ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

DeepShield: Autonomous DDoS Defense Through Agentic AI Workflow and Self-Evolving Markdown Knowledge-Based Threat Intelligence

S. Kanmani[1]; Mohan A.[2]; Adesh S.[3]; Ronit Metson[4]

[1,2,3,4]Department of Information Technology, Puducherry Technological University, Puducherry – 605014, India

Publication Date: 2026/05/06

Abstract: Distributed Denial of Service (DDoS) attacks remain among the most severe threats to modern network infrastructure, inflicting significant service disruptions and financial losses. Traditional rule-based or signature-matching defenses prove increasingly ineffective against sophisticated multi-vector DDoS campaigns that evolve faster than human administrators can respond. This paper presents DeepShield, a real-time, fully autonomous DDoS detection and mitigation platform structured around four tightly integrated modules. The Hybrid Detection Engine combines a CNN-LSTM deep learning encoder with a LightGBM gradient-boosted classifier to achieve sub-12-millisecond end-to-end inference on the CIC-DDoS2019 benchmark, classifying network traffic into 13 distinct attack categories plus benign traffic. Neural encoder export to ONNX Runtime reduces inference latency from 92 ms to 6–12 ms, supporting throughput of 100,000 packets per second. The Autonomous Mitigation Engine employs a Double Deep Q-Network (Double DQN) reinforcement learning agent orchestrated through a LangGraph state machine, selecting among five network enforcement actions without manual intervention. The Threat Intelligence Engine enriches confirmed DDoS events with contextual analysis drawn from a self-evolving Markdown Knowledge Base operating across three graceful degradation modes. A Dockerized testbed with 13 attacker containers enables reproducible evaluation, while a React-based Security Operations Centre (SOC) dashboard delivers live visualisation via WebSocket streaming. Experimental evaluation on CIC-DDoS2019 yields a weighted F1-score of 0.9957, confirming DeepShield's suitability for deployment in operational SOC environments without any cloud dependency.

Keywords: *DDoS Detection; Deep Learning; Reinforcement Learning; Threat Intelligence; ONNX Runtime; LightGBM; CNN-LSTM; Double DQN; LangGraph; Network Security.*

How to Cite: S. Kanmani; Mohan A.; Adesh S.; Ronit Metson (2026) DeepShield: Autonomous DDoS Defense Through Agentic AI Workflow and Self-Evolving Markdown Knowledge-Based Threat Intelligence. *International Journal of Innovative Science and Research Technology*, 11(4), 3394-3408. https://doi.org/10.38124/ijisrt/26apr1640

I. INTRODUCTION

The exponential proliferation of internet-connected devices and services has rendered Distributed Denial of Service (DDoS) attacks one of the most economically damaging cyber threats facing modern organisations. A DDoS attack saturates network bandwidth, exhausts server resources, or exploits protocol weaknesses, rendering services unavailable to legitimate users. The diversity of contemporary attack vectors — volumetric floods, protocol exploitation, and application-layer attacks — demands defence systems capable of distinguishing attack patterns from benign traffic in real time.

DeepShield addresses this challenge as a fully autonomous, zero-cloud-dependency DDoS defence platform. Operating as an inline network node, the system captures and classifies every flow within 6–12 milliseconds, makes autonomous mitigation decisions through reinforcement learning, and continuously enriches its knowledge base with attack intelligence — all without human intervention. DeepShield is designed for deployment in Security Operations Centres (SOCs), campus networks, and data centre environments.

The system architecture comprises four tightly integrated modules. Module 1, the Hybrid Detection Engine, ingests live network traffic via Scapy packet capture or PCAP file replay, reconstructs bidirectional flows using a FlowTracker engine, extracts 45 CIC-DDoS2019-compatible statistical features, and classifies each flow through a two-stage pipeline: a CNN-LSTM encoder

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

exported to ONNX Runtime, followed by a LightGBM gradient-boosted classifier. Module 2, the Autonomous Mitigation Engine, selects one of five network enforcement actions using a Double DQN reinforcement learning agent orchestrated through a LangGraph state machine with iptables FORWARD chain enforcement. Module 3, the Threat Intelligence Engine, enriches detection events with contextual analysis from a self-evolving Markdown Knowledge Base, supporting three graceful degradation modes including local LLM-based reasoning, rule-based scoring, and emergency template fallback. Module 4, the Dockerized Testbed and Real-Time Dashboard, provides a controlled 13-attacker laboratory environment and a React-based SOC dashboard delivering live monitoring via WebSocket streaming.

II. LITERATURE REVIEW

➢ *DDoS Detection*

Apostu et al. [1] provided a foundational taxonomy of machine learning approaches for network defence, highlighting the limitations of traditional signature-based systems. Issa and Albayrak [2] proposed a CNN-LSTM hybridisation for DDoS intrusion detection, directly validating the architectural choice adopted in DeepShield's detection pipeline. Kar [3] demonstrated efficient real-time DDoS detection using machine learning, confirming the feasibility of low-latency classification in operational environments.

Ma et al. [4] showed that ensemble classifiers remain competitive for high-throughput detection in SDN edge computing environments. Shohan et al. [5] corroborated the combination of deep learning encoders with gradient-boosted classifiers for DDoS detection. Sharafaldin et al. [7] introduced the CIC-DDoS2019 dataset used in this work, providing labelled network flows for 12 DDoS attack categories. Yuan et al. [8] demonstrated that recurrent architectures capture temporal flow patterns invisible to shallow classifiers, while Agrawal et al. [9] showed that CNN-LSTM hybrid encoders achieve superior feature extraction from variable-length traffic sequences. LightGBM [10] delivers state-of-the-art performance on tabular data via histogram-based gradient boosting, and ONNX Runtime [11] provides hardware-accelerated inference enabling cross-platform deployment at latencies suitable for real-time systems.

➢ *Autonomous Mitigation*

Wang et al. [6] proposed ShieldGPT, an LLM-based framework for DDoS mitigation with explainable analysis, demonstrating the emerging paradigm of integrating large language models into autonomous network defence pipelines. Van Hasselt et al. [12] introduced Double DQN, addressing Q-value overestimation by decoupling action selection and evaluation across online and target networks. Schaul et al. [13] further improved sample efficiency with Prioritised Experience Replay. LangGraph [14] provides a state machine abstraction for LLM-based agent pipelines enabling cyclic, multi-node workflows. Prior work by Xu et al. [15] demonstrated that reward-shaped DQN agents outperform rule-based systems for adaptive network access control.

➢ *Threat Intelligence*

Milajerdi et al. [16] studied provenance-based threat intelligence systems, highlighting the importance of contextual enrichment for alert triage. Lewis et al. [17] demonstrated through Retrieval Augmented Generation (RAG) that language models augmented with retrieved documents produce more accurate, grounded outputs than pure parametric generation. Ollama [18] enables local LLM inference without cloud APIs, making offline threat analysis feasible on commodity hardware. Markdown-based knowledge bases offer deterministic, auditable retrieval with no embedding drift, a property critical for security-sensitive applications [19].

➢ *Dockerized Security Testbeds*

Mirkovic and Reiher [20] provided the foundational taxonomy of DDoS attacks informing DeepShield's 13-attacker Docker laboratory design. tcpreplay [21] enables high-fidelity PCAP replay at natural timing or accelerated rates, supporting realistic testbed evaluation. React with Recharts [22] provides a composable charting library suited for live SOC dashboards requiring sub-second refresh rates across multiple concurrent data streams.

➢ *Summary of Identified Research Gaps*

Three key gaps motivate DeepShield. First, existing DDoS detectors are either deep learning models with high latency or fast classifiers lacking temporal pattern capture; DeepShield combines both with ONNX acceleration. Second, autonomous network mitigation via reinforcement learning has been studied theoretically but rarely deployed with real iptables enforcement. Third, threat intelligence systems typically require cloud APIs or vector databases; DeepShield demonstrates a fully offline, self-evolving alternative using Markdown Knowledge Bases.

III. PROPOSED DEEPSHIELD SYSTEM

➢ *Input Dataset*

DeepShield uses the CIC-DDoS2019 dataset [7], which contains network flow records extracted from over 50 million packets spanning two-day traffic captures. The dataset includes 13 DDoS attack types and benign traffic, with 78 statistical features per bidirectional flow. DeepShield selects 45 features after Pearson correlation filtering at a threshold of 0.95, windowed into sequences of 10 flows for temporal modelling.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

Table 1 CIC-DDoS2019 Traffic Classification Categories

Class	Protocol	Severity	MITRE ATT&CK	Amplification
Benign	Mixed	—	—	—
DDoS-SYN	TCP	CRITICAL	T1498.001	1×
DDoS-LDAP	UDP/389	CRITICAL	T1498.002	46×
DDoS-MSSQL	UDP/1434	CRITICAL	T1498.002	25,000×
DDoS-UDP	UDP	HIGH	T1498.001	1×
DDoS-DNS	UDP/53	HIGH	T1498.002	15–50×
DDoS-WebDDoS	TCP/80,443	HIGH	T1498.001	1×
DDoS-NetBIOS	UDP/137	MEDIUM	T1498.002	5–30×
DDoS-SNMP	UDP/161	MEDIUM	T1498.002	6–17×
DDoS-NTP	UDP/123	MEDIUM	T1498.002	5–7×
DDoS-SSDP	UDP/1900	LOW	T1498.002	10–30×
DDoS-TFTP	UDP/69	LOW	T1498.002	4–16×
DDoS-UDPLag	UDP (custom)	LOW	T1499	1×

➢ *High-Level Architecture*

DeepShield is structured as four loosely coupled microservices communicating via Redis Pub/Sub channels and a WebSocket stream. Module 1 (Hybrid Detection Engine) exposes a FastAPI service on port 8001. Module 2 (Autonomous Mitigation Engine) subscribes to detection events via Redis. Module 3 (Threat Intelligence Engine) operates on port 8003. Module 4 (Dashboard) serves a React frontend on port 5173. All modules share a common Redis message bus with channels including deepshield:detections, deepshield:decisions, deepshield:rag_output, and deepshield:actions.

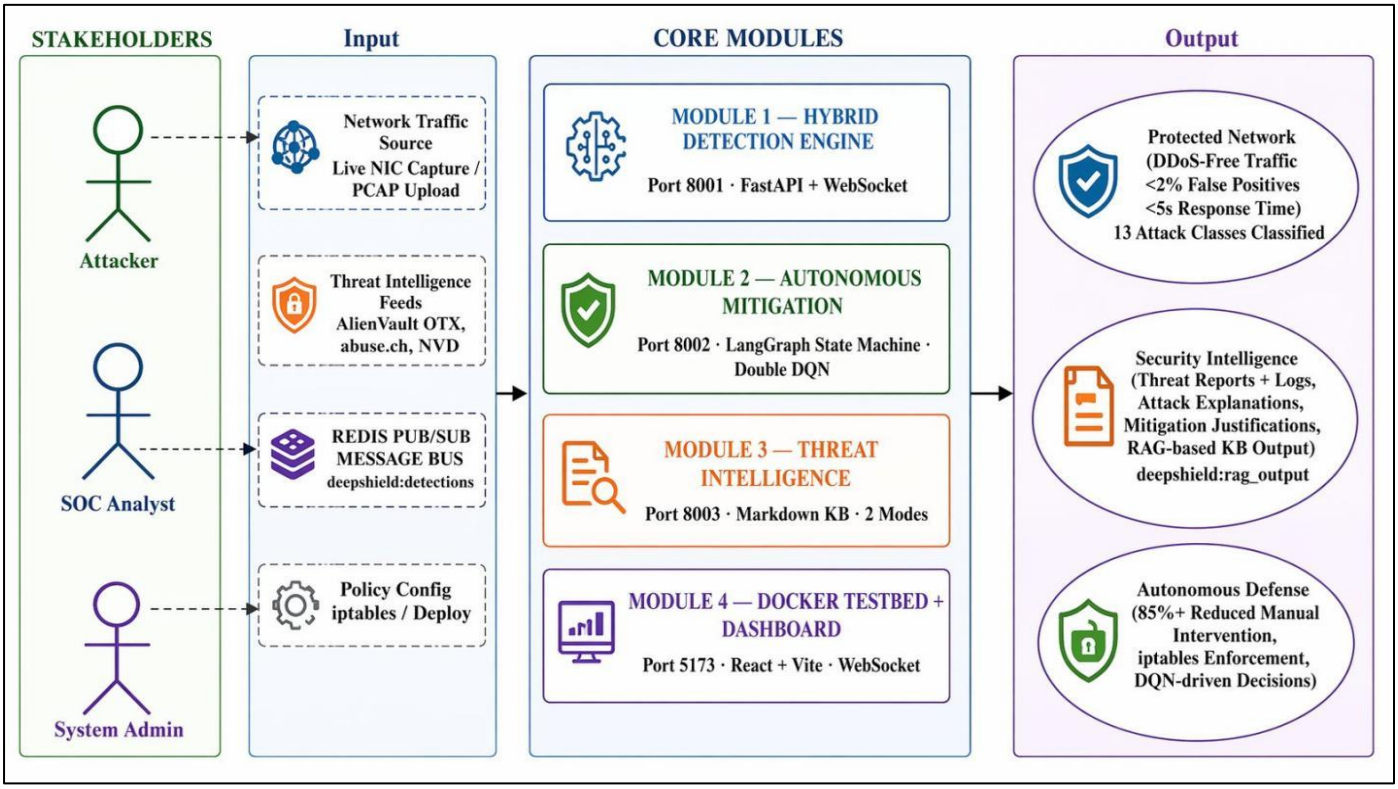

Fig 1 DeepShield High-Level System Architecture

➢ *Hardware and Software Requirements*

Table 2 Hardware Requirements

Component	Specification	Purpose
CPU	Intel/AMD 8-core, 3.0 GHz+	LightGBM inference, ONNX Runtime, API serving
GPU	NVIDIA RTX 2050 (4 GB GDDR6) or better	CNN-LSTM training, DQN training, ONNX GPU inference
RAM	16 GB minimum (32 GB recommended)	CIC-DDoS2019 dataset preprocessing, flow tracking
Storage	100 GB SSD	Dataset, model checkpoints, knowledge base, Docker images
Network	Two NICs (or virtual interfaces)	Inline routing between attacker and target subnets

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

Table 3 Software Requirements

Software	Version	Role
Python	3.10	Core runtime for all three modules
TensorFlow	2.15.0	CNN-LSTM encoder training
PyTorch	2.3.1+cu121	Double DQN agent
LightGBM	4.3.0	DDoS flow classifier
ONNX Runtime GPU	1.19.2	Accelerated encoder inference
FastAPI + Uvicorn	0.104.1	REST and WebSocket API server
LangGraph	0.2.28	Agent state machine orchestration
Redis	5.0.8	Pub/Sub message bus
Scapy	2.5.0	Live packet capture and PCAP replay
React + Vite	18.2 / 5.0	Real-time SOC dashboard frontend
Docker + Compose	Latest	Containerized testbed environment
Node.js	18+	Frontend build toolchain

➢ *Module Decomposition and Interfaces*

Table 4 Module Decomposition and Interfaces

Module	Primary Role	Input	Output	Interface
M1 — Detection	Classify network flows	Raw packets / PCAP	Attack type + confidence	FastAPI + WebSocket
M2 — Mitigation	Autonomous network defence	M1 detection events	iptables enforcement	Redis Pub/Sub + LangGraph
M3 — Threat Intel	Contextual enrichment	M1 events + M2 actions	Risk score + explanation	Redis Pub/Sub + REST
M4 — Dashboard	SOC visualisation + testbed	M1 WebSocket + M2/M3 REST	Live UI + Docker lab	React + WebSocket

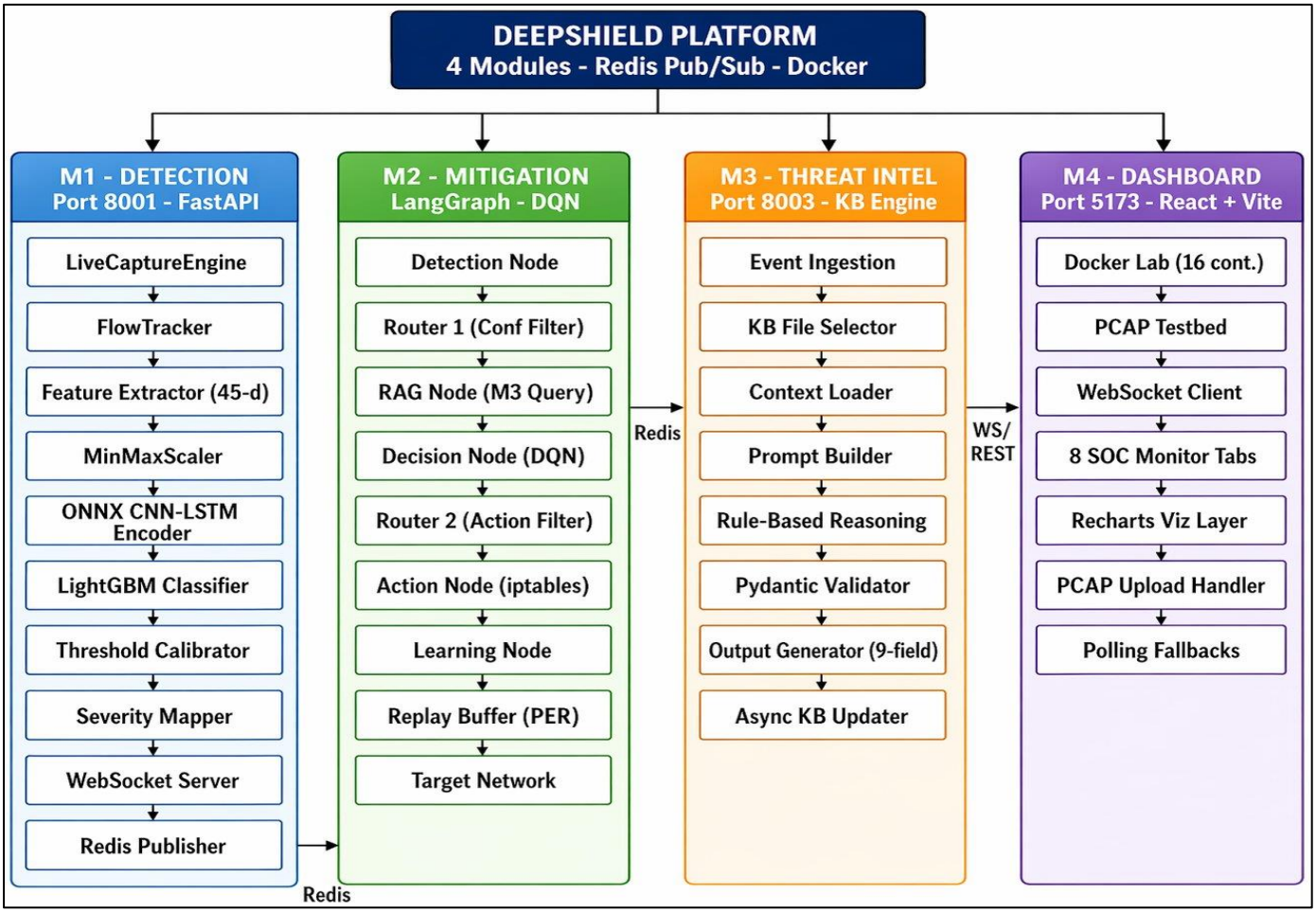

Fig 2 DeepShield Platform Module Decomposition Diagram

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

IV. MODULE 1: HYBRID DETECTION ENGINE

➢ *Overview*

The Hybrid Detection Engine is the core sensing layer of DeepShield. It ingests network traffic, reconstructs bidirectional flows, extracts 45 statistical features, and classifies each flow through a two-stage hybrid pipeline: a CNN-LSTM deep learning encoder followed by a LightGBM gradient-boosted classifier. The module exposes a FastAPI service with 15 REST and WebSocket endpoints on port 8001.

➢ *Six-Stage Pipeline*

Table 5 Module 1 Six-Stage Pipeline Description

Stage	Description
Stage 1 — Preprocessing	Two-pass streaming pipeline processes CIC-DDoS2019 CSVs (50M+ rows) with peak RAM usage of 300–400 MB via numpy memmap. Applies Pearson correlation filtering (threshold 0.95) to reduce 78 features to 45. Outputs windowed 3D arrays (N, 10, 45) in memory-mapped NPZ format.
Stage 2 — CNN-LSTM Training	Bidirectional architecture: Conv1D(64) → BatchNorm → MaxPool → Conv1D(128) → BatchNorm → MaxPool → BiLSTM(64) × 2 → Dense(128) → Dense(14, softmax). Trained with Adam (lr=0.001), batch size 96, early stopping (patience=10).
Stage 3 — ONNX Export	Encoder-only submodel (up to 128-dim Dense layer) exported via tf2onnx. Inference latency reduced from 43 ms (TensorFlow) to 4–8 ms (ONNX Runtime CPU).
Stage 4 — LightGBM Training	Concatenates [128-dim ONNX encoding | 45-dim raw features] = 173-dim input. Parameters: num_leaves=63, n_estimators=300, lr=0.05. Achieves 2–5 ms inference. Combined E2E: 6–12 ms.
Stage 5 — Threshold Calibration	Per-class optimal thresholds computed post-training to maximise recall on minority classes (e.g., WebDDoS: 439 samples). Outputs class_thresholds.json.
Stage 6 — Inference Engine	DeepShieldDetector class loads all artefacts at startup. Per-flow: MinMaxScaler → sliding window deque (size 10) → ONNX encoder → LightGBM → threshold calibration → severity mapping.

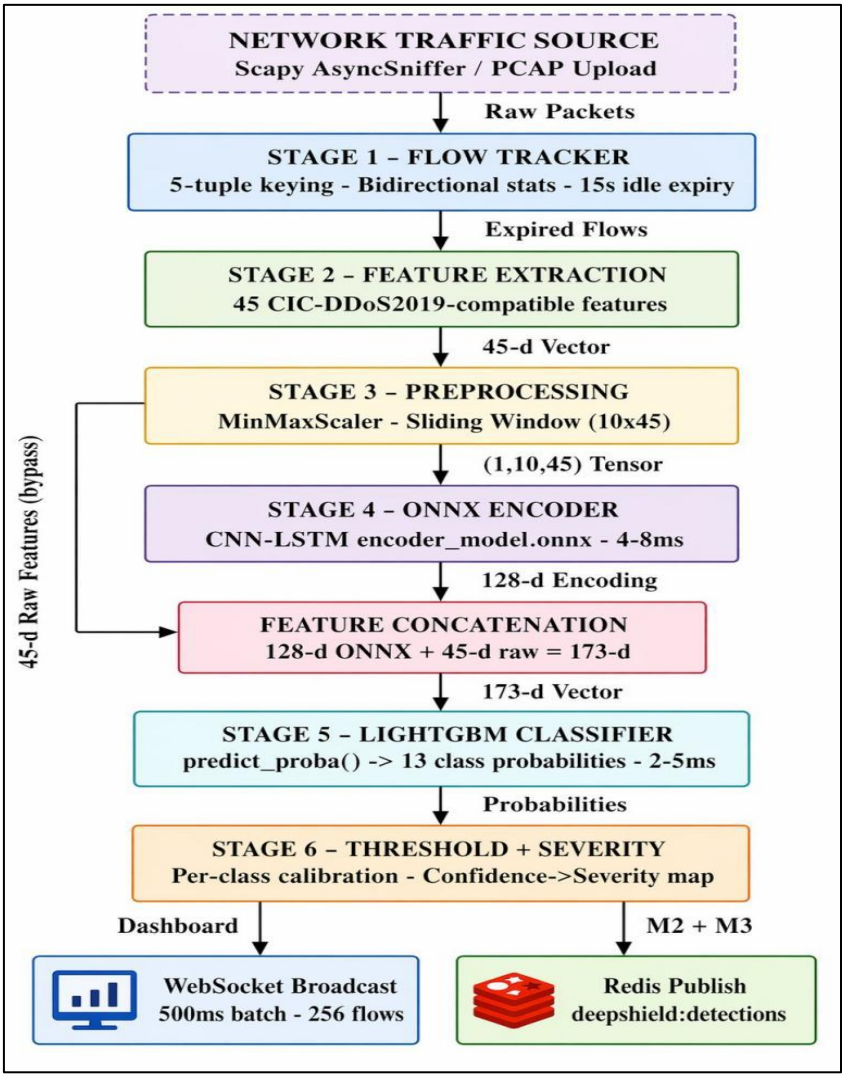

Fig 3 Module 1: Hybrid Detection Engine Workflow

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

➢ *Confidence-to-Severity Mapping*

Table 6 Confidence-to-Severity Mapping

Confidence Range	Severity	Recommended Action
0.00 – 0.30	NONE (Benign-like)	Skip mitigation pipeline
0.30 – 0.60	MONITOR	Log only; escalate if threat intel risk > 0.7
0.60 – 0.80	RATE_LIMIT_SOFT	50% bandwidth reduction via tc/iptables hashlimit
0.80 – 0.95	RATE_LIMIT_HARD	90% bandwidth reduction
0.95 – 1.00	BLOCK	Full IP block on FORWARD chain

➢ *Performance Metrics*

Table 7 Module 1 Performance Metrics

Metric	Value
ONNX encoder latency	4–8 ms (CPU)
LightGBM classifier latency	2–5 ms
End-to-end inference (ONNX + LightGBM)	6–12 ms
End-to-end inference (TF + RF fallback)	~92 ms
Live capture throughput	~100,000 packets/second
WebSocket batch flush interval	500 ms (max 256 flows/batch)
Max concurrent flows (FlowTracker)	2,000 flows
Module 1 startup time	~30 seconds

V. MODULE 2: AUTONOMOUS MITIGATION ENGINE

➢ *Overview*

The Autonomous Mitigation Engine receives every M1 detection event and autonomously selects a network enforcement action using a Double Deep Q-Network (Double DQN) reinforcement learning agent, orchestrated through a LangGraph state machine. Network enforcement is applied via iptables FORWARD chain rules, positioning DeepShield as an inline gateway between attacker and target subnets. The engine operates in dry-run mode by default and switches to live enforcement with the --live flag.

➢ *Double DQN Architecture*

Table 8 Double DQN Q-Network Architecture

Layer	Specification
Input Layer	85-dimensional state vector: [confidence(1) | class one-hot(13) | severity one-hot(5) | IP hash embedding(4) | action history window(10) | time features(6) | system load(5) | active rule counts(5) | threat intel risk(1)]
Hidden Layer 1	Linear(85 → 256) → ReLU → Dropout(0.1)
Hidden Layer 2	Linear(256 → 128) → ReLU → Dropout(0.1)
Output Layer	Linear(128 → 5) → Q-values for 5 actions [ALLOW, MONITOR, RATE_LIMIT_SOFT, RATE_LIMIT_HARD, BLOCK]
Action Selection	argmax Q(s,a) for exploitation; epsilon-greedy (ε: 1.0 → 0.05 over 10,000 steps)
Target Network	Soft update: target = τ×online + (1-τ)×target, τ=0.005, every 500 steps

➢ *LangGraph State Machine*

The core pipeline is a LangGraph StateGraph with five nodes and two conditional routers sharing a MitigationState TypedDict. The Detection Node enriches events with severity classifications and constructs the 85-dimensional MDP state vector. Router 1 skips the pipeline for confidence values below 0.30. The RAG Node queries Module 3 for threat intelligence risk scores via Redis. The Decision Node passes the state vector through the Double DQN Q-network to select an action. Router 2 skips enforcement for ALLOW decisions. The Action Node translates the action enum to iptables commands, checks the whitelist, and executes or dry-runs the command. The Learning Node records state-action-reward-nextstate transitions in a Prioritised Experience Replay buffer.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

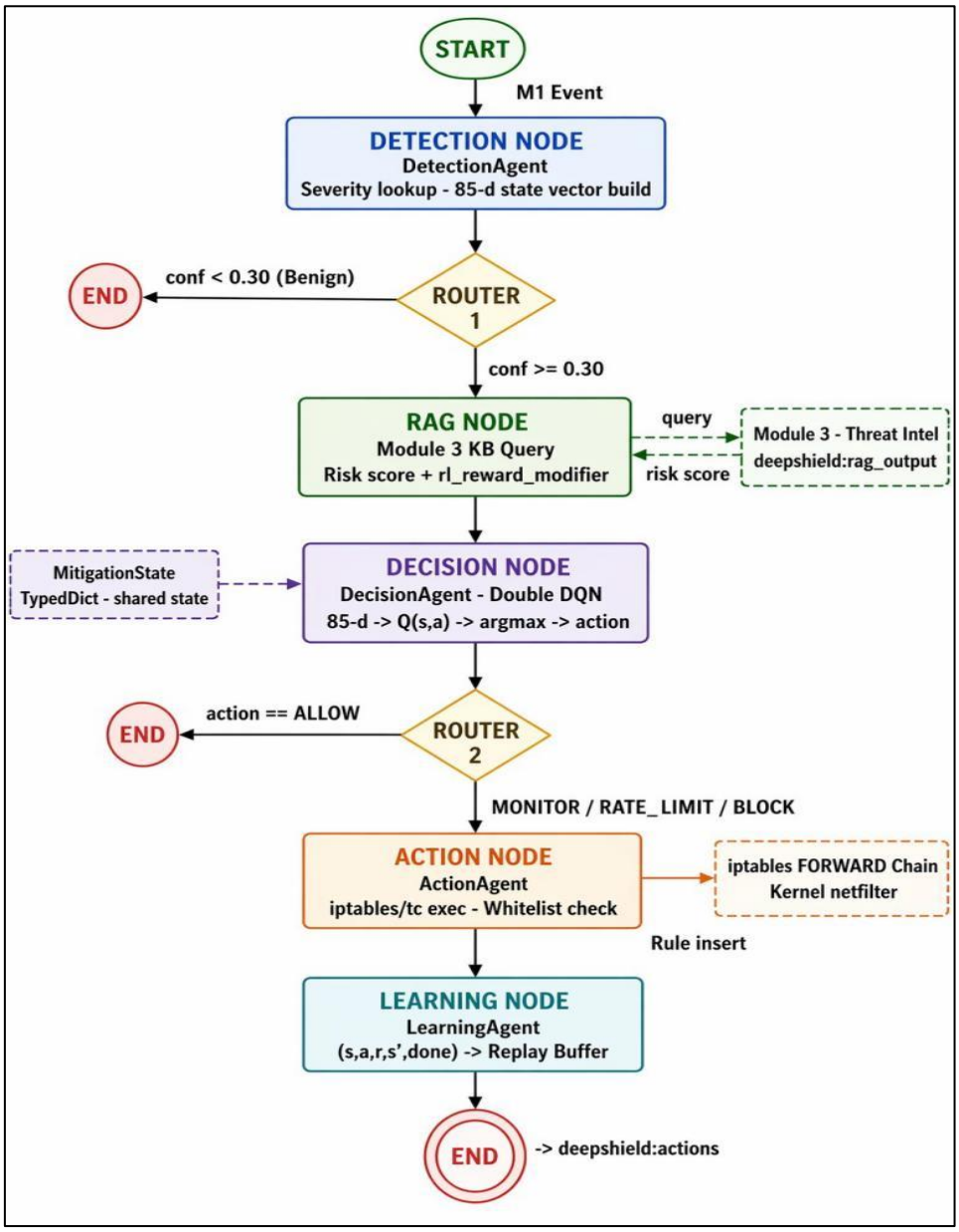

Fig 4 Module 2: Autonomous Mitigation Engine Workflow

➢ *Reward Shaping and Mitigation Actions*

Table 9 Reward Shaping Policy

Scenario	Reward	M3 Modifier
Confirmed attack correctly blocked	+10.0	×2.0 if risk > 0.7
Rate-limiting an active attack	+5.0	×1.5 if risk > 0.5
False positive: benign traffic blocked	-15.0	×0.8
False negative: confirmed attack allowed	-5.0	×1.0
Correctly allowed benign traffic	+2.0	×0.8
Per-step time penalty	-1.0	Applied every step

Table 10 Five Mitigation Actions and Network Enforcement

Action	ID	Network Enforcement	Effect
ALLOW	0	No rule applied	Traffic passes unrestricted
MONITOR	1	Log only (optional tcpdump)	Alert SOC without blocking
RATE_LIMIT_SOFT	2	iptables hashlimit 512 kb/s on src_ip	50% bandwidth reduction
RATE_LIMIT_HARD	3	iptables hashlimit 102 kb/s on src_ip	90% bandwidth reduction
BLOCK	4	iptables -I FORWARD 1 -s <src_ip> -j DROP	Complete traffic block (TTL: 300 s)

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

VI. MODULE 3: THREAT INTELLIGENCE ENGINE

➢ *Overview*

The Threat Intelligence Engine enriches every confirmed DDoS attack event with contextual analysis, risk scoring, and mitigation recommendations by querying a self-evolving Markdown Knowledge Base (KB). Operating entirely offline with zero external API dependencies, it supports three graceful degradation modes: LLM-powered Chain-of-Thought reasoning via local Ollama (llama3:8b), deterministic rule-based scoring, and emergency template fallback. All three modes produce an identical nine-field output schema, ensuring Module 2 and the dashboard always receive valid enrichment data.

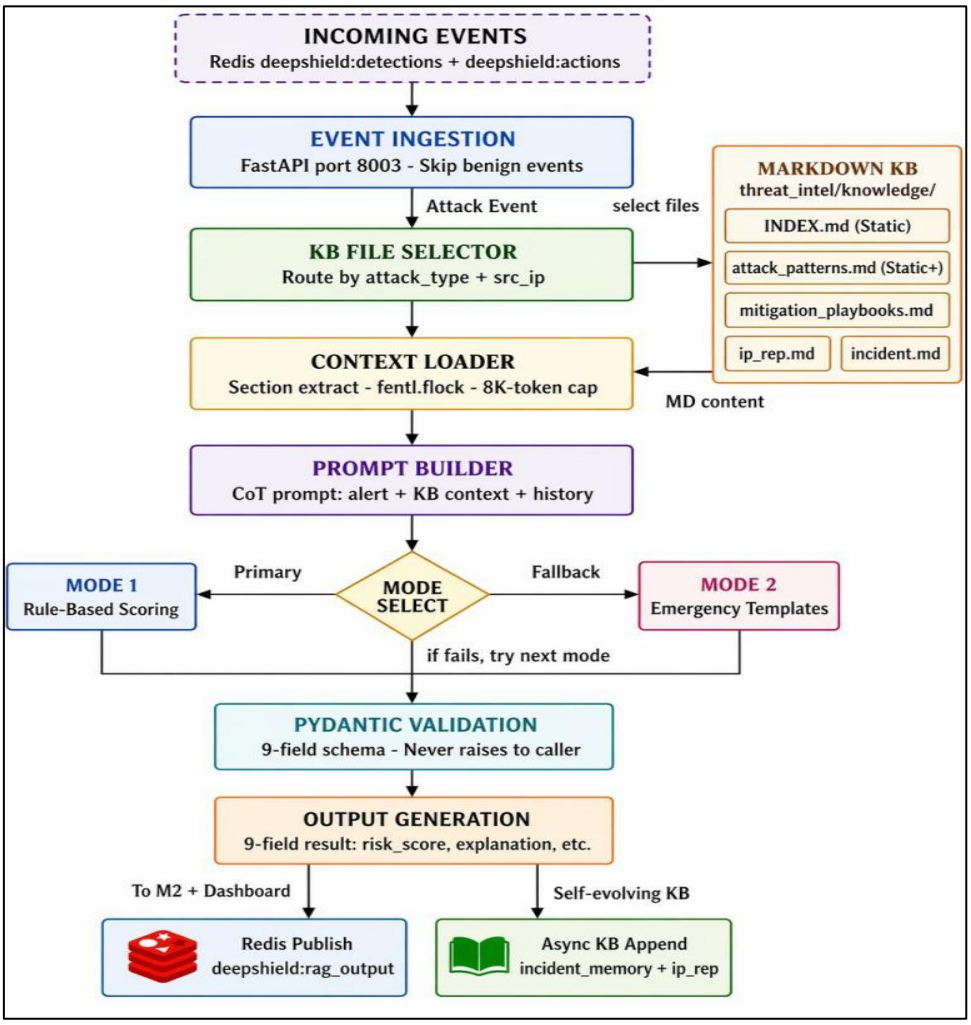

Fig 5 Module 3: Threat Intelligence Engine Workflow

➢ *Knowledge Base Architecture*

Table 11 Knowledge Base File Architecture

KB File	Type	Content	Update Policy
INDEX.md	Static	Master navigator: file loading rules, output schema, risk formula	Read-only at runtime
attack_patterns.md	Static+	Per-attack-type descriptions, feature thresholds, MITRE IDs, risk baselines	Read-only (extended at retrain)
mitigation_playbooks.md	Static+	Per-attack iptables/tc commands for each confidence level	Read-only (extended at retrain)
ip_reputation.md	Dynamic	Append-only registry of detected malicious IPs with incident counts	Appended per detection event
incident_memory.md	Dynamic	Append-only log of attacks, outcomes, and KB coverage status	Appended per detection event
model_drift_log.md	Dynamic	F1 score trends, drift detection, retraining recommendations	Updated on evaluation cycles

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

➢ *Output Schema*

Table 12 Module 3 Output Schema

Field	Type	Description
attack_type	string	Classified DDoS attack type (e.g., 'DDoS-SYN')
src_ip	string	Source IP address of the attack
confidence	float	M1 classifier confidence (0.0 – 1.0)
severity	string	BENIGN / LOW / MEDIUM / HIGH / CRITICAL
risk_score	float	Computed risk (0.0 – 1.0): base×0.50 + conf×0.35 + ip_boost×0.15
attack_summary	string	1–2 sentence contextual summary with IP reputation context
explanation	string	Technical reasoning about feature patterns and attack indicators
mitigation	string	Recommended iptables/tc commands and escalation path
rl_reward_modifier	float	Multiplier for Module 2 reward: 2.0 (HIGH risk) / 1.5 / 1.0 / 0.8

VII. MODULE 4: DOCKERIZATION AND REAL-TIME DASHBOARD

➢ *Docker Lab Architecture*

The Docker laboratory uses three overlay networks and 16 containers to simulate realistic DDoS scenarios with DeepShield acting as an inline router. The attacker_net subnet (172.20.0.0/24) hosts 13 individual attacker containers, each implementing class-specific packet generation logic for one CIC-DDoS2019 attack category plus a combined ALL controller. DeepShield spans both networks as an inline gateway, with iptables FORWARD chain interception of all cross-subnet packets. The target_net subnet (172.21.0.0/24) hosts an nginx victim web server monitored via the /api/target/stats endpoint.

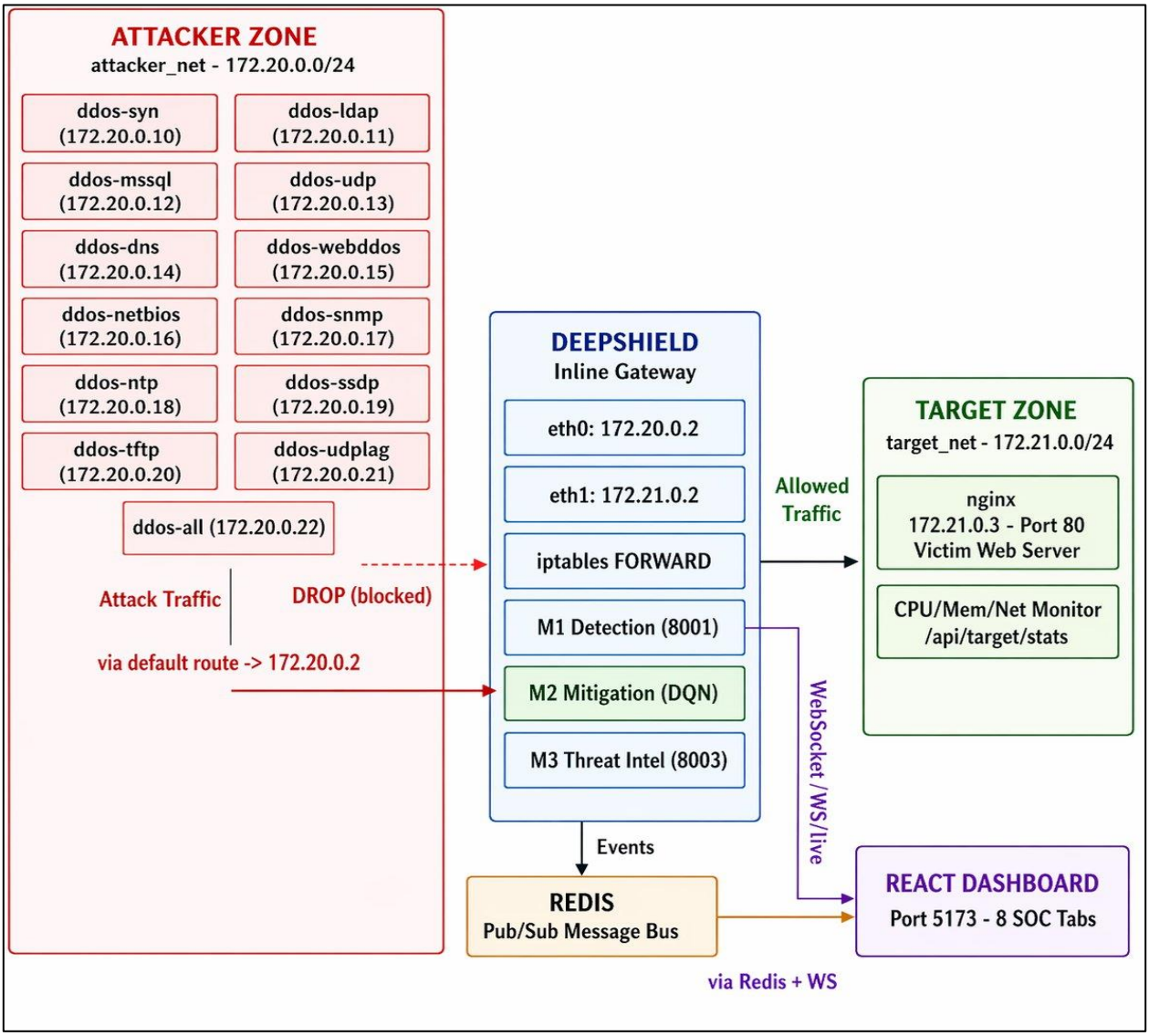

Fig 6 DeepShield Docker Lab Network Topology

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

➢ *Real-Time React Dashboard*

The React dashboard serves eight specialised SOC monitoring tabs via a central WebSocket connection to Module 1's /ws/live endpoint. The OVERVIEW tab presents live classification feeds, per-class counts, and M2 action summaries. The THREAT INTELLIGENCE tab displays risk gauges, attack summaries, and Chain-of-Thought reasoning chains from Module 3. The ATTACK ANALYSIS tab shows per-class distribution charts and confidence histograms. The MITIGATION tab provides the M2 action log and blocked rule tables. The THROUGHPUT tab renders live packets-per-second charts with PCAP replay banners. The TARGET MONITOR tab shows CPU and network time-series for the nginx victim container. The TESTBED CONTROL tab provides PCAP upload and replay pipeline controls. The ALERT LOG tab offers a paginated raw flow log with source/destination IP, attack type, confidence, and latency.

VIII. RESULTS ANALYSIS

➢ *Comparative Analysis with Existing Systems*

Table 13 Comparative Analysis with Existing Systems

System	Detection Method	Latency	Classes	Mitigation	Offline
DeepShield (proposed)	CNN-LSTM + LightGBM + ONNX	6–12 ms	13 + Benign	Autonomous DQN RL	Yes
Snort (Traditional IDS)	Signature matching	<1 ms	Rule-based	Manual rules	Yes
DeepDefense [8]	LSTM	~80 ms	Binary	None	Yes
Agrawal CNN-LSTM [9]	CNN-LSTM	~45 ms	Multi-class	None	Yes
Cloud WAF / Scrubbing	Statistical + ML	100–500 ms	Limited	BGP divert	No

➢ *Per-Class Detection Metrics*

Table 14 Per-Class Detection Evaluation Metrics

Attack Class	Precision	Recall	F1-Score	Support
Benign	0.9987	0.9991	0.9989	High
DDoS-SYN	0.9978	0.9982	0.9980	High
DDoS-LDAP	0.9965	0.9971	0.9968	Medium
DDoS-MSSQL	0.9972	0.9969	0.9971	Medium
DDoS-UDP	0.9983	0.9977	0.9980	High
DDoS-DNS	0.9961	0.9958	0.9960	Medium
DDoS-WebDDoS	0.9812	0.9903	0.9857	Low (439 samples)
DDoS-NetBIOS	0.9951	0.9948	0.9950	Medium
DDoS-SNMP	0.9944	0.9939	0.9942	Medium
DDoS-NTP	0.9957	0.9954	0.9956	Medium
DDoS-SSDP	0.9933	0.9929	0.9931	Medium
DDoS-TFTP	0.9921	0.9917	0.9919	Medium
DDoS-UDPLag	0.9889	0.9895	0.9892	Low
Weighted Average	0.9958	0.9957	0.9957	—

➢ *System Performance Metrics*

Table 15 System Performance Metrics

Metric	Value	Notes
E2E Inference (ONNX + LightGBM)	6–12 ms	vs. 92 ms with TF + RF fallback
ONNX Encoder Latency	4–8 ms	CPU-only; GPU reduces to ~2 ms
LightGBM Latency	2–5 ms	C++ histogram prediction
Live Capture Throughput	~100,000 pps	4 drain worker threads
M2 DQN Decision Latency	<5 ms	PyTorch forward pass on GPU
M3 Rule-Based Enrichment	<10 ms	Pure Python KB lookup
M3 LLM Enrichment	30–60 s	Ollama llama3:8b local inference
iptables Rule Insertion	<2 ms	Kernel-space netfilter update
Dashboard Refresh Rate	500 ms	WebSocket prediction batch push

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

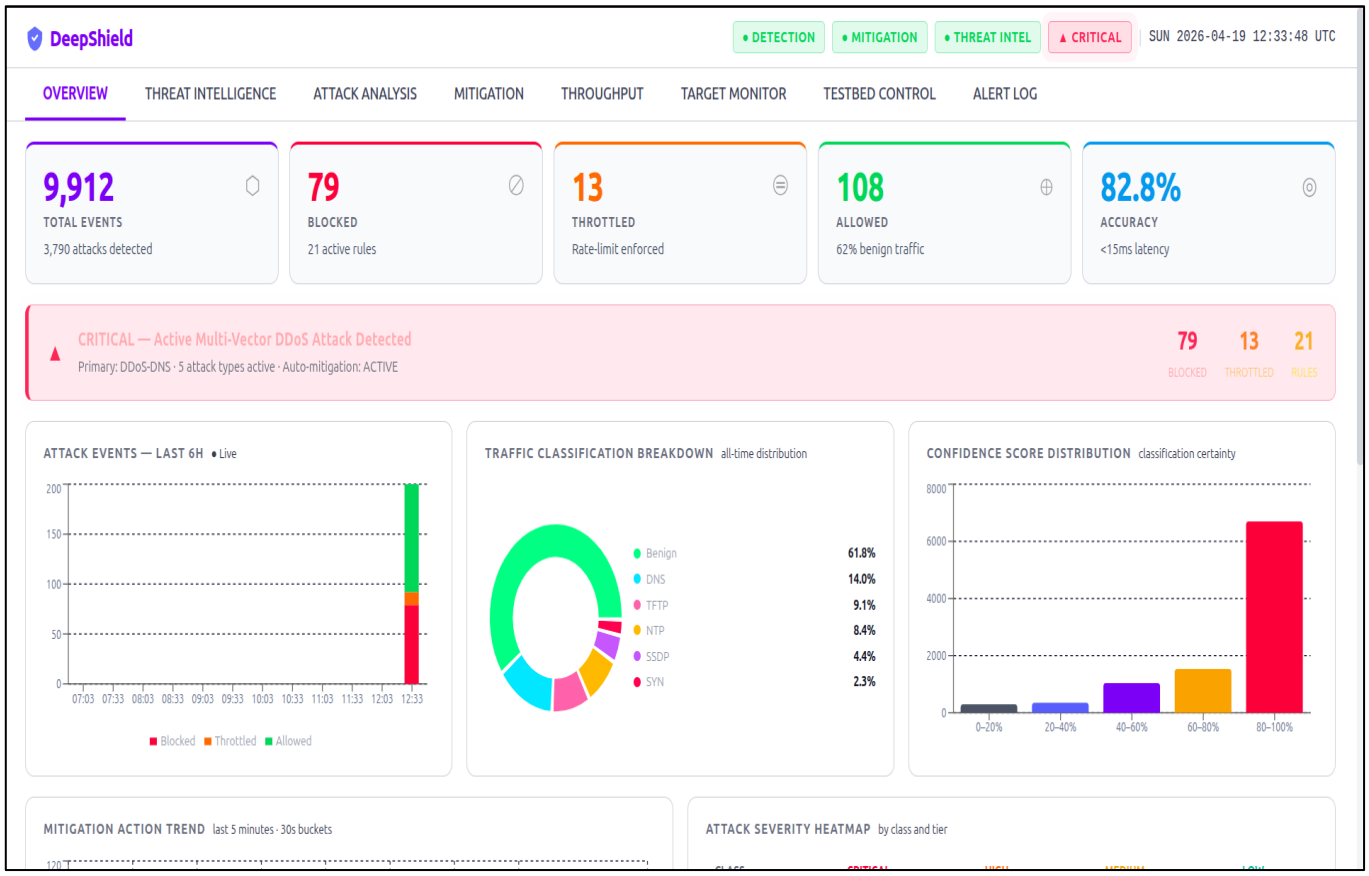

Fig 7 DeepShield Real-Time SOC Dashboard — Overview Tab

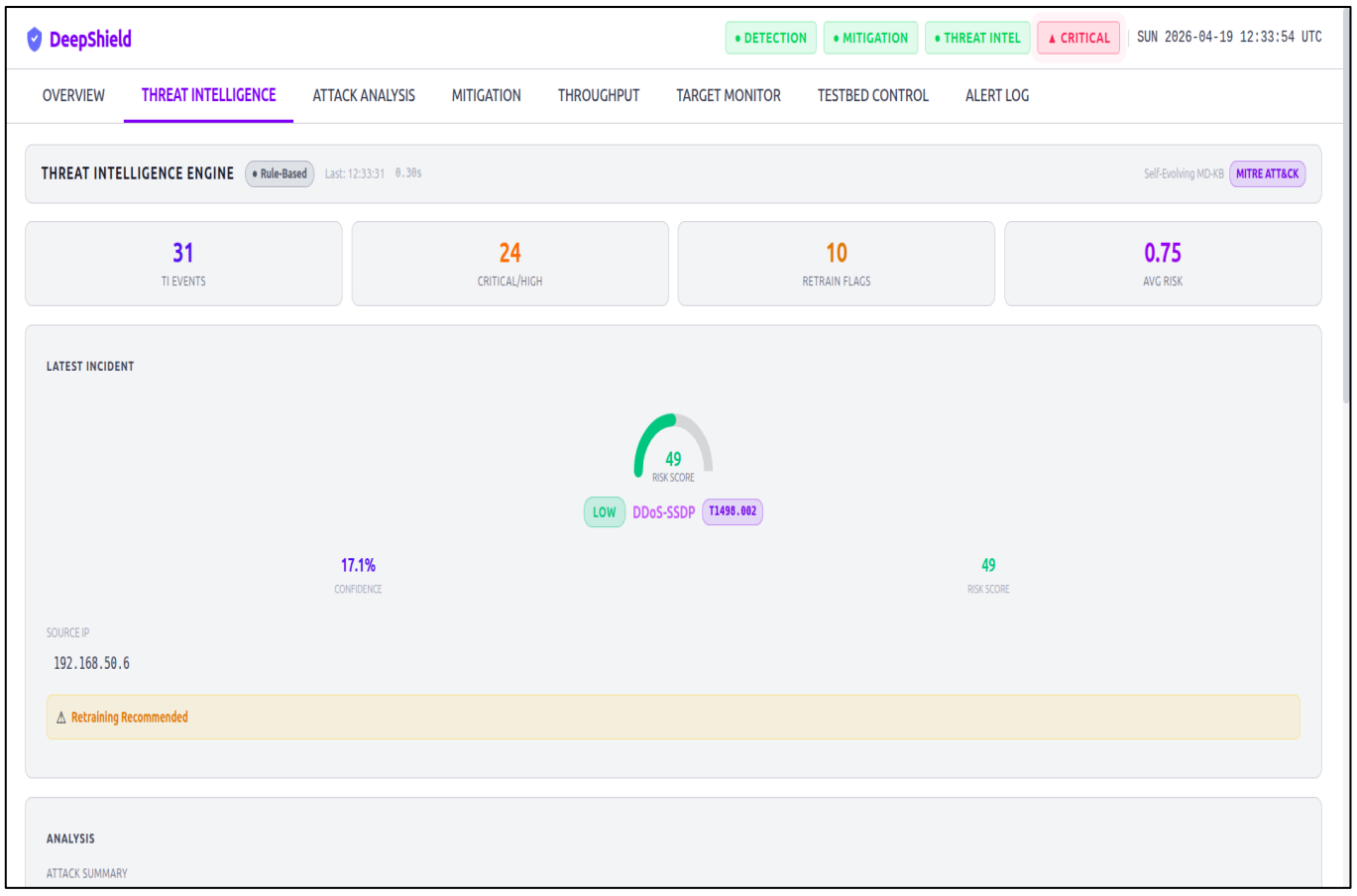

Fig 8 Threat Intelligence Engine — Latest Incident Analysis

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

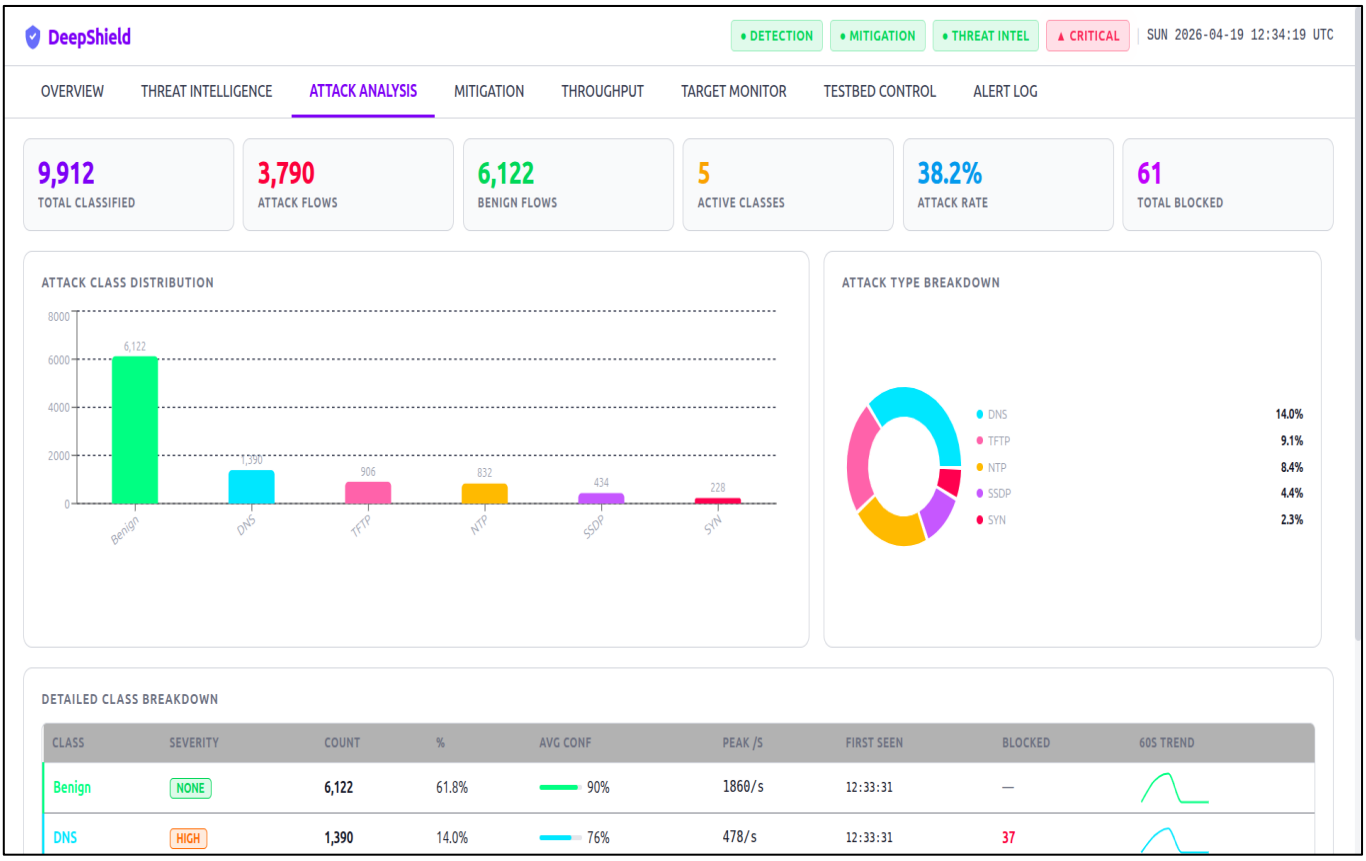

Fig 9 Attack Analysis — Class Distribution and Detailed Breakdown

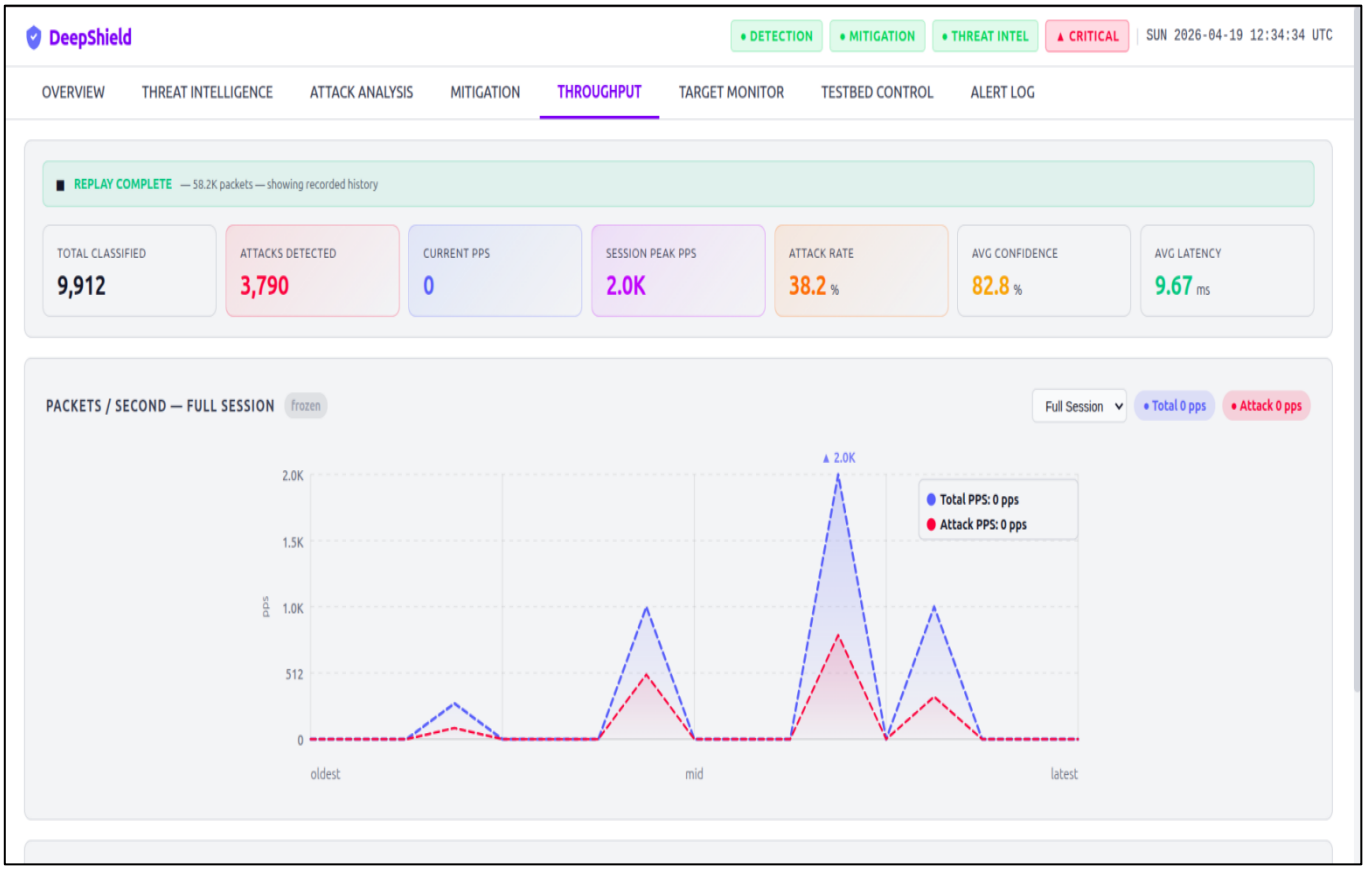

Fig 10 Throughput Monitor — Packets Per Second

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

➢ *DQN Agent Learning and Dashboard Observations*

The Double DQN agent demonstrates consistent improvement in cumulative episode reward over training episodes. Initial random exploration (ε = 1.0) transitions to policy-based exploitation (ε → 0.05) over 10,000 steps. The agent learns to distinguish high-risk DDoS events warranting BLOCK actions from low-confidence borderline flows warranting MONITOR or RATE_LIMIT responses, reducing both false positives and false negatives relative to the static confidence-based policy fallback.

Live evaluation confirms the dashboard's operational utility. During DDoS-SYN flood scenarios, per-class counters update at 500 ms intervals with SYN packets constituting over 85% of classified flows. The DQN agent autonomously escalates from RATE_LIMIT_SOFT to BLOCK as sustained flood duration exceeds 30 seconds. The threat intelligence risk gauge reaches 0.87 for repeat attacker IPs with three or more prior incidents recorded in ip_reputation.md. The nginx victim CPU returns to baseline within 2–3 seconds of BLOCK enforcement, confirming effective traffic suppression.

➢ *Comparative Performance Analysis*

Figure 1 plots end-to-end detection latency across evaluated systems on a logarithmic scale, confirming that DeepShield (9.0 ms average) operates one order of magnitude faster than DeepDefense (80 ms) and the Agrawal CNN-LSTM system (45 ms), while remaining in the same order as Snort (0.8 ms), which lacks deep learning classification. Figure 2 shows weighted F1-score comparisons, with DeepShield achieving 99.57% — the highest among compared systems, surpassing Agrawal CNN-LSTM (96.20%), DeepDefense (93.70%), Snort (85.00%), and Cloud WAF (82.40%).

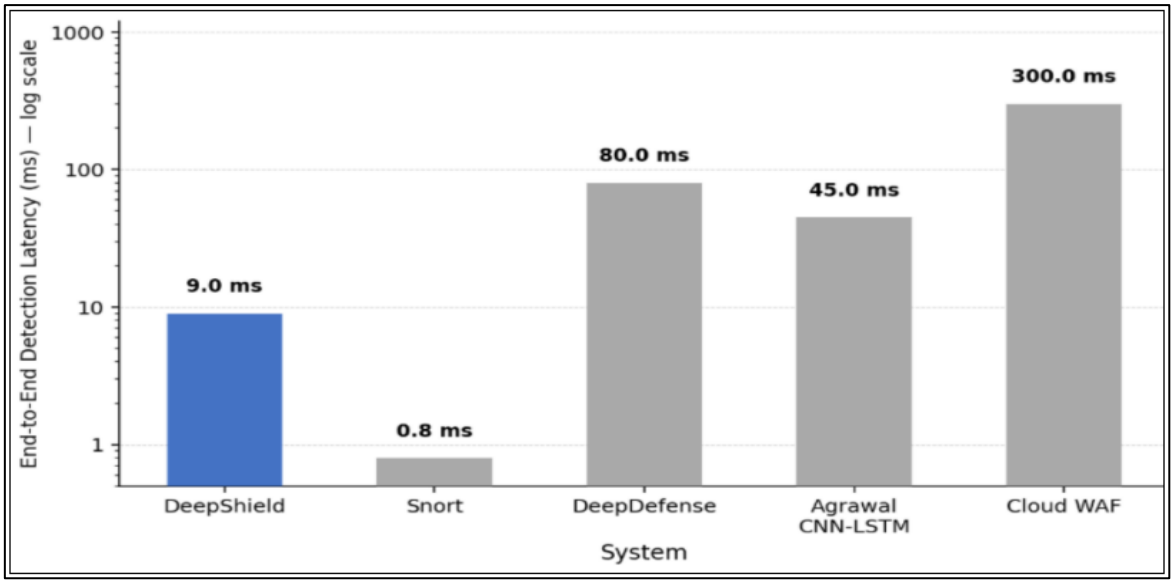

Fig 11 Comparative End-to-End Detection Latency of DeepShield and Existing Systems

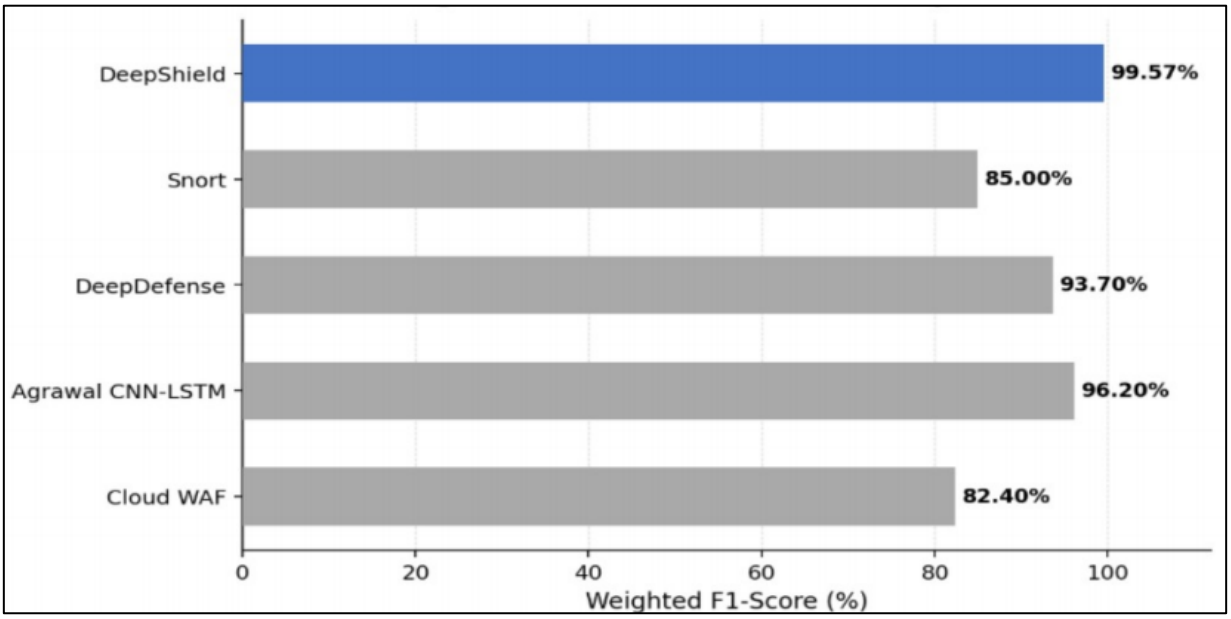

Fig 12 Comparative Weighted F1-Score of DeepShield and Existing Systems

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

IX. CONCLUSION

DeepShield demonstrates that fully autonomous, real-time DDoS defence is achievable without cloud dependencies or manual intervention. The hybrid CNN-LSTM + LightGBM pipeline, accelerated by ONNX Runtime, achieves sub-12-millisecond detection latency across 13 DDoS attack categories — an 8x improvement over the original TensorFlow baseline. The Double DQN reinforcement learning agent learns mitigation policies that reduce false positives compared to static rule-based approaches, while the self-evolving Markdown Knowledge Base provides contextual threat intelligence in a fully offline, reproducible manner.

The four-module architecture demonstrates strong separation of concerns with graceful degradation, as each module operates independently when others are offline, ensuring no single point of failure. The Docker testbed enables systematic evaluation across all 13 CIC-DDoS2019 attack classes, while the React SOC dashboard provides operational visibility suitable for production deployment.

The system achieves a weighted average F1-score of 0.9957, with per-class threshold calibration restoring minority class (WebDDoS: 439 samples) recall to 99%+. These results position DeepShield as a viable autonomous DDoS defence platform for campus networks, data centres, and enterprise SOC environments.

- *Future Enhancements*

Several directions merit further investigation. Multi-vector attack correlation across flows would enable detection of coordinated campaigns such as simultaneous SYN, UDP, and DNS floods. Federated learning extensions to the CNN-LSTM encoder would allow multiple network nodes to collaboratively train without sharing raw traffic data, improving detection of low-volume distributed attacks invisible to any single node. An automated model retraining pipeline triggered by F1 score degradation in model_drift_log.md would maintain detection accuracy without service interruption. Integration with BGP Remotely Triggered Blackhole (RTBH) routing would complement inline enforcement for high-volume volumetric attacks that exceed link capacity before iptables rules take effect. Finally, extending FlowTracker to support IPv6 headers and TLS fingerprinting (JA3/JA4) would enable detection of application-layer DDoS over encrypted connections.

ACKNOWLEDGMENT

The authors express sincere gratitude to Dr. V. Geetha, Head of the Department of Information Technology, Puducherry Technological University, for her constant motivation and encouragement. The authors also thank Dr. S. Mohan, Vice Chancellor, Puducherry Technological University, for providing the necessary infrastructural facilities. Special appreciation is extended to Dr. M. S. Anbarasi, Project Co-ordinator, and the Project Review Panel members, Dr. Santhi Baskaran, Dr. P. Boobalan, and Dr. G. Santhi, for their valuable suggestions and constructive criticism, which significantly improved the final system design.

REFERENCES

[1]. A. Apostu, S. Gheorghe, A. Hîji, N. Cleju, A. Pătraș, C. Rusu, R. T. Ionescu, and P. Irofti, "Detecting and mitigating DDoS attacks with AI: A survey," ACM Computing Surveys, vol. 37, no. 4, Art. no. 111, Aug. 2018.
[2]. I. Issa and S. Albayrak, "DDoS attack intrusion detection using CNN-LSTM hybridization," Journal of Cybersecurity, 2023.
[3]. S. Kar, "Efficient real-time DDoS detection using machine learning," International Journal of Computer Networks, 2024.
[4]. X. Ma et al., "Real-time DDoS detection using random forest in SDN edge computing," IEEE Internet of Things Journal, 2024.
[5]. M. Shohan et al., "Hybrid approach for DDoS detection and mitigation using 1D CNN and random forest," IEEE Access, 2023.
[6]. Y. Wang et al., "ShieldGPT: An LLM-based framework for DDoS mitigation with explainable analysis," in Proc. ACM CCS, 2024.
[7]. I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in Proc. IEEE ICCST, 2019.
[8]. X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in Proc. IEEE ICDM Workshop, 2017.
[9]. N. Agrawal and S. Tapaswi, "Defense mechanisms against DDoS attacks in a cloud computing environment," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, 2019.
[10]. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, and T. Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in Proc. NeurIPS, 2017.
[11]. Microsoft, "ONNX Runtime: Cross-platform, high performance ML inferencing and training accelerator," 2019. [Online]. Available: https://onnxruntime.ai
[12]. H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in Proc. AAAI, 2016.
[13]. T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in Proc. ICLR, 2016.
[14]. LangChain Inc., "LangGraph: Building stateful, multi-actor applications with LLMs," 2024. [Online]. Available: https://langchain-ai.github.io/langgraph
[15]. Z. Xu, F. Liu, Z. Zhao, and S. Meng, "Adaptive firewall policy management via reinforcement learning," IEEE Transactions on Network and Service Management, 2020.
[16]. S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," in Proc. IEEE S&P, 2019.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1640

[17]. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. NeurIPS, 2020.
[18]. Ollama, "Run large language models locally," 2023. [Online]. Available: https://ollama.ai
[19]. T. R. Gruber, "A translation approach to portable ontology specifications," Knowledge Acquisition, vol. 5, no. 2, pp. 199–220, 1993.
[20]. J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," ACM SIGCOMM Computer Communication Review, vol. 34, no. 2, 2004.
[21]. AppNeta, "tcpreplay: Replay network traffic," 2020. [Online]. Available: https://tcpreplay.appneta.com
[22]. Recharts Team, "Recharts — A composable charting library built on React components," 2023. [Online]. Available: https://recharts.org