Leveraging In-band Network Telemetry for Automated DDoS Detection in Production Programmable Networks: The AmLight Use Case

Hadi Sahin*, Jeronimo Bezerra*, Italo Brito*, Renata Frez†, Vasilka Chergarova*,
Luis Fernandez Lopez*‡, Julio Ibarra*,
*Center for Internet Augmented Research and Assessment
Florida International University, Miami, Florida, 33199
†Rede Nacional de Ensino e Pesquisa, Rio de Janeiro, Brazil
‡ Faculty of Medicine, University of São Paulo, São Paulo, Brazil
Emails: see https://ciara.fiu.edu/staff.html

***Abstract*—Programmable data planes have provided great flexibility in defining the behaviors of packet forwarding switches, routers, and network interface cards (NICs). The In-band Network Telemetry (INT) technology further increased network operators' potential to manage packet flows by enabling real-time and customizable monitoring of packets without creating much overhead on the network. These recent advancements in networking technology have generated significant research interest and activity, including studies on INT-based DDoS detection and mitigation mechanisms. However, in practice, INT technology has not been fully realized yet, especially in detecting network anomalies in real-time. There is also a gap in the literature that provides a comparative evaluation of INT-based solutions against existing alternatives. In this paper, we aim to implement a holistic real-time INT-based DDoS detection mechanism. The proposed mechanism will retrieve INT data from the network, analyze it using machine learning (ML) models in real-time, and send the information to the control plane. We will also compare the performance of using INT to detect DDoS attacks against sFlow-based detection.**

***Index Terms*—P4 programmable plane, INT, DDoS detection, Machine Learning**

I. Introduction

One of the significant challenges in network security is Distributed Denial of Service (DDoS) attacks, which aim to disrupt the normal functioning of a system, rendering it incapable of delivering legitimate services. This is accomplished by either exhausting the system's resources or overwhelming its response mechanisms [1]. Over the past decade, there has been a significant increase in cybersecurity incidents, with DDoS and malware attacks emerging as the most prevalent threats [2]. For example, in the AmLight Research and Education (R&E) network[1] alone, 193 major and 346 low-to-medium DDoS attacks were detected since March 2020. These threats are rapidly evolving, becoming increasingly sophisticated and challenging to counter [2]. As a result, defensive strategies on production networks must adapt and remain dynamic to effectively combat emerging threats.

[1]AmLight is an intercontinental network that provides production network infrastructure for research and education in the United States, Latin America, and South Africa.

Simultaneously, we have witnessed significant advancements in network management. Software-defined networking (SDN) has enabled centralized and software-based management of networks, providing greater flexibility and programmability [3]. The Protocol-independent Packet Processors (P4) programming language further enhanced this flexibility by allowing customization of packet processing logic [4]. Building on this infrastructure, In-band Network Telemetry (INT) was introduced for real-time monitoring of the network at the packet level [5]. INT provides granular data that can be used for various tasks, including traffic engineering, Quality of Service (QoS) management, and security monitoring [6].

Our previous work described the implementation of Software-Defined Networking (SDN) and In-band Network Telemetry (INT) technologies on the AmLight Network [6], [7]. AmLight later leveraged INT technology to detect and record microbursts [8]. Building on these foundations, this paper focuses on addressing security threats, particularly Distributed Denial of Service (DDoS) attacks, on the AmLight network. We explore the potential of INT to detect these threats and enhance overall network security. INT offers sub-second visibility into network operations, enabling faster and precise threat detection and response [6]. By utilizing this granular, real-time data, cybersecurity detection capabilities can be significantly improved.

On the other hand, INT data presents its own challenges in terms of storage and processing of the data, especially in a network with more than 1.2 Tbps of aggregated international capacity [6]. In the AmLight network, one minute of INT data would amount to 30 GB of data, and includes telemetry information on more than 80 million packets. Given the novelty and challenges of this technology, there is a lack of knowledge of the real-time application of INT in DDoS threat detection. From a research perspective, although there are studies that utilize INT data in ML-based DDoS detection [9], [10], a test of the concept with real data is lacking, and to our knowledge, an automated INT detection mechanism has not been proposed.

This paper aims to address this gap in the literature and provide a proof of concept for using INT in automated DDoS detection. As the AmLight network has already deployed INT, we have the unique opportunity to leverage this technology specifically for DDoS detection. In future work, we also plan to scale this mechanism for implementation in the production AmLight network.

In this paper, we make the following contributions:

Firstly, we compare DDoS detection based on INT data with alternative methods that utilize sampled data. We specifically focus on sFlow data, which is widely utilized in the industry for intrusion detection systems (IDS). We use production data retrieved from the AmLight network to compare sFlow and INT performances in detecting DDoS attacks.

Secondly, we propose an automated DDoS detection mechanism designed to detect and report DDoS attacks at line rate. This mechanism employs machine learning (ML) models to predict normal and attack flows.

Finally, we discuss the challenges and limitations of using INT-based IDS and how our findings will guide its deployment on a real production network.

This paper is organized as follows: The next section provides background information on networking and intrusion detection systems and discusses related work. Section III introduces the proposed DDoS automated detection mechanism. Section IV discusses the experimental setup and results. Section V provides a discussion of challenges of using INT data. Finally, Section VI concludes the paper.

II. Background and Related Work

A. Network Measurement and Monitoring

1) sFlow: sFlow is a network measurement tool that employs proxy reporting through device-level sampling to provide comprehensive network status information. It utilizes two distinct sampling methods. The first is packet-count based, which samples packet information at equal intervals of packet counts. The second is time-based, sampling packet data at equal time intervals [11].

The sFlow monitoring architecture consists of two key components. The first is the sFlow Agent, positioned on switches or routers, which collects sampling information. The second is the sFlow Collector, which acts as the analyzer, receiving and processing the data transmitted by the sFlow Agent [12].

sFlow provides valuable insights into network performance and behavior, and is widely used in the industry for traffic monitoring and anomaly detection. On the other hand, there are not many studies that utilize sFlow in their analysis [13].

2) INT: In-band Network Telemetry (INT) enhances network monitoring by using programmable data plane technology to gather real-time information from data packets as they travel through the network. This approach has various applications, such as INT itself, In-situ Operations, Administration, and Maintenance (IOAM), Alternate Marking Performance Measurement (AM-PM), and Active Network Telemetry (ANT) [5].

INT is built on the P4 language. While it is possible to directly utilize P4 to extract telemetry data, INT provides a standard method to embed and extract telemetry data within packet headers. It involves embedding telemetry information into packets as they traverse the network; therefore, it has minimal impact on packet processing performance [12].

There are three types of switches involved in the process: source, transit, and destination/sink (See Figure 1). At the source switch, an INT header is inserted that specifies what telemetry data should be collected. Based on these instructions, telemetry data is added as metadata at the transit switches. At the sink switch, the metadata is extracted and sent to the INT collector [9], [14].

Despite its advantages, INT has some limitations. By including telemetry data, it reduces the payload ratio for normal packets and increases the switch's workload due to gathering and processing of telemetry data [5]. However, as demonstrated in [6], it is possible to effectively mitigate and minimize these drawbacks within the network.

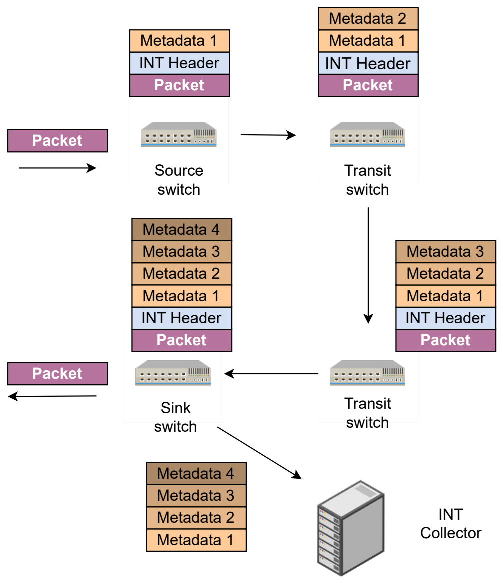


Fig. 1: INT data collection.

B. DDoS attacks

Denial of Service (DoS) attacks aim to overwhelm a target's resources with malicious traffic, preventing it from serving legitimate requests [1]. When this type of attack is carried out by multiple distributed sources, it is called a Distributed Denial of Service (DDoS) attack. Examples of such attacks include SYN flood, MSSQL, DNS amplification, and NTP amplification attacks [15].

To provide more detail about these attacks, let's examine one common type of TCP based attack: the SYN flood. This attack exploits the TCP handshake process. In the TCP protocol, a client initiates a connection by sending a SYN packet to a server. The server responds with a SYN-ACK packet to acknowledge the request and indicate readiness to establish the connection.

The client then completes the handshake by sending an ACK packet.

In a SYN flood attack, the attacker sends a flood of SYN packets, but never responds with the final ACK packet. This leaves the server with numerous half-open connections, consuming its resources and preventing it from handling legitimate requests.

C. The Intrusion Detection System (IDS)

Intrusion Detection Systems (IDS) are critical defense and security mechanisms that monitor and analyze network traffic to identify unusual or potentially malicious activities within a network. In terms of utilized detection techniques these systems can be broadly classified into three categories: signature-based, anomaly-based and hybrid techniques.

The signature-based technique relies on the careful analysis and recording of previous malicious activities [16]. Once sources of malicious activities are well-established and the patterns are recognized, they are used to identify, block, and eliminate similar malicious flows. This method is highly accurate for detecting known threats. However, it is less effective against new or unknown threats and requires constant updates to maintain its effectiveness.

In anomaly-based techniques, normal and expected behavior of a network is defined based on historical network traffic data. This network profile is then used to manage flows and identify abnormal behavior. Based on this technique, an IDS can perform well against unseen threats; however, it is prone to false alarms as network behavior can show quite varying patterns, which could make it difficult to capture all normal flow behaviors [16].

Finally, the hybrid techniques combine and exploit the strengths of both methods to provide a more robust defense against intrusions. For instance, one way to combine these methods is to use threat records to identify abnormal behavior, and thus recognize normal patterns in a network [16].

IDS is a general term for all defense mechanisms against intrusions. In this paper, we implement an anomaly-based technique and utilize machine learning models to differentiate normal flows from malicious behavior and detect DDoS threats. Therefore, we will specifically refer to DDoS detection systems rather than using the general term IDS.

D. Related Work

Nam et al. [9] proposed a network intrusion detection system (IDS) based on the ONOS SDN controller, where INT data is used as input to the RNN-based machine learning detection. They implemented their model in a Mininet environment and gathered INT data. Then they used this data for training and testing. They achieved an F1-score of 92.41% with their test set. Although the paper utilizes INT data for detection, the data is not from a production environment, and the detection process is not automated.

Cao et al. [14] presented a DDoS detection and mitigation scheme that employs spatial-temporal graph convolutional network models. They used INT data to gather spatial and temporal information on the network state. Their models require data from every switch at specific time intervals, necessitating a larger dataset. To address this challenge, they implemented equal interval sampling. While their work outlines a comprehensive detection and mitigation framework, it does not address the implementation of an automated detection and mitigation system.

Aslam et al. [17] proposed a real-time DDoS detection and mitigation application in an SDN-based environment. The application consists of four modules: the main method, detection, mitigation, and flow-rule generation modules. The main method module connects the application to the ONOS controller and Python server and also manages the communication between all modules. It processes the flows into features and statistics and sends them to the detection module. The attack detection module deploys the ML models, the mitigation module traces the origin of the attack, and the flow-rule generation module creates flow rules to discard malicious flows. This paper is a great example of an automated DDoS application. However, it utilizes data retrieved through the OpenFlow protocol. As a result, the number of features that can be derived from this method may be somewhat limited.

Some studies utilize the P4 language to extract packet-level data and analyze network activity. Seufert et al. [18] investigate how this method can be applied to monitor large-scale networks with high volumes. They explore the use of packet-level data in traffic classification, Quality of Service (QoS), IoT device classification, and intrusion detection tasks. Their work offers valuable insights into the parameters required to deploy such mechanisms on a real production network. This study could be further developed by including the use of INT technology.

Musumeci et al. [19] employed the P4 language to extract network traffic information for detecting DDoS attacks. They compared packet mirroring, header mirroring, and P4 metadata extraction, demonstrating that P4 metadata extraction significantly reduces feature extraction and processing time for ML prediction. Similarly, Diana et al. [20] used the P4 language to extract data and analyze DDoS attacks on a 5G network. They designed and implemented an IDS system that detects live DDoS attacks and mitigates the problem. Their paper provides a comprehensive road map for creating an autonomous IDS based on P4. Although both of these studies are similar to ours in terms of using packet-level data and providing automated detection, they extract information directly from the switches using the P4 language, rather than utilizing INT data. As discussed earlier, our experience with INT in the Amlight network gives us the unique opportunity to complement these studies with INT-based DDoS detection.

III. Proposed Automated DDoS Detection Mechanism

DDoS attacks can be short-lived and unpredictable, requiring continuous monitoring of the network. Detection of DDoS attacks should also be quickly followed by taking appropriate measures. Thus, a dynamic and automated process is needed to defend against DDoS attacks.[2] The proposed mechanism will continuously extract packet information, process the data, and feed it into machine learning (ML) models for anomaly detection. This mechanism was developed by expanding on [20]'s work.

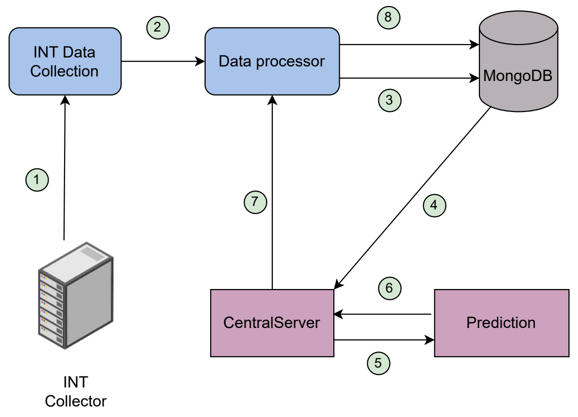


Fig. 2: Automated DDoS detection mechanism.

Figure 2 presents the proposed automated DDoS detection mechanism. It is composed of four modules: *INT data collection*, *Data processor*, *CentralServer*, and *Prediction*, as well as the database. Below, we discuss them in further detail.

1) INT Data Collection: This module gathers INT data from the *INT Collector* (1). It primarily runs a Python script that reads the INT telemetry header, metadata, and the IP header information. Then, the module sends this information to the *Data processor* module (2). The following packet-level features are collected from the IP header:

- Source and destination IP addresses
- Source and destination ports
- Protocol: TCP or UDP
- Packet length

From the INT header and metadata we retrieve the following:

- Queue occupancy: Queue depth when the packet is removed from the queue.
- Ingress time: The 32-bit timestamp in nanoseconds of when the packet enters a switch
- Egress time: The 32-bit timestamp in nanoseconds of when the packet exits a switch

2) Data Processor: This module runs a JavaScript script to process the packet INT data. The module creates packet-level data such as *Inter arrival time* by taking the difference between consecutive *Ingress time*. It also creates flow-level features based on *Flow ID*. Drawing on the literature, *Flow ID* is defined by a five-tuple variable consisting of: *Source IP address, Destination IP address, Source port, Destination port, and Protocol* [17].

[2]We should note that in this paper, we do not address mitigation.

The module keeps a counter for *the Number of packets*, *Flow duration*, and *Total packet size* by aggregating them based on the *Flow ID*. Moreover, with these variables the module creates flow-level variables such as *Packets per second* and *Packet size per second*.

The module first searches if there is any record of the *Flow ID* for the incoming packet. If there is no record, it creates a new entry with packet information, default values of flow-level information (which are mostly 0 at initiation), and the timestamp of the new record. If there is already a record for the *Flow ID*, it updates all flow-level information but replaces all packet-level data with the data from the new packet.

After data is processed, it is sent to the database for storage (3). This data processing step primarily helps us create flow-level features and enhance our capabilities to differentiate abnormal flows from normal flows. It also reduces the workload for the database, as we only keep one record for each flow at a given time.

Additionally, the *Data Processor module* receives predictions from ML models from the *CentralServer* (7). It aggregates them into one prediction label and sends it to the database with the timestamp and *Prediction Latency*, measured by the difference between prediction time and the time of the packet's registration (8).

3) CentralServer: This module continuously communicates with the database to check whether there is an update in the records (4). It does not consider new entries with new *Flow IDs*, but focuses on existing records from their first update to the end. If there is an update, it sends this information to the *Prediction* module (5).

The *CentralServer* module also continuously listens to the *Prediction* module for predictions and retrieves them whenever they are available (6). It then sends these predictions to the *Data processor* module for an aggregated prediction (7).

4) Prediction: This module primarily provides ML predictions. When initialized, it uploads the pre-trained ML models and the coefficients of scaler transformation, which are used to standardize the feature values to unit variance. The module receives packet- and flow-level information from the *CentralServer* when there is an update in the records (5). The module then standardizes this new feature set and feeds it to the pre-trained models for prediction. These predictions are subsequently retrieved by the *CentralServer* (6).

IV. Experimental Evaluation

In this section, we have two experimental stages. In the first, we evaluate the effectiveness of INT and sFlow data for DDoS attack prediction across various machine learning models using AmLight production data and simulated attacks. In the second, we present and discuss the results of experiments conducted with our proposed automated DDoS detection mechanism, implemented on a physical testbed.

A. Evaluation Metrics

We use machine learning models to analyze network data and differentiate attack flows from normal flows. To measure performance, we utilize conventional ML metrics that leverage True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The following metrics, with their respective formulas, are used [21]:

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN)$$
$$\text{Recall} = TP/(TP + FN)$$
$$\text{Precision} = TP/(TP + FP)$$
$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$$

In all four of these metrics, the minimum value is 0 and the maximum value is 1, with 1 being the perfect score.

We also employ a *Confusion matrix* metric, which is a two-by-two matrix of positives (P) and negatives (N) [22]. It shows how many data points fall into each quadrant, either as a quantity or a percentage. PP and NN quadrants indicate correctly classified data, while PN and NP represent misclassified data.

B. DDoS Prediction Using INT and sFlow Data

In this experimental analysis section, we compare the effectiveness of INT and sFlow data in detecting DDoS attacks. While INT offers advantages over other monitoring tools in terms of its granularity and minimal burden on the network while gathering data, our focus here is on INT's performance relative to sFlow in identifying DDoS attacks.

One important difference between INT and sFlow is that INT gathers information from every packet, while sFlow uses sampling. In our production environment, sFlow monitors network flows by selecting 1 out of every 4,096 packets.

This analysis will provide insights into the strengths and limitations of both INT and sFlow in network security applications, considering the nature of the collected data and whether sampling is applied or not.

1) Data Collection: The INT and sFlow data were collected from a subnet of the AmLight network, specifically focusing on traffic interacting with a production web server. We captured all traffic to this server from June 6 to June 11, 2024, recording all normal flows during this period. However, we used a smaller set of INT data for training, as it was sufficient to achieve reliable model performance.

TABLE I: Simulated Attack Flows

Attack Type	Date	Attack Episode
SYN Scan	06.10.2024	13:24:02 - 13:57:03
SYN Scan	06.10.2024	16:30:51 - 16:35:20
UDP Scan	06.10.2024	16:36:20 - 16:53:00
UDP Scan	06.10.2024	16:56:45 - 16:59:99
SYN Flood	06.10.2024	20:48:01 - 20:49:01
SYN Flood	06.10.2024	20:52:11 - 20:54:12
SYN Flood	06.11.2024	20:13:31 - 20:15:31
SYN Flood	06.11.2024	20:16:41 - 20:17:01
SYN Flood	06.11.2024	20:17:17 - 20:17:37
SlowLoris	06.11.2024	20:27:37 - 20:28:37
SlowLoris	06.11.2024	20:29:12 - 20:31:12

We also generated traffic by simulating various attack types, including TCP SYN scan, UDP scan, TCP SYN Flood, and SlowLoris. These attacks varied in duration. We simulated attack flows using the *Hping* tool [23] and performed the SlowLoris attack using a Python script as described by [24]. Table I details the attack flows, including their types, dates, and specific times. For example, the first row shows that we conducted a *SYN Scan* attack on the target from 13:24:02 for approximately 33 minutes.

2) Feature Selection: We utilized all available information from both sFlow and INT data. Therefore, we preserved all packet-level details while also generating flow-level statistics. The goal was to create as many differentiating features as possible for normal and attack flows. As discussed earlier, we defined flows using the five-tuple: *Source IP address, Destination IP address, Source port, Destination port, and Protocol*. Based on the *Flow ID*, we calculate the duration of each flow, the number of packets within it, and the total bytes transmitted. From these primary variables, we then derive additional flow-level metrics including mean and standard deviation of these features.

TABLE II: Features used to detect DDoS attacks

Features	INT	sFlow
Protocol	✓	✓
Packet Size*	✓	✓
Number of packets	✓	✓
Queue Occupancy*	✓	×
Hop Latency*	✓	×
Inter Arrival Time*	✓	✓
Flow rate (Gbit/s)	✓	✓
Packet rate (Packet/s)	✓	✓

Note: * Includes packet-level, cumulative, average, and standard deviation of the variables. The cumulative inter-arrival time denotes flow duration.

Table II displays the features for both INT and sFlow. As can be seen from the table, both network monitoring tools provide similar metrics on network traffic. The main difference is that INT provides additional information on *Queue occupancy* and *Hop latency*. It should be noted that we did not utilize *Hop Latency* in our analysis as we were not able to retrieve it on the same scale for all flow types.

3) Machine Learning Models: We employed various ML models, including Random Forest (RF), K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GNB), and Neural Networks (NN) [21], [22]. The first three traditional ML models require less computational power and training time, while also providing good prediction performance in DDoS detection [21]. We also used a Neural Network for its effectiveness in handling more complex tasks. We implemented a shallow neural network which comprises three hidden layers with 32, 16, and 8 neurons.

The outcome label is binary, with normal flows coded as 0 and attack flows as 1. Therefore, we implemented binary classification tasks using these four ML models [25].

Due to the large volume of data, we focused our INT analysis on specific time frames: June 10th from 1:00 PM to 3:00 PM and June 11th from 7:00 PM to 9:00 PM. It should be noted that we saw no need to use the entire sample, as the model trained well with this subset. For sFlow data, we utilized all

TABLE III: Performance Comparison of ML Models for DDoS Attack Detection using INT vs sFlow Data

Data	Model	Accuracy	Recall	Precision	F1-score
INT	RF	1.0000	0.9999	0.9999	0.9999
sFlow	RF	9.9996	0.9900	1.0000	0.9950
INT	GNB	0.9978	0.9849	1.0000	0.9924
sFlow	GNB	0.9951	0.9925	0.9041	0.9462
INT	KNN*	0.9995	0.9992	0.9984	0.9988
sFlow	KNN	0.9992	0.9875	0.9949	0.9912
INT	NN	0.9997	0.9987	0.9981	0.9984
sFlow	NN	0.9995	0.9875	1.0000	0.9937

Note: * We utilize a smaller training and testing set, one thousandth of the whole sample, for KNN to facilitate easy convergence.

available data from June 6th to June 11th. To train the models, we employed a 90:10 train-test split ratio.

4) Experimental Results: Table III presents a comparative analysis of machine learning model performances using INT versus sFlow data. The table lists results from the four ML models based on whether they utilized INT or sFlow data. As discussed, we provide accuracy, recall, precision, and F1-score as performance metrics.

Table III shows that using sFlow and INT data for detecting DDoS attacks yields consistently similar and high accuracy across all models. The F1-scores for both sFlow and INT data in the RF, KNN, and NN models are above 99 percent. The results for GNB are slightly lower for sFlow; however, it still reaches a significant 95 percent level.

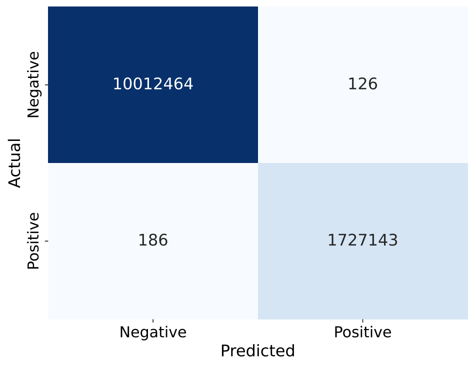


Fig. 3: Confusion Matrix for Random Forest Model Using INT Data

We also drew the confusion matrices from the *RF* model to discuss the misclassified data more closely. Figure 3 illustrates the confusion matrix for the INT data. It shows that out of approximately 1.7 million attack packets, 186 were misclassified, while 126 packets from normal flows were misclassified as attack flows. Similarly, Figure 4 displays the confusion matrix for the sFlow data. As can be seen from the figure, the *RF* model perfectly predicted the normal flows but misclassified 4 of the attack packets.

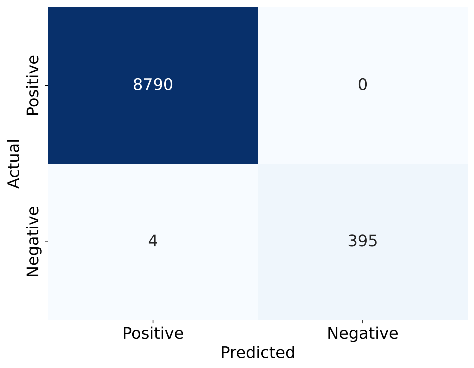


Fig. 4: Confusion Matrix for Random Forest Model Using sFlow Data

To get a better comparison between INT and sFlow, we also ran the analysis with a different training and testing setup. We assigned the flows from June 11, 2024, as the test set; the flows from remaining days were assigned to the training set.

The attack flows from June 11 include both SYN Flood and SlowLoris attack types. Since SlowLoris is not included in the training set, this setup allows us to evaluate how the models perform with a zero-day (unseen) attack, specifically SlowLoris attacks. It should also be noted that SlowLoris attack flows are relatively harder to detect compared to other DDoS attacks, such as SYN and UDP Flood [26]. Thus, this setup provides a more challenging test for sFlow and INT data.

TABLE IV: Performance Comparison of ML Models for DDoS Attack Detection with Zero-day attacks

Data	Model	Accuracy	Recall	Precision	F1-score
INT	RF	1.0000	1.0000	0.9999	1.0000
sFlow	RF	0.9999	1.0000	0.9907	0.9953
INT	GNB	0.9919	1.0000	0.9959	0.9959
sFlow	GNB	0.9959	1.0000	0.6057	0.7544
INT	KNN	0.9988	0.9993	0.9984	0.9988
sFlow	KNN	0.9997	1.0000	0.9550	0.9770
INT	NN	0.9996	1.0000	0.9992	0.9996
sFlow	NN	0.9937	0.0000	0.0000	0.5000

Table IV presents the analysis for zero-day attacks. Given the presence of unseen data in the test set, we expect lower performance from the models. Analysis with the INT data shows high accuracy, exceeding 99 percent for all models. For instance, the results from RF show 100 percent Accuracy, Recall, and F1-score, with a Precision level of 0.9999. Analysis using sFlow data shows relatively lower accuracy, particularly for the GNB and NN models. However, the results from the RF and KNN models achieve the same level of accuracy as those from the INT data.

The results from both Table III and IV show that both INT and sFlow data offer valuable and comparable advantages in detecting DDoS attacks. To further understand the role of sampling in prediction performance and to deduce any

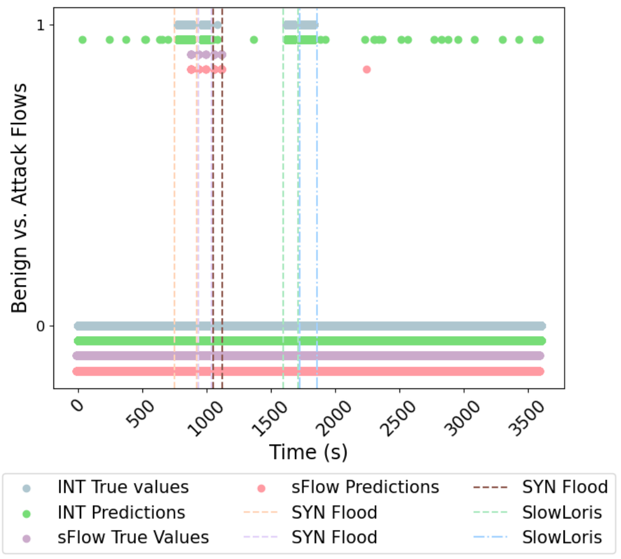


Fig. 5: Real Data versus RF Model Predictions from INT and sFlow Data

insights from comparing the two data sources, we visualized the performance of the INT and sFlow data using the RF model results. Figure 5 presents this comparison. In the figure, the dashed vertical lines indicate the simulated attack episodes. The colored horizontal dots represent the true values and predictions from INT and sFlow, respectively. For instance, in a scenario with 100 percent accuracy, the purple and red dots for the sFlow data should align perfectly.

Figure 5 shows that the predictions from INT successfully identified all attack periods, although some misclassifications were observed. In the figure, the INT true values are represented in light blue, indicating a value of 1 during the attack intervals and 0 otherwise. The predictions are shown in green and should align with the light blue intervals for perfect accuracy. Consequently, any green dots (predictions of 1) outside the attack intervals are considered misclassifications.

In contrast, for sFlow, the actual data is missing from the last two attack episodes. Due to sampling, sFlow did not capture any information on Slowloris, and therefore did not provide any predictions. However, when it did sample data, as seen in the first three periods, it provided strong predictions. Thus, from our analyses, we found that both INT and sFlow provided similar accuracy levels in predicting DDoS attacks. However, INT data has an advantage over sFlow, as it captures all packet data information. Analyses with sFlow data could underperform, if the attack episode is shorter than the sampling rate, or if the sampling interval does not coincide with the attack flows.

Finally, in Table V, we present the top five important features for each ML model. The subscripts *cum*, *avg*, and *std* denote cumulative, average, and standard deviation, respectively. The table shows that across all four models, *Inter Arrival Time*, *Packet Size*, *Queue Occupancy*, and *Protocol* are the most important features for detecting DDoS attacks. However, the

TABLE V: The Five Most Important Features for Detecting DDoS Attacks with INT Data based on model types

Features	RF	GNB	KNN	NN
Inter Arrival Time$_{\text{cum}}$	✓	✓	-	✓
Inter Arrival Time$_{\text{std}}$	✓	-	✓	✓
Packet Size	-	✓	✓	-
Packet Size$_{\text{avg}}$	✓	✓	✓	✓
Packet Size$_{\text{std}}$	✓	-	-	-
Queue Occupancy$_{\text{avg}}$	✓	-	✓	✓
Queue Occupancy$_{\text{std}}$	-	✓	-	-
Protocol	-	✓	✓	✓

variants of these features, individual, cumulative, average or standard deviation, that make the top five differ across models.

C. Automated DDoS Detection

In this experimental analysis section, we discuss results from our proposed automated DDoS detection mechanism. In this mechanism, the network traffic information is collected from live packets using INT tools. The data are then automatically processed into packet-level and flow-level features and stored in a database. This data is subsequently fed to pre-trained models, and the predictions are stored back in the database (See Figure 2).

We implemented this automated DDoS detection mechanism on a physical testbed using INT-enabled switches. In this experiment, our goal is primarily to provide a proof of concept by discussing DDoS prediction accuracy and prediction time.

1) The INT Testbed: The INT testbed consists of three key components: a source agent, a target agent, and a switch. Figure 6 displays the topology of the testbed. The source and target agents are connected through a switch, which regulates the data flow between them. In our network configuration, packets traverse from ports 1 and 2, but also traverse ports 3 and 4 of the switch, with one port acting as the source and the other functioning as the sink. We execute all module scripts from the source agent. Additionally, our INT Data Collection scripts, also running on the source agent, gather INT data from port 5. The source and target servers powered by dual AMD EPYC 7451 24-core processors and 128GB of RAM. Each server utilizes a Mellanox ConnectX-5 network card capable of 100Gbps throughput. The switch is an Edgecore Wedge DCS800.

2) Dataset: In the experiments, we utilized production data from a subnet of the Amlight network, as described in Section IV-B. In order to create a training set, we replayed a segment of the data including both benign and attack flows (See Table I) in the INT TestBed. Then this training set is used to pre-train the ML models offline. We did not train the model with SlowLoris attack types, as we want to use it as an zero-day (unseen) attack in the analysis, as we did in the first experimental section. In the testing phase, we replayed around 2500-packet data for each flow type, including benign and attack flows (see Table VI), to provide live/real-time detection. We ran the following command to replay the flows:

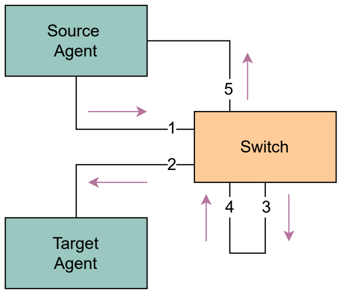


Fig. 6: The INT Testbed Topology

tcpreplay -i ⟨*interface*⟩ *-p* ⟨*number of packets*⟩ ⟨*pcap file path*⟩

3) Machine Learning Models: We aimed to replicate the analysis in Section IV-B on the INT TestBed, however we made some minor changes. We utilized the Random Forest (RF) and Gaussian Naive Bayes (GNB) models without any modifications. Instead of the Neural Network model used in the previous section, we implemented a Multi-layer Perceptron (MLP) classifier from the Scikit-learn Python library [27]. This MLP model functions similarly to the NN, with 3 hidden layers containing 64, 32, and 16 neurons, respectively, but requires less storage for trained model. We did not employ KNN, because of its relatively slower prediction times.

Regarding features used in the models, we utilized those listed in the INT column of Table II except *Hop Latency*. In total, we employed 15 packet-level and flow-level features.

4) Experimental Results from automated DDoS Detection: We conducted experiments in our INT Testbed by replaying normal flows and four types of attack flows. As discussed earlier, anomaly-based DDoS detection techniques are prone to false alarms [16], [28]. To minimize this issue, we employed ensemble voting [29] by combining the outputs from the MLP, RF, and GB models for each packet to create a single result. Specifically, if two or more of the predictions are 1, then it is classified as an attack flow; if they are 0, it is classified as a normal flow. However, we do not classify predictions as attack or normal flow immediately. We wait for three predictions. If two or more of the last three predictions are 1, then it is classified as an attack flow; the same applies for normal flows. For instance, if the last three predictions were [1, 0, 1], the final decision would be 1, indicating an attack flow.

Table VI presents the prediction performance from the automated DDoS detection mechanism. We achieved over 99 percent accuracy in predicting most of the attack types. The accuracy for benign flows is relatively lower, however, it still reaches a respectable 94 percent level. For SlowLoris attack flows, which are treated as zero-day attacks in the tests, the results achieved 97.95 percent accuracy. The second column of the table shows the total number of predicted packets and, among them, the number of misclassified ones. For SlowLoris attacks, 16 out of 779 packets were misclassified as normal flows, while the rest were correctly classified as attack flows.

TABLE VI: Performance Comparison of Automated DDoS Detection Based on Attack Types

Attack Type	Accuracy	Misclassified/ Number of Predicted Packets	Average Prediction Time (s)	Max Prediction Time (s)
UDP Scan	0.9947	14/2628	0.12	0.73
SYN Scan	0.9961	10/2542	0.44	1.81
SYN Flood	0.9984	27/2814	0.09	0.4
SlowLoris	0.9795	16/779	0.05	130.85
Benign	0.9417	136/2331	103.14	734.55*

Note: * Note: For benign flows, we report the 99th percentile of prediction time rather than the maximum value.

As discussed, in the experiments, we replayed each type of attack flow and normal flow as shown in Table VI, and evaluated the models' success in classifying their types. Figures 7a and 7b present the predictions for benign flows and SlowLoris attack flows, respectively. In the data, benign flows are denoted as 0, so a prediction of 1 indicates a misclassification. For attack flows, such as SlowLoris, the reverse is true. Figure 7a shows that the misclassification of benign flows occurs more frequently at the beginning and continues throughout as new flows are registered. On the other hand, the misclassification of SlowLoris attacks happens exclusively at the beginning of flows (Figure 7b). This pattern is also correct for the remaining attack types. Overall, our solution demonstrated strong performance in detecting attack types even for unseen and difficult to detect attack flows.

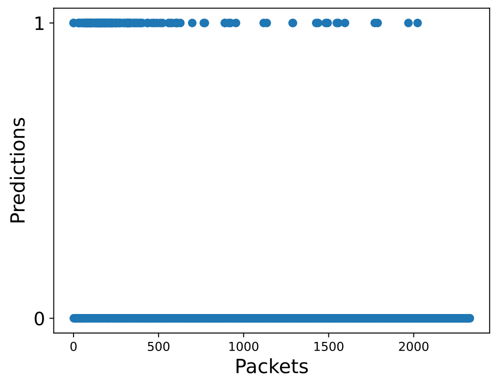
Table VI also shows the average and maximum prediction times for each attack type in the third and fourth columns. For most of the attack types, the prediction times are less than 0.5 seconds on average and remain under 2 seconds. However, the results show higher numbers for SlowLoris and benign flows. This is due to the high volume of these flows. This indicates that scaling up the project would require a mechanism with faster processing capabilities to handle production-level volumes and provide detection in a shorter amount of time.

V. Discussion

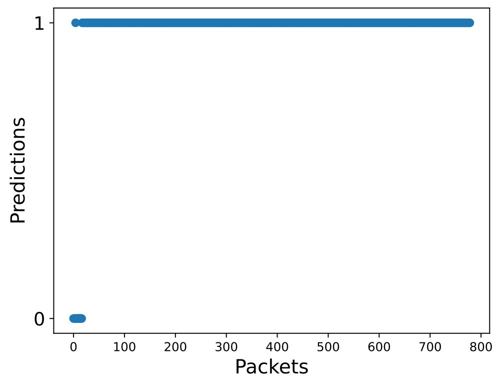
The information extracted from the INT data proved effective in detecting DDoS attacks for both seen and unseen data. The comparison with sFlow data also shows that both tools offer comparable and quite accurate performance. However, due to sampling, sFlow may fail to capture some attack episodes and, as a result, may miss detecting these attacks.

The analysis also shows that an automated detection mechanism can be successfully created based on INT data. The experiments on the testbed provided highly accurate and timely DDoS attack predictions. However, working with INT data poses some challenges. In this section, we will focus on these challenges and discuss how these experiences could help us realize a deployment of an automated DDoS detection on a production network based on INT data.

Effectively storing, processing, and analyzing this data requires significant computational resources and efficient



(a) Benign Flows



(b) Slowloris Attack Flows

Fig. 7: The distribution of predictions from Benign (a) and SlowLoris attack (b) flows. Attack flows are labeled as 1 and Benign flows are labeled as 0.

techniques. In our implementation, we used Python and JavaScript to retrieve and process the data and feed it to ML models. We used much lower packet rate levels than we would observe in attack flows in order to run experiments smoothly. For production-scale implementation, faster processing capabilities will be required. Additionally, the high volume of data poses a challenge if longer periods are needed to account for temporal patterns in the flows. In our implementation, we do not consider any temporal patterns.

There is already ongoing research on how to optimize the use of INT data (See [30], [31]). Both in training and detecting, optimized usage of INT may reduce the workload in deployment of the system in a production network. In future work, we are planning to build on these studies to address the challenges associated with high-volume network data.

The second challenge was the lack of a timestamp with day and hour information in the INT data. The timestamp provided by INT is limited to 32 bits in nanoseconds, which effectively restarts every 4.3 seconds. This limitation can make the inter-arrival time derived from INT susceptible to errors, as it is calculated by taking the difference between consecutive packet *egress times*. This issue could pose a greater challenge if we want to construct a longer time frame to detect and control for cyclic behavior in network traffic.

Finally, our setup did not allow us to fully utilize all INT-based variables. We considered a subset of the production traffic for the experiments. Given our network capacity of 100 Gbps, there were very few instances where attack flows had a significant effect on *Queue occupancy* levels.

VI. CONCLUSION

In this paper, we leveraged the INT technology implemented in the Amlight network to detect DDoS attacks. By comparing INT data with sFlow, we first demonstrated that features extracted from INT can be used to accurately detect DDoS attacks in the network. We then implemented an automated DDoS detection mechanism and showed that this mechanism can provide real-time DDoS detection with high accuracy.

For future work, we plan to further explore the use of INT in a limited production environment to identify the requirements needed for implementation in a production network. Our initial goal is to improve the processing capabilities of the DDoS detection system to handle high-volume network traffic. We then aim to eventually scale it up for full implementation in a production network.

ACKNOWLEDGMENT

The AmLight project is supported by the National Science Foundation (NSF Award number OAC-2029283).

REFERENCES

[1] J. Mirkovic and P. Reiher, “A Taxonomy of DDoS Attack and DDoS Defense Mechanisms,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[2] O. I. Falowo, M. Ozer, C. Li, and J. B. Abdo, “Evolving Malware & DDoS Attacks: Decadal Longitudinal Study,” *IEEE Access*, 2024.

[3] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A Survey of Software-defined Networking: Past, Present, and Future of Programmable Networks,” *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, “P4: Programming protocol-independent packet processors,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[5] L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, “In-band Network Telemetry: A Survey,” *Computer Networks*, vol. 186, p. 107763, 2021.

[6] J. Bezerra, I. Brito, A. Quintana, J. Ibarra, V. Chergarova, R. Frez, H. Morgan, M. LeClerc, and A. Paneri, “Deploying per-packet telemetry in a long-haul network: the AmLight use case,” in *2021 IEEE Workshop on Innovating the Network for Data-Intensive Science (INDIS)*. IEEE, 2021, pp. 44–49.

[7] J. Ibarra, J. Bezerra, H. Morgan, L. F. Lopez, D. A. Cox, M. Stanton, I. Machado, and E. Grizendi, “Benefits brought by the use of OpenFlow/SDN on the AmLight intercontinental research and education network,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 942–947.

[8] J. Bezerra, I. Brito, R. Frez, and J. Ibarra, “An adaptive and efficient approach to detect microbursts leveraging per-packet telemetry in a production network,” in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2023, pp. 1–6.

[9] S. Nam, J. Lim, J.-H. Yoo, and J. W.-K. Hong, "Network anomaly detection based on in-band network telemetry with RNN," in *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2020, pp. 1–4.

[10] G. Altangerel and M. Tejfel, "In-network DDoS detection and mitigation using INT data for IoT ecosystem," *Infocommunications Journal: A Publication of The Scientific Association For Infocmmunications (HTE)*, vol. 15, no. SI, pp. 49–54, 2023.

[11] P. Phaal, S. Panchen, and N. McKee, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," Tech. Rep., 2001.

[12] M. Hira and L. Wobker, "Improving network monitoring and management with programmable data planes," 2015, accessed on June 5, 2024. [Online]. Available: http://website-url.com

[13] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013.

[14] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, "Detecting and mitigating DDoS attacks in SDN using spatial-temporal graph convolutional network," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3855–3872, 2022.

[15] 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, https://ieeexplore.ieee.org/abstract/document/8888419/.

[16] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, pp. 447–489, 2019.

[17] N. Aslam, S. Srivastava, and M. Gore, "ONOS flood defender: An intelligent approach to mitigate DDoS attack in SDN," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 9, p. e4534, 2022.

[18] M. Seufert, K. Dietz, N. Wehner, S. Geißler, J. Schüler, M. Wolz, A. Hotho, P. Casas, T. Hoßfeld, and A. Feldmann, "Marina: Realizing ml-driven real-time network traffic monitoring at terabit scale," *IEEE Transactions on Network and Service Management*, 2024.

[19] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted DDoS attack detection with P4 language," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[20] D. Pineda, K. Akkaya, A. Perez-Pons, S. Uluagac, and A. Sahin, "DDoS Attack Detection and Mitigation in 5G Networks using P4 and SDN," *Accepted to The 49th IEEE Conference on Local Computer Networks (LCN) Caen, Normandy, France*, October 8-10 2024.

[21] T. E. Ali, Y.-W. Chong, and S. Manickam, "Machine learning techniques to detect a DDoS attack in SDN: A systematic review," *Applied Sciences*, vol. 13, no. 5, p. 3183, 2023.

[22] M. Najafimehr, S. Zarifzadeh, and S. Mostafavi, "DDoS attacks and machine-learning-based detection methods: A survey and taxonomy," *Engineering Reports*, vol. 5, no. 12, p. e12697, 2023.

[23] Hping Network Tool, "hping3," n.d., software. [Online]. Available: https://www.kali.org/tools/hping3/#hping3

[24] G. Yaltirakli, "Slowloris," *github.com*, 2015. [Online]. Available: https://github.com/gkbrk/slowloris

[25] A. A. Alashhab, M. S. M. Zahid, M. A. Azim, M. Y. Daha, B. Isyaku, and S. Ali, "A survey of low rate ddos detection techniques based on machine learning in software-defined networks," *Symmetry*, vol. 14, no. 8, p. 1563, 2022.

[26] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, and M. Ylianttila, "5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network," *arXiv preprint arXiv:2212.01298*, 2022, https://arxiv.org/abs/2212.01298.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] J. N. Bakker, B. Ng, and W. K. Seah, "Can machine learning techniques be effectively used in real networks against DDoS attacks?" in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–6.

[29] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022.

[30] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic in-band network telemetry," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 662–680.

[31] M. Polverini, S. Sardellitti, S. Barbarossa, A. Cianfrani, P. Di Lorenzo, and M. Listanti, "Reducing the in band network telemetry overhead through the spatial sampling: Theory and experimental results," *Computer Networks*, vol. 242, p. 110269, 2024.