From Detection To Mitigation: A Hybrid ML-Firewall Ddos Defense With Hardware-Assisted Validation

Ayeshna Singh
Electronics and Telecommunications Engineering
Thakur College of Engineering and Technology
Mumbai, Maharashtra

Prachi Sankhe
Electronics and Telecommunications Engineering
Thakur College of Engineering and Technology
Mumbai, Maharashtra

Shlok Yadav
Electronics and Telecommunications Engineering
Thakur College of Engineering and Technology
Mumbai, Maharashtra

Nikhil Tiwari
Electronics and Telecommunications Engineering
Thakur College of Engineering and Technology
Mumbai, Maharashtra

Abstract -- **Distributed Denial of Service (DDoS) attacks continue to pose a significant threat to service availability by overwhelming network and host resources with malicious traffic [1][7]. While machine learning based intrusion detection systems have demonstrated high classification accuracy on benchmark datasets, their direct deployment in operational environments is often limited by false positives, delayed reaction times, and the lack of explainable enforcement mechanisms [4][5].**

This paper presents a hybrid DDoS detection and mitigation system that integrates machine learning based flow classification with active Linux firewall enforcement, supported by independent detection signals from an FPGA-based monitoring module. Network traffic is captured and aggregated into flows, from which engineered features are extracted and classified using a Random Forest model trained on CIC-DDoS2019 data. Instead of relying on a fixed decision threshold, confidence calibration is performed using precision recall analysis to optimize detection reliability under class imbalance [4].

To enhance robustness, ML outputs are combined with heuristic overrides based on packet volume and burst behavior, enabling rapid mitigation through dynamic iptables rule insertion. An FPGA-based SYN flood detector operates in parallel to provide deterministic, low-latency alerts that corroborate software-based detections and motivate hardware-assisted scalability [9][10]. The system further integrates Suricata as an IDS/IPS layer and Splunk for centralized logging and visualization, enabling cross-layer validation of attack events. Experimental results demonstrate consistent detection behavior across software and hardware signals, reduced false alarms, and timely mitigation, highlighting the effectiveness of hybrid, multi-layer DDoS defenses.

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks remain among the most disruptive threats to modern networked systems, targeting availability by saturating bandwidth, exhausting connection tables, or overwhelming processing resources [1][7]. The increasing prevalence of large-scale botnets, particularly those composed of compromised IoT devices, has amplified the scale and diversity of such attacks, making traditional signature-based defenses insufficient [3][8].

Recent research has explored machine learning based intrusion detection systems to identify anomalous traffic patterns by learning statistical characteristics of benign and malicious flows [1][4]. While these approaches often report high accuracy in controlled evaluations, their operational deployment presents challenges. In real networks, traffic is highly imbalanced, attack patterns evolve rapidly, and false positives can lead to unnecessary service disruption [4][5]. Consequently, relying solely on ML-based classification without calibrated decision logic or enforcement context can reduce practical effectiveness.

In parallel, host-based firewalls such as Linux iptables provide a lightweight and readily deployable mechanism for traffic filtering and rate control [2][9]. Firewalls are effective at mitigating small-scale floods and restoring service stability; however, they lack adaptive intelligence and struggle to scale under high-rate or highly distributed attacks. Hardware-based solutions, particularly FPGA-assisted packet processing, have been proposed to address these scalability limitations by enabling parallel, line-rate inspection and filtering [9][10].

Motivated by these observations, this work proposes a hybrid DDoS defense architecture that combines the adaptability of machine learning with the determinism of rule-based firewalls and hardware-assisted detection. Unlike purely analytical studies, the system is implemented and evaluated as an integrated pipeline, incorporating ML-driven classification, heuristic decision logic, active firewall enforcement, FPGA-based signal generation, and IDS/IPS correlation. By grounding detection decisions in multiple, independent signals, the proposed approach aims to improve reliability, reduce false alarms, and provide a clear pathway from software-based defenses toward scalable hardware-assisted mitigation.

II. THREAT MODEL & DESIGN GOALS

This work focuses on volumetric and protocol-level denial-of-service attacks, with particular emphasis on TCP SYN floods, which exploit the asymmetric nature of the TCP three-way handshake to exhaust server-side resources [7]. In a SYN flood, attackers generate a large number of SYN packets often with spoofed source addresses, without completing the handshake, leading to a buildup of half-open connections on the victim system.

The threat model assumes an attacker capable of generating high-rate traffic from one or more sources, including randomized or spoofed IP addresses, targeting a publicly reachable service port. The attacker does not require access to the victim host and operates entirely at the network level. Application-layer attacks and encrypted payload inspection are considered out of scope for the present study.

Based on this model, the system is designed with the following goals:

1. Early and reliable detection: Identify malicious traffic patterns within short observation windows while minimizing false positives under benign traffic bursts [4].
2. Adaptive mitigation: Translate detection outcomes into concrete enforcement actions using inbuilt firewall mechanisms, enabling rapid response without external dependencies [2][9].
3. Hybrid decision-making: Combine ML confidence scores with heuristic rules based on packet volume and burst behavior to improve robustness against evasion and dataset bias [5].
4. Hardware compatibility: Incorporate FPGA-based detection signals as an independent and deterministic reference, enabling validation of software decisions and motivating future hardware offloading for scalability [9][10].
5. Operational visibility: Maintain comprehensive logging and monitoring through IDS/IPS alerts and centralized dashboards to support analysis, debugging, and trust in automated mitigation [6].

These goals collectively guide the design of a practical, multi-layer DDoS defense that balances adaptability, determinism, and deployability.

III. SYSTEM ARCHITECTURE

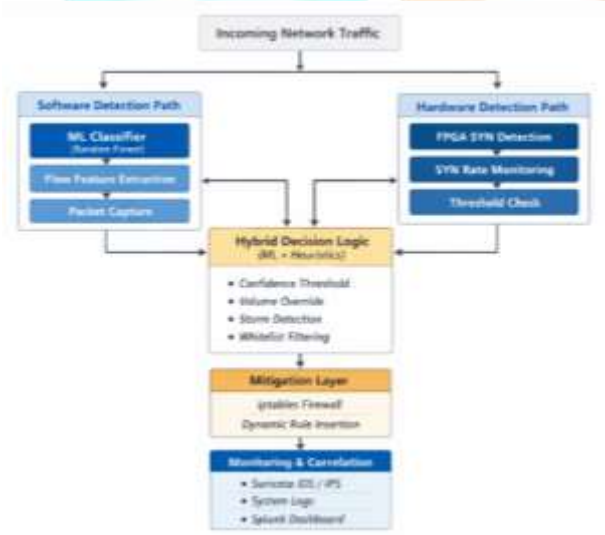

Figure 1: System architecture block diagram

The proposed system follows a modular architecture in which detection, decision making, enforcement, and monitoring are decoupled but tightly coordinated. Each component is designed to contribute a specific signal to the overall mitigation pipeline.

A. Traffic Observation and Flow Aggregation

Incoming packets are captured at the host network interface and grouped into flows over a fixed time window. Flow keys are defined as a tuple of source IP, destination IP, destination port, and protocol.

flow_key = (src_ip, dst_ip, dst_port, protocol)

This flow-based abstraction reduces per packet overhead while preserving temporal and protocol-level characteristics relevant to DDoS attacks, as supported by prior flow monitoring studies [1][2].

B. Machine Learning Detection Engine

For each completed flow window, a feature vector is constructed and passed to the ML classifier. The classifier outputs a probabilistic score rather than a binary label, enabling downstream decision logic.

attack_score = RF.predict_proba(flow_features)[1]

Random Forest models are used due to their robustness under noisy and correlated feature sets, a property validated in several ML-based DDoS detection studies [1][4].

C. Firewall Enforcement Subsystem

The Linux Netfilter framework is used as the enforcement layer. Rather than statically configured rules, firewall entries are inserted dynamically in response to detection decisions.

Key properties of this subsystem include:

- Immediate packet drop after rule insertion
- Stateless enforcement behavior
- Minimal dependency on external controllers

Host-based firewalls have been shown to provide effective mitigation for short-lived or moderate-scale floods, though they require intelligent triggers under large-scale attacks [2][9].

D. FPGA Assisted Detection Module

An FPGA-based detection module operates independently of the software stack. The module performs packet-level inspection and maintains counters for SYN packets and per source rates.

Detection logic follows a simple threshold comparison:

```
if syn_count > T_syn:
    raise_alert()
```

As discussed in FPGA mitigation literature, such deterministic logic enables low latency detection and consistent behavior under high packet rates [9][10]. In this work, FPGA outputs are used for validation and correlation, not direct enforcement.

E. Monitoring and Correlation Layer

Suricata is deployed as an IDS/IPS to generate alerts and flow logs, while Splunk aggregates logs from the ML detector, firewall, and IDS.

Correlation across these layers allows verification of detection consistency, as recommended in operational monitoring studies [6]. This design improves trust in automated mitigation decisions.

IV. HYBRID DETECTION & MITIGATION LOGIC

The hybrid logic module is responsible for translating detection signals into mitigation actions. Decisions are made using both probabilistic and rule-based inputs.

A. Decision Inputs

The following inputs are considered for each flow:

- ML attack confidence score
- Packet count per flow
- Number of concurrent flows in the observation window
- Source IP whitelist status

This multi-input design reflects recommendations from hybrid detection frameworks in prior work [5].

B. Confidence Based Classification

Each flow is first evaluated against the calibrated ML threshold τ.

```
if attack_score ≥ τ:
    candidate = malicious
```

Threshold calibration using precision–recall analysis ensures reliability under class imbalance, as emphasized in [4].

C. Heuristic Overrides

Two overrides are applied to capture extreme attack behavior:
Volume override

```
if packet_count > P_max:
    candidate = malicious
```

Storm override

```
if total_flows > F_max and packet_count ≤
2:
    candidate = malicious
```

These heuristics capture volumetric and distributed attack patterns reported in DDoS literature [1][7].

D. Whitelisting and Safety Controls

Before enforcement, the source IP is checked against a trusted whitelist.

```
if src_ip in WHITELIST:
    ignore_flow()
```

Whitelisting is essential to prevent self-induced denial-of-service and is widely recommended in firewall-based mitigation systems [2][5].

E. Enforcement Workflow

Confirmed malicious sources are blocked using dynamic firewall rule insertion.
iptables -A INPUT -s <src_ip> -j DROP
This approach enables rapid mitigation with minimal system overhead and aligns with enforcement strategies used in prior host-based and SDN mitigation systems [1][9].

V. MACHINE LEARNING DETECTION PIPELINE

The machine learning component of the proposed system is responsible for estimating the likelihood that an observed network flow corresponds to a DDoS attack. The design of this pipeline follows established practices in flow-based intrusion detection, while incorporating calibration steps required for operational deployment under highly imbalanced traffic conditions.

A. Dataset Selection and Feature Engineering

Traffic data for training and evaluation was drawn from the CIC-DDoS2019 dataset, which contains labeled benign and attack flows generated under controlled yet realistic scenarios. This dataset has been widely used in prior DDoS detection studies due to its diversity of attack types and rich flow-level statistics [1][4].
From the dataset, only BENIGN and SYN flood traffic were selected to align with the threat model described in Section II. Each flow is represented using 21 engineered features that capture temporal, volumetric, and protocol-specific behavior. These include flow duration, forward and backward packet counts, packet length statistics, inter-arrival times, and TCP flag counts.
A simplified representation of the feature vector for each flow is shown below:

```
F = [
  src_port, dst_port, protocol,
  flow_duration,
  total_fwd_packets, total_bwd_packets,
  fwd_pkt_len_mean, fwd_pkt_len_std,
  flow_packets_per_sec,
  flow_iat_mean, flow_iat_min,
  fwd_iat_mean,
  min_pkt_len, max_pkt_len,
  avg_fwd_segment_size,
  syn_count, ack_count,
  psh_count, rst_count, fin_count
]
```

These features are consistent with those employed in earlier ML-based DDoS detection works, where flow aggregation was shown to be effective in distinguishing attack behavior from normal traffic bursts [1][4].

B. Data Preprocessing and Class Imbalance Handling

Raw flow records often contain undefined or extreme values caused by short-lived connections or capture artifacts. All infinite and NaN values were removed prior to training to ensure numerical stability.
A key challenge in DDoS detection is extreme class imbalance. In real deployments, benign traffic vastly outnumbers attack traffic, yet misclassifying benign flows has a high operational cost. As discussed in [4], relying on accuracy alone in such conditions leads to misleading performance estimates.
To address this, the training set was balanced using random undersampling of the majority class, while the test set was intentionally kept imbalanced to reflect real-world traffic distributions. This approach has been adopted in several prior studies to prevent the classifier from becoming biased toward the majority class during training [1][5].
Feature scaling was applied using standard normalization to ensure that attributes with larger numeric ranges did not dominate the learning process. The transformation is defined as:
$x\_scaled = (x - \mu) / \sigma$
where μ and σ denote the mean and standard deviation computed from the training data.

C. Model Selection and Training

A Random Forest classifier was selected for flow classification due to its robustness to feature correlation, resistance to overfitting, and strong empirical performance in DDoS detection tasks [1][4]. The ensemble nature of Random Forests allows multiple decision trees to vote on the classification outcome, improving generalization under noisy traffic patterns.

Model training follows the procedure:

```
rf_model = RandomForestClassifier(
    n_estimators=50,
    random_state=42,
    n_jobs=-1
)
rf_model.fit(X_train_scaled,
y_train_balanced)
```

Rather than producing a hard class label, the trained model outputs class probabilities using the predict_proba function. This probabilistic output is essential for threshold calibration and hybrid decision-making.

D. Threshold Calibration Using Precision–Recall Analysis

In operational DDoS detection, the cost of false positives and false negatives is inherently asymmetric. Blocking benign traffic degrades service availability, while delayed or missed detection increases the impact of an ongoing attack. As emphasized in prior classifier evaluation studies, decision thresholds must therefore be tuned explicitly rather than fixed at the default value of 0.50, particularly under severe class imbalance [4].

To address imbalance during training, Random Under-Sampling was applied to the training set, forcing the classifier to learn discriminative characteristics of benign and attack traffic with equal representation. The test set was intentionally kept imbalanced to reflect realistic deployment conditions, consistent with evaluation practices reported in earlier ML-based DDoS studies [1][5].

Threshold calibration was performed using precision–recall analysis on the imbalanced test set. The F1 score was computed across all candidate thresholds, and the threshold that maximized F1 was selected for deployment:

```
precision, recall, thresholds =
precision_recall_curve(y_test, y_probs)
f1_scores = 2 * (precision * recall) /
(precision + recall + 1e-9)
best_threshold =
thresholds[np.argmax(f1_scores)]
```

Across repeated experiments, the optimal threshold was consistently observed in the range $\tau \approx 0.08$ to 0.09. This relatively low threshold prioritizes sensitivity to attack patterns, enabling early detection of malicious flows. While such a threshold may increase the risk of false positives, the proposed system mitigates this risk through hybrid decision logic and independent FPGA-based validation. As a result, only alerts corroborated by heuristic conditions or hardware signals lead to persistent firewall enforcement, balancing detection sensitivity with operational stability.

E. Integration with Online Detection

During live traffic monitoring, extracted flow features are scaled using the same parameters learned during training. The classifier outputs an attack confidence score for each flow, which is then forwarded to the hybrid decision logic described in Section IV.

By separating probabilistic detection from enforcement decisions, the system avoids brittle behavior and enables informed mitigation based on both statistical learning and rule-based reasoning. This separation aligns with hybrid mitigation architectures proposed in prior work, where ML serves as an advisory component rather than a sole authority [5].

VI. FPGA-ASSISTED DETECTION MODULE

The FPGA-assisted detection module is incorporated as a parallel and independent component within the overall defense architecture. Its purpose is not to replace machine learning or firewall-based mitigation, but to provide a deterministic and low-latency detection signal that can corroborate software-based decisions and motivate hardware-assisted scalability. This design choice aligns with prior studies that emphasize hardware offloading for simple, high-frequency detection tasks rather than complex classification logic [9][10].

A. Role of FPGA in the Detection Pipeline

As discussed in FPGA-based DDoS mitigation literature, hardware logic is particularly effective for identifying primitive attack indicators such as excessive SYN rates or abnormal packet bursts [9]. In the proposed system, the FPGA operates as a lightweight monitoring module that observes incoming TCP traffic and extracts protocol-level signals that are computationally expensive to track in software at high packet rates.

The FPGA does not perform flow classification or mitigation. Instead, it performs three core functions:

1. Parsing incoming Ethernet and IP packets.
2. Identifying TCP SYN packets using flag inspection.
3. Maintaining counters over short observation windows.

By restricting the FPGA role to deterministic operations, the design avoids complexity while ensuring predictable timing behavior.

B. Hardware Detection Logic

At the hardware level, incoming packets are parsed to extract TCP header fields. SYN packets are identified using a simple condition on the TCP flags field.

```
if TCP.SYN == 1 and TCP.ACK == 0:
    syn_counter = syn_counter + 1
```

Counters are maintained over a fixed time interval. At the end of each interval, the observed SYN count is compared against a predefined threshold.

```
if syn_counter > T_syn:
    alert = 1
else:
    alert = 0
```

This threshold-based detection mechanism is consistent with FPGA designs reported in [9], where simplicity and determinism are favored over adaptive logic to guarantee reliable operation under high traffic volumes.

C. Integration with the Host System

The FPGA detection module operates alongside the host system and communicates detection events through software-visible interfaces. During experimental evaluation, detection alerts generated by the FPGA were observed in real time and correlated with traffic generation parameters.

The host system does not automatically enforce mitigation based on FPGA alerts alone. Instead, FPGA outputs are treated as an independent validation signal that can be compared against:

- ML confidence scores
- Firewall-triggered logs
- IDS alerts generated by Suricata

This separation of detection and enforcement follows the design philosophy advocated in hybrid defense frameworks, where multiple weak signals are combined to increase overall reliability [5].

D. Experimental Validation of FPGA Detection

To validate the FPGA module, controlled SYN flood traffic was generated targeting the FPGA-equipped system. The packet generation rate was varied systematically, and the observed SYN counts were recorded at both the traffic generator and the FPGA output.

Detection accuracy was evaluated using the following metric:

Detection Accuracy (%) = (Detected SYN / Sent SYN) × 100

Across all tested scenarios, the FPGA counters accurately reflected the number of injected SYN packets, and alerts were triggered consistently once the configured threshold was exceeded. These results confirm that the FPGA module provides reliable, low-latency detection of protocol-level attack behavior, consistent with observations reported in prior hardware-based DDoS studies [9][10].

E. Design Implications

While the FPGA module in this work is limited to basic SYN flood detection, its integration demonstrates a clear pathway toward scalable hardware-assisted defenses. By offloading repetitive packet-level operations to hardware, the system reduces reliance on CPU-bound processing and improves resilience under increasing traffic rates.

As highlighted in recent hardware offloading research, such modular designs enable future extensions, including flow aggregation, rate limiting, or policy enforcement in hardware, without disrupting existing software pipelines [10].

VII. IDS, IPS, AND MONITORING SYSTEM

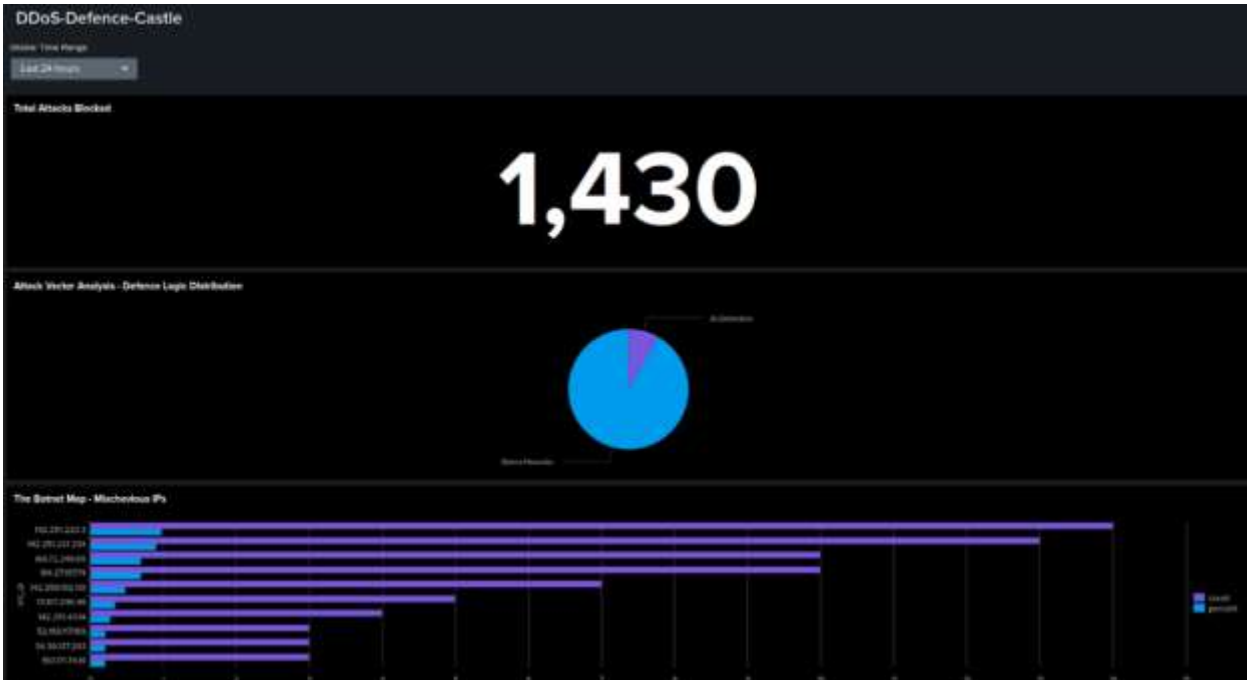

Figure 2: Centralized monitoring dashboard summary

The proposed system integrates intrusion detection, prevention, and monitoring components to ensure that detection decisions are observable, verifiable, and auditable. Rather than treating mitigation as a black-box action, the architecture explicitly exposes detection and enforcement events through an IDS/IPS layer and a centralized monitoring framework. This design choice follows recommendations from operational security studies, which emphasize visibility and correlation as essential requirements for trustworthy automated defenses [6].

A. Role of Suricata as IDS and IPS

Suricata is deployed as both an intrusion detection system and an intrusion prevention system within the monitoring plane. It inspects network traffic independently of the ML pipeline and generates alerts based on protocol violations, abnormal flow behavior, and predefined detection rules. While Suricata does not directly control mitigation in this design, it provides an important secondary signal that reflects traffic anomalies from a signature and rule-based perspective.

Suricata generates structured JSON logs containing flow records, alerts, and protocol metadata. These logs include fields such as source and destination IPs, ports, protocol identifiers, timestamps, and alert classifications. The presence of these structured logs allows precise alignment between IDS alerts and events observed by the ML detector and FPGA module.

Conceptually, Suricata operates as follows:

packet_in → protocol_decode → rule_match → alert/log

By maintaining this separation, the system avoids coupling ML decisions to signature-based logic, a limitation noted in earlier IDS-centric mitigation frameworks [1][5].

B. Log Generation and Semantics

Detection and mitigation events across the system generate logs at multiple layers:

1. The ML-based detector logs flow-level decisions and confidence scores.
2. The firewall logs rule insertions and blocked source addresses.

3. Suricata logs alerts and flow summaries.
4. The FPGA module produces threshold-based detection events.

Each log entry includes a timestamp and source identifier,

Figure 3: Protocol distribution during attack window, showing TCP-dominant traffic consistent with SYN flood behavior.

enabling cross-layer correlation. For example, a firewall block event is logged using a structured message format:

action=block src_ip=<IP> confidence=<score> reason=<trigger>

Such explicit logging semantics are critical for post-event analysis and debugging, as emphasized in monitoring-oriented studies [6].

C. Centralized Monitoring Using Splunk

Splunk is used to aggregate and visualize logs generated by the detection and mitigation pipeline. Logs from Suricata, the ML detector, and the firewall are ingested into a centralized index, enabling real-time queries and dashboards.

Typical queries used during evaluation include:

- Counting blocked IP addresses over time
- Visualizing active flows and alert frequency
- Identifying top destination ports and protocols during attack windows

An example query used to monitor mitigation activity is shown below:

source="/var/log/ddos_guard.log" "action=block"

Similarly, Suricata flow logs are used to visualize traffic dynamics:

source="/var/log/suricata/eve.json" event_type="flow"

Such dashboards provide operational insight into attack progression and mitigation effectiveness, supporting the role of monitoring frameworks highlighted in [6].

D. Cross-Layer Correlation of Detection Signals

A key strength of the proposed system lies in its ability to correlate detection signals across independent components. During attack scenarios, the following pattern is typically observed:

- A spike in SYN packets detected by the FPGA counters
- Elevated ML confidence scores for short-lived flows
- Suricata alerts indicating abnormal TCP behavior
- Firewall rule insertion events targeting malicious sources

By aligning these events temporally, the system verifies that mitigation decisions are not based on a single noisy signal. This cross-validation approach reflects best practices recommended in hybrid detection and mitigation architectures, where corroboration across layers improves reliability and reduces false positives [5][9].

E. Operational Benefits

The integration of IDS, IPS, and monitoring components ensures that automated mitigation remains transparent and controllable. Administrators can inspect detection decisions, verify their causes, and assess their impact through centralized dashboards. This visibility is particularly important in DDoS defense systems, where overly aggressive blocking can disrupt legitimate traffic.

By combining ML-based inference, deterministic hardware signals, and rule-based IDS alerts within a monitored environment, the proposed system balances automation with observability. This balance addresses a key limitation identified in prior ML-centric approaches, where lack of explainability hindered real-world adoption [4][6].

VIII. EXPERIMENTAL EVALUATION & RESULTS

This section evaluates the proposed hybrid DDoS detection and mitigation system under controlled attack scenarios. The objective of the evaluation is not to benchmark raw throughput, but to assess detection consistency, mitigation behavior, and cross-layer agreement between software and hardware signals. This aligns with the experimental scope adopted in prior practical DDoS studies that emphasize system behavior and reliability over synthetic performance metrics [1][5].

A. Experimental Setup

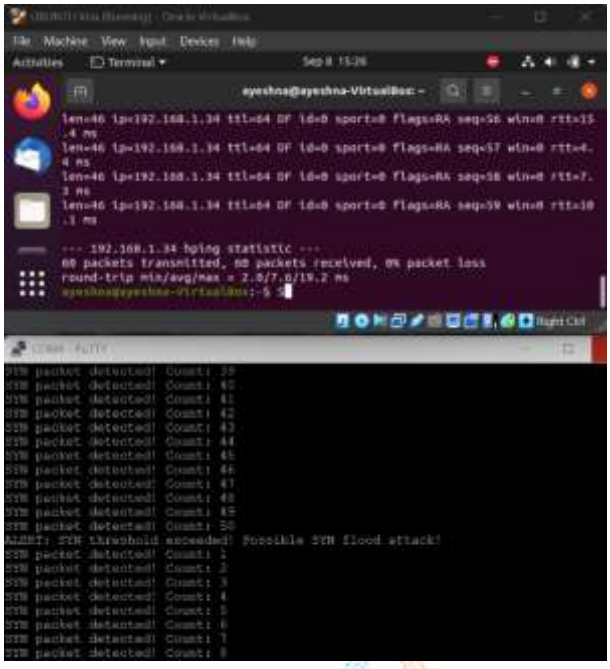

Figure 4: controlled SYN flood generation and corresponding real-time SYN packet detection

The evaluation environment consists of a Linux-based host system running the ML detector, firewall, IDS, and monitoring stack, along with traffic generators used to simulate SYN flood attacks.

Traffic generation parameters were configured to vary packet rates and source behavior while remaining within a controlled private network. Attack traffic was generated using standard packet generation tools capable of producing TCP SYN packets at configurable rates. Packets targeted a known service port on the victim system. Source addresses were randomized in selected experiments to emulate distributed and spoofed attack behavior, consistent with the threat model described earlier.

All components operated simultaneously during evaluation:

- The ML detector processed flow windows and generated confidence scores.
- The firewall dynamically inserted blocking rules based on hybrid decisions.
- The FPGA module monitored SYN packet rates independently.
- Suricata generated IDS alerts and flow logs.
- Splunk aggregated logs across all subsystems.

B. Detection Consistency Across Components

One of the primary evaluation criteria was consistency across detection signals. During SYN flood scenarios, the FPGA module reliably detected threshold violations corresponding to elevated SYN packet rates. These hardware alerts were observed to coincide with increased ML confidence scores for short-lived flows characterized by high SYN counts and minimal handshake completion.

Suricata alerts indicating abnormal TCP behavior appeared within the same time windows, providing an additional independent confirmation. Firewall block events followed shortly thereafter, triggered by the hybrid decision logic once confidence and heuristic conditions were satisfied.

This convergence of signals demonstrates that the system does not rely on a single detection mechanism. Instead, mitigation decisions emerge from agreement across probabilistic, heuristic, and deterministic indicators, a design principle advocated in hybrid defense frameworks [5][9].

C. Mitigation Behavior and Response Characteristics

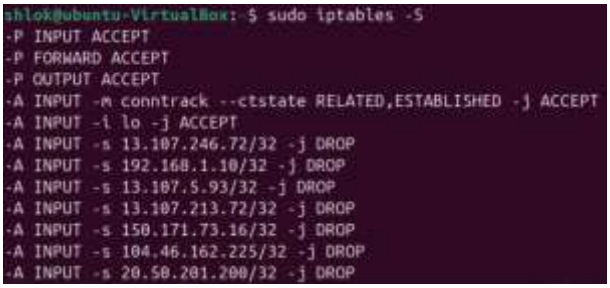

Figure 5: Dynamic firewall rule insertion

Once a source address was classified as malicious, firewall rules were inserted dynamically to drop subsequent packets from that source (Fig 5.). The use of iptables enabled immediate enforcement without requiring service restarts or external controllers.

In volumetric attack scenarios, the volume-based override ensured that extremely high packet counts triggered mitigation even when ML confidence had not yet crossed the calibrated threshold. In distributed attack simulations involving many low-packet flows, the storm detection heuristic effectively elevated suspicious flows for mitigation, preventing evasion through traffic fragmentation. These behaviors reflect known attack patterns described in DDoS literature [7].

Importantly, the whitelist mechanism prevented blocking of trusted infrastructure addresses and internal services. This safety control was essential in maintaining system availability during evaluation, reinforcing recommendations from firewall-based mitigation studies [2][5].

D. False Positives and Stability Observations

False positives were evaluated qualitatively by observing system behavior under benign traffic bursts. Short-term spikes in legitimate traffic did not result in persistent firewall blocks when ML confidence remained below the calibrated threshold and heuristic conditions were not satisfied.

The use of precision–recall based threshold tuning played a critical role in this behavior. As discussed in prior ML evaluation studies, lowering the decision threshold indiscriminately can significantly increase false positives under realistic traffic distributions [4]. By selecting a threshold optimized for F1 score on an imbalanced test set, the system achieved a stable trade-off between sensitivity and specificity.

No self-blocking events were observed during evaluation, validating the effectiveness of the whitelist and safety checks embedded in the hybrid logic.

E. Correlation and Observability

Centralized monitoring via Splunk enabled real-time visualization of attack progression and mitigation actions. Dashboards displaying blocked IP counts, flow statistics, and IDS alerts provided a coherent view of system state during attack and recovery phases.

Temporal alignment of logs revealed a clear sequence: traffic surge, detection signals, mitigation, and stabilization. This observability is essential for operational trust, as emphasized in monitoring-focused studies [6]. Administrators can trace each enforcement action back to its triggering conditions, reducing ambiguity and supporting post-event analysis.

F. Summary of Observations

The experimental evaluation demonstrates that:

1. ML-based detection, FPGA counters, and IDS alerts exhibit strong temporal agreement during attack scenarios.
2. Hybrid decision logic enables timely mitigation without excessive false positives.
3. Firewall-based enforcement is effective when guided by calibrated and corroborated detection signals.
4. Monitoring and logging provide transparency and confidence in automated mitigation decisions.

These results support the feasibility of the proposed hybrid architecture as a practical DDoS defense framework, consistent with findings reported in prior hybrid and hardware-assisted mitigation studies [1][5][9].

IX. DISCUSSIONS & LIMITATIONS

The proposed system demonstrates that combining machine learning, heuristic logic, and hardware-assisted detection can improve reliability in DDoS mitigation. However, several limitations remain and are discussed here to contextualize the results.

First, the evaluation focuses primarily on TCP SYN flood attacks. While SYN floods are representative of protocol-level denial-of-service behavior, the current feature set and heuristics are not designed to detect application-layer or reflection-based attacks. Extending the detection pipeline to additional attack classes would require both feature augmentation and retraining, as noted in prior DDoS surveys [3][7].

Second, the machine learning model is trained on CIC-DDoS2019 traffic, which, although widely used, cannot fully capture the diversity of real-world network behavior. As reported in earlier ML-based detection studies, supervised models remain sensitive to dataset bias and concept drift, necessitating periodic retraining and threshold reassessment in deployment environments [4][5].

Third, the hybrid decision logic relies on empirically selected thresholds for volume-based and storm-based detection. While these heuristics improve robustness under extreme attack conditions, static thresholds may be susceptible to evasion by adaptive attackers. This limitation is inherent to hybrid systems that combine learned and rule-based components [1][5].

Finally, the FPGA-assisted module is intentionally limited to basic protocol-level detection. Although this design ensures deterministic behavior and low latency, it does not exploit the full potential of hardware acceleration. Throughput analysis and deeper hardware integration remain outside the scope of this study but are essential for evaluating scalability, as emphasized in prior FPGA offloading work [9][10].

X. CONCLUSION & FUTURE WORK

This paper presented a hybrid DDoS detection and mitigation framework that integrates machine learning–based flow classification, heuristic decision logic, dynamic Linux firewall enforcement, FPGA-assisted detection, and IDS/IPS monitoring. The system demonstrates that mitigation decisions grounded in multiple independent signals are more reliable than ML-only or rule-only approaches [1][4][9].

Experimental evaluation under controlled SYN flood scenarios showed consistent detection behavior across software and hardware components, effective mitigation through dynamic firewall rule insertion, and strong observability through centralized monitoring. These results indicate that hybrid architectures provide a practical pathway toward deployable DDoS defenses.

Future work includes extending detection to additional attack types, introducing adaptive thresholding mechanisms, and deepening FPGA integration for flow aggregation or inline enforcement. Deployment in higher-throughput environments, such as cloud or edge networks, would further validate scalability and operational feasibility [5][10].

REFERENCES

[1] F. Khashab, J. Moubarak, A. Feghali, and C. Bassil, "DDoS Attack Detection and Mitigation in SDN using Machine Learning,"

[2] T. Jili and N. Xiao, "DDoS Detection and Protection Based on Cloud Computing Platform," Journal of Physics: Conference Series, ICCSCT 2020, vol. 1621, 012005, 2020

[3] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," Telecommunication Systems, 2019

[4] "Machine Learning Techniques used for the [DDoS detection]" (IRJET / comparative study), International Research Journal of Engineering and Technology, 2017.

[5] "SDMTA: Attack Detection and Mitigation Mechanism for DDoS Vulnerabilities in Hybrid Cloud Environment,"

[6] P. Sabjan and T. Mounika, "Nginx Web Server With AWS Cloud Watch Monitoring Service," International Journal of Research in Engineering, IT and Social Sciences, vol. 14, Issue 06, June 2024

[7] S. A. Varma and K. G. Reddy, "A Review of DDoS Attacks and its Countermeasures in Cloud Computing,"

[8] T. Wang, X. Xie, L. Zhang, C. Wang, L. Zhang, Y. Cui, "ShieldGPT: An LLM-based framework for DDoS mitigation" APNet 2024

[9] A. J. Ibrahim and S. R. Repas, "DDOS Attack Mitigation Based on FPGA Implementation," *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–6, Jul. 2024, doi: https://doi.org/10.1109/icecet61485.2024.10698517.

[10] Alessandro Rivitti, A. Tulumello, Giacomo Belocchi, and G. Bianchi, "Decentralizing DDoS Protection via Efficient Hardware Offloading," pp. 49–54, Jul. 2024, doi: https://doi.org/10.1109/hpsr62440.2024.10635987