Learn to Detect Attacks Without Seeing Attacks

Tal Barzilay* Tomer Laor*

July 28, 2023

Abstract

Machine learning (ML) has demonstrated its effectiveness in identifying malicious network flows, addressing the core problem of network intrusion detection systems (NIDS). However, existing NIDS algorithms rely on training datasets that include known attacks, introducing a bias towards detecting similar zero-day attacks. For instance, attacks such as SYN flooding and port scanning may exhibit similar attributes like the number of packets per second or average packet size, leading to inherent biases when training on one known attack and testing on another. To mitigate this issue, we propose a method that enables the detection of attacks without the need to observe them during the training phase.

Our method aims to address two main scenarios in attack detection:

1. Zero-shot detection of unseen attacks: In this scenario, our algorithm operates autonomously after the training phase, capable of detecting attacks it has never encountered before, thereby reducing reliance on a predefined set of known attacks.

2. Few-shot detection of manually annotated attacks: In this scenario, a security analyst identifies a suspicious network flow and manually annotates it for further investigation. Subsequently, our algorithm leverages this limited set of annotated flows to discover additional suspicious network flows exhibiting similar behavior, effectively assisting the analyst in identifying potential threats.

We evaluate the effectiveness of our proposed method through comprehensive experiments and demonstrate its ability to detect truly unknown attacks while aiding security analysts in their investigative tasks.

*Both authors are considered co-first authors.

1 Introduction

Machine learning (ML) has become a rapidly advancing field with diverse applications, including the realm of network intrusion detection systems (NIDS) [3]. The integration of ML in NIDS has been extensively researched and continues to evolve. One common approach involves training classification models on datasets comprising both benign and malicious network flows. These models aim to identify instances of known attacks, leveraging the abundance of available training data. Additionally, there are few-shot classification models that are trained on datasets containing both benign and malicious network flows. These models are specifically designed to detect "unseen" attacks during the training phase by utilizing only a limited number of examples of these attacks during a memorization phase. This enables the model to generalize and identify new attack patterns beyond those encountered during training.

ML algorithms used in NIDS encompass a range of techniques, including outlier detection algorithms like isolation forest and neural network AutoEncoders. Additionally, classification algorithms such as random forest and neural networks are commonly employed. Outlier detection techniques focus on identifying patterns that deviate from expected behaviors, allowing the detection of anomalies within the data [8]. In other words, these techniques learn to recognize and flag data points that exhibit unusual or suspicious characteristics. On the other hand, classification techniques are designed to address the problem of assigning labels to data points based on their characteristics. These algorithms learn to match input data points with their corresponding class or category, enabling the identification of different types of network flows, including malicious or benign ones.

Meta-learning encompasses a range of algorithms, and in our discussion, we will concentrate on one particular type that involves learning to classify with distance constraints. One approach within this type is exemplified by neural networks

trained with losses like AAMSoftmax [1]. This loss introduce angular additive margin constraints to improve the performance of few-shot verification tasks. By enforcing a specific distance between embeddings representing different classes, these constraints enable the measurement of cosine similarity between embeddings. This measurement provides a valuable means of assessing the similarity or dissimilarity of data points representations.

The similarity between attacks and the presence of attacks in the training dataset pose challenges when evaluating "unseen" attacks. When evaluating the performance of intrusion detection systems on "unseen" attacks, the similarity between these attacks and the ones present in the training dataset can introduce biases. For example, certain attacks may share similar attributes or behaviors, leading to a higher detection rate for attacks that resemble those in the training data. This challenge highlights the need for robust evaluation methods that can accurately assess the system's ability to detect novel and diverse attacks. By considering the challenge of similarity between attacks and the presence of attacks in the training dataset during evaluation, researchers can strive for more reliable and unbiased assessments of NIDS performance on "unseen" attacks.

Our research makes two significant contributions. Firstly, we propose a pipeline for training and evaluating models that do not rely on training with malicious flows. This approach allows for a less biased evaluation, as the models are not influenced by specific known attacks during the training phase. We evaluate the proposed models both in zero-shot scenario and in few-shot scenario. Secondly, we introduce the MetaFlowGuard model, a neural network that leverages meta-learning. By training MetaFlowGuard to classify source IPs of network flows, it learns a general feature extractor that can effectively detect malicious flows. This enables MetaFlowGuard to serve as an efficient and adaptable detector for various types of attacks.

2 Related Works

Numerous papers have explored the application of machine learning (ML) techniques to enhance Network Intrusion Detection Systems (NIDS).

Yu et al. [10] leveraged a Convolutional Neural Network (CNN) to classify flows as either benign or malicious, and further identify the specific malware type for malicious flows. They worked with the CIC-IDS2017 and CSE-CIC-IDS2018 datasets and proposed two models. The first model is a standard classification model trained with cross-entropy loss, while the second model is a siamese model designed for few-shot classification. Their findings showed that the siamese model outperforms the normal classification model for attacks with limited training data.

Wang et al. [9] employed BYOL (Bootstrap Your Own Latent) for self-supervised learning, which enables label-free training. BYOL involves running the same input through two neural networks, where each network receives a different perturbation (augmentation) of the input. The loss function minimizes the difference between the two networks' outputs. They reshaped the feature vector of each flow into a matrix as part of their preprocessing step. To use BYOL, they proposed a data augmentation strategy specifically tailored for NIDS datasets, incorporating techniques like horizontal flip, vertical flip, random crop, and their novel random-shuffle augmentation. Random-shuffle augmentation randomly shuffles vectors in the matrix that represents the flow. The authors trained their model on the UNSW-NB15 dataset and fine-tuned it for other datasets, such as NSK-KDD, KDD CUP99, CIC IDS2017, and CIDDS_001. Their neural network trained with BYOL produced effective embeddings, and they further trained a logistic regression classifier to determine whether a flow is benign or malicious and identify its corresponding malware class.

Sarhan et al. [6] introduced a training strategy for a neural network to learn to map network features into semantic attributes. During the inference stage, they applied a nearest neighbor algorithm between the incoming network flow's embedding and the embeddings of malicious flows they previously trained on. They conducted experiments using the UNSW-NB15 and NF-UNSW-NB15-v2 datasets. The authors introduced a new metric called Zero-Day Detection Rate (Z-DR), representing the detection rate of a specific malicious class when it is excluded from the dataset. However, this metric may have limitations, as it only excludes the specific malicious class while other attacks that may have been influenced by or influenced this attack remain in the training dataset. They also examined the Wasserstein Distance between each malicious class and the other malicious classes, confirming the presence of similar and dissimilar malicious classes.

Lotfi et al. [5] proposed an intrusion detection system that utilizes a deep neural network and leverages self-supervised contrastive learning (SSCL) to achieve accurate results even with a limited number of labeled data. The primary aim of

SSCL is to learn transferable knowledge from unlabeled data, which can then be effectively applied to downstream tasks. Their method involved creating two augmented versions of the data, x_i and x_j, and passing them through an encoder block to generate two corresponding representations. The key objective was to minimize the distance between these representations. They conducted the experiment on the UNSW-NB15 dataset and achieved an accuracy of 94.05%. This performance showed an improvement of 4.22% over previous attempts with similar methodologies.

Komisarek et al. [4] proposed a Network flow-based cyber anomaly detection system that utilizes transfer learning to adapt sketchy data structures. Their approach aims to extract generic and universal features from the data and leverage domain adaptation principles to enhance classification quality in zero- and few-shot scenarios. A key goal of their work is to empower system operators to avoid extensive model training when there are changes in the network environment or a need to migrate detectors from one network to another. This capability is achieved by transforming flows from one network to another. Through their experiments, they achieved an impressive F1 score of 0.99, outperforming the best-performing related methods, which obtained an F1 score of 0.97.

3 Methodology

Our methodology aims to accomplish two main objectives: zero-shot detection of unseen attacks and few-shot detection of annotated attacks. To achieve these goals, several key steps are involved, including defining the train-test split, preprocessing the data, selecting relevant features, training the model, and evaluating its performance.

The high-level overview of our methodology is presented in Figure 1. Notably, our model is trained exclusively on benign flows and subsequently evaluated using both malicious and benign flows. In the few-shot scenario, it is crucial to ensure that the chosen malicious flows are memorized by the model prior to evaluation. This memorization step allows the model to effectively identify similar flows during the testing phase.

3.1 Data

The train-test split is outlined in Figure 2. In our approach, we allocated 80% of the benign flows as the training dataset, while the remaining 20% of the benign flows, along with all the malicious flows, constituted the test dataset. This division ensures that the model is not biased towards the existing known attacks in the cybersecurity field, as it has not been exposed to any attack during the training phase, thus can't learn similar attack to those it will encounter during test.

We performed several preprocessing steps on the dataset. First, we dropped columns that were unique to each packet, such as flow ID and timestamp. Additionally, we removed categorical features such as the protocol column to avoid dealing with categorical features. Next, we filtered out flows from source IPs that appeared only once in the dataset, as they may not provide sufficient information for analysis. To process the destination IP column, we decomposed it into four numbers comprising the IP address. Each number was then scaled using min-max scaling, which involved dividing the values by 255. Any infinite values present in the dataset were replaced with zeros. For the remaining features, excluding those related to the destination IP, we applied standard scaling. We computed the scaling statistics on the training dataset and applied the same transformation to both the training and test datasets.

To ensure that our features do not introduce bias into the learning process, we trained a random forest classifier on the dataset with the task of classifying source IPs. We used the Gini importance measure to determine the feature importance and we filtered features that have a very large Gini importance measure compared to other features to reduce overfitting to the source IP classification problem, since our models will need to be tested on another task, which is malicious flows detection.

3.1.1 IoTID20 dataset [7]

We observed that the top three important features identified by the classifier had importance values ranging from 0.152 to 0.167. However, the importance of the fourth feature dropped significantly to 0.06. This abrupt decrease in importance raised concerns about potential overfitting. In fact, nearly 47% of the feature importance scores were concentrated on the first three features, namely the last number of the destination IP, the source port, and the destination port. These three features can lead to overfitting because certain machines may have limited communication with a small number of other machines, or even just one machine. Additionally, the source port, although dynamically allocated by the operating system, can vary significantly across different machines due to the specific applications running on higher ports. Similarly, the destination port is determined by the application that the machines are communicating with, and some

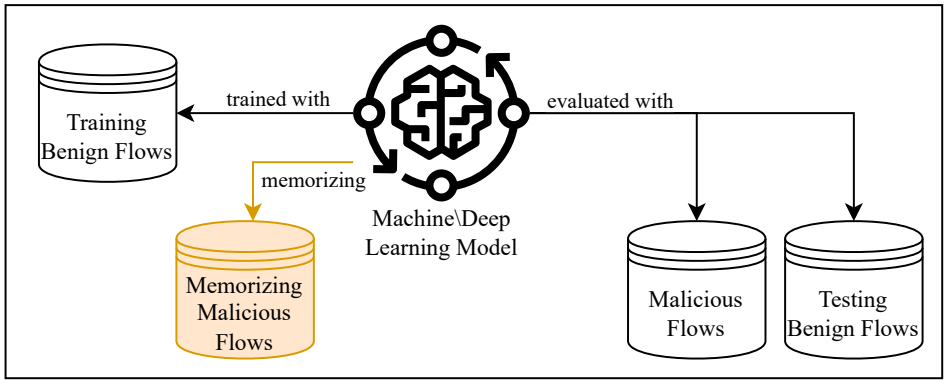

Figure 1: Overview of our method. The model is trained exclusively on benign flows and evaluated using both malicious flows and unseen benign flows. The orange section represents the memorization step, which is specific to the few-shot scenario. In the few-shot scenario, the model follows the same training procedure as the zero-shot scenario. However, prior to testing, few malicious flows are introduced to the model to facilitate memorization and identification of similar flows within the testing data.

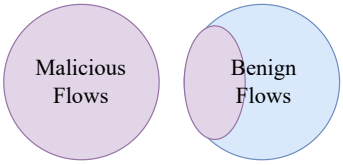

Figure 2: Train-Test Split in Our Method. The train data is highlighted in blue, while the test data is represented in purple. Notably, all the malicious flows are included in the test data.

machines may only use specific communication applications with other machines. To address the overfitting issue associated with these three features, we decided to remove them from the dataset after the preprocessing step. After this removal, we re-evaluated the random forest classifier and found that only 23% of the feature importance scores were concentrated on the first three features. This reduction indicates a reduced risk of overfitting and a more balanced contribution from the remaining features.

3.1.2 MQTT-IOT-IDS2020 dataset [2]

While our main focus was on the IoTID20 dataset [7], we also sought to demonstrate the model's versatility by conducting additional experiments on the MQTT-IOT-IDS2020 dataset [2]. This dataset comprises flows from an IoT network using the Message Queuing Telemetry Transport (MQTT) protocol. During the preprocessing phase, we noticed that there were nine rows lacking source and destination IP information. Consequently, we decided to drop these rows from the dataset.

Unlike the previous dataset, the MQTT-IOT-IDS2020 dataset contained a considerable amount of missing data. To handle this, we conducted the experiment twice. First, we worked with only the rows that had complete data, while in the second experiment, we utilized all rows and replaced the missing data with a value of -1 as [2] proposed.

During the first experiment, we observed significant Gini importance measure in a few columns, namely "tcp_flag_res" (importance of 0.98) and "ip_flag_rb" (importance of 0.55). Consequently, we decided to drop these columns from the analysis. In contrast, the second experiment did not exhibit any columns with substantial Gini importance measure, allowing us to include all available columns in the analysis.

3.2 Data Analysis

We conducted our analysis on the IoTID20 dataset [7], which consists of 625,783 entries. However, for our training purposes, we focused on the benign entries, resulting in a subset of 40,073 records, approximately 6.5% of the entire dataset. Despite working with a significantly reduced portion of the data, we still obtained significant results.

To establish a baseline for our dataset, we trained a DummyClassifier using the most common strategy to predict the source IP from the data. The base rate of the dataset was approximately 78.63%. Our analysis demonstrates improved results compared to this baseline.

We observed that there are 38 unique source IP addresses and 45 unique destination IP addresses within the dataset. Surprisingly, there are 57,985 unique source IP addresses and 478 unique destination IP addresses in the entire dataset. This

indicates that only 38 out of 57,985 are benign source IPs, less than 1% of IPs were present in our subset. This discrepancy may be attributed to the sources utilizing DDoS-like techniques to conduct attacks.

In the subset of benign data, the most frequent source IP appears 31,483 times, while the second most frequent appears only 4,342 times. The fifth most frequent IP occurs 194 times, indicating that a majority of the communication originates from a select few machines. The entire dataset exhibits a slightly less centralized pattern. The most common source IP appears 222,096 times, followed by the next IP at 125,890 occurrences. The tenth IP on the list is the first to have fewer than 1,000 appearances, with 993 entries.

Regarding destination IP addresses in the benign data, the most common destination IP appears 31,491 times, with the next IP occurring only 4,267 times. The fifth most common IP appears 255 times, suggesting a similar distribution pattern between source and destination IPs. Distribution graphs illustrating the IP distribution are presented in Figure 3. When considering the entire dataset, the most common IP appears 164,532 times, followed by the next IP with 143,150 occurrences. To observe IPs with fewer than 1,000 entries, we need to examine the 26th rank, which appears 805 times.

The most frequently occurring IPs in the dataset are 192.168.0.13, 192.168.0.16, and 192.168.0.24, which aligns with expectations. The format 192.168.X.X represents the most common default private IP address. The IP address 192.168.0.13 corresponds to the router's address used by devices connected to the network for sending data requests over the internet. Similarly, 192.168.0.16 and 192.168.0.24 are private IP addresses directly associated with the Wi-Fi network and commonly used to access the router's admin page.

Among the source ports, the most common port by far is 9020, with 31,455 entries, followed by port 80 (HTTP) with only 1,650 entries. Port 443 (HTTPS) ranks eighth with only 339 usages.

Considering the entire dataset, the most common port is 443 (HTTPS), while port 9020 ranks third. In the benign data, we identified 119 unique ports, whereas the entire dataset encompasses 11,205 ports. This difference may be attributed to attacks originating from numerous clients with relatively random ports.

The most common destination port is 49,784, occurring 31,454 times, followed by port 9020 with 4,217 occurrences. Port 443 (HTTPS) ranks third with 1,952 usages, and port 80 (HTTP) ranks fifth with only 61 usages. In the entire dataset, port 9020 is the most common with 114,464 usages, while port 443 (HTTPS) ranks fourth with 55,358 usages, and port 80 (HTTP) ranks eleventh with 9,966 usages. Another notable port is 554, with 43,445 usages, which is assigned to the Real-Time Streaming Protocol (RTSP) used for controlling the streaming of multimedia content, such as audio and video, over a network. This suggests that the network traffic contains media files such as videos and audio.

In the benign dataset, we identified 154 unique destination IPs, whereas the entire dataset contains 1,034 unique destination IPs. It is worth noting that port 9020, despite having no specific protocol commonly associated with it, is one of the most common ports.

Overall, the data analysis reveals interesting insights into the IoTID20 dataset, highlighting the distribution of source and destination IPs, as well as the occurrence of commonly used ports.

3.2.1 Comparison of similar Attack Categories

We will now examine the similarities between records from different sub-categories across all data, including the "Normal" category. To compare the categories, we calculated the "average record" for each category by taking the mean of all numeric fields among the records of the category, we ignored infinite values. Our analysis encompasses a total of 76 features that are being tested.

For each pair of categories and for each feature, we computed a similarity score by taking the smaller ratio between their respective averages, if both are zero we gave a score of 1. We consider a feature to be similar between the two categories if the ratio exceeds 0.98. The results are presented in the heatmap shown in Figure 5.

Our observations reveal that only a few attacks exhibit significant similarities, and out of the ones that do, it mostly consists of feature averages that were both 0. Particularly, the "Mirai-Ackflooding" and "Mirai-HTTP Flooding" categories demonstrate the highest similarity, with 72 out of the 76 tested features being similar. This finding suggests a high degree of similarity between these two attack types.

As can be seen from our model's evaluation in Table 1 and Tabel 3 the performance on these two attack types are nearly identical, displaying similar True Positive Rates (TPR) across various False Positive Rates (FPR). This further supports the notion that these attacks share significant similarities.

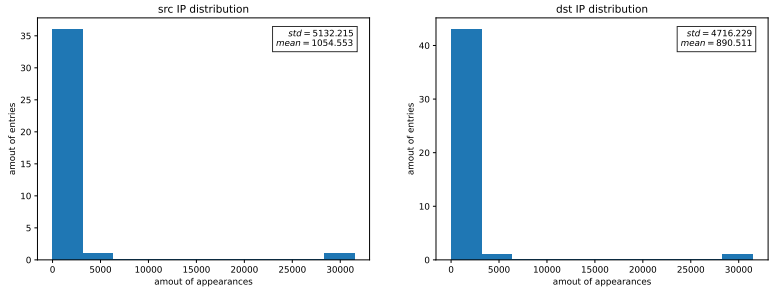

Figure 3: comparison of source and destination IPs in the benign subset.

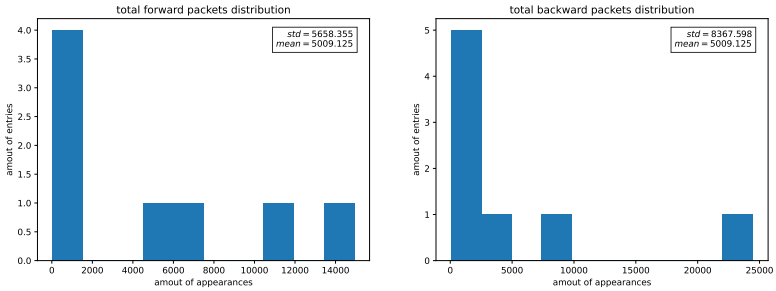

Figure 4: comparison of forward and backward packets in the benign subset.

3.3 Models and Evaluations

In the zero-shot evaluation, our model is tested on both unseen benign flows and malicious flows that it hasn't encountered during training. It aims to detect zero-day attacks without specifically identifying the type of attack, but rather flagging any flow as malicious. In the few-shot evaluation, we randomly select a specified number (k) of flows per attack type. These flows are transformed into embeddings using our neural network algorithms, and the mean embedding is calculated for each attack type. To determine if an incoming embedding represents an attack, we compute the cosine distance between the mean embedding of the corresponding attack and the incoming embedding. If the distance is below a certain threshold, it is predicted as a malicious flow. Otherwise, it is classified as a benign flow. A benign flow is considered a false positive if it is incorrectly predicted as any of the attacks. Conversely, a malicious flow is considered a false negative if it is incorrectly predicted as benign flow. To obtain reliable metrics, we repeat this evaluation process 100 times to account for the variability introduced by the selection of k flows per attack type. This repetition allows us to calculate the standard deviation and mean values for the evaluation metrics.

One-Class SVM is a well-suited algorithm for our work as it is designed for anomaly detection when only one class of data is available for training. It learns the distribution of the training data and identifies out-of-distribution data points during testing. One-Class SVM cannot participate in a few-shot evaluation since it cannot memorize few data points.

AutoEncoder is commonly employed architecture for the detection of malicious entries. By attempting to compress and reconstruct the data, this architecture enables us to identify anomalous patterns in which the reconstruction is different than the data by a large margin. In our approach, we trained the model using Mean Squared Error (MSE) loss. Specifically, we focused on training the model solely on a subset of benign records. If the resulting output loss surpasses a certain threshold, we classify the corresponding record as malicious.

MetaFlowGuard is a neural network trained using the AAMSoftmax loss [1]. This loss function guides the embedding space to ensure that embeddings of the same class have a small cosine distance to their class center while maintaining a large cosine distance to centers of other classes. This property makes it suitable for meta-learning, as it enables the neural network to generate informative embeddings that can be used for few-shot inference through cosine distance measurements. In our approach, we leverage this characteristic and train the neural network to classify source IP

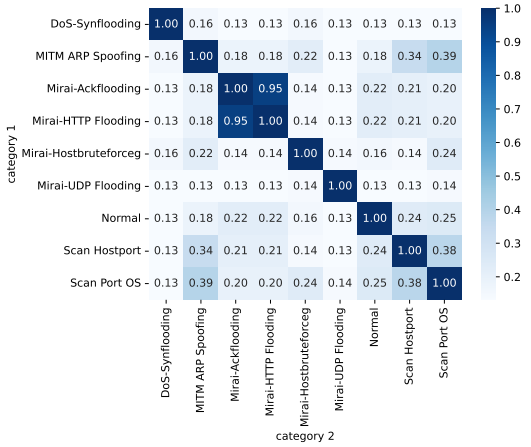

Figure 5: Heat map showing similarities between categories.

flows using the AAMSoftmax loss. Subsequently, we utilize the trained network to create embeddings. Refer to Figure 6 for an overview of the MetaFlowGuard algorithm. In the zero-shot evaluation scenario, we calculate the mean embedding for each source IP based on the training embeddings. An incoming embedding is classified as malicious if its minimum cosine distance to any of the computed embeddings exceeds a specified threshold. The thresholds in Figure 6 are represented by larger, faded circles surrounding the mean embeddings. In the few-shot evaluation scenario, we compute the mean embedding for each attack type using k embeddings per attack. An incoming embedding is classified as malicious if its cosine distance to the mean embedding of an attack falls below a certain threshold.

4 Experimental Setup

The one-class SVM in our implementation utilized the default hyperparameters from the sklearn package, specifically version 1.2.2. The architecture of MetaFlowGuard consists of a straightforward fully connected feed-forward neural network. It comprises a dense layer with 128 neurons and ReLU activation, followed by a dense layer with 256 neurons and ReLU activation. Finally, we have an embedding layer with 512 neurons. During training, MetaFlowGuard was trained for 50 epochs using the Adam optimizer and a learning rate of 0.001. The AutoEncoder architecture comprises a simple fully connected feed-forward neural network. It is composed of multiple layers, with the middle layer responsible for compressing the input to 10% of it's original size. Each layer is separated by a Rectified Linear Unit (ReLU) activation function.

Our algorithms are designed to be deployed with a specific working point, allowing the security analyst to choose their desired false positive rate. This involves tuning the threshold in a manner that achieves the desired false positive rate. It is important to note that the chosen threshold will impact the false negative rate of the algorithm. For instance, in the zero-shot evaluation, a lower distance threshold for identifying malicious flows would result in more false positives, while a higher distance threshold would decrease false positives but increase false negatives. The threshold selection plays a crucial role in balancing the trade-off between false positives and false negatives. We will provide a diverse set of metrics to comprehensively analyze the performance of our method from various perspectives. Firstly, we will provide a ROC plot that illustrates the true positive rate (TPR) for different working points determined by the false positive rate (FPR). This plot allows for a comprehensive analysis of the algorithm's performance across various thresholds. Additionally, we will report the Area Under the ROC Curve (AUC), which provides a single numerical value to compare the performance of different algorithms across all working points. A higher AUC generally indicates better overall performance. Furthermore, we will present the TPR at a specific FPR value, denoted as TPR@FPR=x. This metric focuses on the TPR achieved at a particular false positive rate. Specifically, we will provide results for false positive rates of 0.01 and 0.001, allowing for an assessment of performance at different levels of false positives.

5 Results

5.1 Zero-Shot Evaluation

5.1.1 IoTID20 dataset [7]

The ROC curve and AUC scores depicting the performance of our methods on the IoTID20 dataset can be found in Figure 7. A notable observation is that both the AutoEncoder and the One-Class SVM outperforms MetaFlowGuard at higher false positive rates, while MetaFlowGuard excels at lower false positive rates. Since our algorithm aims to be a part of an intrusion detection system, a higher true positive rate at a low false positive rate, as exhibited by MetaFlowGuard, is more desirable.

Furthermore, the AUC score of MetaFlowGuard is slightly superior to One-Class SVM. Given that relevant working points in intrusion detection systems typically involve low false positive rates, we

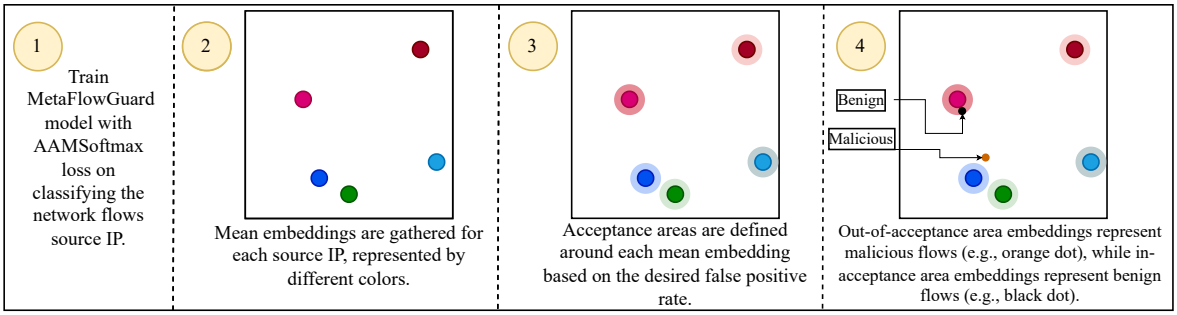

Figure 6: Overview of the MetaFlowGuard algorithm for zero-shot scenario. In the few-shot scenario, step 1 and 3 remains the same. In step 2, mean embeddings are gathered for each malicious class. In step 4, embeddings are considered malicious if they are close to the memorized centers, meaning they are inside the acceptance areas.

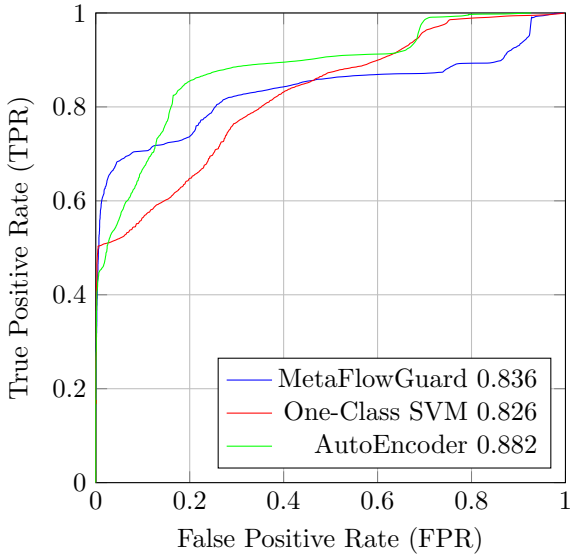

Figure 7: ROC curves for zero-day evaluation on the IoTID20 dataset. The numbers in the legend indicates the AUC score of each algorithm

evaluated the algorithms using specific working points with low false positive rates, as presented in Table 1. Each value in the table represents the true positive rate (TPR) for a specific working point and attack type. The results highlight that MetaFlowGuard generally outperforms the other two, especially for the working point with an FPR of 0.01. MetaFlowGuard surpasses them in detecting every attack at this working point and exhibits superior performance on most attacks for the working point with an FPR of 0.001. Notably, while the One-Class SVM and AutoEncoder yields TPR values close to zero for numerous attacks, MetaFlowGuard demonstrates better generalization capabilities.

5.1.2 MQTT-IOT-IDS2020 dataset [2]

To showcase the versatility of our method, we extended our evaluation to include another dataset. We conducted the experiment twice: first with only rows that had complete data, and secondly, with all rows while replacing missing data with a value of -1 as proposed in [2].

In the first experiment, we had two labels - "mqtt_bruteforce" attack and "Normal." Remarkably, these two labels were completely separable, achieving a perfect score of 100% in classification performance.

For the second experiment, we observed very similar results for False Positive Rates (FPRs) of 0.1 and 0.05, prompting us to report on the better FPR of 0.05. Similarly, the results for FPRs of 0.01, 0.001, and 0.0001 were very close, leading us to report on the better FPR of 0.0001. Our model demonstrated excellent performance on this dataset, achieving an impressive ROC AUC of 0.944. The ROC graph is shown in Figure 8, while Table 2 presents the TPR at different tested FPRs.

The evaluation on the additional dataset further confirms the efficacy of our model in handling diverse data and showcases its robustness across different scenarios.

5.2 Few-Shot Evaluation

To comprehensively assess our model's performance and its ability to handle new and unseen flows, we conducted a Few-shot evaluation. This evaluation simulates a scenario where a cyber analyst encounters a new flow of communication and seeks to trace it, having access to only a few instances marked as the new behavior. Importantly, the Few-shot evaluation is entirely independent of attacks known at the time of training, as no malicious traffic was used in training.

Attack	MetaFlowGuard		One-Class SVM		Autoencoder	
	@0.01	@0.001	@0.01	@0.001	@0.01	@0.001
DoS-Synflooding	0.99	0.91	0.99	0.99	0.99	0.99
MITM ARP Spoofing	0.25	0.17	0.17	0.00	0.01	0.00
Mirai-Ackflooding	0.68	0.30	0.54	0.00	0.47	0.00
Mirai-HTTP Flooding	0.69	0.32	0.53	0.00	0.47	0.00
Mirai-Hostbruteforceg	0.20	0.09	0.08	0.00	0.07	0.01
Mirai-UDP Flooding	0.90	0.31	0.86	0.55	0.78	0.56
Scan Hostport	0.08	0.03	0.04	0.00	0.05	0.00
Scan Port OS	0.04	0.00	0.00	0.00	0.01	0.00
All Attacks	0.58	0.28	0.50	0.27	0.45	0.28

Table 1: Performance comparison of different models on the IoTID20 dataset. The second row denotes true positive rate at false positive rates of 0.01 and 0.001 accordingly.

Attack	MetaFlowGuard	
	@0.05	@0.0001
mqtt_bruteforce	0.53	0.00
scan_A	0.25	0.00
scan_sU	0.68	0.00
sparta	1.00	0.36
All Attacks	0.84	0.24

Table 2: Performance comparison of different models on the MQTT-IOT-IDS2020 dataset. The second row denotes true positive rate at false positive rates of 0.05 and 0.0001 accordingly.

In the Few-shot evaluation, we gathered a predetermined number of labeled examples from each malicious traffic category. Using our MetaFlowGuard model - the same one employed in the zero-shot evaluation. We calculated embeddings for each example, we memorizing and obtained the mean embedding for each category. Subsequently, we compared the embeddings of unlabeled examples to find the most similar memorized embedding.

To score the unlabeled examples, we assigned a score of 1 minus the similarity if the most similar embedding belonged to the benign category, and 1 plus the similarity if it belonged to a malicious category. This approach allows us to emphasize higher scores for more confident malicious detections and lower scores for more confident benign identifications.

For thoroughness and robustness, we tested the evaluation with various memory sizes: 1, 5, 10, 30, 50, and 300. To mitigate the impact of random choices, we repeated the experiment 100 times and averaged the True Positive Rates (TPRs) and False Negative Rates (FNRs) of the examples. The averaged results are presented in the ROC curves shown in Figure 9, and the corresponding TPRs for various FPRs are reported in Table 3.

Our results demonstrate that even a single memorized example can yield valuable information and outperform the zero-shot evaluation at FPRs higher than 0.38. To achieve lower FPRs while maintaining effective detection of malicious flows, more memorized examples are required. Beyond 50 examples, there are diminishing returns, making it less worthwhile to invest additional effort in finding more examples of the new flow, considering the manual and time-consuming nature of this process. The different between memorization size of 30 and 50 is quite small as well. Notably, compare to the zero-shot, with 30 examples, the Few-shot evaluation performs better at FPRs above 0.2, and with 10 examples, it performs better at FPRs above 0.25. We recommend aiming for at least 10 examples, as there are significant performance improvement over having only 1 or 5 examples.

While the Few-shot evaluation shows inferior performance to the zero-shot evaluation at lower FPRs, it excels at higher FPRs, becoming the best performing evaluation overall. This evaluation can serve as a valuable tool for cyber analysts, helping them find more examples that are likely to belong to the same flow and facilitating their research efforts. Furthermore, it can be deployed as an Intrusion Detection System (IDS), assisting in identifying suspicious flows and allowing cybersecurity experts to make informed decisions based on the model's decision, and decide if it is a false alarm or not.

Attack	1-shot		10-shot		30-shot	
	@0.25	@0.2	@0.25	@0.2	@0.25	@0.2
DoS-Synflooding	0.84 ± 0.21	0.78 ± 0.21	0.97 ± 0.06	0.92 ± 0.06	0.99 ± 0.04	0.96 ± 0.04
MITM ARP Spoofing	0.74 ± 0.32	0.64 ± 0.26	0.92 ± 0.18	0.80 ± 0.16	0.98 ± 0.10	0.87 ± 0.11
Mirai-Ackflooding	0.71 ± 0.34	0.56 ± 0.27	0.90 ± 0.21	0.64 ± 0.15	0.98 ± 0.11	0.66 ± 0.06
Mirai-HTTP Flooding	0.71 ± 0.34	0.55 ± 0.27	0.90 ± 0.21	0.64 ± 0.15	0.98 ± 0.11	0.66 ± 0.07
Mirai-Hostbruteforceg	0.71 ± 0.25	0.61 ± 0.20	0.84 ± 0.15	0.67 ± 0.10	0.91 ± 0.08	0.67 ± 0.05
Mirai-UDP Flooding	0.71 ± 0.31	0.53 ± 0.30	0.94 ± 0.13	0.80 ± 0.09	0.99 ± 0.06	0.84 ± 0.05
Scan Hostport	0.73 ± 0.13	0.68 ± 0.11	0.80 ± 0.04	0.75 ± 0.02	0.83 ± 0.03	0.75 ± 0.01
Scan Port OS	0.76 ± 0.14	0.72 ± 0.13	0.84 ± 0.04	0.79 ± 0.02	0.87 ± 0.02	0.79 ± 0.01
All Attacks	0.73 ± 0.25	0.61 ± 0.20	0.90 ± 0.13	0.75 ± 0.09	0.95 ± 0.07	0.78 ± 0.05

Table 3: Performance comparison of different models on the IoTID20 dataset. denoting true positive rate at false positive rates of 0.25 and 0.2 accordingly and displaying the standard deviation of said metrics.

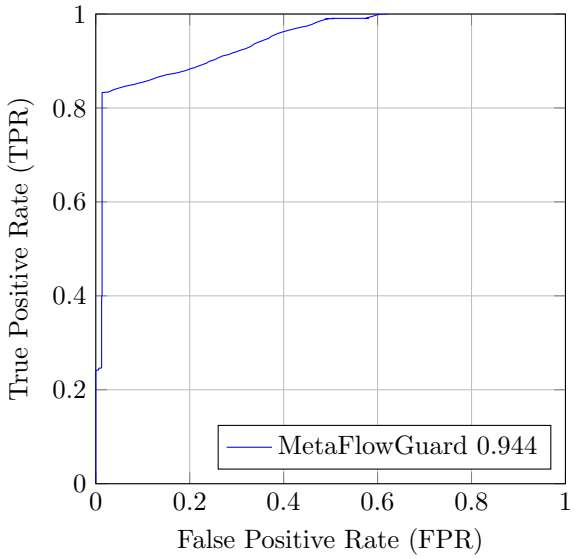

Figure 8: ROC curve for zero-day evaluation on the MQTT-IOT-IDS2020 dataset. The number in the legend indicates the AUC score of the model

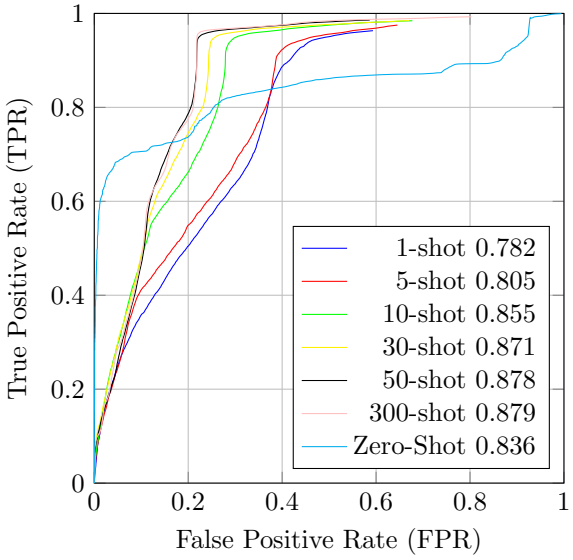

Figure 9: ROC curves for few-shot evaluation (for 1, 5, 10, 30, 50, 300 memory sizes) on the IoTID20 dataset. The numbers in the legend indicate the AUC score of each algorithm. We added the Zero-shot graph as well for comparison.

6 Conclusions

In this paper, we have successfully demonstrated an effective method for detecting malicious flows in an IoT network. A significant highlight of our approach is its ability to achieve impressive results even when trained solely on benign records, making it a robust and practical solution for intrusion detection. We further showcased the versatility of our method by evaluating it on two distinct datasets, where it exhibited promising and valuable capabilities for both cases.

In our evaluation, we compared our model against widely used anomaly detection models, such as One-Class SVM and AutoEncoder. The results showed that our model outperformed these methods at lower False Positive Rates (FPRs),

proving its efficacy in accurately identifying malicious activities. While AutoEncoder excelled at higher FPRs, our model still exhibited strong performance in this regard.

Moreover, we explored the model's generalization capabilities through zero-shot evaluation, where it effectively detected attacks it had never encountered during training. This demonstrates the model's adaptability to handling new and previously unseen threats. Additionally, we conducted a few-shot evaluation, simulating a scenario where a cybersecurity analyst identifies a new flow and seeks to find similar ones. The results indicated that the few-shot evaluation performed better than zero-shot at higher FPRs, while zero-shot had an advantage at lower FPRs.

As part of the few-shot evaluation, we experimented with varying memorization sizes and observed improved results up to a memorization size of 30. Beyond this point, the returns diminished.

In conclusion, our method offers a robust and versatile solution for detecting malicious flows in IoT networks. Its ability to perform well with limited training data, handle new threats, and assist in cybersecurity research makes it a valuable asset for intrusion detection and analysis.

References

[1] Deng, J., Guo, J., Xue, N., and Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 4690–4699.

[2] Hindy, H., Tachtatzis, C., Atkinson, R., Bayne, E., and Bellekens, X. Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset, 2020.

[3] Jordan, M. I., and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science 349*, 6245 (2015), 255–260.

[4] Komisarek, M., Kozik, R., Pawlicki, M., and Choraś, M. Towards zero-shot flow-based cyber-security anomaly detection framework. *Applied Sciences 12*, 19 (2022), 9636.

[5] Lotfi, S., Modirrousta, M., Shashaani, S., Amini, S., and Shoorehdeli, M. A. Network intrusion detection with limited labeled data. *arXiv preprint arXiv:2209.03147* (2022).

[6] Sarhan, M., Layeghy, S., Gallagher, M., and Portmann, M. From zero-shot machine learning to zero-day attack detection. arxiv 2021. *arXiv preprint arXiv:2109.14868*.

[7] Ullah, I., and Mahmoud, Q. H. A scheme for generating a dataset for anomalous activity detection in iot networks. In *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33* (2020), Springer, pp. 508–520.

[8] Wang, H., Bah, M. J., and Hammad, M. Progress in outlier detection techniques: A survey. *Ieee Access 7* (2019), 107964–108000.

[9] Wang, Z., Li, Z., Wang, J., and Li, D. Network intrusion detection model based on improved byol self-supervised learning. *Security and Communication Networks 2021* (2021), 1–23.

[10] Yu, L., Dong, J., Chen, L., Li, M., Xu, B., Li, Z., Qiao, L., Liu, L., Zhao, B., and Zhang, C. Pbcnn: packet bytes-based convolutional neural network for network intrusion detection. *Computer Networks 194* (2021), 108117.