Accepted in IEEE INDISCON 2026

Dynamic Rule Generation and Intrusion Detection for 5G UPF Security using Machine Learning

George Bhokta and Samaresh Bera, *Senior Member, IEEE*
Department of Computer Science and Engineering
Indian Institute of Technology Jammu
Jammu and Kashmir, India - 181221
Email: 2023pcs0028@iitjammu.ac.in, s.bera.1989@ieee.org

***Abstract*—The User Plane Function (UPF) in 5G networks handles high-throughput user data, making it a critical yet vulnerable component in the mobile core of a 5G network. As 5G introduces diverse and low-latency traffic patterns, securing the UPF becomes essential to prevent malicious flows from compromising network performance and reliability. This paper presents a real-time intrusion detection and prevention system for the 5G UPF, which integrates machine learning (ML)-based flow classification with an open-source signature-based detection engine. The proposed system consists of two phases: a) classification of real-time network traffic using ML-based model; and b) dynamic rule generation and placement to act on the classified traffic. In the first phase, we train multiple supervised models, including Random Forest, Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), and Logistic Regression, using an open-source 5G dataset. The second phase, upon detection of malicious traffic, generates dynamic traffic forwarding rules and injects the rules in real-time via UNIX socket, enabling instant rule enforcement. The system is deployed inside a fully containerized 5G environment using Docker and Kubernetes, with custom 5G core and radio access network images. Simulations on 10,000 flows demonstrate 99% detection accuracy and show a 92% reduction in rule count via duplicate suppression. Our approach enhances 5G security by bridging data-driven anomaly detection with low-latency and rule-based mitigation, providing a reproducible and scalable framework for inline defense at the 5G UPF.**

***Index Terms*—5G Security, User Plane Function (UPF), Inline Detection, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Machine Learning, Dynamic Rule Generation.**

I. Introduction

The roll-out of 5G networks brings unprecedented speed and connectivity, but also expands the attack surface in the mobile core, especially at the User Plane Function (UPF), which handles user data traffic. Ensuring security in the 5G user plane is critical, as cyber threats continue to grow in scale and sophistication [1]–[3]. Traditional network intrusion detection systems (IDS), like Suricata [4], rely on static and manually crafted rules (signatures) to detect known malicious patterns. While effective for known attacks, such signature-based approaches struggle to keep pace with rapidly evolving threats and novel attack variants. Manually updating and tuning hundreds of rules is time-consuming and error-prone, creating a need for more adaptive security mechanisms [1]. Conversely, machine learning (ML)-based IDS techniques can identify anomalies or new attack patterns by learning from data. However, they are often deployed in offline analysis or monitoring roles rather than inline prevention [5]. Moreover, the 5G core network uses specialized protocols (such as PFCP) along with traditional network protocols (such as HTTP, TCP-TLS). Consequently, the traditional IDS systems may not be able to recognize specialized 5G protocols, potentially limiting their detection capabilities [6].

In this paper, we propose a dynamic rule generation system that combines the strengths of ML-based detection with the quick response of rule-based engines. Our approach is implemented within a 5G UPF to perform real-time monitoring of user traffic. We leverage ML models to classify traffic flows *on-the-fly* as *benign* or *malicious*. When a malicious flow is identified with high confidence, the system automatically generates a specific rule tailored to that threat and injects it into the IDS/IPS engine running in the network. This allows the system to quickly adapt to react on real-time traffic without waiting for manual rule updates. This also ensures the immediate detection and blocking of any subsequent malicious traffic of similar type. Essentially, the system learns from the traffic and generates signatures in real-time. The key contributions in this work are as follows:

- **End-to-End Containerized 5G Testbed Implementation:** We create a softwarized 5G network testbed using containerized Open5GS [7] and UERANSIM [8], with Suricata [4] as IDS/IPS engine deployed directly inside the UPF pod. The UPF was extended to include an ML inference module and a dynamic rule generation engine, enabling inline traffic analysis and automated mitigation. The system is fully containerized and portable, supporting reproducibility and ease of integration in live deployments.
- **ML and Rule-Based Inline Security for 5G UPF:** We train and evaluate four supervised ML models: Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Random Forest (RF) using 5G-NIDD [9] dataset. Among these, the RF classifier offered the best trade-off between detection accuracy and inference speed, achieving up to 99% accuracy on a 10,000 flow hold-out test set.
- **Dynamic Rule Management:** We develop a lightweight rule update algorithm that maintains in-memory slid-

ing windows of recent malicious activity. The system automatically generates and injects forwarding rules in real-time via UNIX sockets based on service-specific thresholds.

- **Performance Evaluation:** We simulate the detection-to-mitigation pipeline using a replay of 10,000 flows. Our experiments validate the ML classifiers' high accuracy and show that the rule generator achieves a 92% reduction in the number of enforced rules by eliminating duplicates and merging redundant rules.

The remainder of the paper is organized as follows. Section II reviews existing approaches in 5G intrusion detection and dynamic rule adaptation. Section III describes the design of the proposed system with experimental setup. Sections IV details the dataset used and the ML models. We then explain the rule generation strategy and integration of the rule with IDS/IPS engine in Section V. Section VI presents our findings. Finally, we conclude the paper in Section VII.

II. Background and Related Work

Recent works on intrusion detection and prevention in 5G networks [3], [10]–[12] primarily focused on performance analysis of softwarized security functions like IDS and IPS, IoT applications, and 5G network-specific dataset creation to emulate different attack scenarios in 5G. Malik and Bera [10] proposed a Security-as-a-Function framework that integrates security functions directly into the 5G core network using off-the-shelf hardware and open-source software. Their implementation demonstrated that incorporating an Intrusion Prevention System (IPS) like Snort within the UPF can effectively detect and mitigate threats. The authors presented the performance trade-off between softwarized IDS and IPS systems in terms of latency and throughput. Their architecture embeds detection and enforcement directly into network components, demonstrating that real-time security enforcement can be achieved without major latency penalties. Lin et al. [3] proposed an intrusion detection framework in 5G network for IoT applications. The authors deployed the IDS engine inside the UPF to monitor the IoT traffic filtered using IoT protocols, such as MQTT.

The importance of developing datasets that reflect the behavior of real-world 5G user-plane traffic has also been highlighted in recent research. Samarakoon et al. [9] introduced 5G-NIDD, a comprehensive 5G intrusion detection dataset collected from a real 5G test network. The dataset captures realistic benign and malicious traffic flows, including attacks like DoS, DDoS, and port scans, following GTP-U decapsulation and arguments-based flow extraction. Similarly, Tiwari and Bera [13] emulated `SCTP-SYN` flood attacks in the 5G control plane based on the synthesized 5G network packet traces [14]. These efforts addressed the gap in available 5G-specific intrusion detection datasets and provided a valuable benchmark for evaluating ML-based IDS solutions in 5G environments.

The adoption of ML in IDS enables the development of adaptive security mechanisms capable of responding to the sophisticated and evolving threats present in 5G networks. By leveraging ML, IDS can achieve higher detection rates and reduce false positives, thereby enhancing the overall security posture of 5G infrastructures. Researchers have explored the integration of ML techniques to improve detection capabilities using different datasets [15]–[18].

Synthesis: Beyond detection, an important aspect of an intrusion detection system is its ability to respond effectively once a threat is identified. Traditional responses include alerting network administrators or actively terminating malicious connections in IPS. In general, when rules are statically configured in IDS/IPS systems like Suricata, updating them typically requires reloading the entire engine, which introduces overhead, causes temporary downtime, and risks packet loss during the reload window. Hence, static rule updates are unsuitable for high-availability 5G environments where even brief interruptions can degrade service quality. The proposed approach addresses these limitations by dynamically injecting new detection rules based on ML techniques directly into the live instance of the IDS/IPS engine without stopping the network, leveraging Unix socket commands to update rules on-the-fly.

III. System Architecture and Working Principle

A. Architecture

The proposed system augments a standard 5G UPF with components for ML-based traffic analysis and dynamic rule injection. The architecture is designed to detect malicious traffic in real-time and immediately apply mitigation via rule updates in the IDS/IPS engine. Figure 1 depicts the proposed architecture with IDS/IPS engine at the UPF and control-plane signaling interfaces in the 5G network.

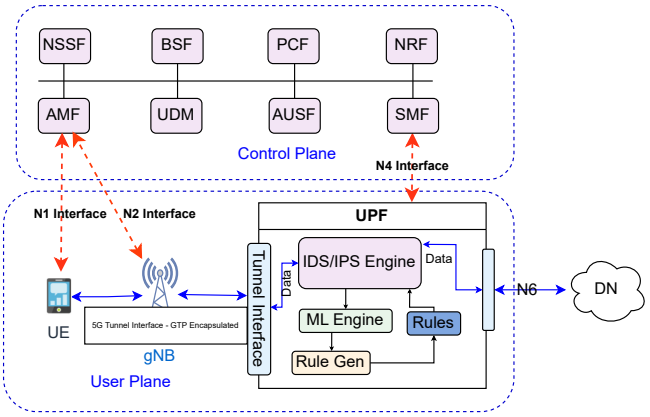

Fig. 1: 5G user-plane with components of the IDS/IPS engine

- **5G UPF:** The UPF routes user traffic between the Radio Access Network (RAN) and the Data Network (DN). We instrument the UPF to enable monitoring of traffic flows. Specifically, the IDS/IPS engine is deployed within the UPF and is configured to monitor the 5G core network interface, which carries decapsulated IP packets from GTP-U tunnels.

- **ML Classification Engine:** This module performs real-time traffic classification using pre-trained ML models. It reads flow metadata directly from the `eve.json` file generated by IDS/IPS detection engine, which includes fields such as source IP, destination IP, source port, destination port, and protocol. These details are used to construct feature vectors (discussed in Section IV-A) for each flow. The ML classifier then outputs a prediction – `Benign` or `Malicious`.
- **Rule Generation Module:** When the ML engine flags a packet or flow as `Malicious`, a new forwarding rule is generated that instructs the system to raise an alert and drop future packets from the same source. This rule is immediately injected into the running IDS/IPS engine. For example, if an IP address `10.0.0.5` is found to be malicious, a generated rule looks like:

  ```
  drop ip 10.0.0.5 any -> any any
  (msg:"ML-detected malicious IP";
  sid:100001; rev:1;)
  ```

  This rule ensures that any further traffic from the malicious IP is both alerted and blocked in real-time.
- **IDS/IPS Engine:** It is deployed to inspect user-plane traffic flowing through the UPF. It supports two operational modes:
  1) **Monitoring mode (IDS):** The engine observes traffic and generates alerts for suspicious or malicious activity but does not intervene.
  2) **Inline mode (IPS):** The engine is placed directly in the packet path (e.g., using Linux `NFQUEUE`) and is capable of actively dropping or rejecting malicious packets in real-time.

B. Working Principle

Figure 2 presents the sequence of operations within the system. We describe the operations below:

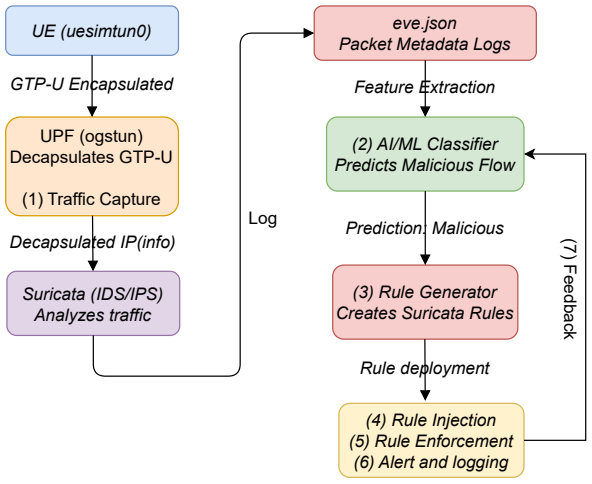

Fig. 2: Information flow within the system

1) **Traffic Capture:** Packets passing through the UPF are captured, either through mirroring or inline processing.
2) **Flow Classification:** Flow metadata is extracted and analyzed by the ML classification engine, which determines whether the traffic is benign or malicious.
3) **Rule Creation:** If a malicious flow is detected, the rule generation module formulates a new rule to match that flow.
4) **Rule Injection:** The new rule is injected dynamically into the running IDS/IPS engine using the control socket.
5) **Enforcement:** The IDS/IPS engine uses the new rule to detect and block subsequent packets that match the malicious signature.
6) **Alert and Logging:** IDS/IPS engine logs all detection and enforcement actions into the `eve.json` file.
7) **Iteration:** The system continues monitoring traffic and updating rules dynamically as new threats are identified.

The architecture and the working principle are designed to be modular and flexible. All components, UPF, ML engine, rule generator, and IDS/IPS engine, can be co-located on the same host or deployed across multiple nodes, depending on the desired scale and deployment environment.

C. Experimental Setup

To evaluate our system in a near-realistic 5G setup, we deployed the entire architecture in a virtualized environment using container orchestration and network emulation tools.

- **Containerization and Orchestration:** All 5G core components, including the UPF, were containerized using Docker. Kubernetes was used for orchestration, and Minikube served as the local Kubernetes cluster manager for testing. Helm was utilized to streamline deployment configurations across pods.
- **5G Core Setup:** We deployed the Open5GS [7] stack in Kubernetes. All control-plane network functions are containerized and managed through Helm charts. Simulated gNB and UE traffic was generated using UERANSIM [8]. Furthermore, we have modified the images for UPF and UE to integrate the desired functionalities. We use Suricata (https://suricata.io/) as the IDS/IPS detection engine. It is initialized with a base rule set and supports dynamic rule updates at runtime via its UNIX socket interface, allowing for real-time adaptability without restarting the engine [4].

IV. Dataset and Machine Learning Models

A. 5G-NIDD Dataset with Features

The 5G-NIDD dataset [9] captures benign and malicious user-plane traffic produced on a 5G testbed. After removing duplicate flow records and keeping only the 19 features that are observable at the UPF in real time (17 numeric statistics + 2 categorical headers + binary label), the working corpus comprises 12,10,974 individual flows. Roughly 39% of the flows are *benign*, while the remaining 61% are labeled *malicious*.

Each flow contains in the dataset contains multiple features and many of the features cannot be captured at the UPF. Therefore, only 18 flow-level features, which can be captured

at the UPF, were retained after removing one redundant attribute. The retained features are as follows:

- **Basic statistics**: Duration of a flow: `Dur`, and packet or byte counts: `TotPkts`, `SrcPkts`, `DstPkts`, `TotBytes`, `SrcBytes`, `DstBytes`.
- **Packet-size means**: `sMeanPktSz`, `dMeanPktSz`.
- **Rates**: `Rate`, `SrcRate`, `DstRate`.
- **TCP timing/flags**: `TcpRtt`, `SynAck`, `Offset`, `AckDat`.
- **Categorical**: protocol (`Proto`) and connection state (`State`).

All selected features are observable at the UPF in real time, making them suitable for inline intrusion detection.

B. Machine Learning Models

Four supervised learning algorithms were trained and evaluated on the 5G-NIDD flow-level dataset. A stratified 80/20 split was used for training and testing, respectively. Numerical features were standardised and categorical features (`Proto`, `State`) were one-hot encoded inside a single `scikit-learn` `ColumnTransformer`. The details of the ML models are as follows:

- **Artificial Neural Network (ANN):** Two fully-connected hidden layers (128 and 64 neurons, ReLU); Trained with Adam (`lr`$= 10^{-3}$) for 20 epochs, mini-batch size 64, using `CrossEntropyLoss`; Test accuracy is 97.66%.
- **K-Nearest Neighbours (KNN):** $k = 5$, Euclidean distance, uniform weighting, `algorithm=auto`; Test accuracy: 98.18%.
- **Logistic Regression (LR):** `penalty='12', C=1.0, solver=lbfgs`, max_iter = 100; Provides a fast linear baseline; Test accuracy: 94.43%.
- **Random Forest (RF):** 100 trees, `criterion=gini`, no depth limit (`max_depth=None`); other parameters left at default; Test accuracy: 97.56%.

V. Dynamic Rule Generation and Suricata Integration

This section presents the integration of trained ML models into the IDS/IPS engine for dynamic rule generation. The procedure starts with a *trigger* for malicious traffic followed by rule generation and injection. The injected rules are periodically managed for suppressing duplicate rules and escalating/consolidating rules beyond a predefined threshold. We discuss each of the steps in the subsequent sections.

A. Triggering Criteria

Rule generation is triggered whenever the ML module labels a flow as `Malicious` (output as 1). For a such flow, we extract the 5-tuple <`SrcIp`, `SrcPort`, `DstIp`, `DstPort`, `Proto`> together with a timestamp and update two light-weight in-memory counters as follows:

- $C_{\text{src}}[\texttt{SrcIp}]$: list of detection timestamps for the specific source IP
- $C_{\text{dst}}[(\texttt{DstIp}, \texttt{DstPort})]$: list of timestamps for that destination service

Entries older than 60 seconds are purged on every update, giving a sliding-window view of recent malicious activity. No disk or external database is required; the cache occupies only a few kilobytes. This ensures the usability of the proposed system and approach across different platforms.

B. Rule Content Strategy

Depending on the counter values, the rule generator chooses one of following three rule templates:

1) **Specific flow rule (5-tuple):** Default choice when the flow is isolated: `drop <Proto> <SrcIp> <SrcPort> -> <DstIp> <DstPort> (msg:"Malicious-flow"; sid:100 001; rev:1;)`
2) **Source-based blocking:** Applied when the same `SrcIp` appears more than or equal to a threshold value $N_{\text{src}=5}$ within the last 60 seconds: `drop ip <SrcIp> any -> any any (msg:"Malicious-block SrcIP"; sid:100 010; rev:1;)`
3) **Destination/service rule (thresholded):** Applied when the pair (`DstIp`, `DstPort`) is hit by more than or equal to a threshold value $N_{\text{dst}=5}$ within the last 60 seconds, indicating a flood: `alert tcp any any -> <DstIp> <DstPort> (msg:"Malicious service flood"; flags:S; threshold:type=threshold,track=dst,count=50, seconds=1; sid:100 020;)`

Rules are injected through Suricata's Unix control socket and become active before the next matching packet arrives. All dynamically generated rules occupy a reserved SID range 100,000+ and include the "Malicious-detected" tag for easy identification. Content-inspection keywords are intentionally omitted because the ML pipeline operates on flow-level features.

C. Injection Mechanism

The generated rules are delivered through Suricata's Unix control socket (`/var/run/suricata/suricata-command.socket`). For each flow that the ML engine labels as `Malicious`, the rule generator constructs the appropriate rule string (as discussed in Section V-B and issues the JSON command {`"command":"add-rule","rules":"<rule_text>"`}.
As the socket operates on the live rule tree, the injected rule becomes effective instantaneously; no engine restart or `kill -USR2` reload is required. All dynamically created rules are assigned SIDs from a reserved range (100,000 to 200,000) to avoid clashes with the base rule set shipped with Suricata.

D. Rule Management

To keep the live ruleset compact and avoid redundant entries, for every newly detected malicious flow the rule generator executes the following:

1) **Duplicate suppression:** An identical 5-tuple rule is not re-injected if its SID is already present in the dynamic range, tracked through an in-memory `seen_flow` set.
2) **Escalation/consolidation:**

- If $C_{\text{src}}[\texttt{SrcIp}] \geq 5$ within the window, all 5-tuple rules for that source are replaced by a single *source-blocking* rule (`drop ip <SrcIp> any -> any any`).
- If $C_{\text{dst}}[(\texttt{DstIp}, \texttt{DstPort})] \geq 10$, the generator substitutes the per-flow rules with one *destination-threshold* rule protecting the service against floods.

Together, these measures ensure that the dynamic ruleset remains both *precise* (avoiding redundant entries) and *bounded* in size. Figure 3 shows the flow-chart for the rule generation and update procedure of the proposed scheme.

Algorithm 1 Dynamic Rule Generator

Inputs: Pre-trained ML model, sliding-window size W, thresholds $N_{\text{src}}, N_{\text{dst}}$

Output: Real-time inline dynamic forwarding rule generation

while tailing `eve.json` **do**
  $r \leftarrow$ next flow record
  if $r.label = 0$ **then continue** ▷ benign
  Update $C_{\text{src}}[r.src\_ip]$ and $C_{\text{dst}}[(r.dst\_ip, r.dst\_port)]$
  Remove timestamps older than W from both caches
  if $|C_{\text{src}}[r.src\_ip]| \geq N_{\text{src}}$ **then**
    type $\leftarrow$ SOURCERULE
  else if $|C_{\text{dst}}[(r.dst\_ip, r.dst\_port)]| \geq N_{\text{dst}}$ **then**
    type $\leftarrow$ DESTRULE
  else
    type $\leftarrow$ FLOWRULE
  if rule with same 5-tuple already exists **then continue**
  Compose Suricata rule according to **type**
  Send `{"command":"add-rule", "rule"}` to Suricata socket

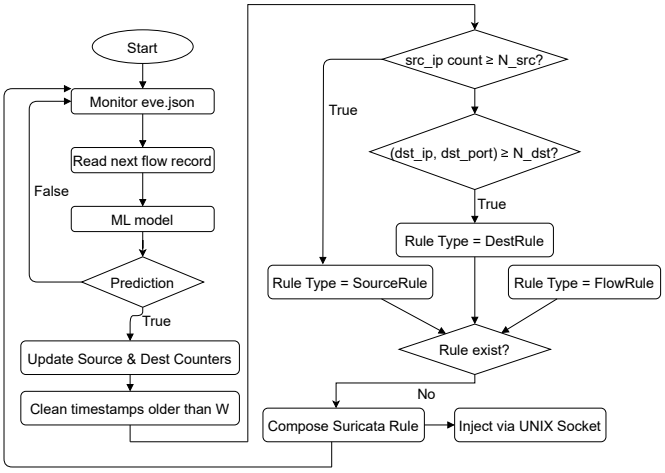

Fig. 3: Flowchart illustrating the dynamic rule generation process. The ML classifier's decision is followed by rule-type selection, duplicate suppression, and UNIX socket injection.

VI. Results and Evaluation

A separate *hold-out set* containing 10,000 previously unseen NetFlow-style records (5,000 malicious and 5,000 benign) was reserved exclusively for final evaluation. These flows were replayed through the complete processing pipeline, including feature extraction, ONNX-based model inference, and dynamic rule generation, to simulate post-deployment behavior. Each record exposed the same 18-dimensional feature vector that would be observable at the UPF following GTP-U decapsulation.

A. ML Model Performance

Two evaluation stages were performed to assess classifier performance. Table I reports training-time metrics from cross-validation on the 5g-NIDD [9] dataset for four candidate models. Random Forest (RF) demonstrated the best trade-off between accuracy and inference speed, while KNN achieved slightly higher accuracy at the cost of significantly increased latency.

TABLE I: Training-time performance comparison of machine learning models on the 5g-NIDD [9] dataset

Model	Precision	Recall	$\mathbf{F}_1$	Accuracy
ANN	0.979	0.970	0.974	0.975
KNN	0.984	0.980	0.982	0.982
LR	0.942	0.94	0.941	0.944
RF	0.984	0.977	0.981	0.976

Selection of Random Forest for deployment: Although KNN attained marginally higher accuracy during training, its inference overhead is impractical for real-time usage due to distance-based computation. Random Forest, on the other hand, offered highly competitive accuracy (97.6%) with fast and deterministic inference. It is robust to noise, supports model interpretability through feature importance ranking, and generalizes well to imbalanced and noisy network traffic. These properties make it well-suited for integration within a real-time packet inspection system at the UPF.

To validate the trained model's ability to generalize on unseen data, the final ONNX-exported version of Random Forest was evaluated on the 10,000 hold-out set. The resulting classification metrics are shown in Table II.

B. Rule-Generation Simulation

To evaluate the dynamic rule generation logic, we performed an offline replay of 10,000 previously unseen flow records from the hold-out test set. Each flow was processed using the ONNX-exported Random Forest model for binary classification. Flows labeled as `Malicious` were passed through the Python-based rule engine (Algorithm 1) to simulate real-time decision-making under constraints of duplicate suppression and escalation thresholds.

Since the dataset lacked source and destination IP addresses and ports, we synthesized them deterministically using MD5 hashes of existing flow fields (e.g., `Proto`, `SrcBytes`, `DstBytes`). This allowed consistent 5-tuple construction necessary for rule grouping and escalation logic, while preserving reproducibility.

TABLE II: Test-time classification report for ONNX-exported Random Forest on the 10,000 hold-out set

Class	Support	Prec.	Rec.	$\mathbf{F}_1$
Benign	5,000	0.995	0.993	0.994
Malicious	5,000	0.993	0.995	0.994
Accuracy	0.99	0.99	0.99	0.99
Macro average	10,000	0.994	0.994	0.994

Overall accuracy on hold-out set: **0.994**

Each malicious flow was treated as a candidate for a fine-grained rule. Duplicate 5-tuples were suppressed, and source or destination escalation was triggered if repeated activity was detected within a 60 seconds sliding window. The test used the same threshold values as the deployment system: $N_{\text{src}} = 5$, $N_{\text{dst}} = 10$.

TABLE III: Rule statistics for the 10 k-flow simulation

Metric	Value	Comments
Candidate rules (5-tuple)	5,094	one per malicious flow
After duplicate filter	423	unique flow-keys retained
Final live rules	**423**	all were 5-tuple rules

The experiment confirms that our rule generation engine is functionally correct and achieves a rule reduction of approximately 92% (5,094 candidate flows $\rightarrow$ 423 rules), validating its duplicate suppression capability in practice, as presented in Table III.

VII. Conclusion

This paper presented a real-time intrusion detection and prevention framework for 5G networks that dynamically generates rules based on machine learning (ML) predictions within the User Plane Function (UPF). By integrating pre-trained ML models with existing IDS/IPS engines' inline packet inspection capabilities, our system enables rapid detection and mitigation of malicious flows without requiring manual intervention or engine restarts. We demonstrated that all the flow-level features required by the ML models are observable at the UPF, making live inference feasible. The proposed rule-generation logic—including duplicate suppression, flow escalation, and threshold-based generalization—keeps the dynamic ruleset small and effective. Simulations on open-source dataset confirmed high classification accuracy (up to 99%) and a compact rule footprint, with fewer than 423 rules needed to cover 5,094 malicious flows.

Future extension of this work includes the training and integration of stronger AI/ML models (e.g., Transformers), implementing rule expiration mechanisms, and validating the system's performance on a 5G testbed.

References

[1] C. Hamroun, A. Fladenmuller, M. Pariente, and G. Pujolle, "Intrusion Detection in 5G and Wi-Fi Networks: A Survey of Current Methods, Challenges and Perspectives," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2025.

[2] P. Radoglou-Grammatikis, G. Nakas, G. Amponis, S. Giannakidou, T. Lagkas, V. Argyriou, S. Goudos, and P. Sarigiannidis, "5GCIDS: An Intrusion Detection System for 5G Core with AI and Explainability Mechanisms," in *Proc. of IEEE Globecom Workshops*, 2023, pp. 353–358.

[3] T. N. Linh Le, B. A. Salem, E. A. Ahad, N. Aitsaadi, and X. Du, "5G-IoT-IDS: Intrusion Detection System for CIoT as Network Function in 5G Core Network," in *Prof. of the IEEE Globecom*, Dec. 2023, pp. 4773–4778.

[4] Devzery, "Guide to Suricata: Network Security, IDS, IPS, and NSM," 2023. [Online]. Available: https://suricata.io/

[5] Y.-E. Kim, Y.-S. Kim, and H. Kim, "Effective Feature Selection Methods to Detect IoT DDoS Attack in 5G Core Network," *Sensors*, vol. 22, no. 9, p. 3456, 2022.

[6] P. Körning and M. Nilsson, "Intrusion detection systems in 5G core networks," Master's thesis, Mälardalen University, 2024.

[7] "Open5GS: Open source 5G core network," Accessed on Jan. 2025. [Online]. Available: https://open5gs.org/

[8] "UERANSIM: 5G UE and RAN simulator," Accessed on Jan. 2025. [Online]. Available: https://github.com/aligungr/UERANSIM

[9] 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 over 5G Wireless Network," 2022. [Online]. Available: https://dx.doi.org/10.21227/xtep-hv36

[10] S. Malik and S. Bera, "Security-as-a-Function in 5G network: Implementation and Performance Evaluation," in *Prof. of the SPCOM*, 2024, pp. 1–5.

[11] A. Alam, A. Umer, I. Ullah, and A. Alsayat, "AI-enabled cybersecurity framework for future 5G wireless infrastructures," *Scientific Reports*, vol. 16, no. 1, p. 7055, Feb. 2026.

[12] N. Wehbe, H. A. Alameddine, and C. Assi, "HTTP/2 DoS Attacks in 5 G Networks: Impact Analysis and Anomaly Detection," *IEEE Trans. Mob. Comput.*, pp. 1–15, 2026.

[13] A. Tiwari and S. Bera, "Dataset of handshake flooding on 5g control plane," 2025. [Online]. Available: https://dx.doi.org/10.21227/06yy-4j95

[14] S. M. S. Subramanian, M. R. K. Kanagarathinam, and K. M. S. Sivalingam, "Packet Traces from Large Scale 5G Performance Testing," 2024. [Online]. Available: https://dx.doi.org/10.21227/5mne-2m31

[15] P. L. S. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, "Machine and Deep Learning Solutions for Intrusion Detection and Prevention in IoTs: A Survey," *IEEE Access*, vol. 10, pp. 121 173–121 192, 2022.

[16] K. Noor, A. L. Imoize, C.-T. Li, and C.-Y. Weng, "A Review of Machine Learning and Transfer Learning Strategies for Intrusion Detection Systems in 5G and Beyond," *Mathematics*, vol. 13, no. 7, p. 1088, 2025.

[17] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of Machine Learning for Anomaly-based Intrusion Detection Systems in the CICIDS2017 Dataset," *IEEE Access*, vol. 9, pp. 22 351–22 370, 2021.

[18] L. Zomlot, S. Chandran, D. Caragea, and X. Ou, "Aiding Intrusion Analysis Using Machine Learning," in *Proc. of the Intl. Conf. on Machine Learning and Applications*, vol. 2, Dec. 2013, pp. 40–47.