Beyond Static Traffic: A Dynamic and Phase-Aware SDN Framework for Evidence-Grounded Intrusion Detection Dataset Generation

Dr. S.Bharath Bhushan, Ganga Bharath, Pathepur Dikshith, Badveli Guru Pala Nivesh Reddy, and Chitteti Durga Sai

***Abstract*—Research on Intrusion Detection in Software-Defined Networking (SDN) requires traffic datasets that accurately represent changes in network behavior and state. In this work, we introduce a dynamic and phase-aware SDN traffic generation and dataset creation framework through the use of Mininet, Open vSwitch, Ryu, and OpenFlow 1.3. The framework includes support for line and star topology, controlled stable, constrained, and recovery phases, and security-related benign and synthetic workloads. Network telemetry information and timestamped execution logs are used to construct traceable observations.**

H14 dataset is composed of 216 observation intervals partitioned into run-disjoint sets for training, validation, and test. The prospective evaluation for H23 employed 144 additional intervals from 16 experimental runs to compare frozen classification models. Gradient Boosting obtains the maximum balanced accuracy of 0.8958 for probing. H28 network-fidelity evaluation reveals an increase in mean RTT from 6.0508 ms to 32.8780 ms and reduction in TCP Bulk Goodput from 9.2999 Mbps to 1.9001 Mbps while constrained, followed by recovery towards baseline values. The results show that SDN-based traffic generation is controllable, dataset is traceable, and network reaction to phase is measurable. But independent malicious attack validation and generalization to other unknown attacks have not been verified.

***Index Terms*—Software Defined Networking, Intrusion Detection System, Dynamic Traffic Generation, Creation of Dataset, Mininet, Ryu Controller, OpenFlow, Phase Aware Labeling, Network Telemetry, Cybersecurity.**

I. Introduction

THE Software-defined Networking (SDN) is a new networking paradigm that allows for centralization, programmability, and flexibility of network management through the separation of control plane and data plane. OpenFlow protocol enables communication between SDN controllers and forwarders and thus makes network behavior configurable and controllable [1]. The benefits of the architecture of SDN such as programmability, network automation, simplification of network management, and adaptability have been well-studied in literature [2], [3]. Despite these benefits, there is a growing problem of security due to the complexity of programmable networks. Intrusion Detection Systems (IDS) are crucial in detecting any suspicious behaviors and defending SDN infrastructures from network attacks. The centralized view offered by SDN controllers provides means of collecting the flow-level information and detecting any anomalous network activities. The control and data plane could also become targets for various attacks such as the denial of service, access attacks, and abnormal behavior of network traffic [6]. Previous research on SDN included such topics as OpenFlow-based attacks detection, monitoring of traffic anomalies, and secure approaches in SDN networks [7]–[9]. In addition, machine and deep learning techniques were explored in relation to intrusion detection [12], [13]. Nevertheless, any good intrusion detection system requires adequate training and evaluation datasets. Building realistic intrusion detection datasets is a challenging problem. Various datasets such as UNSW-NB15, CICIDS2017, and InSDN have been created for the research in network security [16]–[18]. The dataset generation procedures have put the spotlight on realistic traffic, controlled attacks, and ground truth information [14]. The representation of dynamic environment and time coincidence of the traffic behavior needs more research attention. Limitations such as low data quality, labeling errors, and experiments design have also been described by the existing works [19]–[22]. The listed limitations can influence the validity of the future intrusion detection assessments because of the independence assumption about observations from the traffic sessions. In network environment, there exists dynamics in bandwidth, latency, packet loss, topology, and intensity of traffic. Such dynamics can affect observed network statistics even in the absence of any malicious action. Distinguishing between normal operational dynamics and security-related actions is an important aspect in the construction of datasets for intrusion detection. The literature on concept drift sheds light on problems related to changing distributions in data [29]. In the context of machine learning for network security, it has been previously established that realistic experiments are key to good evaluation results [26], [30]. In order to cope with the challenges listed above, this paper proposes the use of a phase-aware SDN traffic generation and dataset creation framework. The framework employs Mininet, Open vSwitch, and the Ryu SDN controller in order to create a controllable experimental setting. Mininet is used for virtual network experiments [4], while Open vSwitch is used for programmable forwarding [5]. The framework supports several different network topologies, the controllable switch between stable, constrained, and recovery phases, as well as various benign and attack-related traffic scenarios. Network observations are gathered via OpenFlow flow statistics, RTT measurements, UDP traffic monitors, and experiment timestamps.

II. Related Work

Studies on SDN Security have moved from traditional traffic monitoring and rule-based attacks to data-based intrusion detection using machine learning and deep learning approaches. The programmability and centralized view of SDN facilitate the gathering of traffic statistics and the use of network security mechanisms. But the reliability of these mechanisms is highly dependent on the quality of traffic observations, the diversity of network environment, and labeling of dataset [6], [19], [28].

A. From Conventional Intrusion Detection Datasets to Dynamic SDN Traffic Generation

Most of the early work on SDN security focused on traffic monitoring and attack detection through programmable network elements. Braga et al. [7] provided DDoS flooding detection in an OpenFlow-based traffic information context. Mehdi et al. [8] addressed the problem of traffic anomalies detection in SDNs, whereas Giotis et al. [9] used OpenFlow and sFlow to achieve scalable traffic anomalies detection and mitigation. On the other hand, Shin et al. [10] proposed AVANT-GUARD for more efficient flow management and security at the switch level, whereas Wang et al. [11] suggested FloodGuard for mitigating denial-of-service attacks in SDN settings. As a result of the growing use of data-based security approaches, the focus has now shifted to machine learning based intrusion detection. The work of Tang et al. [12] examined the role of deep learning in network intrusion detection in SDN, while the DeepIDS framework [13] extended the scope of deep learning to detect malicious flows. These approaches need appropriate datasets for their training, testing, and generalization evaluation.

There are several benchmark datasets that have been developed for intrusion detection studies. Shiravi et al. [14] offered a systematic approach to the generation of realistic intrusion detection datasets. UNSW-NB15 was introduced by Moustafa and Slay [16], while Sharafaldin et al. [17] provided a dataset generation methodology related to CICIDS2017. The dataset of Elsayed et al. [18] is called InSDN, and it is specifically designed for SDN environment.

Although they have proven useful, the benchmark datasets should be analyzed based on the environment they were created and tagged in. The characteristics of network-based intrusion detection datasets were reviewed by Ring et al. [19]. Errors and issues in tagging associated with dataset were studied by Engelen et al. [20], Liu et al. [21], whereas problems related to experimental design in benchmark dataset were raised by Flood et al. [22]. They indicate the need to investigate the reliability of the ground-truth, measurement, reproducibility of experiment setup.

Moreover, any change in intensity of the traffic flow, network configuration, bandwidth, latency, packet loss can make difference in the observations without any presence of malicious behavior. Learning from concept drift is also discussed in concept drift literature [29]. Hence, conducting experiments that capture the change in both network conditions and traffic behaviors is useful.

The reviewed studies have shown much progress in SDN security monitoring, intrusion detection, and benchmark dataset generation. However, there are still difficulties in modeling dynamic network environment, traffic transitions, and accurate labeling of ground truth. This is why the necessity to design a phase aware SDN dataset construction framework arises.

B. This work

The current work is focused on building an adaptive framework for generating SDN traffic in a phase-aware manner for constructing a dataset.

For implementing the proposed framework, we use the combination of Mininet, Open vSwitch, and Ryu SDN controller. The Mininet tool allows network emulation with high reproducibility [4]. The Open vSwitch tool gives us an opportunity to work with programmable forwarding and implement network operations through OpenFlow [5]. The proposed framework uses line and star network topologies and three network-condition phases: stable, constrained, and recovery.

As opposed to the dataset construction which is done only using one fixed network condition, our approach is based on the combination of the network-condition transitions and traffic scenarios. Our traffic workloads consist of benign communication generation, authorized traffic burst, bounded TCP reconnaissance, UDP traffic simulation, TCP connection churn, and low-rate synthetic probing.

Our framework captures several types of network evidence such as OpenFlow flow statistics, round-trip time, UDP sender and receiver data, and timestamped experiment events. Then we use these evidence in order to label the phases and workload execution intervals.

One aspect of significance in this research paper is the isolation of workload execution evidence from the assumption of successful attacks. The methods like scenario validation, synchronization of timestamps, transition-aware labels, and telemetry data validation are used to enhance the traceability of the generated dataset.

Furthermore, this paper introduces the use of repeating experiment settings using various topologies and random seeds. It enables the planned assessment of intrusion detection models using independent experiments instead of only relying on random combination of observations.

C. Research gap

There have been numerous studies that have built foundations in SDN security monitoring, intrusion detection, and benchmark dataset generation. Nevertheless, there are still some relevant methodological questions regarding the creation of benchmark datasets in dynamic SDN environment.

First of all, it is impossible to conclude automatically on the behavior of intrusion detection algorithms during controllable changes in bandwidth, latency, packet losses, and network topology using existing benchmark datasets. Such datasets can contain multiple types of attacks, but the influence of network conditions is not considered. Next, the time alignment between network switch events and traffic activities needs to be properly considered. The observational windows may overlap some changes to network states or beginnings and endings of

TABLE I
SUMMARY OF RELATED WORK ON SDN INTRUSION DETECTION AND DATASET CONSTRUCTION

S.No	Authors	Year	Paper Title	Aim	ML/DL Model	Dataset	SDN Controller	Attack Type	Evaluation Metrics	Accuracy / Best Result	Limitations
1	Rodrigo Braga, Edjard Mota, Alexandre Passito [7]	2010	Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow	Detect flooding attacks through SDN flow statistics.	OpenFlow-based detection	SDN experimental traffic	NOX	DDoS flooding	Detection performance, traffic statistics	Demonstrated lightweight OpenFlow-based DDoS detection.	Focused on flooding rather than diverse attack behaviors.
2	Syed A. Mehdi, Junaid Khalid, Syed A. Khayam [8]	2011	Revisiting Traffic Anomaly Detection Using Software Defined Networking	Investigate traffic anomaly detection using SDN capabilities.	Anomaly detection methods	Experimental network traffic	SDN/OpenFlow environment	Traffic anomalies	Detection effectiveness	Demonstrated feasibility of SDN-assisted anomaly detection.	Not focused on dynamic dataset construction.
3	Seungwon Shin, Vinod Yegneswaran, Phillip Porras, Guofei Gu [10]	2013	AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks	Improve flow management and network protection.	N/A	Experimental SDN traffic	OpenFlow environment	Flow-based attacks	Flow management, security performance	Demonstrated switch-level security enhancements.	Primarily a protection mechanism rather than an IDS dataset.
4	K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris [9]	2014	Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments	Detect and mitigate network anomalies using combined monitoring.	Flow-based anomaly analysis	Experimental SDN traffic	OpenFlow	Large-scale traffic anomalies	Detection and mitigation scalability	Demonstrated integrated anomaly monitoring and mitigation.	Does not focus on transition-aware dataset labeling.
5	Haopei Wang, Lei Xu, Guofei Gu [11]	2015	FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks	Protect SDN forwarding and control operations against DoS.	N/A	Experimental SDN traffic	OpenFlow environment	DoS attacks	Attack resistance, performance overhead	Demonstrated DoS prevention through SDN extensions.	Focused on attack prevention rather than dataset diversity.
6	Tuan A. Tang, Lotfi Mhamdi, Des McLernon, Syed A. R. Zaidi, Mounir Ghogho [12]	2016	Deep Learning Approach for Network Intrusion Detection in Software Defined Networking	Apply deep learning to flow-based intrusion detection.	Deep Neural Network	NSL-KDD	SDN-oriented architecture	Normal and malicious traffic	Intrusion classification performance	Demonstrated DNN-based detection using six selected features.	Uses a benchmark dataset not generated from dynamic SDN phases.
7	Tuan A. Tang et al. [13]	2020	DeepIDS: Deep Learning Approach for Intrusion Detection in Software Defined Networking	Develop deep learning-based intrusion detection in SDN.	DNN, GRU-RNN	NSL-KDD	POX	Network intrusions	Accuracy, throughput, latency	80.7% DNN accuracy; 90% GRU-RNN accuracy.	Relies on NSL-KDD rather than phase-aware SDN traffic data.
8	Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, Ali A. Ghorbani [14]	2012	Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection	Generate realistic and reproducible intrusion detection datasets.	N/A	Profile-based benchmark traffic	N/A	Multiple attack scenarios	Dataset realism, completeness, reproducibility	Introduced a systematic profile-based dataset-generation approach.	Not designed specifically for OpenFlow telemetry and SDN transitions.
9	Nour Moustafa, Jill Slay [16]	2015	UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems	Construct a modern benchmark containing normal and attack traffic.	Benchmark evaluation	UNSW-NB15	N/A	Nine attack categories	Traffic features, class distribution	Released 2,540,044 records and nine attack categories.	Not specifically constructed for dynamic SDN network conditions.
10	Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani [17]	2018	Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization	Create a realistic intrusion detection benchmark with modern attacks.	Machine learning evaluation	CICIDS2017	N/A	DoS, DDoS, brute force, web attacks, infiltration	Traffic features, classification performance	Produced labeled benign traffic and seven attack categories.	SDN-specific controller telemetry and phase transitions are outside its focus.
11	Mahmoud S. Elsayed, Nhien-An Le-Khac, Anca D. Jurcut [18]	2020	InSDN: A Novel SDN Intrusion Dataset	Develop a benchmark specifically for SDN intrusion detection.	Eight ML-based IDS techniques	InSDN	SDN testbed	Multiple SDN attack categories	Machine learning detection performance	Introduced an SDN-specific dataset and evaluated eight ML techniques.	Controlled phase-wise bandwidth and delay transitions are not its central focus.
12	Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, Andreas Hotho [19]	2019	A Survey of Network-Based Intrusion Detection Data Sets	Compare existing network intrusion detection datasets.	N/A	Multiple existing IDS datasets	N/A	Various attack categories	Dataset suitability criteria	Identified 15 properties for systematic dataset assessment.	Survey study; does not generate new SDN experimental traffic.
13	Gints Engelen, Vera Rimmer, Wouter Joosen [20]	2021	Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study	Identify and correct dataset construction and labeling issues.	Machine learning benchmarking	CICIDS2017	N/A	Recorded benchmark attacks	Label correctness, feature reliability	More than 20% of original traffic traces were reconstructed or relabeled.	Focused on correcting one existing benchmark dataset.
14	Lisa Liu, Gints Engelen, Timothy Lynar, Daryl Essam, Wouter Joosen [21]	2022	Error Prevalence in NIDS Datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018	Investigate errors affecting intrusion detection datasets.	Dataset auditing	CICIDS2017, CSE-CIC-IDS2018	N/A	Multiple recorded attack types	Error prevalence, label reliability	Documented dataset-quality problems affecting evaluation reliability.	Analyzes existing datasets rather than generating dynamic SDN traffic.
15	Robert Flood, Gints Engelen, David Aspinall, Lieven Desmet [22]	2024	Bad Design Smells in Benchmark NIDS Datasets	Investigate dataset design problems and experimental biases.	Dataset audit methods	Seven benchmark NIDS datasets	N/A	Multiple attack categories	Design quality, experimental bias	Identified six design-smell patterns across seven datasets.	Provides dataset audits rather than a dynamic SDN traffic framework.

workloads involved in security activities. Without considering boundaries like these, a single categorization might lead to ambiguity in ground truth. Past analyses of dataset errors and experiment design [20]–[22] have pointed out the need for validating such assumptions.

Thirdly, executing a traffic generating script does not guarantee the desired behavior from the network to be observed. Proper construction of datasets would consider the distinction between the timing and results of the initialization, observation of traffic generated, measurement of network response, and independent observation of effects of an attack on a network when using synthetic scenarios instead of attack workloads.

Lastly, conventional random division of strongly related traffic data might generate optimistic model generalization results. Sommer and Paxson [26] pointed out challenges of realistic network intrusion detection with the use of machine learning. Arp et al. [30] emphasized the need for proper evaluation of computer security.

Based on these facts, the present research work is aimed at analyzing the use of dynamic SDN testing, traffic behavior simulation, ground-truth generation based on evidence, and phase-aware telemetry validation. It is important not to state that previous datasets do not possess all these capabilities, but rather to examine their use in one single transparent and repeatable approach for intrusion detection.

III. PROPOSED DYNAMIC AND PHASE-AWARE SDN FRAMEWORK

This approach will allow generating different and consistent network traffic while maintaining correlation among the network conditions, workload behavior, and ground truth for the intrusion detection. In contrast with the fixed configuration

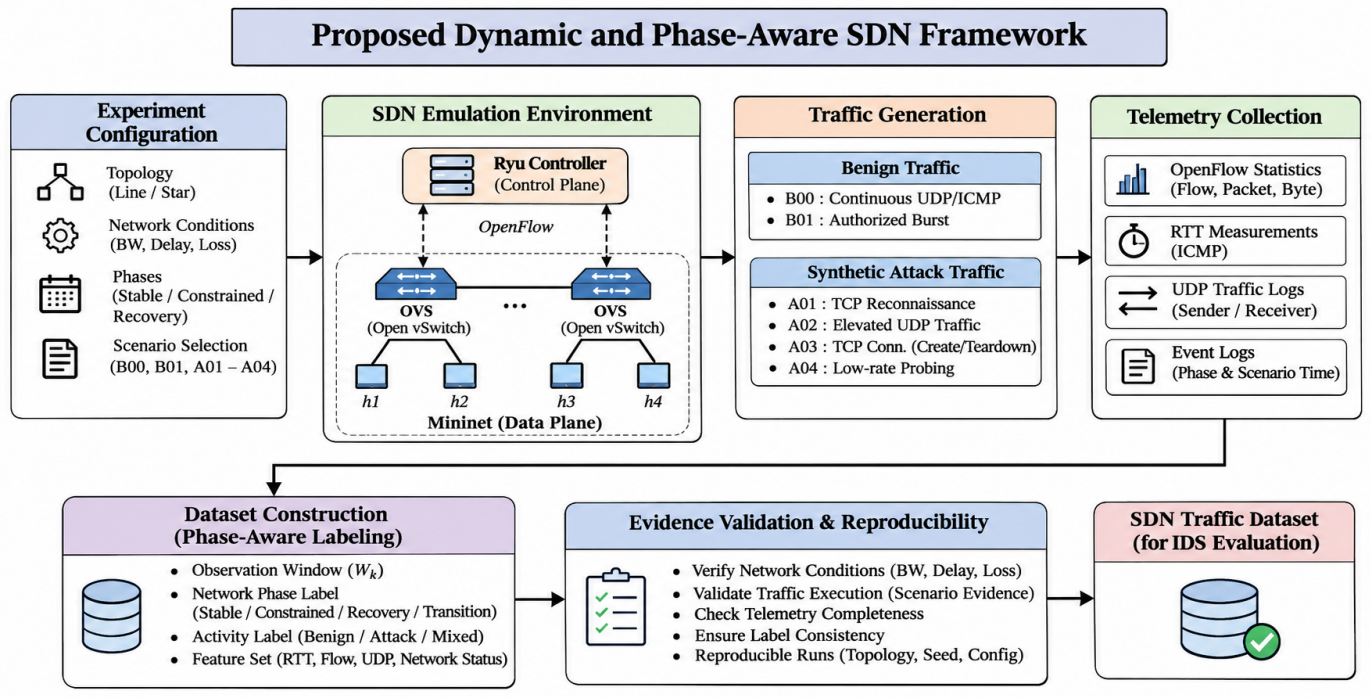


Fig. 1. Proposed Dynamic and Phase-Aware SDN Traffic Generation and Dataset Construction Framework.

approach, this framework takes into consideration the introduction of controlled changes in network topology, bandwidth, propagation delay, packet loss, and traffic volume.

Network emulation, programmable traffic generation, multi-source telemetry collection, temporal data set generation, and evidence-based validation are included in the architecture. They provide a structured procedure to prepare SDN data sets that can be used further for intrusion detection evaluation.

A. Overall Framework Architecture

The suggested architecture includes five components connected among each other: experiment configuration, SDN network emulation, traffic generation, telemetry collection, and dataset generation and validation.

Mininet is utilized to create virtual SDN topologies and Open vSwitch to get OpenFlow-compatible forwarding devices [4], [5]. Forwarding process is managed by the Ryu controller with the use of OpenFlow communication. All experiments are carried out under predefined network conditions, and their results are stored with the use of timestamped experiment events.

The overall framework can be expressed as

$$\mathcal{F} : \mathcal{C} \xrightarrow{\mathcal{E}} \mathcal{N} \xrightarrow{\mathcal{G}} \mathcal{T} \xrightarrow{\mathcal{M}} \mathcal{X} \xrightarrow{\mathcal{L}} \mathcal{D}, \tag{1}$$

where $\mathcal{C}$ represents the experiment configuration, $\mathcal{N}$ the emulated SDN environment, $\mathcal{T}$ the generated traffic, $\mathcal{X}$ the collected telemetry, and $\mathcal{D}$ the constructed dataset. The operators $\mathcal{E}$, $\mathcal{G}$, $\mathcal{M}$, and $\mathcal{L}$ represent network emulation, traffic generation, measurement, and labeling, respectively.

The main components of the framework are illustrated in Fig. 1.

TABLE II
CONTROLLED SDN NETWORK CONDITIONS

Stage	BW (Mbps)	Delay (ms)	Loss (%)
Stable	100	2	0
Constrained	20	30	0.5
Recovery	100	2	0

B. Dynamic SDN Topology and Network Conditions

In order to study network behavior in different conditions of operation, the experimental setup provides two types of network topology: line and star.

The network consists of virtual hosts and OpenFlow-compatible switches connected to the Ryu controller.

The experiments involve three network-condition stages: stable, constrained, and recovery. The stages reflect controlled variations in the network links' parameters and not malicious behavior.

The network-condition state is described by

$$N_p = (B_p, D_p, L_p), \qquad p \in \{S, C, R\}, \tag{2}$$

where B_p is link bandwidth, D_p is configured delay, and L_p is packet loss. Here S, C, and R stand for stable, constrained, and recovery stages, respectively.

The network-condition setup is presented in Table II.

During the stable phase, the network operates under the reference configuration. The constrained phase introduces lower bandwidth, higher delay, and controlled packet loss. The recovery phase restores the original link configuration.

Link parameters are configured through traffic-control mechanisms and checked against their intended settings. This

procedure distinguishes requested network changes from verified network configurations.

C. Benign and Synthetic Attack Traffic Scenarios

Traffic diversity is achieved using six controlled load experiments defined as

$$\mathcal{S} = \{B_{00}, B_{01}, A_{01}, A_{02}, A_{03}, A_{04}\}. \tag{3}$$

The benign loads include constant communication via UDP and ICMP (B_{00}) and traffic bursts (B_{01}). These loads are examples of normal network operations at various levels of traffic.

Four security-related scenarios are bounded synthetic traffic behaviors. A_{01} generates TCP reconnaissance on a limited port range. A_{02} generates controlled elevated UDP traffic. A_{03} involves TCP connection establishment and tear-downs repeatedly, while A_{04} generates probing to a controlled service.

The experiments are carried out in an isolated experimental environment. Metadata for each experiment includes the experiment type, its execution periods, and any generator or endpoint information.

More significantly, there is the distinction between traffic generation evidence and evidence showing a successful network attack. In the ongoing pilot, A_{01} and A_{02} are generator-level evidence, while A_{03} and A_{04} involve endpoint-level verification. The security labels assigned, therefore, indicate controlled synthetic events and not independently demonstrated real world attacks.

D. Multi-Source Network Telemetry Collection

The model incorporates various types of measurement data collected from different sources that describe the network behavior during each experiment.

OpenFlow flow-table statistics include packet and byte counter information. ICMP-based measurements include round-trip time observations with timestamps. UDP sender and receiver log files include application level traffic measurements and experiment event files indicate network phases and workload boundaries.

The feature vector representation for the k^{th} observation window is given by

$$\mathbf{x}_k = [R_k, F_k, P_k, U_k, C_k], \tag{4}$$

where R_k is the set of available RTT features, F_k flow-table statistics, P_k packet and byte counter features, U_k UDP traffic measurements, and C_k network condition metadata.

Each feature set may include several statistics, depending on the available observations in the corresponding window.

Given n_k valid RTT measurements in a window, the average RTT value is given by

$$\overline{RTT}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} RTT_i, \qquad n_k > 0. \tag{5}$$

In the case where no valid RTT sample exists, the respective statistic stays undefined rather than being artificially filled.

UDP goodput from the receiver side is given by

$$G_k = \frac{8B_k^{rx}}{\Delta t_k}, \tag{6}$$

where B_k^{rx} stands for the number of application bytes that were received in the course of an observation period Δt_k. This metric stands for application goodput and must not be confused with physical bandwidth of the link.

Control plane statistics combined with application traffic measurements and network timing data allow for multi-dimensional analysis of SDN behavior. The importance of flow-level monitoring for SDN security applications has also been shown earlier [7], [9].

E. Phase-Aware Dataset Labeling

Temporal labeling must be accurate since one observation window can fall within the scope of several network phases or workload intervals.

Let the observation window be defined as

$$W_k = [t_k, t_k + \Delta t), \tag{7}$$

where t_k denotes the start time of the window and Δt is the length of the window.

The network phase label is found based on the verified phase events. The window that falls within one verified network phase is labeled with that phase. Windows that span phase transitions are explicitly labeled as transitions.

The network phase label can be written as

$$q_k = \begin{cases} p, & W_k \subseteq I_p, \\ \text{TRANSITION}, & W_k \text{ overlaps several phases}, \end{cases} \tag{8}$$

where I_p is the verified time interval of network phase p.

The workload activity label is found based on

$$y_k = \begin{cases} 0, & \text{verified benign interval}, \\ 1, & \text{verified synthetic security interval}, \\ m, & \text{mixed or boundary interval}. \end{cases} \tag{9}$$

In this case, m represents measurements that cannot be uniquely attributed to an activity type. Activity labels are constructed using observed workload intervals instead of changes in traffic intensity only. Even increased traffic of authorized benign activity is classified as such. Measurement quality attributes remain distinct from activity labels. Observable measurement data fall into categories OBSERVED, TRANSITION, or PARTIAL. Such an approach prevents the use of missing telemetry or changes in the condition of the network as evidence of malicious traffic presence. It is well recognized in previous intrusion detection research that reliable labeling and dataset construction is important. [14], [20]–[22]

F. Evidence Validation and Reproducibility

The final criterion ensures that there is consistency between the experimental setup, workload execution, telemetry, and dataset labeling.

All experiments have structured reports and raw data measurements, such as time-stamped event logs, flow data, RTT logs, UDP sender and receiver measurement, and scenario-verification evidence.

The evidence validation process makes sure that the right network phases were applied, that the SDN controller was connected throughout the experiment, that enough measurements were taken, and that the necessary workload conditions were met.

For experiment r, the evidence-consistency indicator is given by

$$V_r = V_r^N V_r^T V_r^E V_r^L, \tag{10}$$

where V_r^N is the indicator of verified network conditions, V_r^T is the indicator of validated telemetry, V_r^E is the indicator of sufficient scenario execution evidence, and V_r^L is the indicator of consistent temporal labels. All indicators represent a binary validation condition.

Consequently, $V_r = 1$ only if all validation conditions are met. An unmet condition means that the run needs to be reviewed.

Reproducibility is achieved through the use of explicit experiment configurations, topology choice, random seed values, run directory separation, and validation reports.

Furthermore, a repeat experiment evaluation procedure will be employed to investigate the generalizability of any downstream intrusion detection techniques across different traffic-generation experiments. The training and test sets will be separated across complete runs of the experiment to minimize the chance of correlated observations in both sets.

This approach aligns with the best practices suggested for the evaluation of machine learning systems in practical security scenarios [26], [30].

In summary, the proposed methodology combines controlled SDN experimentation, diverse traffic generation, phase-based labeling, and evidence-based validation into an end-to-end dataset construction procedure. It focuses on generating traceable network observations that may be used for further intrusion detection experiments.

IV. DATASET CONSTRUCTION AND EXPERIMENTAL METHODOLOGY

The proposed methodology should enable to generate, collect, tag and validate the SDN network traffic under different operational conditions. This process involves a combination of the controlled network setup and workload generation that is either benign or synthetic security-related.

The proposed methodology includes several steps, namely SDN environment configuration, traffic scenario creation, dynamic network phase control, telemetry gathering from various sources, feature extraction, labeling based on the evidences, validation and preparation of the datasets for IDS evaluation.

A. Experimental Environment and Configuration

The experimental environment is set up with the help of Mininet, Open vSwitch, and the Ryu controller. Mininet supplies the virtual network infrastructure, Open vSwitch is used for programmable packet forwarding, while Ryu controls network behavior using OpenFlow 1.3 [4], [5].

Two network topologies, line and star, are used to induce different network structures. Both topologies are configured with virtual hosts and OpenFlow enabled switches.

Three random seeds, 42, 123, and 2026, are provided to facilitate multiple runs of the experiment under controlled conditions.

Experiment configuration is defined by

$$E_r = (T_r, S_r, Z_r, P_r). \tag{11}$$

Here T_r stands for the chosen network topology, S_r is the traffic scenario, Z_r is the random seed, and P_r is the sequence of phases with specific network conditions.

The planned experimental space is composed of six traffic scenarios, two topologies, and three random seeds. Thus, the total number of planned runs is

$$N_{\text{runs}} = N_S \times N_T \times N_Z = 6 \times 2 \times 3 = 36. \tag{12}$$

This 36-run structure is the initially intended experimental design. The campaigns being evaluated are unique: Campaign H14 has 24 runs, Campaign H23 has 16 extra runs, and Campaign H28 has 12 network-fidelity runs. These numbers must not be considered as a single evaluation of 36 runs.

This configuration determines the intended experimental coverage. Whether a run will be included into the final dataset depends on successful validation of the evidence.

B. Traffic Scenario Design

There are six traffic scenarios that are deemed both legitimate network traffic as well as synthetic security behaviors. The labels used for this set of traffic scenarios are B00, B01, A01, A02, A03, and A04.

The B00 traffic scenario creates continuous benign traffic through UDP and ICMP traffic to simulate normal behavior. B01 creates benign UDP bursts without creating malicious activity.

The next set of scenarios is related to traffic scenarios that create security behavior. The A01 creates bounded TCP reconnaissance attacks to certain ports. The A02 creates high UDP traffic in a controlled manner. The A03 creates multiple TCP connection attempts. Finally, A04 creates low TCP probe attempts against the controlled test service.

These traffic scenarios configurations are summarized in Table III.

The experimental workload scenarios are bounded and performed in the isolated SDN testbed. Workload execution logs are recorded for each scenario in order to track the start and end times of individual workloads. When there is only generator-side data available for particular scenarios, this provides the necessary evidence for traffic generation but not for the success of the attack itself. This approach is used in dataset labeling.

TABLE III
CONTROLLED TRAFFIC GENERATION SCENARIOS

ID	Category	Traffic Behavior
B00	Benign	Continuous UDP and ICMP communications
B01	Benign	Authorized UDP traffic burst
A01	Synthetic	Bounded TCP probing
A02	Synthetic	Controlled UDP traffic increase
A03	Synthetic	Repeated TCP connection churn
A04	Synthetic	Low-rate TCP service probing

C. Dynamic Network Phase Execution

Each experiment proceeds through the order of stable, constrained, and recovery phases.

The stable phase gives the reference network conditions. In the constrained phase, the amount of bandwidth is decreased, while the delay and packet loss are increased. The recovery phase allows for restoring the initial network settings.

Let the sequence of network conditions for experiment r be

$$P_r = \{P_S, P_C, P_R\}, \tag{13}$$

where P_S, P_C, and P_R stand for stable, constrained, and recovery conditions, correspondingly.

Each phase has a configured period of its execution,

$$I_p = [t_p^{start}, t_p^{end}), \tag{14}$$

where t_p^{start} and t_p^{end} are the recorded start and end time moments of phase p.

The network parameters are set up via traffic control. The checks of configurations and records of events are used for evaluation of the achieved phase conditions. Time taken by each phase is compared with the configured time for that phase. Significant deviation indicates a need to reconsider the run rather than accepting it unquestioningly.

D. End-to-End Dataset Construction Procedure

Data set creation starts following experiment execution and telemetry acquisition.

First of all, the framework detects what measurements were available from each experiment execution. They are OpenFlow polling measurements, ICMP round-trip time samples, UDP applications measurements, and experiment events timestamps.

Then the measurements are arranged into a dataset based on the common timestamp representation. Observation window is characterized by its starting time and length.

For experiment r, the data set can be represented as

$$\mathcal{D}_r = \{(\mathbf{x}_k, y_k, q_k, u_k)\}_{k=1}^{N_r}, \tag{15}$$

where $\mathbf{x}_k$ is the network features extracted, y_k is the activity label, q_k is the network phase label, u_k is the observation quality, and N_r is the number of windows created.

Each observation also maintains experiment identifier and timestamp so that the connection between observation and its source telemetry could be established.

Data set is comprised of the retained experiments data sets:

$$\mathcal{D} = \bigcup_{r \in \mathcal{R}_{valid}} \mathcal{D}_r, \tag{16}$$

and where $\mathcal{R}_{valid}$ denotes the set of runs that pass verification checks.

The run ID is kept in order to ensure that observations from different runs do not lose context.

E. Telemetry Processing and Feature Extraction

Network telemetry is analyzed for traffic characteristic extraction from an observation window.

RTT values are combined to compute latency statistics. UDP sender and receiver entries are leveraged to assess the application-level traffic pattern.

Flow-table data and packet/byte counts from OpenFlow polling are available. The difference between two valid counts of a cumulative variable provides an estimation of the measurement interval traffic activity.

For a cumulative counter C, the increment between two valid counters is

$$\Delta C_k = C(t_{k+1}) - C(t_k), \tag{17}$$

assuming that both counts refer to the same counter, and there was no reset or discontinuity.

For a valid measurement period, the counter rate is

$$R_k^C = \frac{\Delta C_k}{t_{k+1} - t_k}. \tag{18}$$

Negative counter difference, missing timestamp and discontinuity not confirmed by OpenFlow polling result are reported, instead of interpreting them as a legitimate traffic flow rate.

In the telemetry pre-processing phase, each measurement time window is scanned for presence of any missing observations. Any missing telemetry values are marked as missing data rather than considering them as zeroes in the network traffic [19]–[21], [23].

F. Ground-Truth Labeling and Quality Assessment

Ground-truth labels are determined based on the documented execution intervals of workloads and verified network-phases events.

Window-labeling differentiates between benign network traffic, attack-synthetic activity, and windows having transitional or mixed content.

For window W_k and workload interval I_j, temporal overlap between them is defined as

$$O_{k,j} = \frac{|W_k \cap I_j|}{|W_k|}, \tag{19}$$

where $|W_k|$ denotes duration of the observation window and $|W_k \cap I_j|$ denotes its overlap with the workload interval.

Window which is fully included in one activity interval can be labeled according to the activity, provided that corresponding evidence of execution is available.

Window that overlaps with more than one activity interval is marked as transitional or mixed observation.

Separate quality factor is computed according to the availability and temporal consistency of the measurements:

$$u_k \in \{\text{OBSERVED}, \text{TRANSITION}, \text{PARTIAL}\}. \quad (20)$$

OBSERVED specifies a window with measurements that match the desired specifications subject to validation criteria.

TRANSITION represents a window that has been impacted by changes in the network conditions or is subject to time-based boundaries.

PARTIAL specifies a window for which there is insufficient measurement data.

Network Condition Tag, Activity Tag and Quality Indicator are kept separate. Consequently, an innocent traffic burst will not get labeled as malicious based on the presence of a high traffic rate.

This dataset-labeled approach addresses concerns about dataset labeling reliability and validity highlighted in [20]–[22].

G. Experiment Reproducibility and Validation

Reproducibility is achieved via explicit configuration files, fixed random seeds, network topology identifiers, phase configurations, and output folders separated per each experimental run.

Each run maintains its own telemetry, event logs, and evidence obtained from the scenario verification.

The validation process consists of checking the network configuration, controller connectivity, phase execution, telemetry availability, scenario evidence, and consistency between timestamps and labels.

Validation status of experiment r is defined by the equation:

$$V_r = \begin{cases} 1, & \text{when all required checks pass;} \\ 0, & \text{when one or several checks fail.} \end{cases} \quad (21)$$

Run is considered valid only in case the required checks confirm that it can be used to analyze the dataset.

Runs containing timing, measurement, or scenario evidence problems are kept for further analysis and cannot be considered independently verified just because their collectors executed successfully.

This method ensures traceability between the records of the dataset and their experimental origins.

H. Evaluation Using Run-Disjoint Synthetic Workload

The H14 dataset is split into train, validation and test parts via non-overlapping run identifiers used during experiment runs. Each part includes 72 observations, each from eight different run IDs. Such approach decreases the probability of information leakage due to correlation between observations.

Frozen classifiers were tested on the H23 prospective dataset, including 144 observation intervals from additional 16 experimental runs without retraining or reselecting anything. The assessment process focuses on simulated attacks rather than actual verified malicious attacks.

By $\mathcal{R}_{tr}$, $\mathcal{R}_{val}$, and $\mathcal{R}_{te}$ we will denote run IDs sets of training, validation and test partitions, correspondingly.

Condition of run-disjointness is stated via the following equality:

$$\mathcal{R}_{tr} \cap \mathcal{R}_{val} = \mathcal{R}_{tr} \cap \mathcal{R}_{te} = \mathcal{R}_{val} \cap \mathcal{R}_{te} = \varnothing. \quad (22)$$

The H23 evaluation showed that balanced accuracy, recall, and false positives were obtained. For clarity, precision, recall, and F1 score are defined as To complete the discussion, recall and precision can be defined by

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}, \quad (23)$$

$$F_1 = \frac{2PR}{P + R}. \quad (24)$$

where TP, FP, and FN denote true positive, false positive, and false negative instances, respectively. Precision and F1 score are mentioned only for defining a metric and not as an additional result reported in H23 evaluations. The prospective analysis and run-level separation lead to a better evaluation of synthetic workloads' classification. Yet, these results cannot prove the detection of verified malicious attacks and generalization to unseen families of attacks [26], [27], [30].

V. RESULTS AND DISCUSSION

This part shows the results of the experiments conducted within the context of the designed SDN framework. This framework is characterized by dynamism and phase awareness. The results of the experiment concentrate on the process of dataset creation, classification of artificial probing traffic, and the behavior of the network under controlled conditions.

A. Dataset Construction and Validation

There are 216 observations and 20 attributes in the H14 dataset. The dataset is split into training, validation, and test subsets, where each subset includes 72 observations corresponding to eight different experimental runs.

In each of the subsets there are 60 observations that belong to class 0 and 12 observations that belong to class 1. In total, in the dataset there are 180 class-0 observations and 36 class-1 observations.

There is no overlap between subsets at the experimental run level, which means that observations from one experimental run do not appear in more than one subset. This reduces the risk of information leakage due to correlations in traffic observations.

Class 1 refers to synthetic probing activities rather than a verified malicious act.

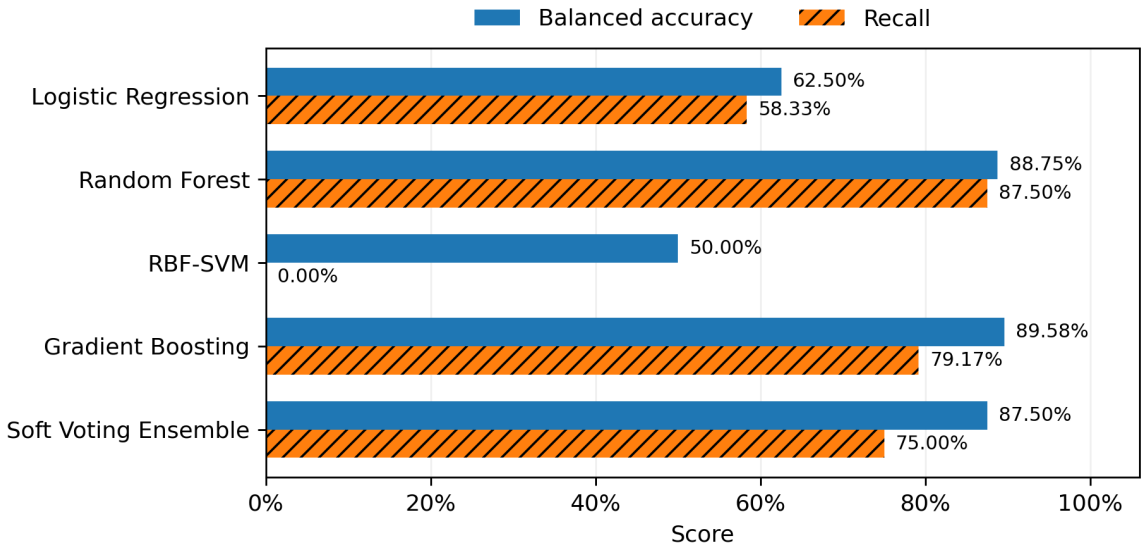


Fig. 2. Comparison of balanced accuracy and recall across five frozen classification models evaluated on H23 synthetic probing data.

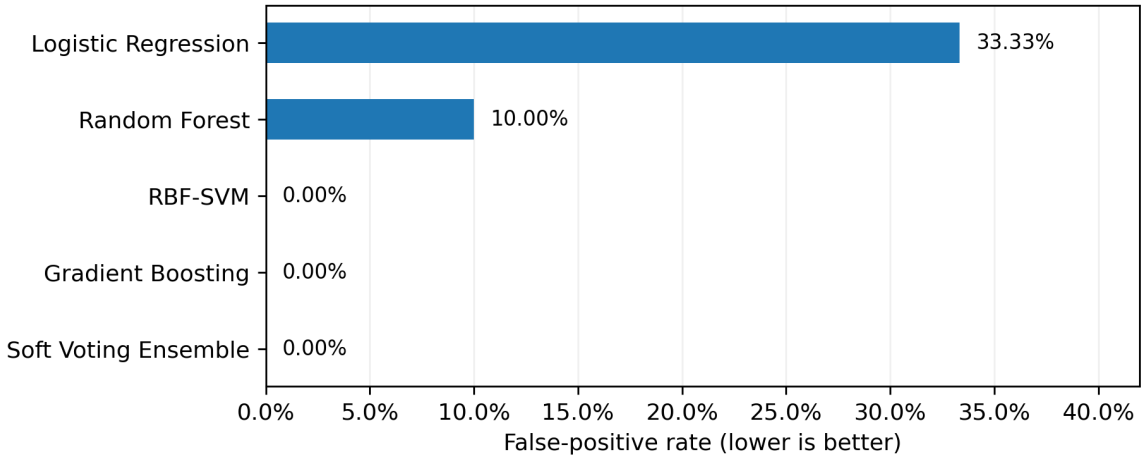


Fig. 3. Comparison of false positive rates across the five frozen classification models in the H23 prospective evaluation.

TABLE IV
PROSPECTIVE FROZEN-MODEL PERFORMANCE ON SYNTHETIC PROBING ACTIVITY

Model	Bal. Acc.	Recall	FPR
Logistic Regression	0.6250	0.5833	0.3333
Random Forest	0.8875	0.8750	0.1000
RBF-SVM	0.5000	0.0000	0.0000
Gradient Boosting	0.8958	0.7917	0.0000
Soft Voting Ensemble	0.8750	0.7500	0.0000

B. Prospective Model Performance

The H23 prospective evaluation was conducted on frozen classification candidates using 144 observation intervals from 16 new runs. These runs were different from those in the original H14 data set. No retraining or reselection was done to evaluate the candidate models.

Table IV summarizes the balanced accuracy, recall, and false positive rate results from the evaluation.

Figures 2 and 3 illustrate the differences in balanced accuracy, recall, and false positive rate among the frozen classification models.

The best performance of Gradient Boosting was achieved at 0.8958 for balanced accuracy, while the highest recall was 0.8750 for Random Forest. The Gradient Boosting and Soft Voting Ensemble did not produce any false positives in the reported prospective evaluation.

The result shows differences between classification techniques when classifying the testing synthetic workload. But, the original winner chosen from the validation process was not noted down in the report. Moreover, data points within one run are correlated, and Statistical significance between the evaluated models has not been established.

C. Comparison with an Existing SDN Dataset Study

The proposed framework is compared with the InSDN dataset study by Elsayed et al. [18]. InSDN stresses the

TABLE V
METHODOLOGICAL COMPARISON BETWEEN INSDN AND THE PROPOSED FRAMEWORK

Criterion	**InSDN [18]**	**Proposed Framework**
Research Focus	SDN Intrusion Detection Dataset Generation	Phase-Aware SDN Dataset Construction
SDN Controller	ONOS	Ryu with OpenFlow 1.3
Testbed	Mininet and Open vSwitch in a virtualized network	Mininet and Open vSwitch with line and star topologies
Number of dataset records	343,939 flows, 83 attributes	216 H14 observations, 20 columns
Types of traffic	Benign traffic and various generated attacks	Benign and constrained synthetic security-relevant workloads
Network scenarios	Stable-constrained-recovery network fidelity protocol not stated	Stable-constrained-recovery configurations explicit
Machine-learning model	Eight machine-learning algorithms with cross-validation	Five frozen models tested in prospective setting on 144 H23 observations
Dataset division	80/20 division and five-fold cross-validation reported	Run-independent train/validation/test divisions
Network performance metrics	Phase-specific RTT and TCP goodput recovery assessment not performed	RTT and TCP goodput were analyzed under 36 phase-specific H28 network environments
Scope of labeling	Labels for generated attack categories	Synthetic workload labels; independent malicious ground truth is not verified

TABLE VI
MEASURED NETWORK BEHAVIOR IN TERMS OF OPERATIONAL PHASES

Phase	**N**	**RTT (ms)**	**Loss (%)**	**Goodput (Mbps)**
Stable	12	6.0508	0.00	9.2999
Constrained	12	32.8780	0.00	1.9001
Recovery	12	6.2758	0.00	9.2628

importance of the construction of a dedicated SDN intrusion detection dataset with various attack types. On the contrary, the current study explores controlled condition transition, run-disjoint evaluation, and evidence-based synthetic workload labeling.

Thus, it appears to be the case of different experimental setups rather than comparative intrusion detection performance. The datasets, observation units, classes of traffic and the evaluation process itself differ between the two studies. Therefore, the classification results are incomparable as performance metrics.

D. Network Fidelity and Phase Transitions

The H28 replication experiment consisted of 12 experimental runs for 36 network-phase conditions. The measurements include stable, constrained, and recovery phases; with 12 conditions observed within each phase.

Table VI shows the resulting measures in terms of measured round-trip time (RTT), ICMP packet losses, and TCP bulk goodput.

As follows from the table, the constrained phase caused an increase in the average measured mean RTT value from 6.0508 ms to 32.8780 ms. Meanwhile, the TCP bulk goodput experienced a decrease from 9.2999 Mbps to 1.9001 Mbps.

After returning to the reference SDN settings, the average RTT value was restored to 6.2758 ms, whereas the TCP bulk goodput reached 9.2628 Mbps. The measured goodput of TCP can thus not be seen as separately calibrated link capacity. Likewise, the lack of ICMP packet losses cannot be seen as establishing perfect loss fidelity, since the number of probes per condition was small.

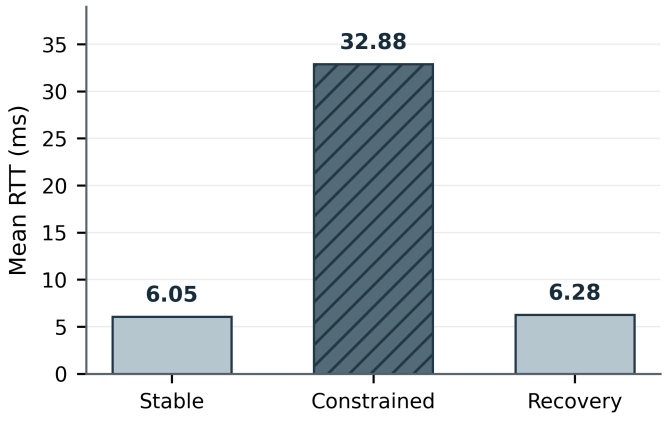


Fig. 4. Measured mean RTT across stable, constrained, and recovery phases in the H28 network-fidelity evaluation.

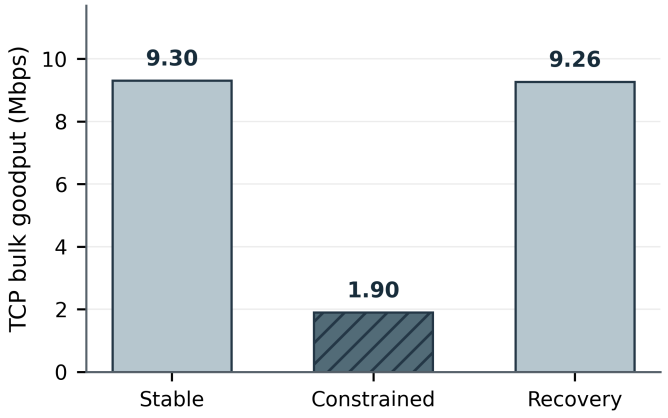


Fig. 5. Measured TCP bulk goodput across stable, constrained, and recovery phases in the H28 network-fidelity evaluation.

E. Discussion and Scientific Limitations

The results of the experiment further prove the feasibility of creating traced traffic observations within an SDN network in a controlled environment. Partitioning and prospective evaluation of run-disjoint data sets provide an additional means for synthetic workload classification evaluation.

Independent verification of ground truth is still incomplete.

The evidence-readiness assessment revealed that all 12 reviewer records were missing adjudication fields. Thus, the classification results presented are relevant only to synthetic probe attacks and cannot prove the detection of verified malicious attacks or generalization to unseen attack families.

In summary, the findings validate a focused experimental contribution to SDN traffic generation, traceable data set construction, and network-fidelity measurement.

VI. Conclusion and Future Work

In this paper, we have developed a dynamic and phase-aware framework for SDN-based generation of traffic and construction of intrusion detection datasets. The framework uses Mininet, Open vSwitch and Ryu to create network traffic under different operational scenarios. The framework is capable of handling multiple network topologies, benign and artificial security-related workloads, multi-source telemetry and phase-aware dataset labeling.

The results of the evaluation show that the proposed approach is feasible under controlled lab conditions. The H14 dataset included 216 observation periods divided into mutually exclusive training, validation, and test sets. Evaluation on 144 additional H23 periods showed differences between frozen classification models for synthetic probing activity.

Network fidelity experiments also showed that constrained network scenarios led to the increase of round-trip time and reduction of TCP bulk goodput. Further recovery measurements were approaching the original stable-phase observations.

However, the current results are limited to controlled synthetic workloads. Independent ground truth for malicious attacks has not yet been created, and generalization to previously unseen malicious attack families has not been demonstrated.

Future works will include independent ground truth adjudication, additional validation of attacks families, service impact measurement, and broader experiments in different network conditions and topologies. Evaluation in future will include the examination of the generalization and statistical robustness of intrusion detection models in use of independently validated data sets. Overall, the proposed framework offers a structured basis for conducting experiments to create traceable SDN traffic data sets.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, *OpenFlow: Enabling innovation in campus networks*. ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[2] 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 Commun. Surveys Tuts., vol. 16, no. 3, pp. 1617–1634, 2014. [Online]. Available: https://doi.org/10.1109/SURV.2014.012214.00180

[3] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, *Software-defined networking: A comprehensive survey*. Proc. IEEE, vol. 103, no. 1, pp. 14–76, 2015. [Online]. Available: https://doi.org/10.1109/JPROC.2014.2371999

[4] B. Lantz, B. Heller, and N. McKeown, *A network in a laptop: Rapid prototyping for software-defined networks*. In Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw., 2010. [Online]. Available: https://doi.org/10.1145/1868447.1868466

[5] B. Pfaff et al., *The design and implementation of Open vSwitch*. In Proc. 12th USENIX Symp. Networked Syst. Design Implement. (NSDI), pp. 117–130, 2015. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff

[6] S. Scott-Hayward, S. Natarajan, and S. Sezer, *A survey of security in software defined networks*. IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 623–654, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2453114

[7] R. Braga, E. Mota, and A. Passito, *Lightweight DDoS flooding attack detection using NOX/OpenFlow*. In Proc. 35th IEEE Conf. Local Comput. Netw. (LCN), pp. 408–415, 2010. [Online]. Available: https://doi.org/10.1109/LCN.2010.5735752

[8] S. A. Mehdi, J. Khalid, and S. A. Khayam, *Revisiting traffic anomaly detection using software defined networking*. In Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, vol. 6961, pp. 161–180, Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-23644-0_9

[9] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, *Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments*. Comput. Netw., vol. 62, pp. 122–136, 2014. [Online]. Available: https://doi.org/10.1016/j.bjp.2013.10.014

[10] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, *AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks*. In Proc. ACM SIGSAC Conf. Comput. Commun. Security, pp. 413–424, 2013. [Online]. Available: https://doi.org/10.1145/2508859.2516684

[11] H. Wang, L. Xu, and G. Gu, *FloodGuard: A DoS attack prevention extension in software-defined networks*. In Proc. 45th IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), pp. 239–250, 2015. [Online]. Available: https://doi.org/10.1109/DSN.2015.27

[12] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, *Deep learning approach for network intrusion detection in software defined networking*. In Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM), 2016. [Online]. Available: https://doi.org/10.1109/WINCOM.2016.7777224

[13] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. El Moussa, *DeepIDS: Deep learning approach for intrusion detection in software defined networking*. Electronics, vol. 9, no. 9, Art. no. 1533, 2020. [Online]. Available: https://doi.org/10.3390/electronics9091533

[14] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, *Toward developing a systematic approach to generate benchmark datasets for intrusion detection*. Comput. Secur., vol. 31, no. 3, pp. 357–374, 2012. [Online]. Available: https://doi.org/10.1016/j.cose.2011.12.012

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, *A detailed analysis of the KDD CUP 99 data set*. In Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications, pp. 1–6, 2009. [Online]. Available: https://doi.org/10.1109/CISDA.2009.5356528

[16] N. Moustafa and J. Slay, *UNSW-NB15: A comprehensive data set for network intrusion detection systems*. In Proc. Military Commun. Inf. Syst. Conf. (MilCIS), pp. 1–6, 2015. [Online]. Available: https://doi.org/10.1109/MilCIS.2015.7348942

[17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, *Toward generating a new intrusion detection dataset and intrusion traffic characterization*. In Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), pp. 108–116, 2018. [Online]. Available: https://doi.org/10.5220/0006639801080116

[18] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, *InSDN: A novel SDN intrusion dataset*. IEEE Access, vol. 8, pp. 165263–165284, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3022633

[19] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, *A survey of network-based intrusion detection data sets*. Comput. Secur., vol. 86, pp. 147–167, 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.06.005

[20] G. Engelen, V. Rimmer, and W. Joosen, *Troubleshooting an intrusion detection dataset: The CICIDS2017 case study*. In Proc. IEEE Security Privacy Workshops (SPW), pp. 7–12, 2021. [Online]. Available: https://doi.org/10.1109/SPW53761.2021.00009

[21] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, *Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018*. In Proc. IEEE Conf. Commun. Network Security (CNS), pp. 254–263, 2022. [Online]. Available: https://doi.org/10.1109/CNS56114.2022.9947235

[22] R. Flood, G. Engelen, D. Aspinall, and L. Desmet, *Bad design smells in benchmark NIDS datasets*. In Proc. IEEE 9th European Symp. Security Privacy (EuroS&P), pp. 658–675, 2024. [Online]. Available: https://doi.org/10.1109/EuroSP60621.2024.00042

[23] M. Sarhan, S. Layeghy, and M. Portmann, *Towards a standard feature set for network intrusion detection system datasets*. Mobile Netw. Appl., vol. 27, pp. 357–370, 2022. [Online]. Available: https://doi.org/10.1007/s11036-021-01843-0

[24] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, *Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset*. Future Gener. Comput. Syst., vol. 100, pp. 779–796, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.05.041

[25] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, *CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment*. Sensors, vol. 23, no. 13, Art. no. 5941, 2023. [Online]. Available: https://doi.org/10.3390/s23135941

[26] R. Sommer and V. Paxson, *Outside the closed world: On using machine learning for network intrusion detection*. In Proc. IEEE Symp. Security Privacy, pp. 305–316, 2010. [Online]. Available: https://doi.org/10.1109/SP.2010.25

[27] A. L. Buczak and E. Guven, *A survey of data mining and machine learning methods for cyber security intrusion detection*. IEEE Commun. Surveys Tuts., vol. 18, no. 2, pp. 1153–1176, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2494502

[28] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, *Survey of intrusion detection systems: Techniques, datasets and challenges*. Cybersecurity, vol. 2, Art. no. 20, 2019. [Online]. Available: https://doi.org/10.1186/s42400-019-0038-7

[29] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, *A survey on concept drift adaptation*. ACM Comput. Surveys, vol. 46, no. 4, Art. no. 44, 2014. [Online]. Available: https://doi.org/10.1145/2523813

[30] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, *Dos and don'ts of machine learning in computer security*. In Proc. 31st USENIX Security Symp., pp. 3971–3988, 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp