Identifying Network Intrusions Using Deep Learning

Nicholas Bashour
Stanford University

nbashour@stanford.edu
github.com/nick-bash/network-intrusion-detector

Abstract

Cybersecurity researchers and practitioners have increasingly looked to advanced machine learning methods to help identify nefarious network attacks. While significant research has been done historically, much of the work has been tested using datasets from the late 1990's and early 2000's. This work is an examination of fully-connected deep learning architectures on modern intrusion detection datasets. Experimentation led to significant success in correctly identifying benign network activity, but challenges remain in improving the accuracy of identification of network attacks. An analysis of feature saliency maps and the time-based nature of network attacks provides insight into the future direction of intrusion detection research efforts, e.g. the use of recurrent neural networks (RNNs).

1. Introduction

In an increasingly digital world, organizations across the globe are facing the ever more challenging task of defending their digital assets and infrastructure. Malicious actors are progressively more pernicious – be it through infrastructure attacks, ransomware, or supply chain compromises.

With the rise of cheap computing resources and significant advances in machine learning algorithms, cybersecurity researchers have increasingly leveraged deep learning and other related algorithms to bolster the defenses of digital networks. This project aims to contribute to the work of such researchers by testing deep learning approaches on recently modernized intrusion detection datasets.

2. Related Work

Significant research about the application of machine learning to network intrusion detection and prevention exists.

Khraisat and Alazab provide a recent survey, discussing in detail the state of intrusion detection research in the Internet of Things (IoT) ecosystem [1]. Their work is a robust overview of various intrusion detection methodologies, including supervised, unsupervised, reinforcement, and deep learning methods. Further, their analysis highlights continuing challenges in defending IoT systems. Li conducts another survey into the state of AI in cybersecurity, expanding on the current work with an assessment of the risk posed by adversarial attacks on AI-based cybersecurity defense systems [2].

Broadly, there have been many attempts to research intrusion detection classifiers using advanced machine learning techniques. Moon et al. created a decision tree leveraging behavior analysis to research mechanisms to identify advanced persistent threats (APTs) [3]. APT's are attempts at hacking into a network through social engineering and backdoor exploits, such as the recent and infamous SolarWinds breach. Chowdhury et al. developed a model joining a convolutional neural network (CNN) with a support vector machine (SVM) and 1-nearest neighbor classifier for few-shot intrusion detection. Their analysis suggests their methods outperformed other research, when tested against the well-known KDD '99 and NSL-KDD datasets [4]. Meng et al. attempted to create a CNN to identify binaries as malware using their 'gene sequences' – representations of their code fragments decoded from binary to assembly using IDA Pro [5].

This work principally relies on the dataset created by Sharafaldin, et al, at the Canadian Institute for Cybersecurity in 2018 [6], described below. The research of Sharafaldin et al. explored various machine learning algorithms, such as K-Nearest Neighbors, Random Forest, and ID3, to predict network intrusions in the dataset they constructed. The research presented in this work thus serves as a deeper exploration of a recently modernized intrusion detection dataset, focused on deep learning techniques.

3. Dataset and Features

Sharafaldin et al. recently created a dataset known as the Intrusion Detection Evaluation Dataset (CIC-IDS2017) which serves as the basis for this work. As they outline in detail, the CIC-IDS2017 dataset was created to supplant various datasets that had been relied upon in cybersecurity literature for research to improve intrusion detection techniques [6]. The foremost of these historical datasets are the KDD Cup 1999 (a database of 18M instances of network activity, simulated in a military environment in 1999) and the NSL-KDD (a refined version of the KDD Cup 1999 data). Sharafaldin et al. cite various shortfalls of the KDD data and other data common in the literature, such as a lack of traffic diversity, limited variety of network

attacks, and incomplete network configuration.

The CIC-IDS2017 dataset was created by simulating a variety of attacks over the course of a week in a realistic testbed infrastructure. The simulated network included 4 attacker and 10 victim machines, with 25 users, using multiple protocols, such as HTTPS, HTTP, FTP, SSH, and email. The attacking machines attempt various attacks over the simulation, including brute force, Heartbleed, Botnet, DoS, DDoS, and infiltration attacks, for a total of 14 unique attack types.

The size of the CIC-IDS2017 dataset is significant, with a total of 78 features measured across over 2.8 million instances of network flows. A flow represents "a sequence of packets with the same values for Source IP, Destination IP, Source Port, Destination Port and Protocol (TCP or UDP)." Along with basic data around the network flow, such as the destination port, the flow duration, and the total number of packets, the CIC-IDS2017 dataset captures a significant number of other features using CICFlowMeter. CICFlowMeter is an open-source tool that extracts data from network flows in .pcap files. Some of these features include statistical measures like the max, min, mean, and standard deviation of packet length; the flow's active / idle time; and the interarrival time between flows. See Table 1 in the appendix for a full list of the feature set.

The dataset was split into training and test data as shown in Figure 1. The input features were stacked into an input vector to drive the analysis. Given significant variation across the input features, the values of training and test data were normalized during pre-processing.

Label	# Samples	% of Total	Training	% of Total	Testing	% of Total
Benign	2,273,097	80.3%	2,272,097	80.3%	1,000	0.0%
Attacks						
Bot	1,966	0.1%	966	0.0%	1,000	0.0%
DDoS	128,027	4.5%	127,027	4.5%	1,000	0.0%
DoS GoldenEye	10,293	0.4%	9,293	0.3%	1,000	0.0%
DoS Hulk	231,073	8.2%	230,073	8.1%	1,000	0.0%
DoS Slowhttptest	5,499	0.2%	4,499	0.2%	1,000	0.0%
DoS slowloris	5,796	0.2%	4,796	0.2%	1,000	0.0%
FTP-Patator	7,938	0.3%	6,938	0.2%	1,000	0.0%
Heartbleed	11	0.0%	-	0.0%	11	0.0%
Infiltration	36	0.0%	-	0.0%	36	0.0%
PortScan	158,930	5.6%	157,930	5.6%	1,000	0.0%
SSH-Patator	5,897	0.2%	4,897	0.2%	1,000	0.0%
Web Attack - Brute Force	1,507	0.1%	507	0.0%	1,000	0.0%
Web Attack - Sql Injection	21	0.0%	-	0.0%	21	0.0%
Web Attack - XSS	652	0.0%	-	0.0%	652	0.0%
Total Attacks	557,646	19.7%	546,926	19.3%	10,720	0.4%
Total	**2,830,743**	**100.0%**	**2,819,023**	**99.6%**	**11,720**	**0.4%**

Figure 1: An Overview of the CICIDS-2017 Data and the Training / Test Split

4. Methods

To label network activity as either benign or one of 14 unique attack types, the input feature vector was run through a fully connected network. Several permutations of model architecture, including different network depth, training epochs, number of hidden nodes, and dropout layers, were tested to improve the models' results.

With a softmax layer as the final output, the preliminary loss function chosen was the categorical cross entropy loss. However, initial results showed the model was performing significantly better at classifying benign network activity than network attacks. To attempt to address this shortfall, weights were added to the categorical cross entropy loss calculation to penalize the cost of network attacks in an amount proportional to the ratio of the frequency of the benign label to the frequency of the attack label in the training data.

Given the significant number and complicated nature of the feature set, a saliency map was created after completion of model training to help identify which input features are most important in driving the softmax classification.

The open-source deep learning libraries Tensorflow and Keras were used to implement the model [7, 8]. A Github repository named Saliency-Maps-in-TF-2.0 was used to calculate the saliency maps [9].

5. Experiments, Results, and Discussion

Ultimately, over thirty different model architectures were tested. The primary adjustments to architecture were with respect to model depth, the number of hidden nodes, the number of epochs trained, and the incorporation of dropout layers.

Given the potential cost of a network intrusion, the author believes that an enterprise which employs a network intrusion detector in a production setting would be much more sensitive to 'false negatives' – undetected network attacks' – than to 'false positives' – benign activity mistakenly identified as nefarious. As such, model experimentation was conducted with the goal of optimizing for attack label accuracy conditional upon achieving 95% benign label accuracy.

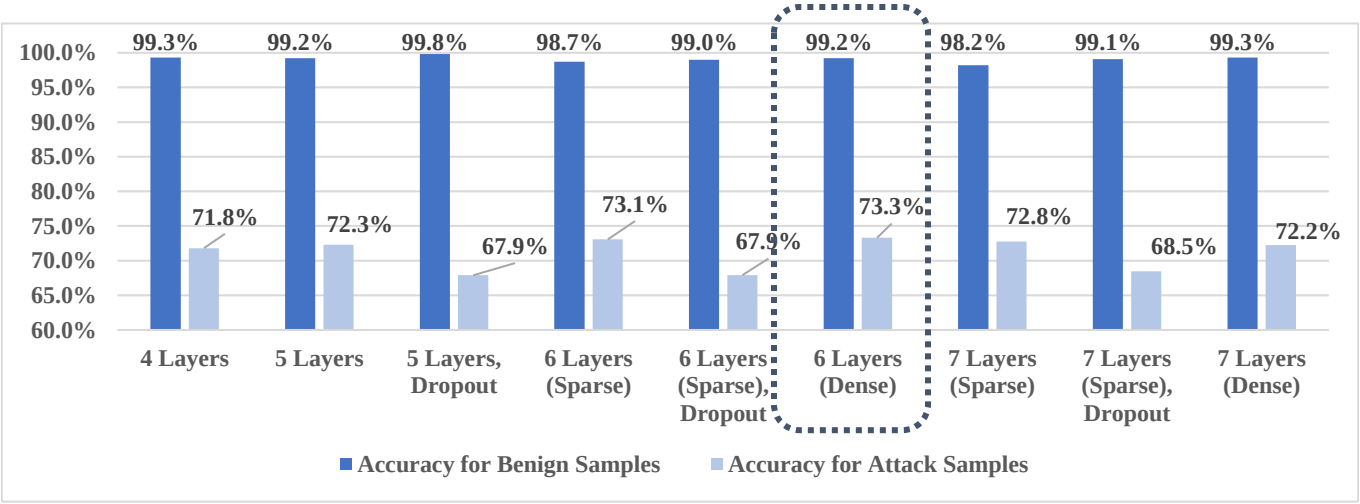

Figure 2: Accuracy for Benign and Attack Samples of Select Models. The *6 Layers (Dense)* model is highlighted due to its performance on the attack samples. All models shown were trained on 50 epochs.

	# nodes							Dropout Rate	
Model Name	**(L = 1)**	**(L = 2)**	**(L = 3)**	**(L = 4)**	**(L = 5)**	**(L = 6)**	**(L = 7)**	**(L = 1)**	**(L = 2)**
4 Layers	40	30	20	15	-	-	-	-	-
5 Layers	50	40	30	20	15	-	-	-	-
5 Layers, Dropout	50	40	30	20	15	-	-	20%	20%
6 Layers (Sparse)	50	40	30	25	20	15	-	-	-
6 Layers (Sparse), Dropout	50	40	30	25	20	15	-	20%	20%
6 Layers (Dense)	60	50	40	30	20	15	-	-	-
7 Layers (Sparse)	50	40	30	25	20	15	15	-	-
7 Layers (Sparse), Dropout	50	40	30	25	20	15	15	20%	20%
7 Layers (Dense)	70	60	50	40	30	20	15	-	-

Figure 3: Architecture of Models in Figure 2

While the models frequently identified benign activity with over ~99% accuracy, the best performer reached only ~73% accuracy on the attack data. This model, a six-layer neural net, is referred to as the *6 Layers (Dense)* model. A summary of the results and an overview of the model architectures are presented in Figures 2 and 3 (these figures show a handful of the better performers, not an exhaustive list of all tested models).

To address the discrepancy in attack and benign accuracy rates, an extra penalty was added to the categorical cross entropy loss calculation for attack labels. The baseline penalty was the ratio of the frequency of the benign data relative to each attack label. Using the *Six Layers (Dense)* model as the reference, this baseline was scaled up by factors [1] of 10 through 10,000. The results of this experimentation are presented in Figures 4 and 5.

Unfortunately, adding this additional penalty to the loss function did not show significant promise in improving identification of network attacks. As shown in Figures 2 and 3, neither did the incorporation of dropout layers in the network. These results were both surprising and frustrating, but an exploration of the saliency maps of the input features may provide some insight into the challenge of predicting attack types accurately.

As shown in Figure 6 in the Appendix, saliency maps of the different labels demonstrate significant consistency in the importance of input features in classifying different labels. By and large, the most important features in predicting labels were the numbers of forward packets, backward packets, subflow packets, and packets with at least 1 byte of data. Only a handful of labels show a material reliance on other features, such as the use of PSH flags to identify certain DoS and web attacks.

One interpretation of this shared importance in features across labels is that gradient descent struggled to make

[1] As shown in Figures 4 and 5, the Loss Function Attack Label Weight represents the factor by which the ratio of benign labels to the attack label was scaled in the modified categorical cross entropy lost. E.g., the model in the last row of Figure 4 penalized a mis-classified attack at 10,000 times the ratio of the frequency of the benign label in the test set to the frequency of the attack label in the test set.

Loss Function Attack Label Weight[1]	# Epochs	Accuracy for Benign Samples	Accuracy for Attack Samples
Reference Model: Six Layers (Dense)			
n/a	50	99.2%	73.3%
Models with Loss Function Weights			
1	25	99.3%	71.8%
10	25	98.4%	72.2%
100	25	98.9%	72.2%
500	25	99.0%	72.3%
1,000	5	99.3%	66.2%
1,000	10	98.8%	72.9%
1,000	15	99.6%	71.5%
1,000	25	98.6%	72.7%
1,000	50	99.2%	71.6%
5,000	25	99.2%	71.5%
10,000	25	99.6%	72.1%

Figure 4: Results of Changes in Epochs and Loss Function Attack Label Weights[1] to *Six Layer (Dense)* Model

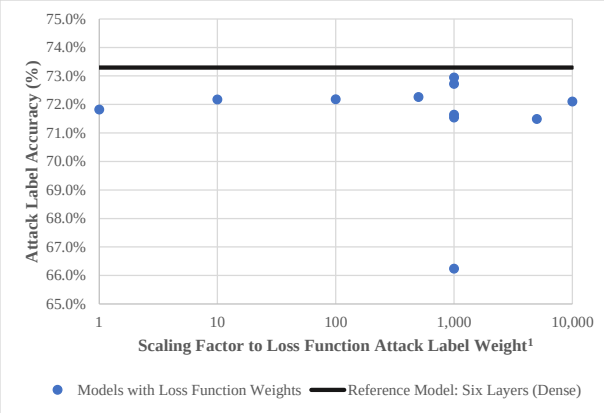

Figure 5: Attack Label Accuracy for Models with Loss Function Weights *(Logarithmic X-axis)*

meaningful progress and distinguish labels across the vector space after a certain amount of training. As Figure 7 shows, the training did in fact see materially diminishing returns to gradient descent after 10 epochs. This, of course, is complicated by the significantly disproportionate frequency of benign activity to attack activity, especially for some attack types like Heartbleed and SQL Injections, which have only a few dozen instances.

Another interpretation is that the feature set needs to be supplemented with new, creative features to further help identify attacks. Relatedly, an important limitation in the use of the CICIDS-2017 dataset was that, to construct this model, it was necessary to ignore the destination port to which network flows were targeted. This step was taken because, in many cases within the test data created by Sharafaldin et al., a large majority of an attack was targeted at a handful of destination ports. While this is likely a realistic simulation of real-world attacks, including this feature would have made the neural network's classification task trivial. In a production environment, a model could be adjusted to identify activity which repeatedly targets the same port over a short time frame (signaling a potential attack) using a recurrent architecture or attention mechanism; unfortunately, the structure of the CICIDS-2017 dataset didn't allow for such analysis.

6. Conclusion and Future Work

This work illustrated firsthand the challenge that cybersecurity researchers and practitioners face in identifying malicious network activity: in any given environment, there is a very significant volume of activity, the large majority of it is benign, and identifying attacks in an automated fashion requires finding 'needles in a haystack.'

The challenge of improving classification of malicious network activity can be addressed in the future with further study of the vector space in which benign and malicious network activity exists. Additionally, leveraging the insight of expert researchers and practitioners would help to identify new, creative features to be used in attack classification. Further, having access to more granular, timestamped network flows (such as activity right before an attack) would allow for the use of advanced neural network architecture like recurrent models. Although likely to be computationally intensive, using time-series data could result in improvements in understanding and identifying realistic network attacks.

References

[1] Khraisat, A., Alazab, A. "A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy, Attacks, Public Datasets, and Challenges", *Cybersecurity*, 2021.

[2] Li, Jian-hua. "Cyber Security Meets Aritificial Intelligence: A Survey", *Frontiers of Information Technology & Electronic Engineering*, 2018.

[3] Moon, D., Hyungjin, I, Kim, I. "DTB-IDS: An Intrusion Detection System based on Decision Tree using Behavior Analysis for Preventing APT Attacks," *The Journal of Supercomputing*, 2017.

[4] Chowdhury, MMU., Hammond, F., Konowicz, G., Xin, C., Wu, H., Li, J. "A Few-shot Deep Learning Approach for Improved Intrusion Detection", *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, 2017.

[5] Meng, X., Shan, Z., Liu, F., Zhao, B., Han, J., Wang, H., Wang, J.. "MCSMGS: Malware Classification Model Based on Deep Learning," *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2017.

[6] Sharafaldin, I., Lashkari, AH, Ghorbani, AA. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", *ICISSP*, 2018.

[7] TensorFlow Developers. TensorFlow (Version v2.1.3), 2021.

[8] Chollet, F. Keras (Version v2.3.0), 2021.

[9] Rizwan, U. Saliency-Maps-in-TF-2.0, 2020, GitHub repository, https://github.com/usmanr149/Saliency-Maps-in-TF-2.0.

Appendix

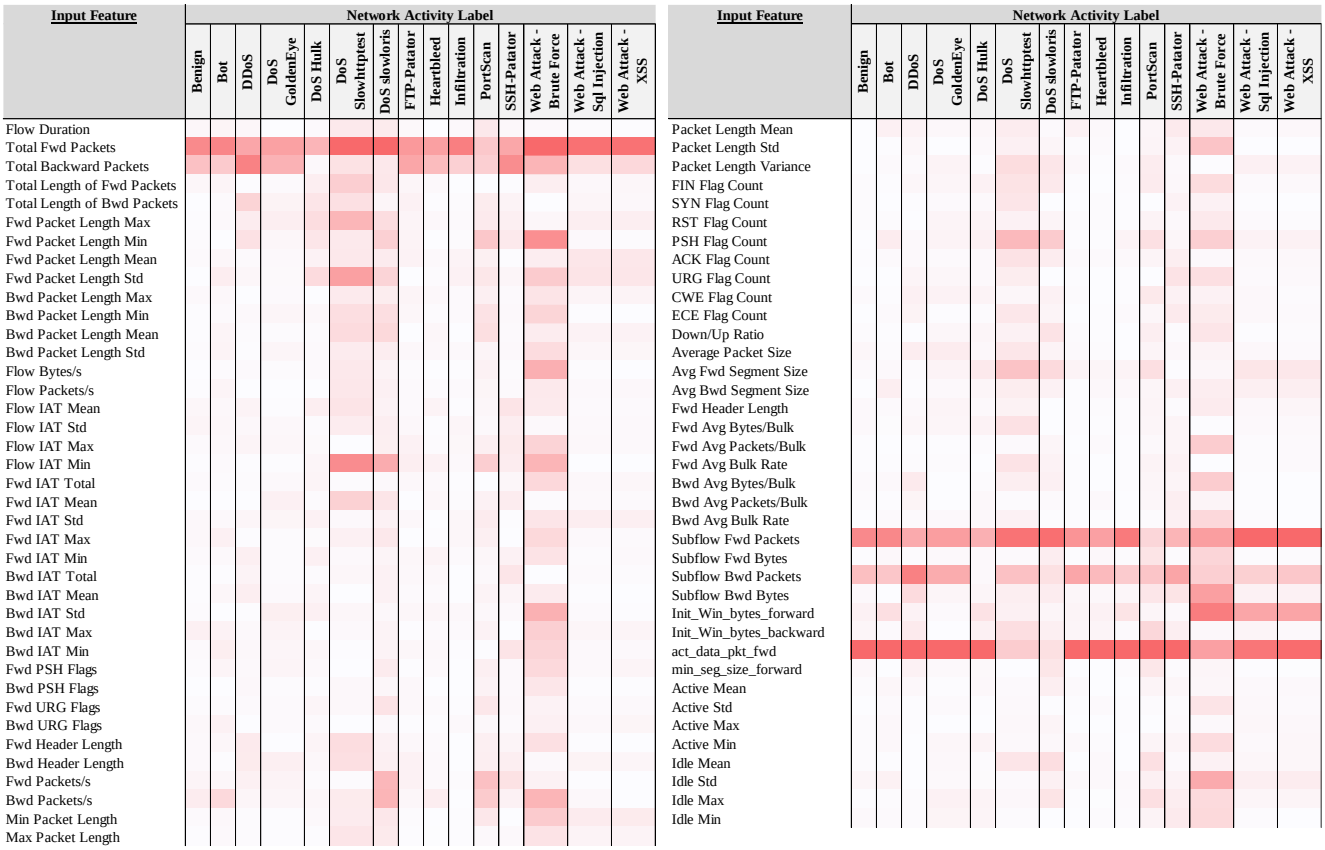

Figure 6: Saliency Map for all 77 input features. Computed on the 14 test samples which had the highest activations for the *Six Layer (Dense)* model.

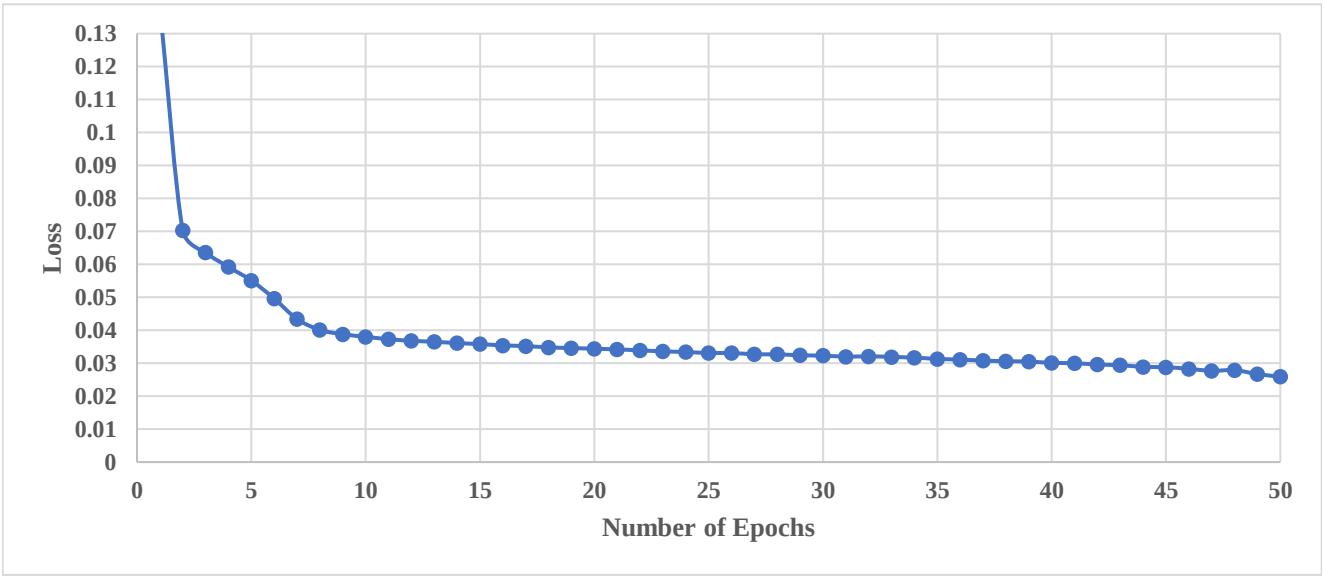

Figure 7: Loss function of the *Six Layer (Dense)* model over 50 epochs.

Table 1: A list of all input features

- Flow duration
- Total Forward Packet
- Total Backward packets
- Total Length of Forward Packet
- Total Length of Backward Packet
- Forward Packet Length Min
- Forward Packet Length Max
- Forward Packet Length Max
- Forward Packet Length Max
- Backward Packet Length Min
- Backward Packet Length Max
- Backward Packet Length Mean
- Backward Packet Length Std
- Flow Byte/s
- Flow Packets/s
- Flow Interarrival time (IAT) Mean
- Flow IAT Std
- Flow IAT Max
- Flow IAT Min
- Forward IAT Min
- Forward IAT Max
- Forward IAT Mean
- Forward IAT Std
- Forward IAT Total
- Backward IAT Min
- Backward IAT Max
- Backward IAT Mean
- Backward IAT Std
- Backward IAT Total
- Forward PSH flag
- Backward PSH flag
- Forward URG flag
- Backward URG flag
- Forward Header Length
- Backward Header Length
- Forward Packets/s
- Backward Packets/s
- Min Packet Length
- Max Packet Length
- Packet Length Mean
- Packet Length Std
- Packet Length Variance
- FIN Flag Count
- SYN Flag Count
- RST Flag Count
- PSH Flag Count
- ACK Flag Count
- URG Flag Count
- CWR Flag Count
- ECE Flag Count
- Down/Up Ratio
- Average Packet Size
- Avg Forward Segment Size
- AVG Backward Segment Size
- Forward Header Length
- Forward Avg Bytes/Bulk
- Forward AVG Packet/Bulk
- Forward AVG Bulk Rate
- Backward Avg Bytes/Bulk
- Backward AVG Packet/Bulk
- Backward AVG Bulk Rate
- Subflow Forward Packets
- Subflow Forward Bytes
- Subflow Backward Packets
- Subflow Backward Bytes
- Init_Win_bytes_forward (# of bytes sent in the initial forward window)
- Init_Win_bytes_backward (# of bytes sent in the initial backward window)
- Act_data_pkt_forward (# of packets with at least 1 byte of TCP data payload in the forward direction)
- Forward Minimum Segment Size
- Active Min
- Active Mean
- Active Max
- Active Std
- Idle Min
- Idle Mean
- Idle Max
- Idle Std