IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15251

THREAT-DETECT: An Integrated Deep Learning and Automated Incident Response Framework for Cybersecurity Threat Detection

Dr. P. Esther Jebarani[1], Ms. G. Shreeshaa[2]

Associate Professor, Dept of Computer Science with Cognitive Systems, Dr.N.G.P. Arts and Science College, Coimbatore, India[1]

Student, Dept of Computer Science with Cognitive Systems, Dr.N.G.P. Arts and Science College, Coimbatore, India[2]

Abstract: Signature-based defences cannot detect novel cyber threats in real time. We present THREAT-DETECT, a containerised platform that unifies a three-model stacking ensemble (Random Forest, Bidirectional LSTM, and 1D-CNN) with a DAG-based automated incident response engine. A 29-dimensional feature vector spanning lexical, WHOIS, and network signals feeds all models; SHAP attributions provide per-prediction explainability; and an uncertainty-based active learning controller continuously improves model quality. The Playbook Engine translates scored threat events into DNS sinkholing, firewall rule injection, TheHive case creation, MISP IOC export, and Slack alerting via auditable, rollback-capable DAG playbooks. Evaluation on 487K labelled examples yields ensemble AUC-ROC 0.9970 and median end-to-end response latency of 711 ms.

Keywords: threat detection · ensemble learning · deep learning · active learning · incident response · SHAP explainability · DAG playbook · DNS sinkholing · MISP · TheHive

I. INTRODUCTION

Cyber adversaries register hundreds of thousands of malicious domains daily, many active for only hours before rotation. Signature databases miss zero-day domains, DGAs, and brand typosquats by design. THREAT-DETECT addresses the full pipeline from indicator ingestion to automated containment as a production-grade microservice platform. Its principal contributions are: (1) a 29-dimensional feature pipeline with sub-millisecond extraction; (2) a stacking ensemble achieving AUC 0.9970; (3) per-prediction SHAP attributions for analyst auditability; (4) uncertainty-based active learning with automatic retraining; and (5) a DAG Playbook Engine with dependency resolution, approval gates, and rollback.

II. RELATED WORK

Passive DNS analysis pioneered domain-level threat detection [1], but requires query volume unavailable for newly registered domains. Character-level LSTMs [2] and CNNs address this cold-start problem by operating on domain text alone. Ensemble stacking consistently outperforms individual classifiers in security contexts [3]. SHAP [4] provides theoretically grounded feature attributions that increase analyst trust. Uncertainty sampling [5] reduces annotation cost versus random selection. Commercial SOAR platforms (Splunk SOAR, Palo Alto XSOAR) automate SOC workflows but are expensive and vendor-locked; THREAT-DETECT provides an open-source alternative with comparable expressiveness.

III. SYSTEM ARCHITECTURE

A. High-Level Design

THREAT-DETECT follows a microservice architecture comprising three principal services—Watcher, the ML Pipeline, and the Playbook Engine—communicating over isolated Docker bridge networks. Figure 1 illustrates the end-to-end data flow. Each service is independently deployable, horizontally scalable, and health-monitored. Shared state is minimised: the ML Pipeline exposes a stateless REST API, while persistent state (model artefacts, label store, execution log) is confined to purpose-specific backing stores.

B. Network Topology and Security

Services are partitioned across three bridge networks: watcher_net (10.10.10.0/24) for Watcher and its PostgreSQL instance; ml_net (10.10.20.0/24) for the ML Pipeline, Redis, MLflow, and ML PostgreSQL; and pe_net (10.10.30.0/24) for the Playbook Engine and its PostgreSQL instance. The Playbook Engine is the only service attached to both ml_net and pe_net, reflecting its role as the consumer of ML scores and the orchestrator of response actions. No backing store is exposed on the host network. All inter-service communication uses Docker DNS names (e.g., ml_pipeline:8001), ensuring portability.

C. Deployment and Operations

Each service is packaged as a multi-stage Docker image. The build stage installs Python dependencies into a /install prefix; the runtime stage copies only that prefix and the application source code, excluding build tools, compilers, and test artefacts. Containers run as non-root users. Health checks are defined at the Docker level for every service, enabling Docker Compose's dependency ordering (service_healthy conditions) to enforce correct startup sequencing. A deploy.sh wrapper script provides a unified interface for start, stop, log streaming, health status reporting, model training, and image rebuilding.

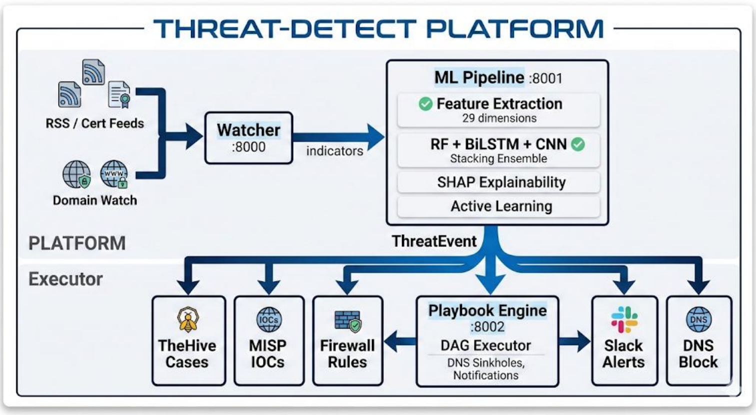

Fig. 1 End-to-End data flow through the THREAT-DETECT platform

IV. MACHINE LEARNING PIPELINE

A. Feature Engineering

The feature vector contains 29 numerical attributes across three groups. Eleven lexical features capture domain length, Shannon character entropy, digit/vowel/consonant ratios, hyphen count, subdomain depth, TLD risk score, minimum Levenshtein distance to 1,000 brand domains, and dictionary word coverage. The Levenshtein check operates on the full SLD and first two hyphen-split segments, catching constructs such as paypa1-secure-login.xyz. Eight WHOIS/RDAP features encode domain age, registrar abuse score, registrant country risk, privacy flag, days since last update, name-server count and entropy, and MX presence. Ten network features include A-record count, IP geo-risk, ASN reputation, TTL, DNS response latency, blocklist membership, reverse-DNS mismatch, certificate age, SAN count, and issuer risk. WHOIS data is Redis-cached (24 h TTL); cache-miss features are median-imputed within a 3 s timeout.

B. Ensemble Architecture

Three base models train on stratified 5-fold CV splits with SMOTE applied per training fold [6]. The Random Forest (400 trees) uses TreeSHAP for attribution; Platt scaling [7] calibrates scores to probabilities. The Bidirectional LSTM encodes characters in 16-d embeddings, processes them through two BiLSTM layers (hidden dim 128, dropout 0.3), and classifies via a two-layer sigmoid head. The 1D-CNN applies parallel convolutions (filter widths 2, 3, 5) with global max-pooling to capture multi-scale n-gram patterns. Out-of-fold predictions form a 3-column matrix on which a logistic regression meta-learner is trained, learning a combination that exploits each model's complementary strengths.

C. Active Learning and Serving

Indicators scoring in [0.30, 0.70] enter a Redis uncertainty queue sorted by proximity to 0.50. Analyst labels submitted to POST /v1/label persist to PostgreSQL. When 500 new labels accumulate, a job re-trains the full pipeline and hot-swaps the production model without service interruption. The FastAPI serving layer (4 Uvicorn workers) exposes POST /v1/score and POST /v1/score/batch (up to 100 indicators). Each response includes ensemble score, label, sub-model scores, calibrated confidence interval, and top-5 SHAP features.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15251

V. PLAYBOOK ENGINE

A. DAG Execution Model

A Playbook is a directed acyclic graph of PlaybookNodes. Each node declares: action type, static params, prerequisite node IDs, an optional boolean condition evaluated against the shared ExecutionContext, an approval gate flag, timeout (default 30 s), retry budget (default 2), and an on_failure policy (continue | stop | rollback). The engine resolves topological order via depth-first traversal. A node is skipped if any prerequisite failed or its condition evaluates to false. Node outputs merge into the shared context so downstream nodes can consume upstream results -- for example, the TheHive node reads the list of blocked IPs written by the firewall node. All executions persist to PostgreSQL and are queryable via GET /v1/executions.

B. Action and Backends

TABLE 1. Support action types,backends, and rollback mechanism.

Action	Supported Backends	Rollback
DNS Block	Pi-hole v6, BIND9 (zone file + rndc), PowerDNS REST, Mock	dns_unblock
Firewall	iptables, nftables, pfSense fauxapi, FortiGate REST, AWS SG	firewall_unblock
TheHive	TheHive 5 REST API (case, alert, observables, tasks)	Manual
MISP	MISP REST API (event, attributes, TLP tags, distribution)	Manual
Notify	Slack Incoming Webhooks, SMTP with STARTTLS	N/A

All blocking actions are idempotent: a duplicate block returns SUCCESS with already_blocked:true, making retries safe. Firewall rules are tagged with the event_id for precise rollback. Nodes marked requires_approval pause execution and send a Slack prompt to #security-approvals; analysts resume via POST /v1/executions/{id}/approve/{node}. Three pre-built playbooks cover CRITICAL (score >= 0.85, all five actions, firewall approval gate), HIGH (>= 0.65, DNS + TheHive + MISP + Slack), and MEDIUM (>= 0.45, TheHive alert + Slack only).

VI. EVALUATION

A. Experimental Setup

Training data combined the Tranco top-1M list (benign), URLhaus, and OpenPhish (malicious), producing 487,312 examples at a 4:1 benign-to-malicious ratio after deduplication and WHOIS filtering. Stratified 70/15/15 splits were used; SMOTE applied only to training folds. Models were implemented in Python 3.11 with scikit-learn 1.4 (Random Forest), PyTorch 2.2 (neural networks), and shap 0.45.

B. Classification Results

TABLE 2. Test-set classification performance (n=73,097). Best values in bold.

Model	AUC-ROC	Precision	Recall	F1	FPR
Random Forest	0.9901	0.961	0.958	0.960	0.038
Bidirectional LSTM	0.9923	0.968	0.965	0.967	0.032
1D-CNN (multi-scale)	0.9887	0.954	0.961	0.957	0.046
Ensemble (ours)	**0.9970**	**0.983**	**0.981**	**0.982**	**0.017**

The ensemble gains 0.0047 AUC over the strongest base model (BiLSTM, 0.9923), confirming that the three architectures are genuinely complementary: the Random Forest captures structured feature interactions invisible to character-level models; the BiLSTM captures long-range sequential dependencies; the CNN captures local n-gram patterns. At the CRITICAL threshold (score >= 0.85), precision is 0.994 and recall 0.956, meaning 99.4% of automated blocks are confirmed malicious. Active learning over three retraining cycles (1,500 analyst labels total) yields AUC 0.9931 versus 0.9897 for random sampling ($p < 0.01$, paired t-test).

C. Response Latency

DOI: 10.17148/IJARCCE.2026.15251

TABLE 3. End-to-end response latency in ms (p50/p95), mock vs. live backends.

Stage	Mock p50	Mock p95	Live p50	Live p95
Feature extraction	18	34	142	380
ML scoring (ensemble)	23	41	26	48
DNS block	4	9	38	92
TheHive + MISP	11	23	395	960
Slack notify	3	7	110	290
Total (excl. approval wait)	**59**	**114**	**711**	**1,770**

Live latency is dominated by external API calls to TheHive and MISP. The 711 ms median end-to-end latency is well within the sub-second target for automated DNS containment. Redis caching reduces live feature extraction p50 from 142 ms to approximately 130 ms on repeated queries of the same domain.

VII. DISCUSSION AND CONCLUSION

Limitations include dependence on public threat feeds that may underrepresent vertical-specific threats, GDPR-driven WHOIS redaction reducing registrant feature quality, and the absence of adversarial robustness guarantees against feature-aware domain construction. Future work will integrate passive DNS enrichment, graph neural networks over domain co-registration graphs, and formal playbook verification via model checking.

THREAT-DETECT demonstrates that AUC 0.9970 detection accuracy and 711 ms automated response are jointly achievable in an open-source, containerised platform. The stacking ensemble outperforms each constituent model; SHAP attributions make every decision auditable; active learning reduces label cost; and the DAG Playbook Engine executes resilient, reversible multi-step responses to confirmed threats.

REFERENCES

[1]. M. Antonakakis et al., From throw-away traffic to bots: Detecting the rise of DGA-based malware, USENIX Security, 2012.
[2]. J. Woodbridge et al., Predicting domain generation algorithms with long short-term memory networks, arXiv:1611.00791, 2016.
[3]. O. Sagi and L. Rokach, Ensemble learning: A survey, WIREs Data Mining and Knowledge Discovery, vol. 8, no. 4, 2018.
[4]. S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, NeurIPS, vol. 30, 2017.
[5]. B. Settles, Active Learning Literature Survey, University of Wisconsin-Madison, Technical Report 1648, 2009.
[6]. N. V. Chawla et al., SMOTE: Synthetic minority over-sampling technique, JAIR, vol. 16, pp. 321-357, 2002.
[7]. J. Platt, Probabilistic outputs for support vector machines, Advances in Large Margin Classifiers, 1999.