Importance Analysis of Micro-Flow Independent Features for Detecting Distributed Network Attacks

Samuel Kopmann and Martina Zitterbart

***Abstract*—Network infrastructures are critical and, therefore, subject to harmful attacks against their operation and the availability of their provided services. Detecting such attacks, especially in high-performance networks, is challenging considering the detection rate, reaction time, and scalability. Attack detection becomes even more demanding concerning networks of the future facing increasing data rates and flow counts. We thoroughly evaluate eMinD, an approach that scales well to high data rates and large amounts of data flows. eMinD investigates aggregated traffic data, i.e., it is not based on micro-flows and their inherent scalability problems. We evaluate eMinD with real-world traffic data, compare it to related work, and show that eMinD outperforms micro-flow-based approaches regarding the reaction time, scalability, and the detection performance. We reduce required state space by 99.97%. The average reaction time is reduced by 90%, while the detection performance is even increased, although highly aggregating arriving traffic. We further show the importance of micro-flow-overarching traffic features, e.g., IP address and port distributions, for detecting distributed network attacks, i.e., DDoS attacks and port scans.**

***Index Terms*—Network intrusion detection, machine learning, traffic monitoring, traffic aggregation.**

I. Introduction

NETWORK infrastructures are critical and, therefore, subject to harmful attacks against their operation and the availability of their provided services. Detecting such attacks, especially in high-performance networks, is challenging considering, for example, detection rate, reaction time, and scalability concerning the number of flows and high data rates.

Network intrusion detection systems (NIDSs) monitor and classify arriving network traffic, to detect malicious activities and raise the alarm if an attack is occurring. The application of machine learning (ML) in this context has become possible due to the increased processing power available for training and inference. Therefore, ML approaches have recently become increasingly popular [1], [2], [3] in detecting attacks against network infrastructures.

ML-based NIDSs no longer require hand-crafted attack signatures, as they learn them from previously monitored traffic. Nevertheless, sophisticated traffic monitoring is critical for developing scalable and well-performing solutions. Monitoring must not be expensive regarding memory and computational resources to adapt to high data rates but needs to capture as many detection-relevant traffic characteristics as possible. A common approach of ML-based NIDSs is monitoring and classifying arriving traffic on the granularity of micro-flows [4], which are identified by the five-tuple ⟨source IP address, destination IP address, source port, destination port, transport protocol⟩. These NIDSs match header fields of arriving packets according to the five-tuple and calculate features per micro-flow, e.g., the mean packet length, serving as input for an ML model.

Micro-flow-based NIDS have already shown success for real-world data [5]. However, the buildup of micro-flow context is challenging, considering increasing data rates. The higher the data rate, the less time is available for the NIDS to match arriving packets. If packets arrive too fast for micro-flow matching, they are not monitored, e.g., due to sampling, and detection-relevant information is potentially discarded. Furthermore, attackers can exploit micro-flow-based classification by creating large amounts of different micro-flows, e.g., spoofing IP addresses, which turns the NIDS into the primary bottleneck. In the scenario of *distributed network attacks*, such as Distributed Denial of Service (DDoS) attacks with thousands of sources (different IP addresses) or port scans with thousands of destinations (different port numbers), attackers do not even need IP spoofing to cancel the NIDS, as distributed attacks inherently go along with large amounts of different micro-flows because the five-tuple contains both source IP and destination port. eMinD [6] addresses these scalability issues by introducing an efficient and *M*icro-flow *in*dependent *D*etector for distributed network attacks, completely *avoiding packet matching and state keeping*.

State keeping and management is an essential part of micro-flow-based NIDS, as the expressiveness of every micro-flow needs to be assessed regarding the amount of collected data. Further, the NIDS needs to decide if monitored micro-flows are still active or already completed, and remove completed flows from monitoring to not waste state space.

Micro-flow-based NIDSs require one classification through the ML model per micro-flow, resulting in computational effort correlated to the micro-flow count. This becomes infeasible

This work was funded by the German Federal Ministry of Education and Research (BMBF), RefNr. 16KIS1142K. This work was supported by funding of the Helmholtz Association (HGF) through the Kastel Security Research Labs (POF structure 46.23.01: Methods for Engineering Secure Systems). The associate editor coordinating the review of this article and approving it for publication was X. Fu. *(Corresponding author: Samuel Kopmann.)*

Samuel Kopmann is with the Institute of Telematics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: samuel.kopmann@kit.edu).

Martina Zitterbart is with the KASTEL Security Research Labs, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: martina.zitterbart@kit.edu).

for future ML-based NIDSs, as on the one hand, traffic volume and micro-flow count increase and on the other hand, employing larger models and privacy-preserving techniques is desired, e.g., homomorphic encryption [7], [8], increasing the classification duration. In addition, the reaction time, i.e., the time from the attack's start until its detection, depends on the micro-flow count. The NIDS must classify every monitored micro-flow for attack detection in the worst case.

Contribution. This work extends eMinD [6] that introduced a reliable and effective NIDS independent of the number of active micro-flows. In addition to eMinD's contributions, we thoroughly evaluate the importance of monitored features in Sections VIII and IX, besides 1) the required state space, 2) the classification frequency, and 3) the reaction time. Furthermore, we show that eMinD 4) improves the detection performance of distributed network attacks compared to micro-flow-based approaches, as it makes micro-flow-overarching traffic features observable. We further show 5) the importance of these newly observable features for the detection of distributed network attacks. Therefore, we provide the following contributions:

1) eMinD has a *predictable and strictly limited state space*. It uses only a fixed set of features independent of the micro-flow count, reducing the state space by 99.97%.
2) eMinD has a *strictly limited classification frequency*. It performs one classification periodically, reducing the amount of classifications by 99.98%.
3) The reaction time of eMinD is *short and fixed by a predefined time interval*. eMinD returns attack detection results periodically on very short time scales and reduces the average reaction time by 90%.
4) eMinD achieves a *detection performance increase* compared to micro-flow-based NIDS with different ML approaches, i.e., Decision Tree, Random Forest, Multilayer Perceptron, and Support Vector Machine.
5) eMinD makes micro-flow-overarching traffic features observable, e.g., unique source IPs and destination ports that are critical for detecting distributed attacks, which is shown in the evaluation.

Evaluation. We use the real-world CIC-IDS2017 [9] data set containing port scan and DDoS traffic to prove the above contributions. We compare eMinD to the approach of Choobdar et al. [10] as they use the same data set and specifically perform DDoS and port scan detection. Furthermore, we use a composition of real-world MAWI backbone traffic [11], [12], [13], [14] and CAIDA DDoS traffic [15] for scalability evaluations as high-bandwidth DDoS scenario. With the *Intrusion Detection Dataset Toolkit* [16], we inject authentic port scans into MAWI backbone traffic to create a data set for scalability evaluations concerning port scan detection.

II. Related Work

We compare eMinD to two different concepts of supervised ML-based NIDS (see Fig. 1), namely *micro-flow-level NIDS* and *packet-level NIDS*. All approaches, including eMinD, process individual packets arriving at an ingress router of the protected downstream network.

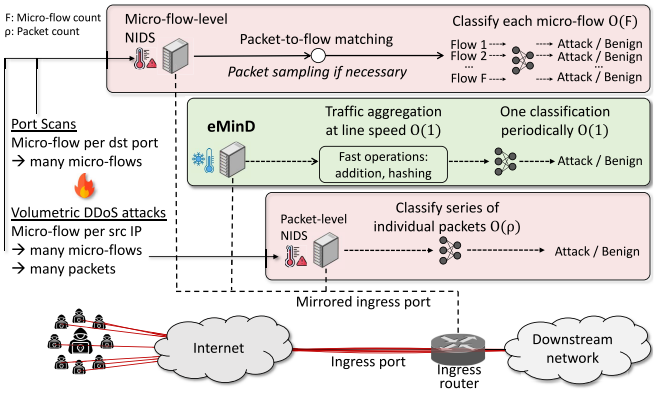


Fig. 1. Comparison of supervised ML-based NIDS approaches.

Micro-flow-level NIDS. Micro-flow-level NIDS [10], [17], [18], [19], [20] classify individual micro-flows. For every incoming packet at the ingress router, the NIDS determines the corresponding micro-flow and updates its state, which consists of calculated features, e.g., flow duration, packet count, and byte count. Packet-to-flow matching is computationally expensive, especially with an increasing amount of active micro-flows, and is computationally infeasible in the context of distributed network attacks, as thousands of different micro-flows are active simultaneously. In contrast, eMinD calculates only one set of traffic features during monitoring, using fast operations like addition, enabling monitoring at line speed.

Even if the NIDS can monitor micro-flows at line speed, the ML model has to perform one classification for every active micro-flow to detect ongoing attacks. Therefore, the reaction time of the NIDS depends on the number of active micro-flows and their duration, which can lead to long reaction times in the worst case. In contrast, eMinD only requires one classification in a given time frame, completely independent of micro-flows.

Micro-flow-level NIDS never consider distributions of IP addresses or port numbers, spanning multiple or all active micro-flows. We show in this work that those *distributions* represent the most important characteristics of *distributed* attacks. As eMinD highly aggregates arriving network traffic, and therefore, widens the scope from five-tuple inspection, it can monitor address distributions and use them for training and classification, which improves detection performance.

Packet-level NIDS. Packet-level NIDS classify raw traffic input [21] or complete packets [22]. These NIDS do not require expensive state keeping per se, but they suffer from inefficiency at high data rates, as the NIDS's maximum possible packet processing rate depends on the classification duration. If the classification speed cannot keep up with line speed, e.g., in the context of volumetric DDoS attacks with high peak data rates, traffic needs to be sampled, and detection relevant information is potentially lost. Furthermore, packet-level NIDS also have only a limited view of address distributions, as they only observe a limited amount of packets. eMinD has neither of those limitations, as traffic is highly aggregated. Aggregation operations, such as addition and hashing, can be performed at line speed, in contrast to

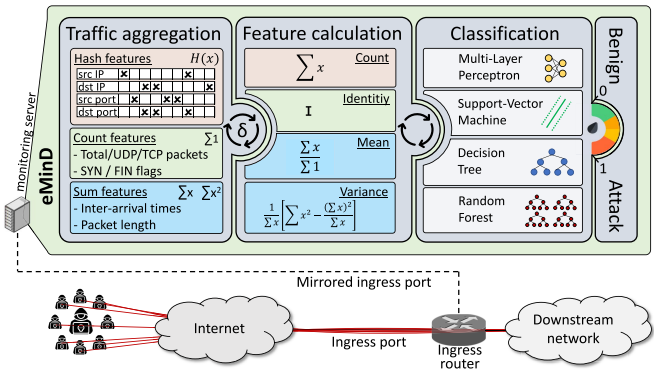


Fig. 2. Illustration of eMinD's workflow from traffic aggregation, over feature calculation to attack detection.

classification, while enabling address distribution observation at the same time.

ML approaches. Supervised ML-based NIDSs employ a variety of different approaches, e.g., decision trees [19], random forests [17], [18], [19], support vector machines, and artificial neural networks [10], [18], [20]. We show the flexibility of eMinD concerning the used ML approach in the evaluation section, which means that one can implement eMinD according to individual constraints regarding the selected ML approach, e.g., the training duration.

III. eMinD in a Nutshell

Fig. 2 shows an overview of eMinD's workflow, which employs the following components consecutively: A. Traffic aggregation, B. Feature calculation, and C. Classification. eMind runs on a monitoring server collocated with an ingress router. Arriving traffic at the ingress router is mirrored to the monitoring server, which calculates a highly aggregated traffic representation that an ML model periodically classifies. Neither micro-flow matching nor management is required.

A. Traffic Aggregation

Monitoring at line speed is key for observing detection relevant traffic features and not discarding information, e.g., through packet sampling. Therefore, eMinD only employs fast operations for traffic aggregation, such as hashing, adding, and multiplying. In contrast, we explicitly exclude the distinction of cases, dividing, or the modulo operation. This ensures that eMinD can keep up with line speed even at very high data rates. We distinguish between three classes of aggregation features, namely *Hash features*, *Count features*, and *Sum features*.

Hash features enable eMinD to estimate how many different feature values occur. eMinD monitors the source and destination IP addresses and port numbers. For each hash feature, eMinD initializes an array of fixed length. All entries are zero at the beginning.

On packet arrival, each hash feature is processed by an efficient hash function, uniformly and deterministically distributing values of the feature space over the array. For every feature, the array field determined by hashing is set to 1. The same feature values have the same hash value and, therefore, the same corresponding field in the array. Different feature values can have different corresponding array fields after hashing. This enables eMinD to *estimate* the number of different feature values by counting the number of 1-entries in the corresponding array. Hash collisions imply underestimations and their probability is determined by the size of the array. Therefore, a trade-off between the array size (required memory) and the estimation's precision needs to be considered. *Hash features* can be monitored at line speed, as computationally efficient hash functions exist [23], e.g., implementing hash functions with XOR operations.

We show in the evaluation section that *Hash features* constitute a significant conceptual improvement over micro-flow-based features regarding the detection of distributed network attacks, as *Hash features* capture distributions of features overarching the scope of micro-flows, such as source IP and destination port distributions, which are strong indicators for DDoS attacks and port scans.

Count features are implemented by the simple increment operation. eMinD counts the total number of packets, the number of UDP and TCP packets, and the TCP SYN and FIN flags. *Count features* are also part of micro-flow-based NIDS but their overarching observation in eMinD enables better detection of distributed attacks, e.g., the total traffic volume to detect DDoS attacks and the total SYN flag count to detect port scans.

Sum features represent traffic aggregates later used to calculate advanced derived features. eMinD cannot calculate derived features, namely *mean* and *variance*, during traffic aggregation, as the required computational operations are too expensive to keep up with line speed at very high data rates. For all monitored *Sum features*, which are *Inter-arrival time* and *packet length*, the sum of all values and the sum of squared values is calculated.

B. Feature Calculation

eMinD employs a periodic timer with period δ seconds, which we denote as time frame. After every time frame, *Hash*, *Count* and *Sum features* from traffic aggregation are read, used to calculate derived features that eMinD feeds into the ML model, and reset afterwards. Each feature used in traffic aggregation is used for one or more feature calculations, which are color-coded in Fig. 2, i.e., *Hashed features* are only counted; *Count features* are not processed further before classification (illustrated by the identity operation in green), while *Sum features* are used to calculate both *mean* and *variance* for the corresponding feature.

In contrast to traffic aggregation, feature calculation can perform more expensive operations, such as division operations (required for mean and variance), because feature calculation is only performed once a time frame and not once a packet arrival.

Time frame δ is chosen before training and is then fixed. The smaller δ is, the shorter is the reaction time of eMinD, as δ implies the lower boundary of the ML model's classification frequency at the end of eMinD's workflow. On the other hand,

TABLE I
TRAFFIC AGGREGATES AND CALCULATED FEATURES' COUNT

	Feature Calculation				
Aggregate	Count	Identity	Mean	Variance	Total
Source IP	✓	-	-	-	1
Destination IP	✓	-	-	-	1
Source port	✓	-	-	-	1
Destination port	✓	-	-	-	1
Total packets	-	✓	-	-	1
UDP packets	-	✓	-	-	1
TCP packets	-	✓	-	-	1
SYN flags	-	✓	-	-	1
FIN flags	-	✓	-	-	1
Inter Arrival Times	-	-	✓	✓	2
Packet lengths	-	-	✓	✓	2
Total	4	5	2	2	*13*

a longer time frame goes along with two benefits. First, eMinD aggregates more traffic during a longer time frame, which leads to more traffic information in the monitored features, providing more information to the ML model and improving detection performance. Second, longer time frames enable the use of more complex ML models that require more time for classification, which may also lead to better detection performance. Therefore, a trade-off between eMinD's reaction time and the attack detection performance needs to be considered.

Feature List

Tab. I summarizes all *Hash*, *Count* and *Sum features* eMinD is monitoring during traffic aggregation and further shows, how the features are processed during feature calculation (compare with Fig. 2). The table also provides information about the total amount of resulting features fed into the ML model. eMinD uses only *one set* of 13 features, which represents a very low memory footprint compared to micro-flow-based NIDS using 84 traffic features *per micro-flow* [10]. Note that hash features can be efficiently implemented with one bit per array field, as each field only stores binary state. To conclude, *eMinD has fixed monitoring state size* independent of the active micro-flow count or the ingress traffic volume.

C. Traffic Classification

The ML model is trained offline in a supervisory manner. After eMinD has calculated derived features from traffic aggregation of the corresponding time frame, they serve as input for the employed ML model. We show the performance of eMinD with *Decision Trees*, *Random Forests*, *Support Vector Machines* (SVM), and *Multi-layer Perceptrons* (MLP) in the evaluation to provide comparability to related work and to demonstrate eMinD's flexibility regarding the ML approach. The output of the ML model represents attack detection during the corresponding time frame, which constitutes a classification of the time frame's traffic. eMinD performs *only one classification per time frame*, independent of the ingress traffic volume and the number of active micro-flows, which guarantees scalability at high data rates while at the same time encouraging the utilization of more complex ML models with longer classification durations.

IV. DATA SETS

This section covers data set descriptions, as well as the preprocessing and training procedure. We selected two different types of data sets. First, we use the CIC-IDS2017 data set, as it contains both DDoS attack traffic and port scans, whose detection is the focus of this paper. It is publicly available and widely used in related work, so results provide comparability. Second, we use real Internet backbone traffic from MAWI and CAIDA to create a data set composition that contains an authentic DDoS attack at high data rates to show scalibility of eMinD. In addition, we inject authentic port scans with the tool *ID2T* into MAWI backbone traffic to evaluate port scan detection in a high bandwidth scenario. ML model training and hyperparameter optimization is performed offline in a supervisory manner. Traffic composition, preprocessing and model training are publicly available [24].

A. CIC-IDS2017 Data Set

The CIC-IDS2017 data set [9] was developed and published by the *Canadian Institute for Cybersecurity* (CIC) in 2017. Traffic has been recorded over five days and contains various different network attacks. However, for training and testing eMinD, we only use a specific part of the data set, namely *Friday afternoon*, as this part of the data set contains both DDoS and port scan traffic. We explicitly use this data set to provide comparability to related approaches, particularly to Choobdar et al. [10], considering detection performance.

B. MAWI and CAIDA2007 Data Set Composition

To evaluate the scalability of eMinD, we use authentic traffic traces recorded in the backbone of the Internet from MAWI [11], [12], [13], [14], as well as DDoS traffic provided by CAIDA [15]. Each of the MAWI traces lasts 15 minutes, resulting in 60 minutes of background traffic in total. From the CAIDA data set, we cut 30 minutes of high-volume DDoS traffic to create a traffic composition of background and attack traffic. Data rates in this traffic composition exceed 1 Gbit/s and contain thousands of attack sources.

C. MAWI and ID2T Data Set Composition

To also evaluate the scalability of eMinD regarding port scan detection, we injected port scans into high-volume MAWI traffic with the *Intrusion Detection Dataset Toolkit* (ID2T) [16]. ID2T enables injecting port scans into background traffic with certain parameters, e.g., start of the scan, duration of the scan and its intensity (in ports per second). Scans with higher intensities are easier to detect than scans with low intensities, which are often referred to as stealthy scans. Injected scans are non-stealthy TCP SYN scans with randomized source and destination ports.

D. Preprocessing

Once a time frame is selected for eMinD, it is fixed during training and testing. Therefore, testing different time frames

requires individual data sets. For the CIC-IDS2017 data set, which is not a self-composition, the start and the duration of the contained attacks is required, as well as the labeled micro-flows, to extract correct parts of the traffic trace and preform correct labeling.

All features are normalized individually before training and testing using *z-score normalization*, which requires the mean and standard deviation of the used data. The mean and the standard deviation is only obtained from training data and then used for the normalization of test data, ensuring that no information is gained about testing data through normalization.

V. Evaluation Goals and Setup

The evaluation of eMinD focuses on the detection of DDoS attacks and port scans and is three-fold:

1) We evaluate eMinD's *detection performance* (Section VI) of DDoS attacks and port scans and compare results with related work. We further evaluate the ML models' classification duration to show that time frames are not too short for periodic attack detection.
2) We conduct a *feature importance* (Section VII) evaluation to show that micro-flow-overarching features, e.g., *Hash features* that are observable through eMinD, are highly important for detecting distributed attacks.
3) We compare eMinD with micro-flow-based NIDS regarding its *scalability* (Section X), i.e., reaction time, classification frequency, state space, and saved micro-flow management.

We restrict time frame space for evaluation, such that $\delta \in \{0.25s, 0.5s, 0.75s, 1.0s, 1.25s, 1.5s, 1.75s\}$. We chose $1.75s$ as upper boundary for the time frames because the median duration of micro-flows in the inspected part of the CIC-IDS2017 data set is $1.6s$. As the compared micro-flow-based approach [10] classifies completed micro-flows, the flow-duration represents the minimum reaction time, providing comparability regarding eMinD's reaction time.

Each hash feature is monitored with an array of size 1000. Experiments concerning the array size are not contained due to space limitations. We monitor the metrics *Accuracy*, *Precision*, *Recall* and the derived *F1-score* during training, hyperparameter optimization, and testing. Training and testing is performed on two disjunct data sets, the *training set* (70 percent of the original data set) and the *test set* (30 percent of the original data set) to guarantee that no training data is used for testing. Both training and testing sets are balanced, i.e., they contain an equal amount of benign and attack samples, ensuring the expressiveness of the accuracy metric.

VI. Detection Performance

For DDoS attacks and port scans, we evaluate eMinD's detection performance on the CIC-IDS2017 data set (Section IV-A) and compositions of data sets with high data rates, i.e., MAWI/CAIDA (Section IV-B) and MAWI/ID2T (Section IV-C). Results are collected from 100 individual test runs.

We compare our results with *Choobdar et al.* [10], as they perform micro-flow-level classification on the same data set (CIC-IDS2017). They explicitly evaluate DDoS and port scan detection performance, which makes their approach comparable with eMinD. We use eMinD's MLP for comparison because the MLP shows the largest similarity to their approach regarding the ML model.

A. DDoS Detection Performance

Fig. 3 shows the DDoS detection performance results on the MAWI/CAIDA and the CIC-IDS2017 data sets. Each column represents individual metrics, while color-coded rows indicate the employed ML model. In addition, each row contains two subrows containing results for either the MAWI/CAIDA (upper subrow) or the CIC-IDS2017 (lower subrow) data set. Black markers indicate the median performance over all test runs, while colored areas fill the range between the 5th and the 95th percentile.

First, inspecting results on the CIC-IDS2017 data set, there is a correlation between the duration of time frames and the detection performance. The longer the time frames, the better the detection performance, which holds for every tested ML model and metric. As the time frame duration determines the amount of aggregated traffic, longer time frames imply more aggregated traffic information contained in individual calculated features, which again results in better detection results. This increase is not observable in the MAWI/CAIDA backbone scenario, as traffic arrives at a Gbit data rate (compare Fig. 10(c)), already providing enough information for *perfect detection even with short time frames*.

In the bottom row of Fig. 3, the performance of [10] is indicated with a blue triangle for each metric. We selected $1.6s$ for the triangles' x-coordinate according to the median reaction time (see Section X-A) of micro-flow-based NIDS in the CIC-IDS2017 scenario. *eMinD achieves better performance for every metric* with the time frame $\delta = 1.75s$. Even for the time frame $\delta = 1.5s$ implying a median reaction time improvement of 0.1s, eMinD achieves better performance for every metric.

For time frames $\delta \geq 1.75s$, eMinD achieves 100 percent performance for all metrics on the CIC-IDS2017 data set. This results hold for all tested ML approaches, indicating eMinDs flexibility regarding the employed ML approach. Improved detection performance implies that an aggregated view of the arriving network traffic leads to better detection of DDoS attacks, compared to the classification of individual micro-flows.

B. Port Scan Detection Performance

Fig. 4 shows the detection performance results for eMinD's port scan detection. Columns and rows follow the same structure as in Fig. 3, except that the upper subrows contain results from MAWI/ID2T (Section IV-C) data sets instead of MAWI/CAIDA (Section IV-B) data sets. Furthermore, the y-scale shows a different value range, namely between 90 and 100%.

The worst result for port scan detection on the CIC-IDS2017 data set (lower subrows) is the recall of 93% achieved by the MLP at $\delta = 0.25s$. All remaining results, independent of the metric and ML model, are better than 93%, while MLP and

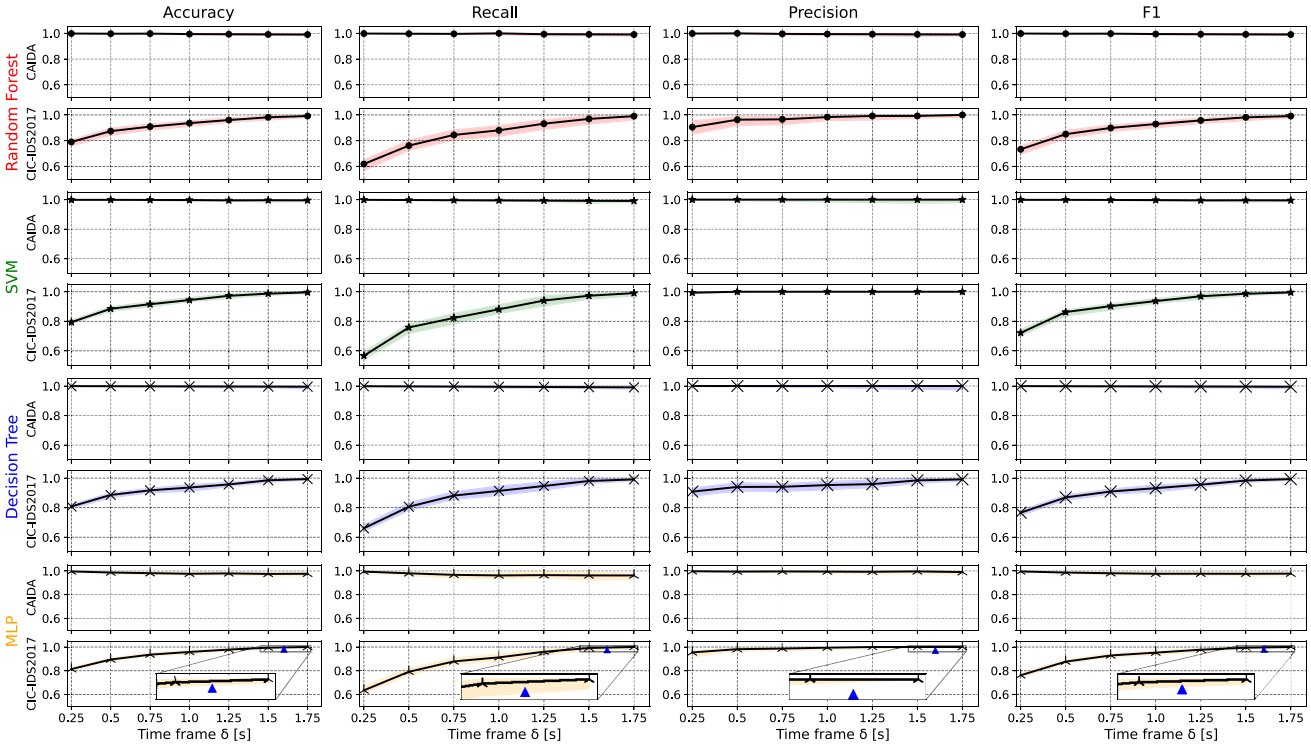


Fig. 3. DDoS detection results for different time frames on CIC-2017 and MAWI/CAIDA data sets with different ML models.

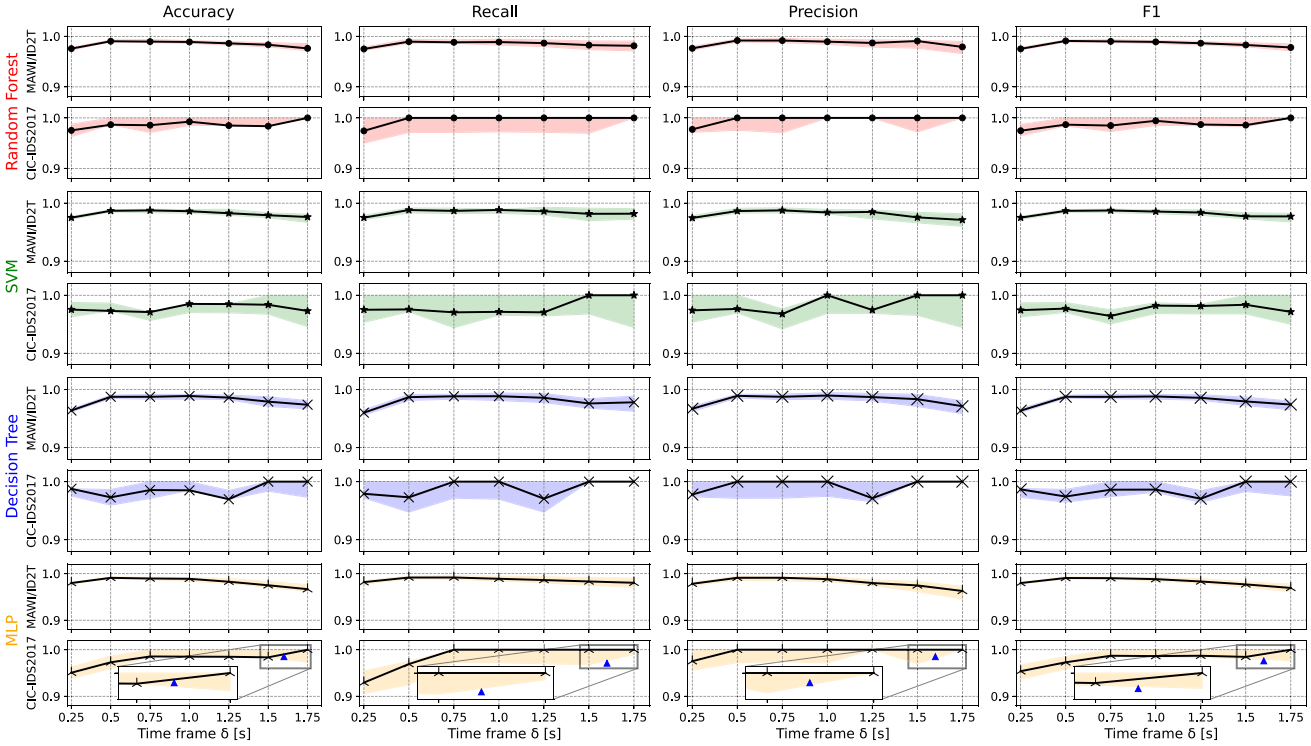


Fig. 4. Port scan detection results for different time frames on CIC-2017 and MAWI/ID2T data sets with different ML models.

Random Forest performing best, with median precision and recall equal to 100% for $\delta \geq 0.75s$.

Results of Choobdar et al. [10] are again displayed as blue triangle in the MLP row for comparison. For every metric, eMinD outperforms the micro-flow-level approach even at all time frames $0.75s \leq \delta < 1.6s$, constituting improved detection performance with an even shorter median reaction time.

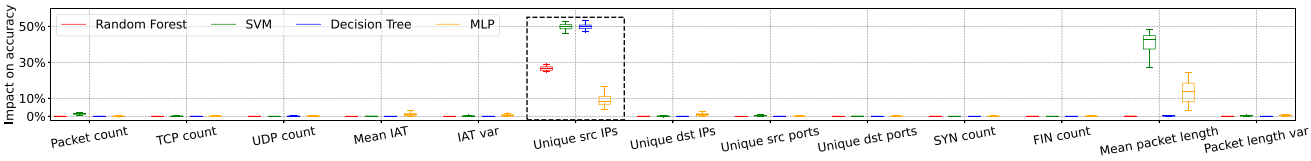


Fig. 5. Feature importance for *DDoS detection* on MAWI/CAIDA data with time frame $\delta = 0.5s$.

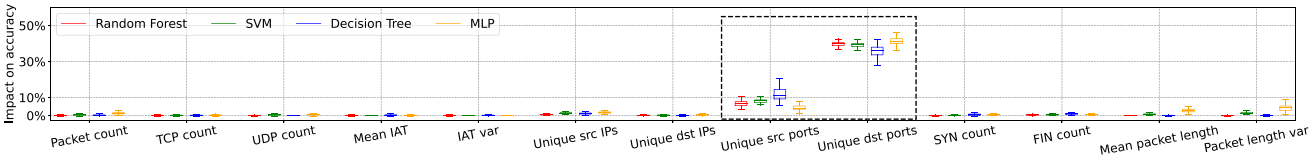


Fig. 6. Feature importance for *port scan detection* on MAWI/ID2T data with time frame $\delta = 0.5s$.

Traffic in the MAWI/ID2T data set contains portscans with an intensity of 1000 ports per second, which means that 1000 attack packets are monitored every second. In contrast, background MAWI traffic contains about 150k packets per second. Therefore, attack traffic constitutes only $\frac{1000}{150k} = \frac{1}{150} = 0.6\%$ of all traffic, in contrast to more than 50% attack traffic in both DDoS scenarios (Section VI-A). This implies an increased difficulty for port scan detection on aggregated traffic compared to DDoS detection, as attack traffic has smaller influence on calculated features. However, the results in Fig. 4 show that eMinD achieves very good results for all metrics and time frames, having its peak detection performance at $\delta = 0.5s$ for all ML models, e.g., 99.8% achieved by the MLP. On the other hand, performance decreases for increasing time frames $\delta > 0.5s$, because attack traffic has less impact on calculated features the more traffic is aggregated due to the imbalance between attack and benign traffic volume.

C. Classification Duration

We have shown in the sections before that eMinD can detect distributed attacks in very short time frames. However, short time frames are only reasonable if the classification duration of the ML model is shorter than the time frame itself. Otherwise, classification cannot keep up with feature calculation (compare Fig. 2), and the reaction time increases. To exclude the classification being the bottleneck, we measured the duration of 1000 classifications by each tested ML model throughout 100 runs and summarized the results in Tab. II, where Q1 and Q3 represent the first and third quartiles.

For every tested ML model, the maximum classification duration (for 1000 classifications!) is between 0.71ms and 22.7ms. This shows that *the classification duration is not a bottleneck* in the eMinD workflow, enabling the usage of computationally more expensive ML models to achieve even better detection performance.

TABLE II
CLASSIFICATION DURATION OF TRAINED ML MODELS

Model	Min	Q_1	Median	Q_3	Max
MLP	1.26ms	1.27ms	1.28ms	1.29ms	2.13ms
Decision tree	0.68ms	0.69ms	0.69ms	0.70ms	0.71ms
Random forest	1.93ms	1.96ms	1.97ms	1.98ms	2.60ms
SVM	21.2ms	22.3ms	22.5ms	22.6ms	22.7ms

VII. FEATURE IMPORTANCE

We evaluated eMinD's features with the *permutation feature importance* score for 100 individual training and test sets. After training, models perform one test run per feature, i.e., 13 runs in the case of eMinD (see Tab. I). In each test run, the values of one feature are randomly shuffled. Shuffling breaks the correlation between the corresponding feature and the target value and dependencies between multiple features. The decrease in the inspected metric, e.g., accuracy, compared to results obtained from unmodified test data, represents the importance of the feature for the ML model. The higher the decrease in the performance metric, the more important the feature is for correct classification. e.g., a feature importance of 50 percent means that a perfect model is not better than guessing after shuffling the corresponding feature.

Fig. 5 shows the *permutation feature importance* score for all tested ML approaches trained on the MAWI/CAIDA DDoS data set with time frame $\delta = 0.5s$. Results show an importance between zero and five percent for most features. However, the estimated number of unique source IP addresses is the most important feature (surrounded by the dashed rectangle) for all models except the MLP, for which it is still important with an average score of 10 percent. For the decision tree and the SVM, it even shows an importance of 50 percent. This confirms the assumption that an aggregated view of the arriving traffic, e.g., *the source address distribution, is highly important for detecting DDoS attacks*, which is not observerd by micro-flow-based NIDS. This result is expected because DDoS attacks are originated from thousands of different sources, leading to an increase in unique source IP addresses during an attack.

The same holds for port scans. Fig. 6 shows the feature importance evaluation for the MAWI/ID2T data set and the time frame $\delta = 0.5s$. Results indicate that *for port scan detection unique source and destination port counts are the most important features*, which are also not observed by micro-flow-based NIDS. This results is also expected, as port scans address many different destination ports that are not necessarily addressed by the benign traffic. Further, port

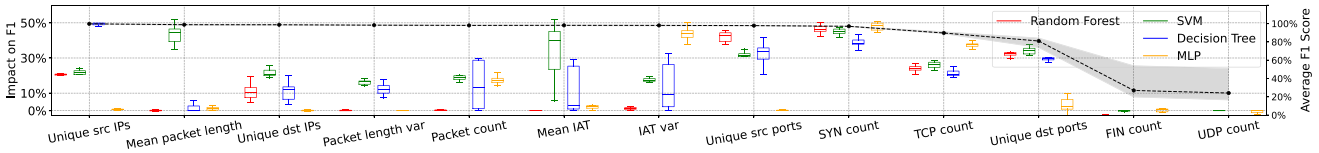


Fig. 7. Feature importance ranking for *DDoS detection* on MAWI/CAIDA data with time frame $\delta = 0.5s$.

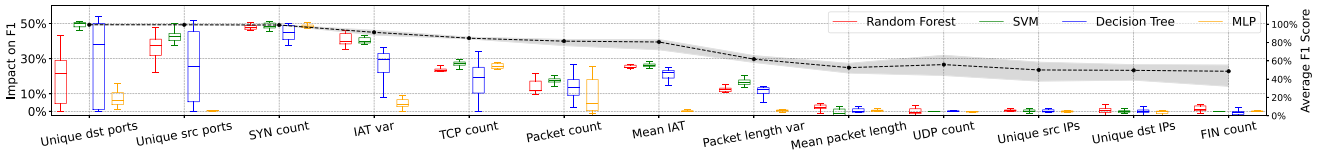


Fig. 8. Feature importance ranking for *port scan detection* on MAWI/ID2T data with time frame $\delta = 0.5s$.

scans are potentially conducted from randomized source ports (as also in the MAWI/ID2T data set), which explains the importance of the unique source port count.

In contrast to micro-flow-based monitoring, eMinD can obtain source IP address and destination port distributions, covering the most important characteristics of DDoS attacks and port scans with its monitoring.

VIII. Feature Ranking

As outlined in the previous section, the feature importance evaluation shows that hash features are the most important features for DDoS attack and port scan detection. However, the employed feature importance metric (permutation feature importance) only indicates those features that are utilized for the classification by the ML model. It is possible that the ML model only uses a small subset of all features for the classification, because this small subset is already sufficient for a qualified decision. On the other hand, this results in a subset of remaining features that are not used by the ML model and therefore show feature importance of zero percent, but might still be sufficient for a qualified decision (*starvation*). To determine the importance of individual features and decide whether an individual feature is unimportant or only suffered from starvation, we employ a *feature ranking*.

The feature ranking approach utilizes an *active feature set* that contains all 13 features at the beginning. The ranking iteratively calculates the *permutation feature importance* for all features in the active feature set, as well as the models' overall performance on the test data (e.g., accuracy) utilizing the active feature set. The feature with the highest importance score is removed from the active feature set and the next iteration starts. This is repeated until the active feature set is empty, resulting in a ranking of all features according to their importance for the ML model to perform a qualified decision.

Fig. 7 and Fig. 8 show the feature importance ranking for DDoS detection on MAWI/CAIDA data and port scan detection on MAWI/ID2T data, respectively. The box plots show the distribution of the feature importance scores for 100 test runs with the corresponding scale on the left y-axis. The dashed black line indicates the average classification performance (F1-score) of the ML models on the test data with the corresponding scale on the right y-axis. The filled gray area around the dashed black line illustrates the range between the quartiles of the classification performance. The x-axis shows the features in the order of their importance for the ML model. The most important feature is located on the left, the least important feature on the right. The active feature set is inclusive from right to left, i.e., the most important feature is removed first, the least important feature is removed last.

A. DDoS Attacks

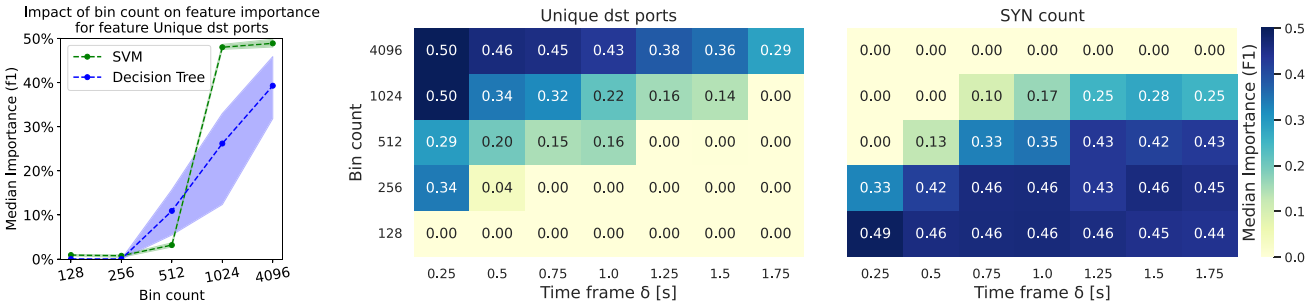
Fig. 7 shows the feature importance ranking for DDoS detection on MAWI/CAIDA data with time frame $\delta = 0.5s$. The most important feature is the source IP address hash feature, followed by packet length mean and the destination IP address hash feature, which was already observed in the feature importance evaluation (see Fig. 5).

In contrast to the feature importance that indicated an importance of zero percent for most of the features, the feature ranking now shows that there are a lot of features that suffer from starvation, as they show an importance larger than zero percent in the feature ranking, while still achieving an average classification performance of 100 percent on the test data with the reduced active feature sets. The average classification performance of the ML models is still 99 percent for the active feature set with 5 remaining features (see *SYN count*). The classification performance first drops below 90 percent when the active feature set is reduced to the four least important features, i.e., *TCP count*, *Unique dst ports*, *FIN count*, and *UDP count*.

In essence, many features are not used by the ML model for the classification, but are still sufficient for a qualified DDoS detection. The reason for this is that the attack volume of the DDoS attack in the MAWI/CAIDA data set is so high that the impact of the attack traffic on most count and sum features, compared to the impact of the background traffic, is so high that the ML model can already make a qualified decision based on these features. However, if there is a feature available in the active feature set that is even more important for the ML model to make a qualified decision, the ML model will focus on this feature instead causing the starvation of other features.

B. Port Scans

Fig. 8 shows the feature importance ranking for port scan detection on MAWI/ID2T data with time frame $\delta = 0.5s$. The three most important features are the destination port



(a) Impact of bin count ($\delta = 1s$) (b) Heatmaps illustrating the interdependence between hash bins, time frames and feature importance

Fig. 9. Hash feature importance evaluation for port scan detection, different numbers of hash bins and time frames.

hash feature, the source port hash feature and the SYN count feature. When comparing the feature importance ranking to the feature importance evaluation (see Fig. 6), the feature importance ranking shows that the SYN count feature suffers from starvation, as it shows an importance of zero percent in the feature importance evaluation, but is still the third most important feature in the feature importance ranking. Furthermore, the SYN count feature is sufficient for a perfect port scan detection, as the average classification performance of the ML models is still 100 percent for the active feature set with 11 remaining features (see *SYN count*). Removing the SYN count feature from the active feature set results in a drop of the average classification performance.

Compared to the feature ranking for DDoS detection, the feature ranking for port scan detection shows that there are less features sufficient for a qualified detection. The reason for this is that port scans are conducted with a lower attack volume (1000 packets per second) compared to the CAIDA DDoS attack (130k packets per second), resulting in a lower impact of the attack traffic on the count and sum features.

IX. Hash Feature Importance

The previous sections showed that hash features are the most important features regarding DDoS attack and port scan detection. They provide an *estimation* about the number of unique IP addresses and ports in the network traffic, which constitutes a significant indicator for *distributed attacks*. However, the quality of this estimation strongly depends on the number of hash bins. The more hash bins are used, the more accurate the estimation becomes. The less hash bins are used, the more hash collisions occur, which leads to an underestimation of the number of unique IP addresses and ports on the one hand, and to an indistinguishability between benign and attack traffic on the other hand.

This section covers the evaluation of the *hash feature importance* in dependence of the number of hash bins and the time frame δ.

A. Impact of the Number of Hash Bins

First, we evaluate the impact of the number of hash bins on the feature importance with the *unique destination ports* feature for port scan detection on the MAWI/ID2T data set. We assume that the importance of this hash feature is directly correlated to the number of hash bins: The more hash bins are used, the more accurate the estimation of the number of unique destination ports becomes, and therefore the port scan detection becomes more accurate. If the number of hash bins is too low, the importance of the feature should be also low, because attack and benign traffic cannot be distinguished.

Fig. 9(a) illustrates the feature importance (F1-score) for the destination ports hash feature for different numbers of hash bins, i.e., 128, 256, 512, 1024, and 4096. The time frame δ is 1.0 seconds. It shows that the previous assumption is correct: The feature importance increases with the number of hash bins and is zero if the number of hash bins is too low to distinguish between benign and attack traffic.

B. Interdependence of Hash Bin Count and Time Frames

Second, we evaluate the interdependence between the number of hash bins, the time frame δ and the hash feature importance on the MAWI/ID2T data set. Fig. 9(b) illustrates the feature importance (F1-score) for the destination ports hash feature and the SYN count feature for different numbers of hash bins and time frames.

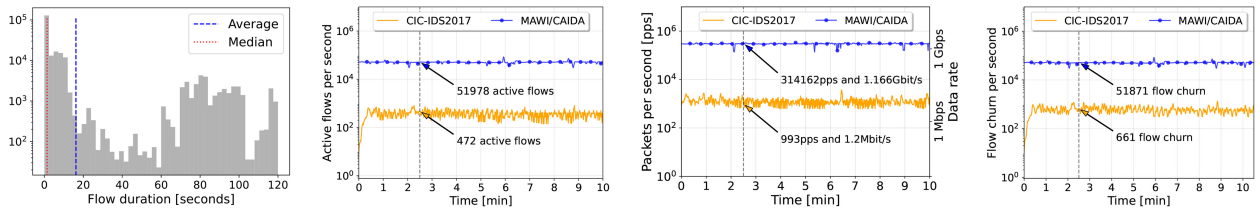
Results show that the hash feature's importance increases with an increasing number of hash bins for all time frames. However, the hash feature's importance decreases with increasing time frames for all numbers of hash bins, as longer time frames lead to more aggregated traffic and, therefore, to more hash collisions, obliberating attack and benign traffic.

The importance of the SYN count feature behaves vice versa: It decreases with an increasing number of hash bins for all time frames, but increases with increasing time frames for all numbers of hash bins. As the SYN count feature is independent of the number of hash bins, this behavior is unexpected. When comparing both heat maps in Fig. 9(b), it seems that the ML model utilizes the SYN count feature to compensate the destination port hash feature's decreasing importance with increasing time frames.

X. Scalability

A. Short Reaction Time

We compare eMinD's reaction time with micro-flow-based approaches that classify complete micro-flows, which is done by related work using the CIC-IDS2017 data set. Therefore,



(a) Flow duration histogram (b) Active micro-flow count (c) Arriving packet count (d) Micro-flow churn

Fig. 10. Traffic statistics from 10 minute sections of CIC-IDS2017 and MAWI/CAIDA data sets during the DDoS attack.

the micro-flow completion duration represents a lower boundary for the reaction time of related work, making it a suitable metric for comparison with eMinD's time frames.

Fig. 10(a) presents the micro-flow duration histogram of all micro-flows in the CIC-IDS2017 data set during the DDoS attack (*Friday afternoon DDoS*) with a logarithmically scaled y-axis. The histogram includes vertical lines, indicating the median (1.6 seconds) and the average (16.4 seconds) micro-flow completion duration.

With the time frame $\delta = 1.5s$, eMinD outperforms micro-flow-based approaches for half of the active micro-flows and even *reduces the average reaction time by 90%* $= 1 - \frac{1.5s}{16.4s}$. In addition, eMinD periodically classifies traffic (for every time frame), making the reaction time predictable in contrast to micro-flow-based approaches that rely on micro-flow completion, which can last two minutes in the CIC-IDS2017 scenario (compare Fig. 10(a)) depending on individual micro-flows.

B. Strictly Limited Classification Frequency

eMinD performs only one classification with the ML model per time frame, while micro-flow-based NIDS require one classification per micro-flow. Therefore, the amount of active micro-flows represents the worst-case number of classifications required for attack detection for a micro-flow-based NIDS. Fig. 10(b) shows the amount of active unidirectional micro-flows for both CIC-IDS2017 and MAWI/CAIDA data sets during a 10-minute interval of the DDoS attack. During the CIC-IDS2017 DDoS attack, there are about 500 active micro-flows on average per second, compared to about 50k in the MAWI/CAIDA scenario.

Even in the small testbed setup of the CIC-IDS2017, eMinD reduces the average amount of classifications from 500 to one classification per second, *saving 99.8% of all classifications*. In the context of the MAWI/CAIDA backbone scenario with high data rates and large amounts of micro-flows, tens of thousands of classifications are saved per second, showing both the need for micro-flow independent NIDS and the scalability of eMinD regarding the number of ML classifications.

C. Predictable and Strictly Limited State Space

We compare the required state space of eMinD to micro-flow-based approaches. State space is the memory necessary to store traffic features fed into the ML model and used for classification. eMinD monitors only a *fixed set of 13 features* (compare Tab. I) and requires only a fixed and predictable amount of memory. In contrast, micro-flow-based approaches monitor one set of features per active micro-flow, resulting in varying memory needs depending on the traffic.

Fig. 10(b) presents the number of active micro-flows per second. Micro-flow-based approaches monitor up to 84 features [10], [25] per micro-flow (500 per second), resulting in a total amount of $42k$ individual features per second $(84\frac{\text{features}}{\text{micro-flow}} \cdot 500\frac{\text{micro-flows}}{\text{s}} = 42k\frac{\text{features}}{\text{s}})$ to be monitored, calculated and processed, e.g., normalized, in the CIC-IDS2017 scenario. Therefore, eMinD *reduces the required state space to 0.03%* $= \frac{13\text{features}}{42k\text{features}}$ of the state space required by a micro-flow-based NIDS. This impact is even more significant in the MAWI/CAIDA backbone scenario, as it contains a lot more active micro-flows. Furthermore, eMinD's monitoring provides *predictability and scalability*, as it does not depend on the number of active micro-flows or the arriving traffic volume.

D. No Micro-Flow State Management

Micro-flow state management consists of two parts: First, matching arriving packets to their corresponding micro-flow to update monitored traffic features. Second, adding new micro-flows to monitoring while removing already completed flows from monitoring to save state space.

Packet-to-micro-flow matching is performed for every arriving packet. Fig. 10(c) shows the number of arriving packets per second. In the context of the CIC-IDS2017 DDoS attack, about 1400 packets need to be matched every second, while in the MAWI/CAIDA backbone scenario, about 314k packets need to be matched every second. *eMinD does not perform five-tuple matching at all*, providing high scalability and saving computational effort.

We denote the sum of appeared micro-flows, i.e., not existing in the previous time frame, and vanished micro-flows, i.e., not existing in the current time frame anymore, as the *micro-flow churn*. Fig. 10(d) presents the churn of micro-flows per second. In contrast to micro-flow-based NIDS, *eMinD completely avoids micro-flow state management* and therefore saves all computational effort for checking appeared or vanished micro-flows, providing scalability, and facing increasing data rates and micro-flow amounts.

E. Discussion and Limitations

eMinD achieves better detection performance than micro-flow-based NIDS, while being very efficient in processing large traffic volumes due to traffic aggregation. However, traffic aggregation has its price, as it goes along with an

information loss, i.e., there are fewer available traffic features and they are not segregated according to micro-flows.

If a micro-flow-based NIDS detects malicious micro-flows, it can derive appropriate network filter rules from the corresponding five-tuple. With eMinD, the only result is whether an attack is ongoing. To mitigate attacks with eMinD, traffic monitoring, and feature calculation must be adapted to derive appropriate information for filter rules. One possibility to preserve mitigation information in eMinD is using prefix-preserving hash features, such as [26], to derive IP-based filter rules.

Nevertheless, *micro-flow independent traffic monitoring* is an essential step towards developing sustainable NIDS facing increasing traffic volumes and micro-flow counts in Networks of the Future.

XI. Conclusion

eMinD is a micro-flow independent network intrusion detection system leveraging supervised machine learning. We compare *eMinD* to related work through evaluation with real-world data sets from the CIC, MAWI, and CAIDA.

eMinD provides scalability regarding the active micro-flow count and the ingress data rate by highly aggregating arriving network traffic, reducing required state space by 99.97% compared to micro-flow-based approaches. Furthermore, the required state space is fixed and a priori known. Simple aggregation operations, such as addition, allow *eMinD* to monitor traffic fast, avoiding discarding traffic, e.g., through sampling. *eMinD* periodically provides attack detection results on very short time scales and reduces the average reaction time by 90%. We further show that micro-flow independent traffic monitoring even improves the detection performance of distributed attacks, namely DDoS attacks and port scans, and we prove the importance of micro-flow-overarching features for their detection, e.g., IP address and port distributions.

References

[1] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, 1st Quart., 2019.

[2] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, p. 4396, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4396

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[4] R. Hofstede et al., "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2037–2064, 4th Quart., 2014.

[5] A. D. Jasim, "A survey of intrusion detection using deep learning in Internet of Things," *Iraqi J. Comput. Sci. Math.*, vol. 3, no. 1, pp. 83–93, 2022.

[6] S. Kopmann and M. Zitterbart, "eMinD: Efficient and micro-flow independent detection of distributed network attacks," in *Proc. 14th Int. Conf. Netw. Future (NoF)*, 2023, pp. 159–167.

[7] G. Lloret-Talavera et al., "Enabling homomorphically encrypted inference for large DNN models," 2021, *arXiv:2103.16139*.

[8] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1333–1345, 2018.

[9] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.

[10] P. Choobdar, M. Naderan, and M. Naderan, "Detection and multi-class classification of intrusion in software defined networks using stacked auto-encoders and CICIDS2017 dataset," *Wireless Pers. Commun.*, vol. 123, no. 1, pp. 437–471, 2022.

[11] "MAWI backbone data," Mar. 1, 2023. [Online]. Available: https://mawi.wide.ad.jp/mawi/samplepoint-F/2023/202303011400.html

[12] "MAWI backbone data," Mar. 2, 2023. [Online]. Available: https://mawi.wide.ad.jp/mawi/samplepoint-F/2023/202303021400.html

[13] "MAWI backbone data," Mar. 3, 2023. [Online]. Available: https://mawi.wide.ad.jp/mawi/samplepoint-F/2023/202303031400.html

[14] "MAWI backbone data," Mar. 4, 2023. [Online]. Available: https://mawi.wide.ad.jp/mawi/samplepoint-F/2023/202303041400.html

[15] 2010, "The CAIDA UCSD 'DDoS attack 2007' dataset," Caida. [Online]. Available: https://www.caida.org/catalog/datasets/ddos-20070804_dataset

[16] C. G. Cordero, E. Vasilomanolakis, N. Milanov, C. Koch, D. Hausheer, and M. Mühlhäuser, "ID2T: A DIY dataset creation toolkit for intrusion detection systems," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2015, pp. 739–740.

[17] K. Gosai, H. Mehta, and V. Katkar, "An intrusion detection using ensemble classifiers," in *Proc. 6th Int. Conf. Comput. Methodol. Commun. (ICCMC)*, 2022, pp. 163–168.

[18] B. Deore, A. Kyatham, and S. Narkhede, "A novel approach to ensemble MLP and random forest for network security," in *Proc. ITM Web Conf.*, 2020, p. 3003. [Online]. Available: https://doi.org/10.1051/itmconf/20203203003

[19] A. A. Sallam, M. N. Kabir, Y. M. Alginahi, A. Jamal, and T. K. Esmeel, "IDS for improving DDoS attack recognition based on attack profiles and network traffic features," in *Proc. 16th IEEE Int. Colloq. Signal Process. Appl. (CSPA)*, 2020, pp. 255–260.

[20] S. Wankhede and D. Kshirsagar, "DoS attack detection using machine learning and neural network," in *Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, 2018, pp. 1–5.

[21] L. Yu et al., "PBCNN: Packet bytes-based convolutional neural network for network intrusion detection," *Comput. Netw.*, vol. 194, Jul. 2021, Art. no. 108117. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001948

[22] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. LISA 13th Syst. Admin. Conf.*, 1999, pp. 229–238. [Online]. Available: http://dblp.uni-trier.de/db/conf/lisa/lisa1999.html#Roesch99

[23] M. Ramakrishna, E. Fu, and E. Bahcekapili, "Efficient hardware hashing functions for high performance computers," *IEEE Trans. Comput.*, vol. 46, no. 12, pp. 1378–1381, Dec. 1997.

[24] "eMinD GitHub repository." [Online]. Available: https://github.com/samkopmann/eMinD

[25] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, pp. 479–482, Jan. 2018.

[26] S. Kopmann and M. Zitterbart, "Toward joining DDoS mitigation and image segmentation," *Datenschutz Datensich-DuD*, vol. 47, pp. 475–477, Aug. 2023.