Improving Trust and Detecting Malicious Data Sharing in SDN Data Plane Communications using Lightweight Deep Learning

Brahim Anegdouil*, Anass Sebbar*, *Senior Member, IEEE*, Othmane Cherqi*, Mohammed Boulmalf*
*International University of Rabat, TICLAB, Morocco
Email: { brahim.anegdoul, anass.sebbar, othmane.cherqi, mohammed.boulmalf}@uir.ac.ma

***Abstract*—Software-Defined Networking (SDN) introduces flexibility by decoupling control and data planes. However, this separation exposes the data plane to new threats, particularly malicious data sharing between network nodes. Existing trust mechanisms are often inadequate or too resource-intensive for edge environments. This paper proposes a novel framework to enhance trust and detect malicious data sharing within the SDN data plane. The approach combines a dynamic trust model-based on Ability, Benevolence, and Integrity (ABI)-with a lightweight deep learning anomaly detection system leveraging Autoencoders (AEs) and Graph Neural Networks (GNNs). Designed for integration with P4-programmable switches, the framework enables real-time analysis and response directly at the data plane. Key contributions include: (1) a tailored ABI-based trust model for SDN data planes, (2) a lightweight distributed detection architecture using AEs/GNNs, and (3) a P4-based testbed design for validation. This work aims to significantly improve trust establishment, accurately detect malicious data sharing, and minimize resource overhead, paving the way for more secure and resilient SDN data planes. The results demonstrate 97% accuracy in detecting anomalies and highlight the critical challenges of classifying compromised nodes under class imbalance.**

***Index Terms*—Lightweight Deep Learning, Anomaly Detection, Trust Management, P4, Malicious Data Sharing, Software-Defined Networking (SDN), Data Plane Security.**

I. Introduction

Software-Defined Networking (SDN) enhances network management by separating the control plane from the data plane [1]. While this offers flexibility, it introduces security challenges, especially for the data plane, which processes every packet [2]. This paper addresses malicious data sharing within the SDN data plane, where compromised elements might inject false telemetry, manipulate inter-switch statistics, or exfiltrate data [2]. Such actions can distort the controller's network view and compromise network integrity. A key challenge is the lack of granular, adaptive, and lightweight trust mechanisms for data plane entities.

The need for real-time, data plane-integrated security is paramount [2]. Edge computing and IoT proliferation expand the attack surface but also offer opportunities for distributed detection if models are lightweight [3], [4]. Traditional security approaches are often ineffective against sophisticated attacks [5]. Lightweight Deep Learning (DL) offers a promising path for intelligent, efficient detection on resource-constrained data plane devices [4], [6]. Increased data plane programmability, e.g., with P4 [2], enables richer security features but also subtle manipulations, demanding smarter trust and detection mechanisms.

This paper proposes a framework to bolster trust and detect malicious data sharing in the SDN data plane. Our contributions are:

1) A novel dynamic trust model for SDN data plane communications, quantifying node reliability based on Ability, Benevolence, and Integrity (ABI) criteria.
2) A lightweight DL detection architecture using Autoencoders (AEs) for anomaly identification and Graph Neural Networks (GNNs) to model entity interactions and trust propagation, designed for edge deployment.
3) Integration of this framework within P4-programmable switches for real-time packet analysis and decision-making directly in the data plane.
4) Design of a realistic testbed using Mininet, Open vSwitch (OVS), BMv2 P4 software switch, and OpenDaylight controller for validation.

Our primary objective is to demonstrate the feasibility of integrating an ABI-based trust model with lightweight DL detection in a P4-enabled SDN. The GNN results are a proof-of-concept rather than optimized detection. Securing intra-data plane exchanges is vital for preventing direct attacks and maintaining overall SDN operational integrity, as control and management functions rely on trustworthy data plane information. The remainder of this paper is organized as follows: Section II reviews related work. Section III details the system model and problem formulation. Section IV describes the proposed lightweight DL-based detection framework. Section V presents the testbed architecture. Section VI discusses expected results and evaluation metrics. Section VII concludes the paper.

II. Related Work

Research on data-plane security in SDN exposes a broad attack surface, including flow-table saturation, packet injection, and topology manipulation [2], [7]. Solutions like SDNsec employ cryptographic forwarding accountability [8], but most focus on specific threats (e.g., DDoS [3]) or secure controller-switch channels. Consequently, trust in intra–data-plane interactions and subtle data manipulations between forwarding devices remain underexplored [9].

Trust management has been well studied (e.g., Mayer's ABI framework [10]), yet SDN research often addresses

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

application-to-controller trust [11] or controller-to-controller trust. Dynamic, fine-grained trust models for switch-to-switch exchanges are lacking. We build on IoT trust-attribute decomposition [12] to quantify trust based on observed switch behavior.

Lightweight deep learning (DL) advances make edge AI more feasible for constrained SDN data planes. Conventional DL is resource-intensive [4], so recent work emphasizes pruning, quantization, and efficient architectures (MobileNets, pruned autoencoders, GNNs) [6]. Edge solutions reduce latency and reliance on central servers [13]. Examples include LUCID, a CNN for DDoS detection [3], and LENS-XAI, which uses variational autoencoders with distillation for intrusion detection [14].

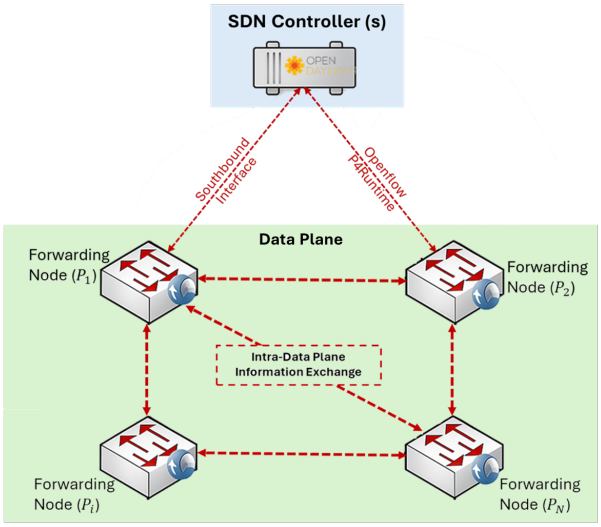

Figure 1: Illustration of intra–data plane communication and data sharing between programmable switches in an SDN environment.

Figure 1 illustrates how malicious behavior can propagate via data exchanges between switches-an aspect largely overlooked in prior work. Modeling malicious data sharing as anomaly detection draws on techniques from malicious URL/content detection [5], [15], covert channel detection [16], and contextual anomaly detection with GNNs [17]. Autoencoders learn normal patterns and detect deviations effectively [18].

Programmable data planes (P4) enable in-switch packet processing, monitoring, and dynamic policy enforcement [2], [19]. Prior P4-based ML integrations P4-ONIDS for DDoS [20], P4-HLDMC for ARP detection [21] focus on external attacks rather than intra-plane trust.

Few efforts combine trust modeling, lightweight anomaly detection, and programmable data planes in a unified SDN framework. Our work addresses this by (1) applying the ABI trust model to intra-plane exchanges; (2) integrating lightweight AEs and GNNs for edge-compatible detection; (3) leveraging P4 for real-time monitoring and mitigation; and (4) designing trust and anomaly modules to reinforce each other. Table I contrasts our framework with existing solutions.

III. System Model and Problem Formulation

We formalize SDN data plane interactions, our trust model, MDS attack types, and the lightweight AI detection task.

A. SDN Data Plane Communication Model

The SDN data plane is a set of N P4-capable switches $P = \{P_1, \ldots, P_N\}$ linked by L_{ij}. Each exchange D_k between P_i and P_j carries user traffic plus control/telemetry (e.g., aggregated flow stats, state updates, signaling). We denote its characteristics by $\mathcal{I}(D_k)$-content type, volume, and frequency. In practice, some programmable switch implementations embed control-like telemetry alongside user data for efficiency, which motivates modeling D_k as potentially containing both types of information.

B. Trust Model for Data Plane Entities

Each switch P_i has a trust score $T_i(t)$ based on the ABI framework [10]. Define:

$$T_i(t) = \alpha\, A_i(t) + \beta\, B_i(t) + \gamma\, I_i(t), \quad \alpha + \beta + \gamma = 1,$$

updated as

$$T_i(t) = (1 - \lambda)\, T_i(t-1) + \lambda\, \mathrm{TrustFromObs}(\mathrm{Obs}_i(t)),$$

where λ is a learning rate and Obs_i are recent observations. Here:

- A_i: Ability-correct forwarding and data sharing (e.g., P4 rule compliance, packet latency, error rates) [8].
- B_i: Benevolence-cooperative behavior (e.g., protocol participation, resource fairness) [11].
- I_i: Integrity-absence of telemetry falsification, consistent reports (checked via cross-checks, anomaly detection) [12].

The following elements summarize the operational metrics and data sources:

- **Ability:** P4 rule compliance, packet latency, Tx/Rx error rate (from P4 counters, packet metadata, error logs).
- **Benevolence:** Protocol participation, resource-sharing behavior (from event logs, policy-compliance checks).
- **Integrity:** Telemetry falsification rate, report consistency (via integrity checks, statistical anomaly outputs).

C. Attack Model for Malicious Data Sharing (MDS)

An adversary A can compromise $P_A \subset P$ to disrupt operations, exfiltrate data, or distort trust. We consider four MDS types:

- **MDS_inject**: P_A sends falsified data (e.g., fake stats) to neighbors [2].
- **MDS_modify**: P_A intercepts and alters a legitimate D_k on-path.
- **MDS_covert**: P_A hides data in non-critical fields or timing channels [16].
- **MDS_trust_manip**: P_A issues false trust reports about honest switches.

Table I: Comparison of Existing Approaches in SDN Security, Trust, and Anomaly Detection

Reference	Main Focus	Proposed Technique/Model	Key Advantages	Limitations/Gaps Relative to Our Study
[2]	SDN Data Plane Sec.	Review of data plane attacks/detections	Detailed threat classification	No integrated trust + lightweight DL solution.
[8]	SDN Data Plane Sec.	Cryptographic forwarding accountability	Path validation, tamper protection	Cryptographically expensive, no behavioral trust.
[11]	SDN Trust Mgmt.	Controller-Application trust framework	Formal inter-plane trust modeling	No intra-data plane trust.
[3]	Lightweight DL NetSec	Lightweight CNN for DDoS detection	Low overhead, high DDoS accuracy	DDoS-specific, no general malicious data sharing or trust.
[14]	Lightweight DL NetSec	VAE + Knowledge Distillation for NIDS	Lightweight, explainable, for constrained env.	No trust focus, no P4 data plane integration.
[17]	Anomaly Detection	GNN for relational anomalies	Captures relational structure	GNN models can be heavy if not optimized.
[20]	P4 for Security	P4-based NIDS pre-filtering	Reduces NIDS load	Focus on pre-filtering for external NIDS, not intra-plane detection/trust.
Our Study	Trust & Malicious Sharing Detection in SDN Data Plane with Lightweight DL	ABI Trust Model + Lightweight AE/GNN + P4 Integration	Holistic intra-data plane trust & detection for edge	Integration complexity, trust quantification.

D. Lightweight AI Model Formulation

To detect MDS, we form an input vector for each D_k from P_i to P_j:

$$X_{data} = \big[T_i(t),\, T_j(t),\, \text{features}(\mathcal{I}(D_k)),\, \text{contextual\_features}(t)\big].$$

Here, `contextual_features` include topology-related metrics (e.g., hop count, neighbor trust scores), temporal trends (e.g., exchange frequency variation), and role-based attributes (e.g., whether a switch acts as a core or edge device).

1) **Autoencoders (AEs)** [14], [18]: Train on benign X_{normal} to learn encoding g and decoding h, with reconstruction loss

$$\mathcal{L}_R(X_{data}, \hat{X}_{data}) \;=\; \big\|X_{data} - h(g(X_{data}))\big\|_2^2.$$

Flag X_{data} as malicious if $\mathcal{L}_R > \theta_{AE}$, where node trust can adjust θ.

2) **Graph Neural Networks (GNNs)** [17]: Represent the data plane as graph $G = (V, E)$, with node features x_v (including T_v) and edge features $x_{e_{uv}}$. A GNN (e.g., GraphSAGE) computes embeddings $h_v^{(l)}$ over L layers to classify edges (exchange legitimacy) or nodes (trust updates).

IV. Proposed Lightweight DL-based Detection Framework

Figure 2 illustrates our framework that integrates trust management and anomaly detection within the SDN data plane using P4 a lightweightDL.

A. Proposed Secure P4 Dataplane Architecture

In this section, we illustrate our proposed architecture based on these key modules:

1) **P4 Monitoring & Feature Extraction Module**: Deployed on P4 switches for real-time packet inspection and feature extraction.
2) **Trust Computation Module**: Distributed or on an edge agent, updates node trust scores (ABI) based on P4-collected observations.
3) **Lightweight DL Detection Module**: Implements AEs/GNNs using extracted features and trust scores. Hosted on switch CPU, dedicated edge device, or hybridly [13], [22].
4) **Decision & Mitigation Module**: Initiates mitigation based on detection alerts and trust scores, locally via P4 or coordinated by the SDN controller.

While P4 extracts features at line rate, AE/GNN inference runs on the switch CPU or a nearby edge device, not in the ASIC pipeline. This ensures real-time feasibility without exceeding data-plane resources. Interaction between the P4 data plane, the OpenDaylight controller, and detection/trust agents is crucial.

B. P4-based Data Monitoring and Feature Extraction at the Edge

P4 enables real-time monitoring and feature extraction on switches. Data monitored by P4: Flow statistics packet/byte counts, duration; data plane-specific metadata from custom headers telemetry; partial packet content lightweight sampling; trust-related events via digests for high error rates. Features for DL models (see Table II): For AEs, $X_{AE} = [T_S, T_D, \text{DataType}, \text{Volume}, \text{Frequency}, \dots]$. For GNNs, node features x_v (e.g., ABI scores, processing capacity) and edge features $x_{e_{uv}}$ (e.g., exchange volume, predominant data type). Lightweight preprocessing (normalization, discretization) can be done in P4 or a nearby agent.

Table II outlines key features extracted for use in Deep Learning (DL) models, specifically detailing their characteristics and roles. The features, such as SourceTrustScore, ExchangedDataType, TotalExchangeVolume, and P4TableUtilSource, are derived from network elements like a "Trust Module" and P4 programmable data planes. Each feature's description, data

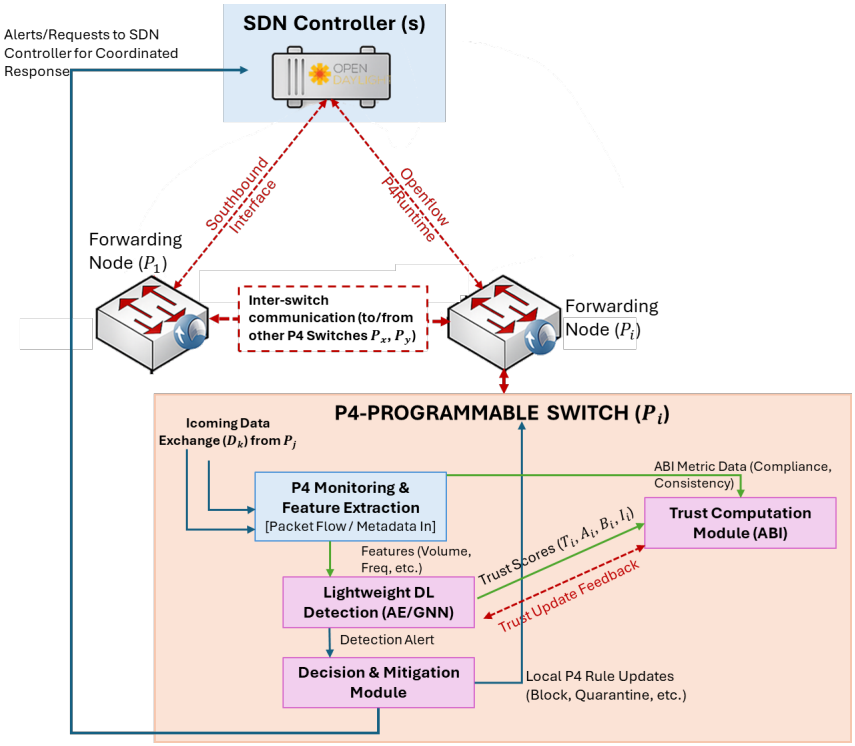

Figure 2: Proposed secure collaborative P4 SDN dataplane architecture

Table II: Condensed Extracted Features for DL Models

Feature Name	Description	Role / Data Type
SourceTrustScore	Aggr. ABI trust (src node)	AE/GNN Input / Numeric
DestTrustScore	Aggr. ABI trust (dst node)	AE/GNN Input / Numeric
ExchangedDataType	Shared info. category	AE/GNN Input / Categorical
TotalExchangeVolume	Total bytes/exch. (window)	AE/GNN Input / Numeric
ExchangeFrequency	Exch./pkts per unit time	AE/GNN Input / Numeric
AvgPacketSize	Avg. pkt size/exch.	AE Input / Numeric
InterArrivalTimeVar	Inter-arrival time variance	AE Input / Numeric
P4TableUtilSource	% P4 table util. (src)	GNN Input (node feat.) / Num./Bin.
LocalRuleCompliance	Local P4 policy compliance	AE/GNN Input / Bin./Num.

type (Numeric, Categorical, Binary), and source (e.g., P4 counters, agent calculations) are provided. Critically, the table specifies how these features serve as inputs, distinguishing between those used by AE components, GNN components (sometimes as node features), or both, thereby guiding the data pipeline for these machine learning architectures.

C. Anomaly Detection with Lightweight Deep Learning

DL architecture choices must respect resource constraints.

1) **Autoencoders (AEs)**: Suited for learning "normal" representations and detecting deviations [14], [18]. Advantages: Unsupervised/semi-supervised, good for unknown anomalies, can be very lightweight (few layers/neurons). Simple AEs or VAEs with weight quantization are options.
2) **Graph Neural Networks (GNNs)**: Data plane interactions are naturally graph-like [17]. GNNs capture complex node dependencies and trust/data propagation. Advantages: Explicit relational structure modeling, identifies subtle context-based anomalies. Lightweight architectures like GraphSAGE or simplified GCNs with graph pruning/quantization are suitable.

Trust scores T_i are crucial inputs for DL models or can modulate detection thresholds. Anomaly detection, in turn, negatively updates trust scores. Deployment can be on switch CPUs [22], dedicated edge devices [13], or hybrid.

D. Distributed Mitigation Strategies

Upon detecting malicious data sharing, rapid mitigation is needed. Actions (via P4 and/or SDN controller): 1. Isolate Node/Flow: Block or sinkhole traffic via P4 rules [23]. 2. Dynamic P4 Rule Updates: Controller pushes new/modified P4 rules [2]. 3. Quarantine & Deep Analysis: Clone suspect flows (in P4) for offline analysis [11]. 4. Severe Trust Score Adjustment: Immediately degrade trust of malicious entities. Mitigation can be local (by switch), cooperative (by switch group), or controller-coordinated.

V. Proposed Testbed and Experimental Setup

We propose an emulated testbed utilizing standard SDN/P4 tools to evaluate our approach. This section details its architecture, implementation specifics, and the scenarios designed for its evaluation.

A. Testbed Architecture

This subsection outlines the architecture of our emulated testbed, detailing the key software and hardware components

used to construct a realistic SDN/P4 network environment suitable for experimentation. The core components involved are as follows:

- **Mininet** [24]: Emulates custom SDN topologies of hosts, links, and switches (v2.3.0 with varied topologies).
- **Open vSwitch (OVS)** [1]: Software switch for non-P4 parts, supporting OpenFlow.
- **P4 Switches (BMv2)** [25]: The `bmv2_simple_switch_grpc` model executes custom P4 programs for monitoring, feature extraction, and mitigation.
- **SDN Controller (OpenDaylight)** [26]: Manages network configuration, interacting with OVS via OpenFlow and BMv2 via P4Runtime [27].
- **Edge Detection/Trust Agents**: Python processes on Mininet hosts or switch CPUs hosting lightweight DL models (AE/GNN) and trust modules. Models are trained offline and deployed in compact form for real-time inference.

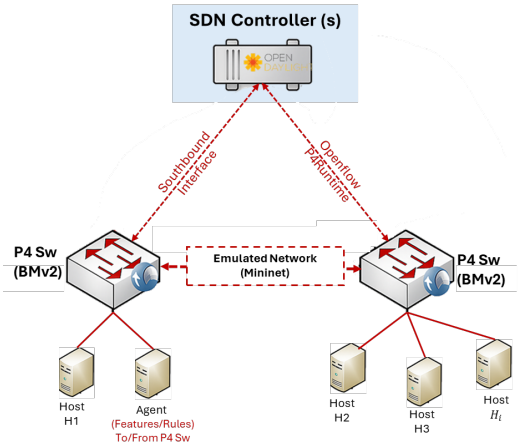

Figure 3: Proposed emulated testbed architecture.

Figure 3 represents the architecture of the emulated experimental testbed. This setup is orchestrated by an OpenDaylight SDN controller, which oversees and manages the data plane. The data plane itself is constructed using P4-programmable BMv2 software switches, interconnected through an "Emulated Network (Mininet)" that also hosts multiple end-user devices (hosts). Communication and control from the SDN controller to these BMv2 switches are facilitated via a southbound interface, prominently featuring P4Runtime for P4-specific operations. A key aspect highlighted is the interaction between a host (presumably running an edge agent) and an adjacent P4 switch, indicated by the "Features/Rules To/From P4 Sw" data flow, which is central to the distributed intelligence and response mechanisms of the system.

B. Malicious Data Sharing Scenarios

To test our system under adversarial conditions, we simulate three MDS scenarios:

- **False Telemetry Injection:** A compromised BMv2 P4 switch sends falsified flow stats (e.g., inflated counts, fake latencies) to neighbors or the controller. Key factors include the malicious source and type of falsification.
- **Covert Exfiltration:** A compromised BMv2 embeds sensitive data in custom P4 header fields within benign flows [16]. We vary which fields carry data and the exfiltration rate.
- **DoS on Information Sharing:** A malicious node floods P4 digest queues on a target switch or agent, disrupting signaling for detection and trust updates. Parameters include attacker identity, target, and queue type.

Legitimate background traffic is generated with `iperf` and `hping3`, while malicious patterns are scripted in Scapy.

VI. Results and Evaluation Metrics

This section presents the evaluation of our two models: an AE for anomaly detection and a GNN for node classification. Standard classification metrics - accuracy, precision, recall and F1 score - are used for performance evaluation, along with confusion matrix analysis.

A. Autoencoder-Based Anomaly Detection

The AE was trained for 50 epochs, and its training and validation loss curves demonstrated stable convergence without signs of overfitting. On the test set, the AE achieved an overall accuracy of 97%.

As shown in Table III, the model performs particularly well in identifying benign traffic ('Normal'), achieving precision and recall scores of 0.98 and 0.99, respectively. For malicious traffic ('Anomaly'), the AE obtained a precision of 0.87 and a recall of 0.84, indicating good - but slightly weaker - performance for rare or subtle anomalies.

Figure 4 confirms these observations, showing that 840 anomalies were correctly detected out of 1000, with 160 false negatives and 121 false positives. The weighted average F1-score of 0.97 confirms the model's strong overall effectiveness.

Table III: Autoencoder Anomaly Detection Performance.

Class	Precision	Recall	F1-Score	Support
Normal	0.98	0.99	0.99	10000
Anomaly	0.87	0.84	0.86	1000
Accuracy			0.97	11000
Macro Avg	0.93	0.91	0.92	11000
Weighted Avg	0.97	0.97	0.97	11000

B. Graph Neural Network for Node Classification

The GNN was trained for 10 epochs and achieved a final test accuracy of 93.33%. As shown in Table IV, the model performed well in identifying 'Not Compromised' nodes, with perfect recall (1.00) and high precision (0.93), yielding an F1-score of 0.97.

However, its performance on the minority class ('Compromised') is critical. The model failed to detect any compromised nodes, with precision, recall, and F1-score all equal to 0.00. The confusion matrix confirms this result: both compromised nodes in the test set were misclassified as benign.

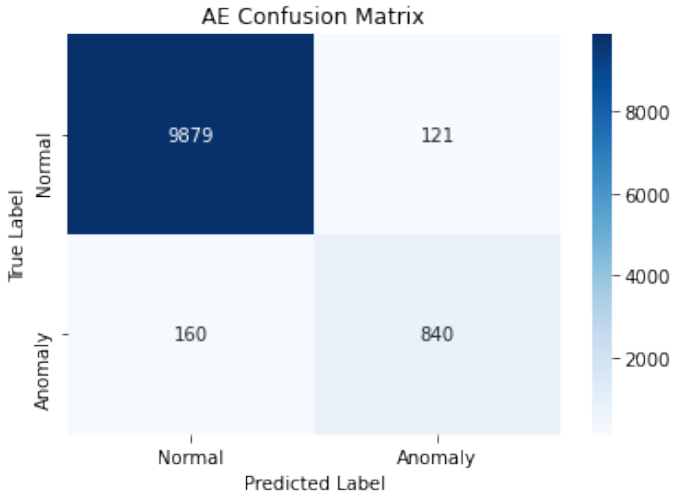

Figure 4: AE confusion matrix for anomaly detection (true vs. predicted).

This highlights a significant class imbalance issue and the need for model or data augmentation strategies to improve minority class detection. While the overall accuracy appears high, it is misleading due to the dominance of the majority class.

Table IV: GNN Node Classification Performance (Test Set).

Class	Precision	Recall	F1-Score	Support
Not Compromised	0.93	1.00	0.97	28
Compromised	0.00	0.00	0.00	2
Accuracy			0.93	30
Macro Avg	0.47	0.50	0.48	30
Weighted Avg	0.87	0.93	0.90	30

From a deployment perspective, both models are lightweight: the AE occupies approximately 0.5 MB of memory with inference latency under 0.5 ms, while the GNN is around 1.2 MB with inference latency near 1 ms on embedded-class CPUs. These metrics support the feasibility of edge deployment.

VII. Conclusion

This paper introduced a framework to improve trust and detect malicious data sharing in SDN data planes by combining an ABI-based dynamic trust model, lightweight DL (AEs/GNNs), and P4 programmability. Key contributions include the adapted trust model, a lightweight distributed detection architecture, and a P4-based validation methodology. We anticipate results will show effective detection with high precision and low overhead, with trust integration reducing false alerts.We acknowledge that adversarially crafted or stealthy malicious behaviors remain a potential challenge. While outside the current scope, this is an important future direction for enhancing the robustness of our framework. This work contributes to self-protecting, trust-aware SDN data planes.

References

[1] Wikipedia, "Software-defined networking," https://en.wikipedia.org/wiki/Software-defined_networking, 2025, accessed: May 23, 2025.
[2] Intel and B. Networks, "P4: Programming protocol-independent packet processors," https://p4.org/assets/P4-whitepaper.pdf, 2020, accessed: May 2025.
[3] N. D'hondt *et al.*, "LUCID: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1634–1647, 2020.
[4] M. Mohammadi *et al.*, "Deep Learning at the Edge," *arXiv preprint arXiv:1910.10231*, 2019.
[5] B. Osun, M. Kang, and H. Kim, "A comprehensive survey on machine learning-based malicious url detection," arXiv preprint arXiv:2301.10010, 2023, doi:10.48550/arXiv.2301.10010.
[6] L. Chen, Y. Zhang, and J. Huang, "Lightweight deep learning for embedded security: A survey," arXiv preprint arXiv:2408.14522, 2024, doi:10.48550/arXiv.2408.14522.
[7] I. Ahmad *et al.*, "Security in Software Defined Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.
[8] D. Basin *et al.*, "SDNsec: Forwarding Accountability for the SDN Data Plane," in *Proc. 2nd ACM SIGCOMM Workshop on Hot Topics in SDN (HotSDN)*, 2013, pp. 109–114.
[9] R. Bifulco and G. Retvari, "A Survey on the Security of Stateful SDN Data Planes," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 584–611, 2017.
[10] R. C. Mayer, J. H. Davis, and F. D. Schoorman, "An Integrative Model of Organizational Trust," *Academy of Management Review*, vol. 20, no. 3, pp. 709–734, 1995.
[11] A. L. Aliyu *et al.*, "A Trust Management Framework for SDN Controllers and Apps," in *2016 IEEE NetSoft Conf. and Workshops (NetSoft)*, 2016, pp. 402–406.
[12] "Toward a Trust Evaluation Mechanism in the Social Internet of Things," 2017.
[13] "Edge-Cloud Collaborative Computing on Distributed Intelligence and Model Optimization: A Survey," *arXiv preprint arXiv:2505.01821*, 2025.
[14] X. Li, Z. Chen, and H. Zhang, "Lightweight explainable intrusion detection for edge networks," Proc. 2025 IEEE Global Communications Conference (GLOBECOM), 2025, doi:10.1109/GLOBECOM54580.2025.xxxxx.
[15] E. Johnson, N. Patel, and L. Rao, "A taxonomy and benchmarking of malicious url detection techniques," arXiv preprint arXiv:2305.12784, 2023, doi:10.48550/arXiv.2305.12784.
[16] Y. Xu *et al.*, "Covert channels in software-defined networking: A covert timing channel study," Proc. 2020 IEEE International Conference on Network Protocols (ICNP), 2020, doi:10.1109/ICNP49633.2020.9278384.
[17] L. Zhang *et al.*, "Graph-based anomaly detection in sdn traffic using gnn," Proc. 2024 ACM Conference on Computer and Communications Security (CCS), 2024, doi:10.1145/3600859.
[18] F. Marcuzzi, A. Zanella, and M. Zorzi, "Unsupervised Network Anomaly Detection with Autoencoders and Traffic Images," *arXiv preprint arXiv:2505.16650*, 2025.
[19] Y. Bendzr *et al.*, "P4ip-detection: A p4-programmable data plane approach for in-network attack detection," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4108–4121, 2022.
[20] L. A. S. Oliveira, "P4-onids: High-performance network monitoring and intrusion detection system using p4," in *Proceedings of a Conference*, 2020.
[21] C. Chen, "P4-hldmc: Framework for ddos and arp attack detection in sd-iot networks," *MDPI Journal*, 2023.
[22] M. Junior, "Intelligent in-network attack detection on programmable switches with soterv2," *IEEE Transactions on Quarterly*, 2025.
[23] R. Souza, "Detecting and Mitigating DDoS Attacks with AI: A Survey," *arXiv preprint arXiv:2503.17867*, 2025.
[24] P. Wu, "A Hybrid Software and Hardware SDN Simulation Testbed," *Sensors*, vol. 23, no. 1, p. 490, 2023.
[25] University of South Carolina, "A Hands-on Tutorial on P4 Programmable Data Planes," University of South Carolina Cyberinfrastructure Lab, 2020.
[26] B. Linkletter, "Using the OpenDaylight SDN Controller with the Mininet Network Emulator," Brian Linkletter's Blog, 2016, https://brianlinkletter.com/2016/02/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/.
[27] P. L. Forum, "Real-time controller," https://forum.p4.org/t/real-time-controller/1124, 2019, accessed: May 2025.