
train4-test1	0.80	0.26
train5-test2	0.80	0.27
train5-test1	0.80	0.27
train4-test2	0.80	0.26
train1-test1	0.70	0.16
train1-test2	0.69	0.16

Table A-14 SGD Linear SVM models' results: best scoring models (F1 score) per label for models trained on the specified number of packets. Scored at the packet level.

Best model/test dataset per label	Label	Training set malicious samples	Training set benign samples	Test Set malicious samples	Test Set benign samples	Precision	Recall	F1	Specificity	FPR	False positives	False negatives
train2-test2	Bot	400000	400000	9141	9141	0.66	0.51	0.58	0.74	0.26	2408	4434
train4-test2	Bot	200000	200000	9141	9141	0.66	0.51	0.58	0.74	0.26	2398	4434
train2-test2	DDoS	400000	400000	1173941	1173941	0.59	0.39	0.47	0.74	0.26	310454	721712
train1-test1	DoS GoldenEye	400000	800000	98438	98438	0.86	0.99	0.92	0.84	0.16	15708	337
train1-test2	DoS GoldenEye	400000	800000	98438	98438	0.86	0.99	0.92	0.84	0.16	15586	337
train2-test2	DoS Hulk	400000	400000	2083577	2083577	0.79	1.00	0.88	0.74	0.26	551204	0
train4-test1	DoS Hulk	200000	200000	2083577	2083577	0.79	0.99	0.88	0.74	0.26	548446	15004
train4-test2	DoS Hulk	200000	200000	2083577	2083577	0.79	0.99	0.88	0.74	0.26	546961	15004
train1-test2	DoS Slowhttptest	400000	800000	16483	16483	0.86	0.96	0.91	0.84	0.16	2609	644
train1-test1	DoS slowloris	400000	800000	39309	39309	0.86	0.95	0.90	0.84	0.16	6294	2012
train1-test2	DoS slowloris	400000	800000	39309	39309	0.86	0.95	0.90	0.84	0.16	6138	2012
train1-test1	FTP-Patator	400000	800000	60477	60477	0.85	0.93	0.89	0.84	0.16	9714	4309
train1-test2	FTP-Patator	400000	800000	60477	60477	0.85	0.93	0.89	0.84	0.16	9565	4309
train1-test1	Heartbleed	400000	800000	45816	45816	0.86	0.99	0.93	0.84	0.16	7332	1
train1-test2	Heartbleed	400000	800000	45816	45816	0.87	0.99	0.93	0.84	0.16	7148	1
train2-test2	Infiltration	400000	400000	9136	9136	0.65	0.50	0.57	0.74	0.26	2407	4582
train4-test2	Infiltration	200000	200000	9136	9136	0.66	0.50	0.57	0.74	0.26	2397	4582
train2-test1	PortScan	400000	400000	296511	296511	0.74	0.75	0.75	0.74	0.26	78539	72877
train2-test2	PortScan	400000	400000	296511	296511	0.74	0.75	0.75	0.74	0.26	78496	72877
train4-test1	PortScan	200000	200000	296511	296511	0.74	0.75	0.75	0.74	0.26	77882	72877
train4-test2	PortScan	200000	200000	296511	296511	0.74	0.75	0.75	0.74	0.26	77810	72877
train1-test1	SSH-Patator	400000	800000	88862	88862	0.86	0.99	0.93	0.84	0.16	14159	23
train1-test2	SSH-Patator	400000	800000	88862	88862	0.86	0.99	0.93	0.84	0.16	14040	23
train1-test2	Web Attack - Brute Force	400000	800000	3484	3484	0.87	1.00	0.93	0.85	0.15	518	0
train1-test1	Web Attack - Sql Injection	400000	800000	12	12	0.86	1.00	0.92	0.83	0.17	2	0
train1-test1	Web Attack - XSS	400000	800000	863	863	0.86	1.00	0.92	0.84	0.16	140	0
train1-test2	Web Attack - XSS	400000	800000	863	863	0.86	1.00	0.92	0.84	0.16	142	0

A.3 Model Explainability Analysis per Algorithm/Model

A.3.1 Principal Component Analysis

Table A-15 PCA eigenvectors for each principal component

Model	ip_ihl-coef	ip_tos-coef	ip_flags-coef	ip_totallen-coef	ip_ttl-coef	tcp_sport-coef	tcp_dport-coef	tcp_do-coef	tcp_flags-coef	tcp_winsize-coef	tcp_opt_mss-coef	tcp_opt_wscale-coef	tcp_opt_sackok-coef
pc 1	0.000	–0.025	0.029	–0.021	0.021	0.703	–0.701	0.018	–0.002	0.096	0.001	0.001	0.047
pc 2	0.000	0.023	–0.028	–0.005	–0.157	–0.028	0.058	0.187	0.000	0.171	0.024	0.025	0.951
pc 3	0.000	0.066	–0.048	0.001	–0.620	0.030	0.091	0.069	0.003	0.728	–0.006	–0.008	–0.254
pc 4	0.000	–0.097	0.078	0.000	0.743	–0.040	0.077	–0.057	–0.001	0.649	0.001	–0.001	0.016
pc 5	0.000	0.803	0.046	–0.005	0.075	–0.405	–0.423	0.035	0.000	0.058	–0.001	0.000	–0.009
pc 6	0.000	0.578	0.071	–0.005	0.095	0.580	0.558	–0.001	0.000	–0.062	0.000	–0.001	–0.002
pc 7	0.000	–0.024	–0.255	–0.044	0.138	0.013	0.016	0.941	0.007	–0.045	–0.004	–0.005	–0.161
pc 8	0.000	0.075	–0.957	–0.030	0.070	0.037	–0.004	–0.263	0.021	0.024	–0.001	0.001	0.030
pc 9	0.000	–0.008	0.044	–0.851	–0.008	–0.015	0.010	–0.030	0.521	–0.001	0.017	0.014	0.000
pc 10	0.000	0.003	–0.002	0.521	0.005	0.009	–0.009	0.017	0.853	–0.001	0.013	0.017	0.001
pc 11	0.000	0.001	–0.002	0.008	0.000	0.000	0.000	0.000	–0.022	0.000	0.994	0.099	–0.027
pc 12	0.000	0.000	0.000	0.003	0.000	0.001	0.000	0.001	–0.020	0.002	–0.100	0.995	–0.024
pc 13	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

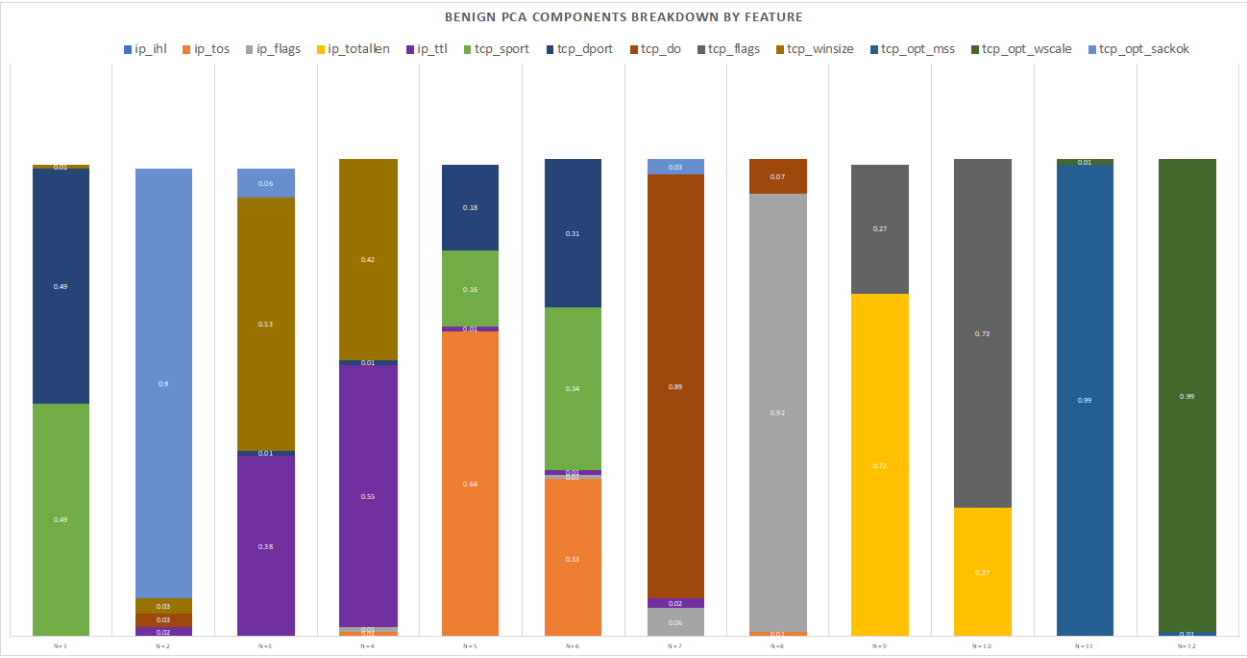


Fig. A-4 Visual depiction of the amount that each feature is used by each principal component, obtained by squaring the unit eigenvectors

Table A-16 PCA models' breakdown of the amount that each feature is used by each model, obtained by squaring the unit eigenvectors and taking a cumulative sum

Model	ip_ihl	ip_tos	ip_flags	ip_totallen	ip_ttl	tcp_sport	tcp_dport	tcp_do	tcp_flags	tcp_winsize	tcp_opt_mss	tcp_opt_wscale	tcp_opt_sackok
PCA_1_pcs	0.00	0.00	0.00	0.00	0.00	0.49	0.49	0.00	0.00	0.01	0.00	0.00	0.00
PCA_2_pcs	0.00	0.00	0.00	0.00	0.02	0.49	0.49	0.03	0.00	0.04	0.00	0.00	0.90
PCA_3_pcs	0.00	0.00	0.00	0.00	0.40	0.49	0.50	0.03	0.00	0.57	0.00	0.00	0.96
PCA_4_pcs	0.00	0.01	0.01	0.00	0.95	0.49	0.51	0.03	0.00	0.99	0.00	0.00	0.96
PCA_5_pcs	0.00	0.65	0.01	0.00	0.96	0.65	0.69	0.03	0.00	0.99	0.00	0.00	0.96
PCA_6_pcs	0.00	0.98	0.02	0.00	0.97	0.99	1.00	0.03	0.00	0.99	0.00	0.00	0.96
PCA_7_pcs	0.00	0.98	0.08	0.00	0.99	0.99	1.00	0.92	0.00	0.99	0.00	0.00	0.99
PCA_8_pcs	0.00	0.99	1.00	0.00	0.99	0.99	1.00	0.99	0.00	0.99	0.00	0.00	0.99
PCA_9_pcs	0.00	0.99	1.00	0.72	0.99	0.99	1.00	0.99	0.27	0.99	0.00	0.00	0.99
PCA_10_pcs	0.00	0.99	1.00	0.99	0.99	0.99	1.00	0.99	1.00	0.99	0.00	0.00	0.99
PCA_11_pcs	0.00	0.99	1.00	0.99	0.99	0.99	1.00	0.99	1.00	0.99	0.99	0.01	0.99
PCA_12_pcs	0.00	0.99	1.00	0.99	0.99	0.99	1.00	0.99	1.00	0.99	1.00	1.00	0.99

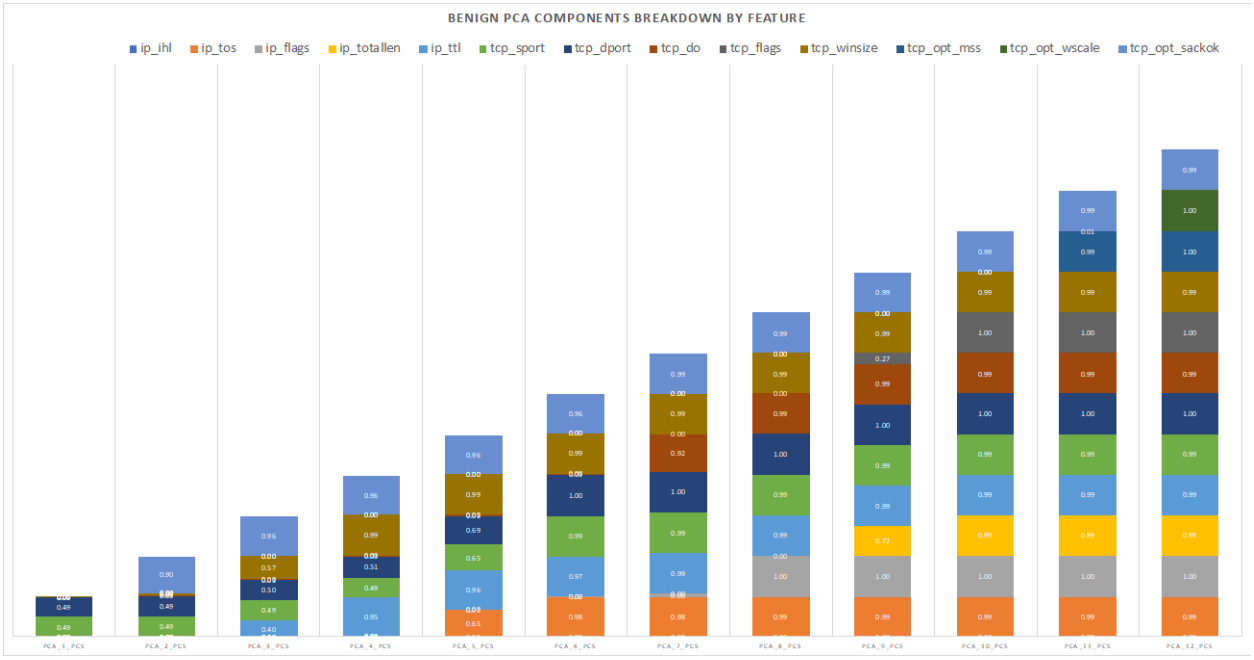


Fig. A-5 Visual depiction of each PCA model's breakdown of the amount that each feature is used by each model, obtained by squaring the unit eigenvectors and taking a cumulative sum

A.3.2 Random Forest

Table A-17 Random Forest feature analysis, breakdown of the amount that each model is influenced by each feature

Model	ip_ihl-coef	ip_tos-coef	ip_flags-coef	ip_totallen-coef	ip_ttl-coef	tcp_sport-coef	tcp_dport-coef	tcp_do-coef	tcp_flags-coef	tcp_winsize-coef	tcp_opt_mss-coef	tcp_opt_wscale-coef	tcp_opt_sackok-coef
train1	0.000	0.005	0.005	0.040	0.311	0.121	0.137	0.044	0.029	0.228	0.032	0.040	0.009
train2	0.000	0.006	0.007	0.044	0.313	0.107	0.123	0.053	0.036	0.243	0.031	0.028	0.008
train3	0.000	0.007	0.008	0.052	0.318	0.105	0.128	0.056	0.043	0.226	0.024	0.029	0.005
train4	0.000	0.007	0.008	0.070	0.318	0.119	0.132	0.038	0.047	0.216	0.020	0.019	0.005
train5	0.000	0.007	0.008	0.073	0.315	0.105	0.136	0.040	0.045	0.225	0.021	0.020	0.006

A.3.3 Logistic Regression

Table A-18 Logistic Regression feature analysis, breakdown of the amount that each model is influenced by each feature

Model	ip_ihl-coef	ip_tos-coef	ip_flags-coef	ip_totallen-coef	ip_ttl-coef	tcp_sport-coef	tcp_dport-coef	tcp_do-coef	tcp_flags-coef	tcp_winsize-coef	tcp_opt_mss-coef	tcp_opt_wscale-coef	tcp_opt_sackok-coef
train1	3.816	–20.516	5.013	–5.273	–3.373	0.172	–0.025	1.902	–0.428	–2.045	–3.113	14.854	0.078
train2	0.610	–21.080	5.070	–4.909	–3.732	0.246	–0.098	1.763	0.255	–2.286	1.017	17.264	–0.021
train3	0.714	–19.705	5.016	–5.685	–3.703	0.303	–0.045	1.489	1.689	–2.226	0.714	16.010	0.021
train4	1.518	–18.794	4.912	–7.271	–3.658	0.437	0.089	0.974	4.434	–2.138	0.887	13.277	0.095
train5	–0.109	–16.483	4.876	–7.239	–3.605	0.433	0.091	1.007	4.356	–2.146	1.636	10.446	0.159

A.3.4 SGD Logistic Regression

Table A-19 SGD Logistic Regression feature analysis, breakdown of the amount that each model is influenced by each feature

Model	ip_ihl-coef	ip_tos-coef	ip_flags-coef	ip_totallen-coef	ip_ttl-coef	tcp_sport-coef	tcp_dport-coef	tcp_do-coef	tcp_flags-coef	tcp_winsize-coef	tcp_opt_mss-coef	tcp_opt_wscale-coef	tcp_opt_sackok-coef
train1	–0.104	–6.225	4.549	–3.471	–3.292	0.172	–0.044	1.922	–0.310	–1.953	–0.652	0.527	0.412
train2	–0.125	–6.729	4.656	–3.561	–3.612	0.274	–0.120	1.783	0.068	–2.247	–0.158	0.546	0.473
train3	–0.154	–6.720	4.623	–4.097	–3.567	0.313	–0.052	1.582	1.048	–2.144	–0.241	0.480	0.399
train4	–0.123	–6.617	4.526	–5.128	–3.587	0.458	0.041	1.063	2.970	–2.075	–0.162	0.542	0.433
train5	–0.233	–6.827	4.523	–5.311	–3.585	0.504	0.127	1.110	2.856	–2.076	–0.384	0.254	0.378

A.3.5 SGD Linear SVM

Table A-20 SGD Linear SVM feature analysis, breakdown of the amount that each model is influenced by each feature

Model	ip_ihl–coef	ip_tos–coef	ip_flags–coef	ip_totallen–coef	ip_ttl–coef	tcp_sport–coef	tcp_dport–coef	tcp_do–coef	tcp_flags–coef	tcp_winsize–coef	tcp_opt_mss–coef	tcp_opt_wscale–coef	tcp_opt_sackok–coef
train1	–0.229	–7.956	3.755	–0.910	–1.461	–0.008	–0.028	3.251	–1.092	–1.873	–1.035	0.599	0.060
train2	–0.261	–8.576	4.256	–2.401	–3.580	0.023	–0.048	0.907	–1.636	–1.090	–0.673	1.338	0.176
train3	–0.181	–8.532	4.328	–3.564	–3.823	–0.008	–0.001	0.399	–0.611	–1.028	–0.494	1.581	0.255
train4	–0.162	–8.584	4.379	–4.038	–3.926	0.076	–0.013	0.133	0.169	–1.038	–0.444	1.561	0.223
train5	–0.097	–8.594	4.366	–4.106	–3.969	0.017	–0.055	0.191	0.131	–1.075	–0.183	1.773	0.323

List of Symbols, Abbreviations, and Acronyms

1D	one-dimensional
AE	Autoencoder
ARL	Army Research Laboratory
CNN	Convolutional Neural Network
DDoS	distributed denial of service
DEVCOM	US Army Combat Capabilities Development Command
DoS	denial of service
FN	false negative
FP	false positive
FPR	false positive rate
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
LSTM	long short-term memory
MAE	mean absolute error
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RFC	request for comment
SGD	Stochastic Gradient Descent
SQL	Structured Query Language
SSH	Secure Shell
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TLS	Transport Layer Security

TN	true negative
TP	true positive
TTL	time to live
UDP	User Datagram Protocol

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
4 (PDF)	DEVCOM ARL FCDD RLA ND M DE LUCIA D KRYCH S RAIO J ELLIS