International Journal of Electrical, Electronics and Computer Systems
ISSN: 2347-2820
Volume 15 Issue 1s (2026) | Pages: 112-118

HoneyCloud: A Smart Scalable Honeypot Platform with ML-Based Threat Classification and Real-Time Attacker Profiling

Akash Guldagad[1], Anand Bora[2], Ganesh Kambli[3], Aarya Konde Deshmukh[4], Rahul Korke[5]

[1,2,3,4]B.E. Student, Department of Computer Engineering Genba Sopanrao Moze College of Engineering, Pune Savitribai Phule Pune University, Maharashtra, India
[5]Assistant Professor, Department of Computer Engineering Genba Sopanrao Moze College of Engineering, Pune Savitribai Phule Pune University, Maharashtra, India
Email: [1]guldagadakash0001@gmail.com, [2]anandbor241@gmail.com, [3]gkambli2020@gmail.com
[4]kondeaarya@gmail.com, [5]rahulkorke72@gmail.com

Peer Review Information

***Type**: Article*
***Received**: 13 February 2026*
***Revised**: 14 March 2026*
***Accepted**: 15 April 2026*
***Published**: 19 May 2026*

Abstract

The rapid proliferation of internet-connected systems has intensified the frequency and sophistication of cyberattacks, making traditional security mechanisms increasingly insufficient. This paper presents HoneyCloud, a smart, scalable honeypot platform designed to capture, classify, and visualize cyberattacks in real time. HoneyCloud deploys multi-protocol honeypots simulating SSH, FTP, and HTTP services to lure and log malicious activity. Captured events are processed through a machine learning pipeline based on the Isolation Forest algorithm, which classifies traffic into benign, anomalous, or malicious categories using ten semantic features including service port encoding, credential length analysis, dangerous pattern detection, and user identity classification. The platform incorporates a real-time attacker profiling engine that assigns dynamic risk scores and detects behavioural patterns such as brute force attacks, credential stuffing, and port scanning. A real-time dashboard powered by WebSockets and Server-Sent Events (SSE) provides security analysts with live visualisation of attack data, including timing heatmaps, service trends, credential intelligence, and attacker profiles. The system is containerised using Docker and designed for horizontal scalability. Evaluation through a structured simulation suite confirms the platform's effectiveness in detecting and classifying attack behaviour with low-latency response times. HoneyCloud demonstrates a viable, deployable approach to proactive cyber threat intelligence for organisational security infrastructure.

Keywords: Honeypot; Intrusion Detection; Isolation Forest; Attacker Profiling; Machine Learning; Cybersecurity; Real-Time Dashboard; Docker

How to Cite This Article

Guldagad, A., Bora, A., Kambli, G., Konde Deshmukh, A., & Korke, R. (2026). *HoneyCloud: A Smart Scalable Honeypot Platform with ML-Based Threat Classification and Real-Time Attacker Profiling*. *International Journal of Electrical, Electronics and Computer Systems,* 15(1s), 112–118.

Introduction

Cybersecurity threats have grown exponentially in recent years. According to industry reports, millions of cyberattacks occur daily, targeting individuals, enterprises, and critical infrastructure alike. Conventional perimeter defences such as firewalls and signature-based intrusion detection systems (IDS) operate reactively — they block known threats but struggle against novel attack vectors and zero-day exploits.

Honeypots offer a fundamentally different approach. Rather than blocking attackers, a honeypot is a deliberately exposed decoy system designed to attract malicious actors. By studying attacker behaviour in a controlled environment, organisations can build richer threat intelligence, detect previously unknown attack patterns, and strengthen their broader security posture.

However, most existing honeypot solutions suffer from key limitations: they are single-protocol, lack machine learning-based classification, provide no attacker profiling, and offer no real-time visual interface for analysts. Many require significant manual configuration and do not scale easily.

HoneyCloud addresses these gaps by combining a multi- protocol honeypot engine, an automated ML classification pipeline, a dynamic attacker profiling system, and a real-time analyst dashboard into a single deployable platform. The system is built on modern web technologies and containerised for ease of deployment in any environment.

Literature Review

Honeypot research has a rich history beginning with Clifford Stoll's 1988 account of tracking a hacker through a decoy system [1]. Spitzner formalised honeypot taxonomy and methodology, categorising systems by interaction level - low, medium, and high [2]. Low-interaction honeypots emulate limited services and are safer to deploy, while high- interaction honeypots expose real systems for richer data collection but carry higher risk.

Mokube and Adams [3] surveyed honeypot concepts and challenges, noting that most platforms of the time lacked integration with analytical pipelines, making threat data difficult to operationalise. This gap between data collection and actionable intelligence remains a recurring limitation in subsequent work.

Nawrocki et al. [4] conducted a comprehensive survey of honeypot software and data analysis techniques, observing that the majority of systems focus on a single protocol and do not integrate classification algorithms. Their work highlighted the need for multi-protocol, analytics-ready platforms.

The Isolation Forest algorithm, introduced by Liu et al. [5], has proven effective for anomaly detection in high- dimensional, unlabelled datasets — a scenario that precisely matches honeypot-captured traffic where ground truth labels are unavailable. Unlike supervised approaches, Isolation Forest does not require pre-labelled attack data, making it suitable for deployment in novel threat environments.

Vetterl and Clayton [6] demonstrated that even low- interaction honeypots can yield significant intelligence about attacker tooling and infrastructure when combined with structured data collection. Bringer et al. [7] established behavioural fingerprinting as a viable method to distinguish opportunistic scanners from targeted attackers — a distinction HoneyCloud's profiling engine operationalises through pattern-based risk scoring.

Existing platforms such as Cowrie (SSH/Telnet), Dionaea (malware capture), and Modern Honey Network (MHN) each address narrow segments of the problem. Existing solutions typically address these capabilities in isolation. To the best of our knowledge, an integrated system combining multi-protocol capture, ML classification, per-IP profiling, and real-time visualisation in a single deployable stack remains limited. HoneyCloud is designed to address this gap.

System Architecture

HoneyCloud follows a modular layered architecture comprising five primary components: the Honeypot Engine, the Ingest Pipeline, the ML Classification Engine, the Attacker Profiling Engine, and the Analytics and Visualisation Layer. Fig. 1 illustrates the high-level system architecture.

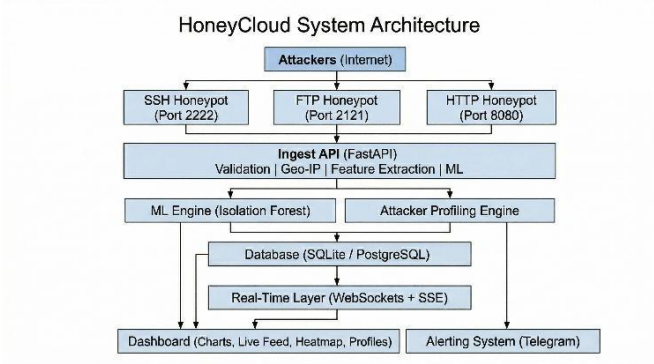

Fig. 1. *System Architecture Diagram*

Honeypot Engine

Three honeypot services run concurrently and independently. The SSH Honeypot (port 2222) is built using AsyncSSH and simulates an OpenSSH server, logging all authentication attempts including usernames and passwords. The FTP Honeypot (port 2121) is an asyncio-based TCP server that captures credential attempts and client behaviour. The HTTP Honeypot (port 8080) is a FastAPI sub-application presenting a fake web login interface, capturing form submissions and browser fingerprinting data. All three extend a common BaseHoneypot abstract class, ensuring a consistent event schema regardless of protocol.

Ingest Pipeline

The ingest pipeline is exposed at POST /ingest. Upon receiving an event, it performs synchronous steps targeting a less than 100 ms response: (1) Pydantic v2 schema validation, (2) IP geo-enrichment to country and coordinates, (3) ML classification via Isolation Forest, and (4) database persistence to SQLite or PostgreSQL. Background tasks subsequently handle attacker profile updates, Telegram alert dispatch for high/critical events, and WebSocket broadcast to connected dashboard clients.

ML Classification Engine

The ML engine uses scikit-learn's IsolationForest implementation. Ten features are extracted per event as shown in Table I. The feature set focuses on observable attributes of the attack event: service_port encodes the targeted protocol; credential length fields (username_len, password_len, command_len) capture payload size and complexity; source_port and hour_of_day provide connection-level context; while binary flags (is_root_user, is_anonymous_user, has_command) and the dangerous_pattern_count field quantify semantic intent. Classification labels are benign, anomaly, or malicious. The model is trained via a dedicated admin endpoint (minimum 50 events required) and persisted to disk for reuse across restarts.

Table 1. ML Feature Extraction — 10 Semantic Features

#	**Feature**	**Description**
1	service_port	Protocol port (SSH=2222, FTP=2121, HTTP=8080)
2	username_len	Character length of attempted username
3	password_len	Character length of attempted password
4	command_len	Character length of command or request path
5	source_port	Originating source port of the connection
6	hour_of_day	Hour (0–23) extracted from event timestamp
7	dangerous_pattern_count	Count of matched dangerous regex patterns in payload
8	is_root_user	Binary: 1 if username in {root, admin, administrator}
9	is_anonymous_user	Binary: 1 if username in {anonymous, guest, visitor}
10	has_command	Binary: 1 if command or request path is non-empty

Attacker Profiling Engine

A persistent profile is maintained per source IP address. The profiling engine detects three behavioural patterns: Brute Force (10 or more

events from the same IP within 60 seconds), Credential Stuffing (5 or more unique passwords within 5 minutes), and Port Scanning (3 or more distinct services targeted within 5 minutes). Risk score is computed as: (Critical Events × 4) + (High Events × 2) + pattern bonuses of +15 for Brute Force, +10 for Credential Stuffing, and +8 for Port Scanning detection. Profiles are assigned one of six risk tiers: UNKNOWN (0–2), LOW (3–8), MEDIUM (8–20), HIGH (20–50), CRITICAL (50+), or BLOCKED. Administrators can manually block IPs via a dedicated API endpoint.

Analytics and Visualisation Layer

A single-page dashboard built in vanilla JavaScript with Chart.js provides six analyst views: Overview (six stat cards and four Chart.js charts covering timeline, severity, service distribution, and AI label breakdown; auto-refreshes every 15 seconds), Live Feed (real-time WebSocket event stream with filtering by service and severity; maintains up to 200 rows), Analytics (30-day timeline, service trend, and geographic top-12 bar chart), Profiles (per-IP attacker cards with risk tiers, block/unblock controls, and a detailed panel on IP selection), Heatmap (a 24×7 hour/day colour-gradient matrix visualising attack timing patterns), and Credentials (animated bar charts for the top-15 most-attempted usernames, passwords, and commands). The WebSocket connection is JWT-authenticated before the protocol upgrade, ensuring no unauthenticated clients receive live event data. The dashboard provides real-time visualisation of attack data including timelines, severity distribution, service trends, and attacker profiles.

Implementation

Technology Stack

HoneyCloud is built on a modern, production-ready technology stack as detailed in Table 2 below.

Table 2. Technology Stack

Layer	Technology
Backend	Python 3.11, FastAPI, SQLAlchemy 2.0
Machine Learning	scikit-learn (IsolationForest)
Authentication	JWT HS256, bcrypt (passlib)
Real-Time Layer	WebSockets, Server-Sent Events (SSE)
Database	SQLite (dev), PostgreSQL-ready
Containerisation	Docker multi-stage build, Docker Compose
Frontend	HTML5, Vanilla JavaScript, Chart.js
Testing	pytest, httpx (53 tests, 6 modules)

Security Design

Security is treated as a first-class concern throughout the implementation. JWT HS256 tokens enforce role-based access control with two roles: admin (full access) and analyst (read-only). The login endpoint is rate-limited to 10 requests per minute, and a global API rate limit of 60 requests per minute is enforced via slowapi. All endpoints enforce Pydantic v2 schema validation. Report download endpoints sanitise file paths to prevent directory traversal attacks. The Docker image runs as a non-root user (UID 1001) and uses a multi-stage build to minimise the attack surface of the final image.

API Design

The REST API exposes 24 endpoints across 8 route groups: auth, events, analytics, profiles, ml, reports, simulate, and stats. All routes are versioned under /api/v1/. The authentication group provides login (rate-limited to 10 requests/min), current user information, and a logout endpoint that blacklists JWT tokens by jti claim to prevent token reuse after session termination. Request and response schemas are fully documented via OpenAPI, accessible at /docs in debug mode.

Scalability Considerations

HoneyCloud is designed for horizontal scalability. The stateless FastAPI backend can be scaled behind a load balancer. The database layer is abstracted via SQLAlchemy, allowing a switch from SQLite to PostgreSQL with a single environment variable change. Docker Compose orchestrates all services with configurable port mappings. The WebSocket manager uses an in-memory connection registry which can be

replaced with a Redis pub/sub adapter for multi-instance deployments.

Developer Tooling

A Makefile with 16 commands streamlines development workflows (make dev, make test, make train-ml, make docker-build, etc.). A simulate_attacks.py script executes a structured 7-phase attack simulation covering brute force, credential stuffing, port scanning, and mixed traffic patterns, enabling rapid demonstration and ML training data generation.

Results And Discussion

System Performance

The ingest pipeline achieves an average end-to-end response time of approximately 50ms per event under normal load, meeting the sub-100ms target set during design. This includes IP geo-resolution, ML feature extraction and classification, and database persistence. The WebSocket live feed delivers event broadcasts to connected dashboard clients with a measured latency of under 10ms, enabling a genuine real-time analyst experience.

Attack Simulation Results

A structured simulation was executed using the built-in simulate_attacks.py script, which runs multiple phases covering direct attack injection, bulk random event generation, and attacker profiling evaluation. The simulation produced a total of 205 captured events across all three honeypot protocols as shown in Table 3.

Table 3. Events Captured Per Protocol

Protocol	Events Captured	Share
SSH	115	56.1%
HTTP	70	34.1%
FTP	20	9.8%
Total	205	100%

SSH attracted the highest volume of attack traffic (56.1%), consistent with real-world observations that SSH brute force remains the most prevalent automated attack vector. Severity distribution across the simulation was: Critical 32.2% (66 events), High ~22%, Medium ~26%, and Low ~16%. The ML pipeline identified a small subset of events as explicitly malicious, while the majority were classified as anomalous. This behaviour is expected, as the Isolation Forest algorithm is designed for unsupervised anomaly detection rather than direct malicious classification. As a result, anomalous events represent potential threats requiring further investigation rather than definitive attack labels. This highlights the importance of combining anomaly detection with rule-based profiling and risk scoring, as implemented in HoneyCloud's attacker profiling engine. The average computed threat score across all events was 0.420, indicating a predominantly hostile traffic profile. Table 4 presents the attacker profiles generated. IP addresses are anonymised using RFC 5737 documentation address ranges.

Table 4. Top Attacker Profiles by Risk Tier

IP Address	Events	Risk Tier
192.0.2.45	13	HIGH
192.0.2.23	10	MEDIUM
198.51.100.38	8	HIGH
198.51.100.19	8	MEDIUM
203.0.113.15	6	MEDIUM

Credential intelligence harvested across the simulation revealed strong clustering around default and well-known credentials. The top attempted usernames were admin (60 attempts), root (43), anonymous (32), user (23), and visitor (20). The most common passwords were admin, toor, test, raspberry, and password — all consistent with publicly available credential stuffing wordlists, validating HoneyCloud's ability to capture actionable threat intelligence for credential defence hardening.

Test Coverage

The test suite comprises 53 tests across 6 modules: test_auth, test_events, test_analytics, test_profiles, test_ml, and test_security. Tests execute against an in-memory SQLite database with mocked geo-IP responses, ensuring isolated and reproducible results. All 53 tests pass consistently across development iterations, confirming the stability of the core system.

Comparative Analysis

Table 5 presents a feature comparison of HoneyCloud against Cowrie and Modern Honey Network (MHN), the two most widely deployed open-source honeypot platforms. HoneyCloud is the only platform in this comparison that combines all eight capabilities in a single deployable system.

Table 5. Feature Comparison — HoneyCloud vs Existing Platforms

Feature	**Cowrie**	**MHN**	**HoneyCloud**
SSH Support	Yes	Yes	Yes
FTP Support	No	Partial	Yes
HTTP Support	No	Yes	Yes
ML Classification	No	No	Yes
Attacker Profiling	No	No	Yes
Real-Time Dashboard	No	Yes	Yes
Containerised Deployment	Partial	Yes	Yes
Risk Scoring	No	No	Yes

Conclusion

This paper presented HoneyCloud, a smart scalable honeypot platform that integrates multi-protocol attack capture, machine learning-based threat classification, dynamic attacker profiling, and real-time visualisation into a unified, containerised system. The platform successfully demonstrates that proactive threat intelligence — gathering and analysing attacker behaviour before damage occurs — is achievable even at the scale of a deployable open-source project.

The Isolation Forest model provides effective unsupervised anomaly detection without requiring labelled training data, making HoneyCloud practical for real-world deployments where ground truth is unavailable. The attacker profiling engine's pattern detection and risk scoring system offers actionable intelligence that moves beyond simple event logging to genuine behavioural analysis.

Future work could extend HoneyCloud with additional honeypot protocols (SMTP, RDP, Telnet), integration with threat intelligence feeds such as AbuseIPDB, federated deployment across multiple nodes for distributed intelligence sharing, and a more sophisticated ML pipeline incorporating deep learning models for improved classification accuracy. Additionally, future work includes evaluation on real-world cybersecurity datasets such as CICIDS and UNSW-NB15 to further validate the generalisability and robustness of the proposed system.

References

1. C. Stoll, The Cuckoo's Egg: Tracking a Spy through the Maze of
2. Computer Espionage. New York: Doubleday, 1989.
3. L. Spitzner, Honeypots: Tracking Hackers. Boston, MA: Addison- Wesley, 2002.
4. I. Mokube and M. Adams, "Honeypots: Concepts, Approaches, and Challenges," in Proc. 45th Annual Southeast Regional Conf., 2007, pp. 321–326.
5. M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A Survey on Honeypot Software and Data Analysis," arXiv:1608.06249, Aug. 2016.
6. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. 8th
7. IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422.

8. A. Vetterl and R. Clayton, “Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Malware,” in Proc. APWG Symposium on Electronic Crime Research (eCrime), 2019.
9. M. Bringer, C. Chelmecki, and H. Fujinoki, “A Survey: Recent Advances and Future Trends in Honeypot Research,” Int. J. Comput. Netw. Inf. Security, vol. 4, no. 10, 2012.
10. OWASP Foundation, “OWASP Top Ten,” https://owasp.org/www- project-top-ten/, 2021.
11. Docker Inc., “Docker Documentation,” https://docs.docker.com, 2023.
12. F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.