Pegah Golchin, Chengbo Zhou, Hengyu Liu, Björn Scheuermann, Ralf Kundel, Tobias Meuser. *Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN*. To appear in Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2024.

Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN

Pegah Golchin *†, Chengbo Zhou *†, Hengyu Liu *,
Björn Scheuermann *, Ralf Kundel *, Tobias Meuser *
*Technical University of Darmstadt, Germany
†These authors contributed equally to this work.
{pegah.golchin, chengbo.zhou, bjoern.scheuermann, ralf.kundel, tobias.meuser}@tu-darmstadt.de
hengyu.liu@stud.tu-darmstadt.de

***Abstract*—Software-Defined Networking (SDN) improves network management and flexibility by separating control and data plane functions. However, the centralized architecture of SDN can increase cybersecurity risks, such as an increased vulnerability to Denial of Service (DoS) attacks. While integrating machine learning (ML) models into Intrusion Detection Systems (IDSs) achieves high detection performance, these ML models must demonstrate strong generalization capabilities across new, previously unseen network traffic patterns, which is crucial for networks with dynamic traffic behavior. In our previously published work, Collaborative ML-based IDS (CML-IDS), different ML models are deployed in both the control and data plane to enhance detection performance while reducing network load and detection time. However, CML-IDS operates as an offline model, where ML models are trained once on a specific network traffic pattern, potentially limiting CML-IDS ability to generalize across diverse and new network traffic patterns effectively. To address this issue, we introduce *COML-IDS*, an online learning framework that automatically updates the ML model in the data plane when the detection performance degrades. Our results demonstrate that *COML-IDS* achieves an average increase of at least 25% in detection performance when encountering new network traffic patterns while reducing the need to forward the necessary flow feature data to the control plane compared to the CML-IDS.**

***Index Terms*—Intrusion Detection Systems, Machine Learning, Software-Defined Networking**

I. Introduction

Software-Defined Networking (SDN) is a network architecture paradigm that separates the control plane from the data plane to enhance network management and provide a comprehensive view of network traffic behavior. However, SDN is susceptible to cyberattacks such as DoS and DDoS attacks, especially when the control plane operates in a reactive configuration [1]. These attacks can target either the control plane, overwhelming it with a large number of new packets in a short period, or the data plane, saturating it with numerous new flow rules. Consequently, an accurate Intrusion Detection System (IDS) is essential for detecting these threats.

IDSs can be categorized into two main types: signature-based and anomaly-based IDS [2]. Signature-based approaches detect known intrusions whose signatures are available in databases, whereas anomaly-based approaches can learn statistical traffic features to detect previously unseen intrusions.

The machine learning (ML)-based IDS is a suitable candidate for the anomaly-based approaches, which can be deployed in the SDN control plane. However, this deployment increases the detection delay and traffic load, potentially overloading the controller. While deploying lightweight ML models in programmable switches can accelerate network attack detection, it may decrease detection performance due to the limited computational resources available in these switches. Our recent work CML-IDS [3] introduced a collaborative framework that balances the strengths of ML-based IDS in both planes. The lightweight ML model in the data plane (DP-IDS) provides a preliminary traffic classification, whereas an ensemble ML model with higher learning capacity is deployed in the control plane (CP-IDS) that targets the classification of the traffic classified by DP-IDS with low confidence.

However, the DP-IDS is pre-trained once and installed in the programmable switch, which can reduce its model confidence for detecting new traffic patterns. This limits generalization performance, which refers to the ML model's ability to accurately classify new network traffic patterns, regardless of whether they share the same distribution as the training data. This capability is crucial because network traffic behavior is dynamic and can change due to concept drifts like modifications in network architecture and management rules [4], [5]. Additionally, the emergence of new types of network intrusions (i.e., zero-day attacks) can present patterns that differ from those in the training data. Therefore, it is essential to continuously adapt the deployed DP-IDS to maintain its effectiveness in detecting both known and unknown attacks.

In this work, we integrate an online learning approach into CML-IDS, naming it *COML-IDS*. The principal objective is to improve the generalization of the CML-IDS, making it adaptable to new network traffic patterns while maintaining high detection performance. Our designed framework addresses the key questions: 1) What conditions should trigger the retraining of an ML-based IDS? 2) How can we mitigate catastrophic forgetting in online learning, where the model prioritizes new

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Pegah Golchin, Chengbo Zhou, Hengyu Liu, Björn Scheuermann, Ralf Kundel, Tobias Meuser. *Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN*. To appear in Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2024.

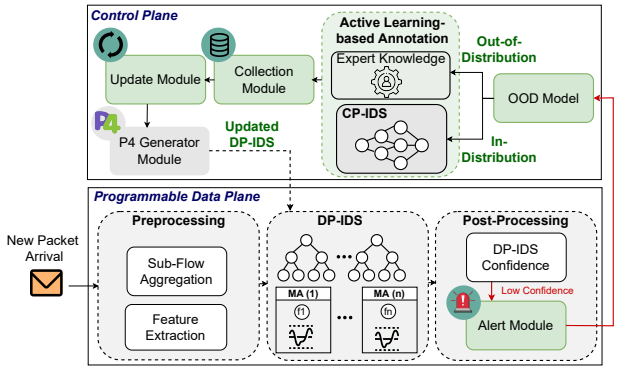


Fig. 1: *COML-IDS* architecture. Various modules are designed to continually adapt the DP-IDS. The contributions of *COML-IDS* are highlighted in green boxes.

patterns at the expense of previously learned ones? 3) How should new network traffic patterns be annotated?

As shown in Figure 1, the DP-IDS classification results are used to trigger the Alert module when they meet the retraining conditions. To prevent catastrophic forgetting, new network traffic patterns are merged with the baseline traffic dataset within the Collection module. An active learning approach is employed to annotate new, previously unseen patterns. Using an Out-of-Distribution (OOD) model, these patterns are classified as either in-distribution or out-of-distribution. If a pattern is in-distribution, CP-IDS is used for annotation; otherwise, expert knowledge is required. The effectiveness of *COML-IDS* is evaluated using three different publicly available network traffic datasets, showing that *COML-IDS* increases detection performance and achieves better generalization than CML-IDS.

The paper is structured as follows: Section II explains related works. The *COML-IDS* architecture is presented in Section III. In Section IV, evaluation results are presented and discussed. Finally, Section V provides a short summary.

II. Related Work

This section provides an overview of deploying ML-based IDS within the SDN architecture and explores the integration of online learning approaches into IDSs.

A. Deploying the ML-based IDS in SDN

Many researchers have deployed various ML models in the SDN control plane to classify network flows due to its sufficient computational resources [6]–[8] Although these models can achieve high detection performance, their detection speed does not match the line rate. Additionally, transmitting flow features to the control plane results in more traffic, which can overwhelm the controller.

To increase detection speed and prevent overwhelming the control plane, researchers have investigated deploying ML models in the programmable data plane [9]–[12]. Given the limited computational resources in P4 switches, some approaches use the P4 switch solely as a flow feature extractor, where the ML-based IDS is deployed on a separate server [9], [13]. Others utilize lightweight ML models structured similarly to the match-action pipeline of the P4 switch [10]–[12]. However, lightweight ML models often have lower detection performance compared to more complex models. To address this issue, CML-IDS [3] combines a lightweight model in the DP and an ensemble ML model in the CP based on the confidence level of the DP model. While CML-IDS balances detection performance, flow transmission, and detection speed, it is trained only once on a specific network traffic pattern. This limitation reduces its adaptability to new, previously unseen network traffic patterns.

B. Online ML-based IDS

To make the ML-based IDS adaptable to dynamic network architectures and concept drift, one effective approach is to use online learning [4], which involves retraining the ML model to continuously adapt to new network conditions. Authors in [14] and [15] proposed an in-network ML model, which was updated regularly.However, updating the model without any conditions can increase network latency and packet transmission. In [16], the authors proposed adaptive flow measurement rules using weighted linear prediction and a strategy from multi-armed bandit problems. However, their focus was not on network intrusion detection.

In our work, we extend the CML-IDS approach by updating the DP-IDS based on specific criteria. These criteria include detecting out-of-distribution flow patterns and monitoring the number of flows classified by the DP-IDS with low confidence. This improves system adaptability and detection performance when encountering dynamic network behavior.

III. COML-IDS Design

In this study, we introduce *COML-IDS*, integrating online learning to augment the generalization capability of CML-IDS [3]. As depicted in Figure 1, *COML-IDS* comprises several *"Baseline Modules"* inherited from CML-IDS, along with new modules that enable online functionality, which are explained in the following sections.

A. Baseline Modules in the Programmable Data Plane

1) ***Preprocessing Module****:* Upon the arrival of a packet, a flow identifier is computed using the CRC32 hash function applied to the packet 5-tuple values (source and destination IP addresses, port numbers, and transport protocol). To avoid detection delays caused by waiting for complete flows, the concept of *sub-flow* is utilized, where the features of the first N packets of a flow are collected and fed to the DP-IDS for traffic classification. Based on the previous findings [12] [3], $N = 8$ achieves the optimal balance between informativeness and detection speed. The subsequent packets of the classified sub-flows are handled according to the classification result (e.g., *forward* if the flow is classified as Benign; *drop* if it is classified as Attack). A total of 59 features are extracted from sub-flows, referred to *complete feature set*. From these, 20 features are identified as the most relevant to the classification using an ensemble feature importance algorithm [6]. These

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Pegah Golchin, Chengbo Zhou, Hengyu Liu, Björn Scheuermann, Ralf Kundel, Tobias Meuser. *Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN*. To appear in Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2024.

selected features constitute the *reduced feature set*. These features are thoroughly explained in [3].

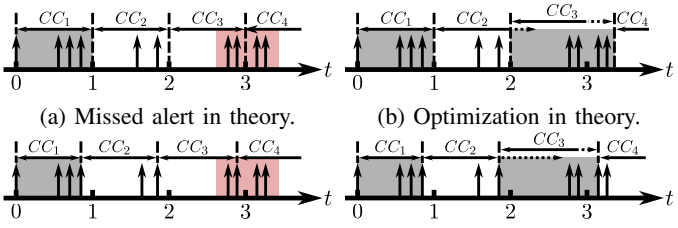
2) ***DP-IDS Model****:* The ML model for DP-IDS must be tailored to suit the programmable switch architecture. Tree-based ML models are suitable for deployment in DP-IDS because their structure shares similarities with the match-action pipeline of switches and does not require complex mathematical computations. In this paper, DP-IDS is implemented using a lightweight Random Forest (RF) comprising three Decision Trees (DTs) with five layers for prototyping. The number of DTs and layers are determined through cross-validation, considering the computational resource constraints of the switch. The RF model is trained with the reduced feature set (§III-A1) to balance computational cost and detection performance. Note that any feasible model for the programmable switch can be used for DP-IDS in *COML-IDS*.

3) ***Post-Processing Module****:* This module accepts the DP-IDS output (Benign or Attack) along with the associated model confidence (MC), which reflects the certainty of the detection result. MC is computed within the programmable switch by averaging the Gini impurities over the reached leaves of the three DTs, resulting in MC values ranging from 0 to 0.5. A lower MC indicates higher confidence in the detection. Subsequently, the MC value is compared to a predefined threshold (MC_{thr}) to determine whether to trust the DP-IDS detection result ($MC \leq MC_{thr}$) or trigger CP-IDS for further processing ($MC > MC_{thr}$). The latter requires that the switch generates a packet (denoted by P_{flow}) carrying the extracted sub-flow features and sends it to CP-IDS. MC_{thr} serves as a factor to adjust the sensitivity for trusting the DP-IDS result. Higher MC_{thr} introduces a less stringent threshold for trust, resulting in more sub-flows classified by DP-IDS and, therefore, less traffic to the control plane. In contrast, lower MC_{thr} leads to relatively higher overall detection performance, as more sub-flows are classified by CP-IDS.

B. Baseline Modules in the Control Plane

1) ***CP-IDS Model****:* CP-IDS is an ensemble ML model comprising an RF, XGBoost, and a Multi-Layer Perceptron (MLP). The final detection result is derived by averaging the prediction outputs of these individual models.

2) ***P4 Generator Module****:* This module is designed to automatically convert the trained RF model for DP-IDS into P4 code and match-action table entries.



(a) Missed alert in theory. (b) Optimization in theory.

(c) Missed alert in switch. (d) Optimization in switch.

Fig. 2: Alert misses and optimization. Upward-pointing arrows: P_{flow} indexed from 1 to 10 along the time axis. CC_n: n-th counting cycle. Gray/Red box: triggered/missed alert.

C. Model Updating Pipeline in COML-IDS

To ensure the ML-based IDS can accurately detect various network traffic patterns resulting from the dynamic behavior of network traffic, the ML model should be updated whenever the classification performance degrades. In *COML-IDS*, the performance degradation is reflected in advance by the increase of the sub-flows classified by DP-IDS with lower confidence. That is equivalent to the increase of the P_{flow} (§III-A3) generated within the switch.

1) ***Alert Module****:* An alert is triggered when the DP-IDS performance degradation occurs, defined by the number of P_{flow} exceeding a specific threshold (Thr) over a time window (Tw). Following each alert, the subsequent modules (§III-C3, §III-C4) are activated for collecting data to construct the training dataset used for retraining DP-IDS. After each successful update, a new counting cycle is initialized to monitor the next potential performance degradation.

Challenge 1: Theoretically, an alert should be triggered whenever the condition is fulfilled using a sliding window. However, implementing a sliding window for online packet counting requires enormous counting cycles depending on the granularity of time measurement and the window size. An intuitive solution is to start counting cycles periodically after completing the previous one. Unfortunately, this solution could lead to alert misses due to the cycle's reset if a significant number of P_{flow} are sent to CP-IDS near the end of a counting cycle. An example is illustrated in Figure 2a. Ten P_{flow} are observed within four counting cycles (CCs). Tw and Thr are set to 1 and 4, respectively. Assume CC_1 starts at time 0 and there is no P_{flow} after the last P_{flow}. We can observe that an alert is triggered within CC_1 since the first four P_{flow} are detected within a single counting cycle. The last four P_{flow} occur within a Tw but spread over CC_3 and CC_4. That leads to a missed alert for them.

Solution: To tackle this challenge, the alert module adopts a two-phase strategy within each counting cycle: the *normal phase*, which is identical to Tw explained above, and the *tolerance phase*, which allows Tw to be exceeded with a specific amount of time (denoted by Tp) while counting the number of P_{flow}. Figure 2b depicts the optimized alert trigger leveraging this two-phase strategy. The tolerance phase is marked with dotted lines. Note that a new counting cycle is initialized after Tw instead of $(Tw + Tp)$ if no alert is triggered in the previous counting cycle (e.g., CC_3 starts at time 2 instead of $(2 + Tp)$). We observe that the last two P_{flow} are counted in the Tp of CC_3, which reaches the Thr within $(Tw + Tp)$, and thus the alert for the last four P_{flow} is successfully triggered. CC_4 starts at the timestamp $(2 + Tw + Tp)$ which is the end of Tp in CC_3, as an alert in the previous counting cycle CC_3 is triggered.

Challenge 2: Initializing the normal and tolerance phases requires retrieving the current clock of the programmable switch at any time. However, the packet processing logic is only triggered when a packet arrives at the switch. This implies that the clock value is only known upon the arrival of a packet.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Pegah Golchin, Chengbo Zhou, Hengyu Liu, Björn Scheuermann, Ralf Kundel, Tobias Meuser. *Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN*. To appear in Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2024.

Solution: Recall that the packet P_{flow} is generated when a sub-flow is classified by DP-IDS with lower confidence. In other words, P_{flow} is created and sent to CP-IDS when the 8th packet of a low MC sub-flow enters the programmable switch and triggers the classification process executed in the ingress pipeline. Therefore, the creation time of P_{flow} can be approximated by the value of the metadata *ingress_timestamp* of the 8th packet of a sub-flow. Figure 2c illustrates the amended start and end of counting cycles for the same traffic pattern without the two-phase strategy. In contrast to the theoretical case in Figure 2a, the duration of a counting cycle becomes elastic instead of being equal to Tw. It is compressed when an alert is triggered (e.g., CC_1), as Thr is reached before the end of Tw, and stretched when no alert is present (e.g., CC_2 and CC_3), since Tw is already exceeded when the last P_{flow} is generated. The counting cycle lasts Tw only when the last P_{flow} is created exactly at the end of the cycle, which happens extremely rarely with nanosecond granularity.

Note that the 4th P_{flow} initializes CC_2 due to the alert trigger, whereas CC_3 and CC_4 are initialized due to the end of the previous counting cycles. Therefore, the 6th and 8th P_{flow} are counted in CC_3 and CC_4, respectively, as they are the first seen P_{flow} in these two cycles. Similarly, the alert for the last four P_{flow} is also missed, the same as the theoretical case. Figure 2d demonstrates the correct triggering by applying the two-phase strategy in the programmable switch. CC_1 starts at timestamp 0, at which time the first P_{flow} is created. An alert is triggered before the end of Tw. Hence, the tolerance phase Tp is not needed for CC_1, and CC_2 is directly initialized subsequently. At the creation time of the 6th P_{flow}, the elapsed time in CC_2 is between Tw and $(Tw+Tp)$. That indicates the tolerance phase is not complete, and CC_3 may be initialized potentially if Thr is not reached in the remaining time of CC_2. Therefore, the switch generates a virtual counting cycle for CC_3 and starts counting the number of P_{flow}. The end of CC_2 is determined at the creation time of the 7th P_{flow}, at which point the elapsed time is larger than $(Tw + Tp)$, and only two P_{flow} are captured. Therefore, no alert in CC_2 is triggered, and the virtual CC_3 becomes a real counting cycle with the state: two P_{flow} are captured, and the elapsed time is equal to the interval between the 6th and 7th P_{flow}. An alert is triggered within CC_3 when the 9th P_{flow} is created in the tolerance phase of CC_3. The alert-triggering procedure with the two-phase strategy for the programmable switch is formulated in Algorithm 1. Note that n_{vir} is initialized with 1, as entering the tolerance phase is triggered by a P_{flow}.

Algorithm 1 Alert-triggering procedure.

Input: 8th packet of a low MC sub-flow f.

Generate a P_{flow}
$Cr_f \leftarrow$ *ingress_timestamp*
$t \leftarrow Cr_f - CC.st$ ▷ $CC.st$: start time of a CC
if $t \leq Tw$ **then**
  $n \leftarrow n + 1$
  if $n \geq Thr$ **then**
    Trigger an alert
    $CC.st \leftarrow Cr_f$ ▷ new CC
    $t \leftarrow 0$; $n \leftarrow 0$; $n_{vir} \leftarrow 1$
  else
    $CC_{vir}.st \leftarrow Cr_f$ ▷ CC_{vir}: virtual CC
  end if
else if $t \leq (Tw + Tp)$ **then** ▷ tolerance phase
  $n \leftarrow n + 1$
  $n_{vir} \leftarrow n_{vir} + 1$
  if $n \geq Thr$ **then**
    Trigger an alert
    $CC.st \leftarrow Cr_f$ ▷ new CC
    $t \leftarrow 0$; $n \leftarrow 0$; $n_{vir} \leftarrow 1$
  end if
else
  $CC.st \leftarrow CC_{vir}.st$ ▷ virtual CC to real CC
  $n \leftarrow n_{vir} + 1$; $n_{vir} \leftarrow 1$;
end if

*2) **Annotation Module**:* Once P_{flow} carrying sub-flow features are forwarded to the control plane for updating DP-IDS, their labels (Attack/Benign) need to be identified to construct the training dataset that is used to retrain the DP-IDS. We consider two scenarios in this work:

Expert Knowledge: In this scenario, the ground-truth label of the sub-flow is used, which means each sub-flow is correctly labeled based on human knowledge. However, it is impractical as human intervention for labeling all sub-flows is required.

Active Learning: To make the system more practical and efficient, we implement an active learning approach [17]. Here, human knowledge is only used for sub-flows that exhibit patterns different from those on which the CP-IDS was originally trained. To assess the distinctiveness of a sub-flow's pattern, we implement an Isolation Forest as an Out-of-Distribution (OOD) model. This method partitions data points randomly and constructs decision trees, effectively isolating anomalies by requiring fewer splits compared to normal data. The OOD model is trained with the same network traffic patterns for training CP-IDS and detects whether a sub-flow is OOD or in-distribution (ID) relative to the training data. Human experts provide ground-truth labels only for the OOD sub-flows, which are uncertain data points. For ID sub-flows, the labels are determined using the output of the CP-IDS. Compared to the *Expert Knowledge* scenario, the need for human intervention is reduced in this scenario, though it may increase noise within training data by annotating the sub-flow incorrectly.

*3) **Collection Module**:* In this module, the annotated sub-flow entries are collected as new network traffic datasets. The training dataset for retraining DP-IDS is built with three segments: 1) the sampled base dataset used for training the original DP-IDS, 2) the sub-flow entries that trigger the alert due to a sudden change in traffic behavior, and 3) a certain amount of sub-flows collected after the alert, as these sub-flows reflect the changes in the traffic pattern and thus provide up-to-date data for updating DP-IDS.

*4) **Update Module**:* This module is responsible for retraining the deployed RF model using the network traffic dataset

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Pegah Golchin, Chengbo Zhou, Hengyu Liu, Björn Scheuermann, Ralf Kundel, Tobias Meuser. *Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN*. To appear in Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2024.

created by the Collection Module. The updated RF model will then be sent to the P4 Generator module to generate compatible match-action table entries and embed the new model in the programmable data plane.

IV. EVALUATION AND DISCUSSION

This section presents the evaluation results of *COML-IDS* on diverse network traffic patterns sourced from various datasets, including CICIDS17 [18], CICDDoS19 [19], and Botnet [20]. The initial training was conducted using 70% of CICIDS17, while the remaining 30% of CICIDS17 and the other datasets were used for testing. For training and updating the DP-IDS and CP-IDS, we use an Ubuntu server with 128GB RAM and 4 CPUs (Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz). The macro-average F1 score reflects the detection performance of *COML-IDS*. The value of MC_{thr} is set to 0.3, as suggested by CML-IDS, that it achieves the best balance between the detection performance and speed.

A. Detection Performance Evaluation

This section highlights the improvement in detection performance and the reduction in sub-flows classified by CP-IDS using the proposed *COML-IDS*, compared with CML-IDS. Figure 3 illustrates these results across three different network traffic patterns. Network traffic patterns from CICIDS17 are considered similar since the model is trained on a portion of CICIDS17. In contrast, the other two datasets represent new, previously unseen network traffic patterns.

According to the results in Figure 3 (left), *COML-IDS* achieves higher detection performance across all network traffic patterns. Although the improvement in detection performance for the similar pattern (i.e., CICIDS17) is modest (3%), the gains for the new patterns are significant: 33% for CICDDoS19 and 31% for the Botnet dataset. This demonstrates that *COML-IDS* can achieve higher generalization and detection performance when encountering new network traffic patterns. Additionally, Figure 3 (right) reveals the percentage of sub-flows classified by CP-IDS among all classified sub-flows in *COML-IDS*. Recall that classifying sub-flows using CP-IDS requires sending P_{flow} (§III-B) to the control plane, which increases the network load. Compared to CML-IDS, *COML-IDS* consumes less network bandwidth for sending P_{flow}, particularly for the new traffic patterns. In other words, more sub-flows are classified by DP-IDS with higher confidence

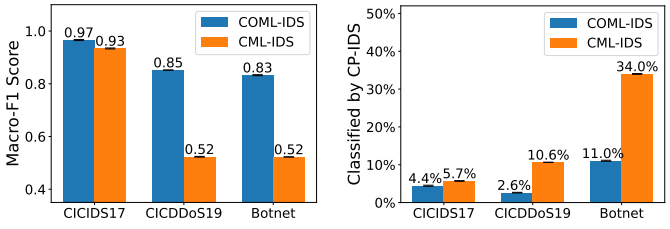


Fig. 3: Comparison of detection performance (left) and network load (right) of *COML-IDS* with CML-IDS across various network traffic patterns.

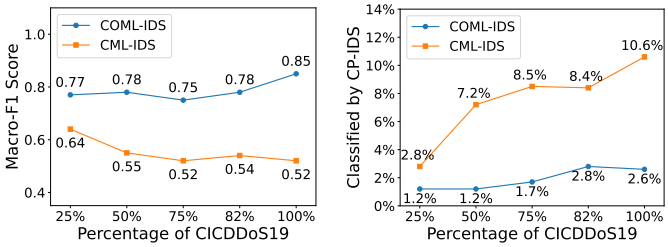


Fig. 4: Effectiveness of continually retraining DP-IDS. Comparing detection performance (left) and network load (right) of *COML-IDS* and CML-IDS across different volumes of received new traffic.

in *COML-IDS*, revealing that online learning can effectively adapt to changes in network traffic.

B. Effectiveness of Online Learning

This section demonstrates the effectiveness of *COML-IDS*, highlighting how online learning can gradually enhance detection performance through continual updates. As shown in Figure 4 (left), when receiving only 25% of the unseen network traffic patterns (i.e., CICDDoS19), *COML-IDS* outperforms CML-IDS by 13%. As the number of received flows increases to 100% of the new traffic patterns, the performance gap widens to a 33% improvement for *COML-IDS*.

According to results, when encountering new network traffic patterns, *COML-IDS* improves by 8% as the volume of new, previously unseen sub-flows increases. In contrast, CML-IDS performance decreases by 12% with the same increase in new sub-flows. Moreover, Figure 4 (right) shows that the sub-flows classified by CP-IDS increase by 7.8% in CML-IDS, while for *COML-IDS*, this increase is only 1.4%. It indicates that online learning can efficiently adapt to changes in network traffic and thus retains lower bandwidth consumption for forwarding traffic features to the control plane.

C. Impact of Active Learning-based Annotation

In *COML-IDS*, annotating newly received network traffic patterns (forwarded due to the alert trigger) is essential for fully integrating our online learning approach. As detailed in §III-C2, an active learning-based annotation mechanism is developed to automate this process. While this mechanism enhances automation, it may reduce annotation accuracy. The out-of-distribution (OOD) model is trained with CICIDS17 network traffic patterns. Therefore, any received network traffic patterns that differ from the training set are considered out-of-distribution and annotated using expert knowledge. In contrast, in-distribution patterns are annotated using CP-IDS.

However, among the sub-flows classified as in-distribution, some may still originate from different distributions, leading to potential incorrect annotations by the CP-IDS. Our evaluation results indicate that for sub-flows with different distributions (i.e., CICIDS19 and Botnet), the number of incorrectly annotated sub-flows by CP-IDS increases by 24%, highlighting a limitation of the active learning-based annotation approach.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Pegah Golchin, Chengbo Zhou, Hengyu Liu, Björn Scheuermann, Ralf Kundel, Tobias Meuser. *Integrating Online Learning with Collaborative Machine Learning for Continuous Intrusion Detection in SDN*. To appear in Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2024.

TABLE I: Detection performance for different scenarios

Scenarios	CICIDS17	CICDDoS19	Botnet
COML-IDS with only Expert knowledge annotation	***96.3%***	***85.7%***	***83.9%***
COML-IDS with Active Learning-based annotation	***96.1%***	***76.3%***	***78.6%***
Baseline (CML-IDS)	***93.6%***	***52.3%***	***52.2%***

To assess the impact of this limitation on overall detection performance, we compare the detection capabilities of *COML-IDS* under two conditions: one solely utilizing expert knowledge for annotation and the other employing an active learning-based approach. Results from CML-IDS are used as a baseline for comparison.

According to the results presented in Table I, the overall detection performance decreases when active learning is employed, as anticipated. However, relying exclusively on expert knowledge for annotation is impractical due to the high cost associated with the process. By integrating active learning, *COML-IDS* maintains higher detection performance compared to the baseline, CML-IDS. Specifically, for the CICDDoS19 dataset, *COML-IDS* with active learning outperforms CML-IDS by 24%, and for the Botnet dataset, this improvement reaches approximately 26%.

V. CONCLUSION

In this paper, we introduce *COML-IDS*, an online learning approach designed to enhance the intrusion detection capabilities of the existing collaborative ML-based IDS (CML-IDS). *COML-IDS* addresses the issue of degraded detection performance caused by the dynamic nature of network traffic, effectively identifying new and previously unseen traffic patterns. By incorporating various modules in both the data plane and the control plane, we ensure the automatic update of the model in the programmable data plane when the alert module is triggered. Our results demonstrate that *COML-IDS* achieves an average detection performance improvement of 32% across new, diverse network traffic patterns when using expert knowledge for annotation and 25% when incorporating an active learning-based annotation. While the active learning approach enhances automation and practicality, it slightly reduces overall detection performance. Additionally, *COML-IDS* reduces the amount of traffic forwarded to the control plane when encountering new, previously unseen network traffic patterns. As this work is implemented on a software-based P4 switch (BMv2), future work could involve deploying this framework on hardware-based programmable switches.

ACKNOWLEDGMENT

This work is funded by the Federal Ministry of Education and Research of Germany (BMBF) through Software Campus Grant 01IS17050 (ML-based NIDS), and the CELTIC-NEXT Flagship Project AI-NET-PROTECT.

REFERENCES

[1] N. Z. Bawany, J. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arabian Journal for Science and Engineering*, vol. 42, pp. 425–441, 2017.

[2] P. Panagiotou, N. Mengidis, T. Tsikrika, S. Vrochidis, and I. Kompatsiaris, "Host-based Intrusion Detection Using Signature-based and AI-driven Anomaly Detection Methods," *Information & Security*, vol. 50, no. 1, pp. 37–48, 2021.

[3] P. Golchin, C. Zhou, P. Agnihotri, M. Hajizadeh, R. Kundel, and R. Steinmetz, "CML-IDS: Enhancing Intrusion Detection in SDN Through Collaborative Machine Learning," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–9.

[4] P. Golchin, J. Weil, R. Kundel, and R. Steinmetz, "Dynamic network intrusion detection system in software-defined networking," in *2nd Workshop on Machine Learning & Networking (MaLeNe), co-located with the 5th International Conference on Networked Systems (NetSys 2023)*, 2023, pp. 1–2.

[5] P. Golchin, N. Rafiee, M. Hajizadeh, A. Khalil, R. Kundel, and R. Steinmetz, "Sscl-ids: Enhancing generalization of intrusion detection with self-supervised contrastive learning," in *2024 IFIP Networking Conference (IFIP Networking)*. IEEE, 2024, pp. 404–412.

[6] P. Golchin, R. Kundel, T. Steuer, R. Hark, and R. Steinmetz, "Improving DDoS Attack Detection Leveraging a Multi-aspect Ensemble Feature Selection," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–5.

[7] P. Golchin, N. Rafiee, and R. Kundel, "A data-driven solution for improving transferability of traffic flow feature selection," in *2024 IFIP Networking Conference (IFIP Networking)*. IEEE, 2024, pp. 592–594.

[8] A. O. Alzahrani and M. J. Alenazi, "ML-IDSDN: Machine learning based Intrusion Detection System for Software-Defined Network," *Concurrency and Computation: Practice and Experience*, 2023.

[9] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine Learning-enabled DDoS Attacks Detection in P4 Programmable Networks," *Journal of Network and Systems Management*, vol. 30, pp. 1–27, 2022.

[10] B. M. Xavier, R. S. Guimarães, G. Comarela, and M. Martinello, "Programmable Switches for in-networking Classification," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[11] J.-H. Lee and K. Singh, "SwitchTree: In-network Computing and Traffic Analyses with Random Forests," *Neural Computing and Applications*, pp. 1–12, 2020.

[12] C. Busse-Grawitz, R. Meier, A. Dietmüller, T. Bühler, and L. Vanbever, "pforest: In-network Inference with Random Forests," *arXiv preprint arXiv:1909.05680*, 2019.

[13] R. N. Carvalho, L. R. Costa, J. L. Bordim, and E. A. Alchieri, "Detecting DDoS Attacks on SDN Data Plane with Machine Learning," in *2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, 2021, pp. 138–144.

[14] M. Zang, C. Zheng, L. Dittmann, and N. Zilberman, "Towards Continuous Threat Defense: In-network Traffic Analysis for IoT Gateways," *IEEE Internet of Things Journal*, 2023.

[15] C. Zheng, Z. Xiong, T. T. Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "IIsy: Hybrid In-Network Classification Using Programmable Switches," *IEEE/ACM Transactions on Networking*, 2024.

[16] C. Liu, A. Malboubi, and C.-N. Chuah, "OpenMeasure: Adaptive Flow Measurement & Inference with Online Learning in SDN," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2016, pp. 47–52.

[17] M. Hajizadeh, S. Barua, and P. Golchin, "FSA-IDS: A Flow-based Self-Active Intrusion Detection System," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2023.

[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[19] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in *2019 international carnahan conference on security technology (ICCST)*. IEEE, 2019, pp. 1–8.

[20] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards Effective Feature Selection in Machine Learning-based Botnet Detection Approaches," in *2014 IEEE Conference on Communications and Network Security*. IEEE, 2014, pp. 247–255.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.