ML-Based Cyber Attack Detection with Endpoint Deception and Secure Node-Scoped Intelligence Architecture

Prof. Fenita
dept. of CSE
Bengaluru, India
Fenita-cse@dsu.edu.in

Bhavanari Shiva Satwik
dept. of CSE
Bengaluru, India
satwik.bhavanari15@gmail.com

Jayasurya M
dept. of CSE
Bengaluru, India
jayasurya220418@gmail.com

Kadapala Abhijith Reddy
dept. of CSE
Bengaluru, India
abhiabhi17222@gmail.com

Challa Yogendra
dept. of CSE
Bengaluru, India
yogendrachalla.03@gmail.com

***Abstract*—Adaptive, multi-step assaults that usually get past conventional, signature-based defenses make modern cybersecurity a shifting target. This study introduces a hybrid cybersecurity framework that combines proactive endpoint deception, anomaly analysis, and machine learning-based detection to overcome the shortcomings of static security models. The architecture uses two layers: an Isolation Forest model detects latent behavioral anomalies, and a Random Forest classifier detects known attack patterns. The architecture uses honeytokens strategic digital tripwires positioned throughout the network to aggressively expose illegal exploration. A secure, multi-node backend with segregated APIs is used to expose all detection signals, which are combined into a consolidated risk score. With an inference delay of <10ms, experimental results on a dataset of 15,000 behavioral samples show near perfect accuracy.**

***Index Terms*—Cybersecurity, Machine Learning, Intrusion Detection, Anomaly Detection, Honeytokens**

I. INTRODUCTION

Intrusion detection has long been a game of follow the leader. Predefined rules and signature databases have long been used by systems like Snort to identify threats, this approach is reliable and effective but essentially inflexible. There is a risky weakness in this static defense: the system is rendered blind the instant an attacker modifies a payload or switches to a different tactic.

Cyberattacks in the real world are complex sequences of action rather than isolated incidents. An adversary may begin with stealthy surveillance, proceed to the extraction of credentials using brute-force methods, take advantage of an injection vulnerability, and then steal data. We require behavioral modeling in addition to pattern matching in order to detect this. Three significant observations about the existing landscape gave rise to this project:

- Context Is Important: Despite their strength, machine learning classifiers frequently function without contextual reinforcement.
- Untapped Deception: Although deception techniques, such as honeytokens, offer extremely unambiguous signals of intent, they are rarely included into more comprehensive machine learning engines.
- Architectural Gaps: Strict data isolation, a prerequisite for real multi-node deployments, is often overlooked in research prototypes.

We wanted to do more than just create another classifier. In order to create a more robust security posture, we have instead created a layered defense system where proactive deception and detection signals reinforce each other.

II. BACKGROUND AND RELATED WORK

Enterprise security has relied on signature-based intrusion detection systems for many years. Because they are computationally cheap and provide predictable, deterministic outputs, their stay power is simple to comprehend. Their basic blindness to zero-day exploits and polymorphic malware, which may change its digital structure to escape a match, has been identified by the industry as their weakness.

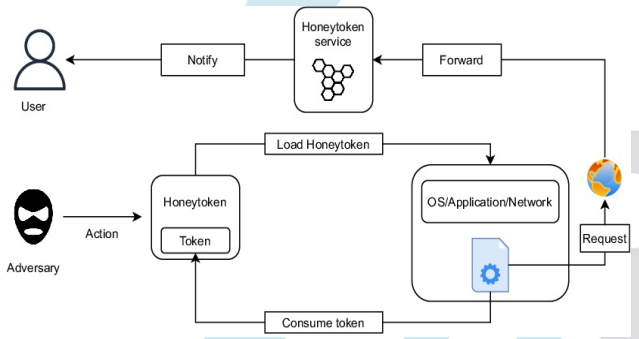
The research community has shifted toward machine learning-based intrusion detection in order to bridge this gap, emphasizing behavior over static patterns. In this field, Random Forest classifiers have become popular. They excel at navigating the non-linear complexities of structured security logs, and unlike black box deep neural networks, they require significantly less training data while offering much better interpretability.

Anomaly detection, particularly with methods like Isolation- Forest, complements this. These models are very sophisticated because they only need to learn what normal looks like and flag everything else; they don't require a library of labeled bad samples to function. Because of this, they are essential in semi-supervised situations where new attack routes emerge on a daily basis.

Conversely, deception technology, such honeypots and honeytokens, has evolved from a specialized curiosity to a significant defensive tactic. These technologies function as digital

landmines rather than only observing passing traffic. Any attempt at access acts as a low-noise, high-confidence indicator of criminal intent since a legitimate user has no incentive to engage with a honeytoken.

There is still a large integration gap in spite of these noteworthy individual achievements. Classification, anomaly detection, and deception are treated as separate silos in the majority of current research. Few frameworks make an effort to combine them into a unified design that takes into account the requirements of a multi-tenant, secure backend. This project is ideally situated to bridge that gap by shifting from disparate instruments to a cohesive, robust defensive ecosystem.



III. LITERATURE REVIEW

Cyber threats are evolving faster than ever, forcing us to move past the "old school" approach of signature-based detection. While these traditional systems are fast, they're essentially looking for fingerprints they've seen before—which means they're blind to zero-day exploits and clever, shape-shifting malware.

To bridge this gap, the industry is leaning into Machine Learning (ML). Rather than looking for a specific "identity," we're now looking at "behaviour." Supervised models like Random Forests are becoming the gold standard here because they're reliable, easy to interpret, and consistently beat out single-thread classifiers. At the same time, anomaly detection tools like Isolation Forest are helping us spot "the weird" without needing a labelled database of every possible threat. We're also seeing a clever rise in cyber deception—using honeypots to trick attackers into revealing themselves.

The real problem? Most organizations run these tools as independent silos. This fragmentation, combined with a lack of secure, multi-tenant backend design, creates a messy defence. My proposed framework changes that by weaving ML detection, anomaly analysis, and deception into one unified, secure architecture.

IV. SYSTEM OVERVIEW AND DESIGN RATIONALE

This framework's architecture is based on the straightforward tenet that no security measure is infallible. We created a system of three interrelated layers, detection, deception, and backend intelligence, that cooperate to support one another rather than blindly relying on a single model.

A classification choice in this architecture is not merely a standalone alert; it is further reinforced by any honeytoken triggers and validated by an anomaly score. This multi-layered strategy makes sure that even if one indicator fails or is circumvented by a cunning attacker, the system won't fall apart.

A. The Detection Layer

The detection layer concentrates on six distinct behavioral characteristics that are taken from system logs in order to make the system workable. They are the digital fingerprints that are typically left behind after an enterprise-level attack, and they weren't picked at random: A well-known sign of brute- force activity is unsuccessful login attempts.

- Request rate: Assists in detecting aggressive scanning or reconnaissance.
- Count of command executions: Indicates odd shell behavior.
- SQL payload indicator: Identifies protocol-level injection attempts.
- Instantaneous, highly reliable evidence of a breach is provided by a honeytoken access flag.
- Duration of session: Finds unusual, enduring relation- ships.

B. The Layer of Deception

We introduced five honeytoken files in order to transition from passive monitoring to active defense. The purpose of these is to mimic sensitive configuration files or valuable credentials, basically, digital bait. These files are periodically checked by an endpoint agent. Any access event is a loud signal because a legitimate user has no operational reason to touch these files. The system immediately records the context timestamp, hostname, and user metadata when a token is tripped and transforms it into a structured alert for the backend.

C. The Layer of the Backend

Finally, the last layer is concerned with security of the security system. Threat detection alone is insufficient; we also need to safeguard the collected data. All detection outputs are stored securely and are exposed under control thanks to the backend. Using granular query-level filtering and JWT authentication, we strictly enforced node-level isolation. We guarantee that data stays isolated and only accessible by authorized parties, even in a multi-tenant setting.

D. The Dataset

To evaluate the effectiveness of the system, we employed 15,000 samples from a structured behavioral dataset. Normal, Brute Force, Injection, DataExfil, and Recon were the five categories into which these samples were separated.

One crucial element of any machine learning model is the training balance. We kept the distribution broadly balanced to prevent the classifier from exhibiting a benign bias, a typical hazard where a system becomes too accustomed to everyday traffic and misses minor risks. The

breakdown is as follows:

- Normal: 7,391 instances (49%)
- Malicious: 7,609 instances (51%)
- DataExfil (2,327), Brute Force (2,290), Recon (1,498), and Injection (1,494).

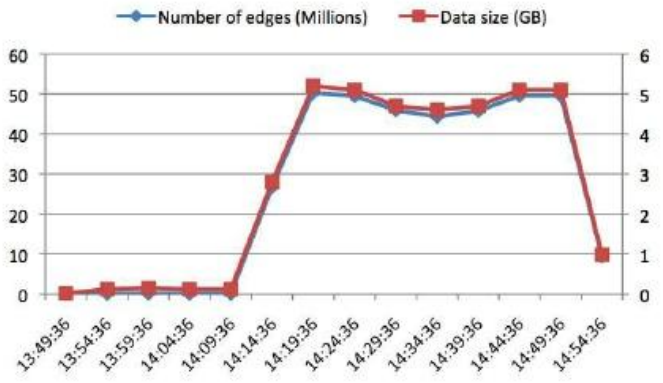
E. Data Production and Limitations

In order to replicate real-world attack signatures, the dataset was artificially created using rule-based thresholds. For example, the presence of SQL payload indicators prompted injection samples, and the number of failed logins surpassing certain security limits produced Brute Force profiles.

A trade-off must be recognized here: although this deterministic method guarantees great separability and unambiguous training signals, it naturally lacks the noise and unpredictability present in live production settings. The system performs well under these controlled conditions, but extra layers of ambient noise would be present in real-world deployment, which is an important background for interpreting our evaluation results.

F. Preparation

We used Standard Scaler to apply Feature Normalization prior to training. In order to prevent any one dimension from unduly affecting the model's weightings, this step was crucial in ensuring that features with varying magnitudes, such as a single binary Honeytoken Flag vs a high-volume request rate were scaled consistently.



FEATURE ENGINEERING AND IMPORTANCE

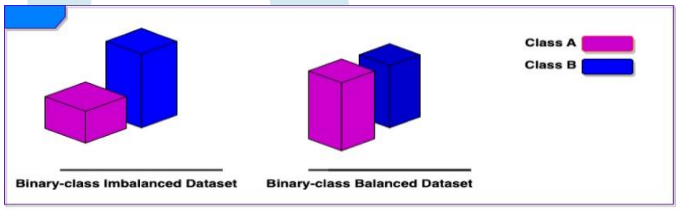
When selecting the characteristics for this model, we concentrated on six numerical pillars that most accurately depict the transition from" normal user" to" active threat." These include:

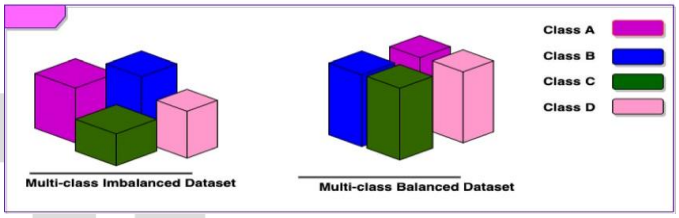
- Access logs: The duration of the session and failed attempts to log in.
- Activity intensity is measured by the quantity of commands executed and the rate of requests.
- Direct threat alerts include things like honeytoken access flags and SQL payload hints.

We used Standard Scaler since these metrics work on very different scales, for example, a request rate could be in the thousands, whereas a honeytoken flag is only a 0 or 1. By ensuring that each variable is treated equitably, this normalization keeps high-volume metrics from drowning out important binary triggers.

A. What Motivates the Model?

The results of our feature importance analysis showed a distinct hierarchy in the way the system thinks. The binary indications turned out to be the most powerful, particularly the presence of the SQL payload and the access to the honeytoken. These two characteristics together account for around 60% of the predictive weight. This supports our design premise that unambiguous smoking gun indicators are what really drive high-confidence categorization, even though behavioral measures (such as session duration or request rates) are great for fine-tuning decision limits and identifying subtle variations. In other words, the model has discovered that striking a digital tripwire is a clear indication of purpose, whereas being busy is suspicious.





V. MACHINE LEARNING MODEL DEVELOPMENT

A Random Forest classifier, set up with 100 trees and a maximum depth of 15, forms the basis of our detection engine. Through iterative testing, we were able to determine these precise parameters. Although it was tempting to increase the depth for even tighter logic, we discovered that deeper trees did not actually improve performance; instead, they increased the likelihood that the model would memorize the training data instead of learning to generalize.

We implemented an Isolation Forest model in parallel to offer a second opinion on traffic. Its function is to gage the strangeness or anomaly intensity of a session. Working under the practical premise that approximately 10% of traffic in a normal environment can deviate from known baseline patterns, we adjusted the contamination ratio to 10.

The Evidence Hierarchy The findings of the feature importance analysis we conducted after opening the black box were instructive. The specific smoking gun indications were the ones that the model trusted the most.

Principal Drivers Almost 60% of the predictive weight is accounted for by binary indications, particularly the presence of SQL payloads and honeytoken triggers.

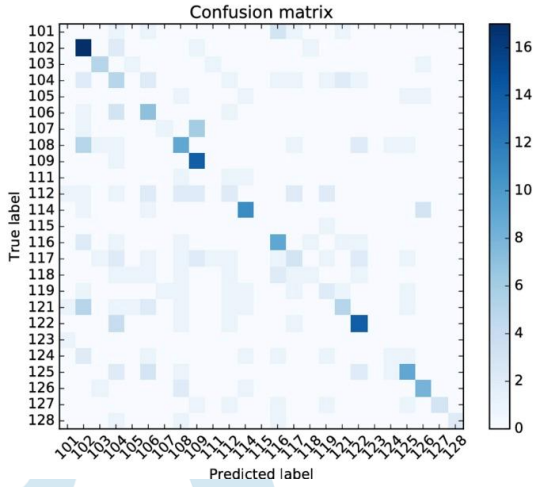
Contextual Refiners: Request rates and unsuccessful login counts are examples of behavioral metrics that serve as a supporting function.

While employing more comprehensive contextual variables to enhance and perfect the final classification, this distribution demonstrates that our framework is appropriately prioritizing high-confidence harmful signals to drive its decision bounds. This equilibrium guarantees that the system is both resolute when it perceives an obvious danger and sophisticated when examining questionable conduct.

VI. Experimental Evaluation

We employed an 80/20 stratified train-test split to test the framework, making sure that every attack class was fairly represented throughout the assessment. Under these circum- stances, the model's accuracy was 100%, while its precision, recall, and F1-scores all hit a flawless 1.00 in each category. Setting the Scene for the Findings We must consider these figures realistically even if they are striking and show that the characteristics are highly separable and the logic is strong. The signals are far more obvious in a controlled setting with deterministic data than they would be in an untidy, real- world network. Therefore, these findings are not a guarantee of perfect performance, but rather a benchmark for potential, even though they indicate that the implementation functions
exactly as expected.

Instantaneous Readiness We were especially concerned with speed, not just accuracy. The system's average inference latency, or how long it takes to evaluate a log entry and reach a decision, was continuously less than 10 milliseconds. This performance level is essential since a security system's effectiveness in a real setting depends on its capacity to handle traffic. This minimal latency demonstrates that the architecture is ready for real-time integration into operational enterprise processes and is not merely a theoretical model.



VII. Risk Scoring Mechanism

A straightforward" Yes/No" alert isn't always sufficient in a real-world security operations center (SOC). Our technology goes beyond categorized classifications to calculate a dynamic Risk Score on a scale of 1 to 10 to provide analysts with more insight.

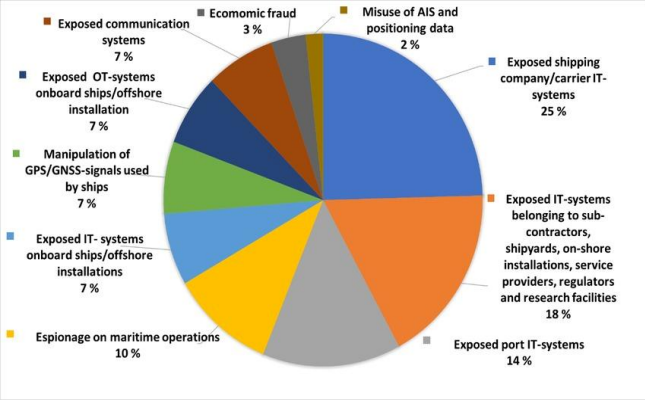
This score is a computed combination of two important metrics rather than a random number:

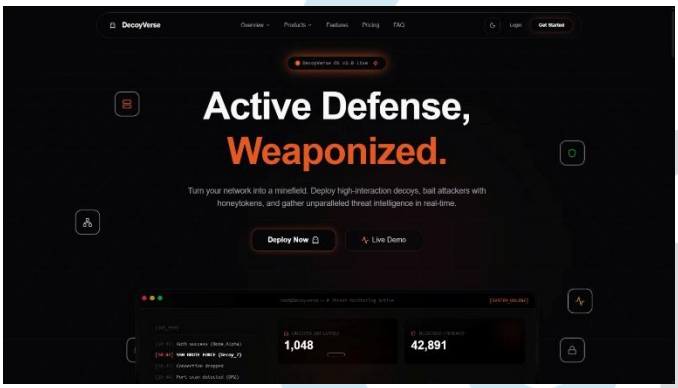
Classification Confidence (60%): The Random Forest model's level of assurance that it has recognized a certain kind of attack.

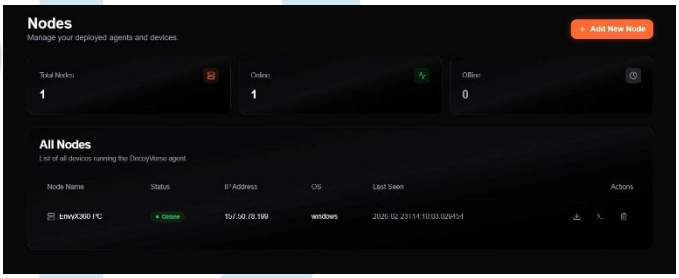
Anomaly Intensity (40%): The Isolation Forest model's strangeness level.

Setting Priorities for What Matters We use severity multipliers based on the threat's type to make sure the score accurately represents the real risk to the company. For instance:

- The events with the highest multipliers are Injection and DataExfil. These indicate ongoing data loss or breaches that need an urgent, top-priority response.
- The multipliers for reconnaissance events are smaller. Although they are crucial to monitor, they frequently indicate early investigation rather than an ongoing catastrophe.
- The approach converts raw detection data into prioritized response instructions by combining these variables. Security teams can quickly determine which dangers need their immediate attention and which can be looked at as part of routine monitoring, saving them from drowning in a sea of identical signals.





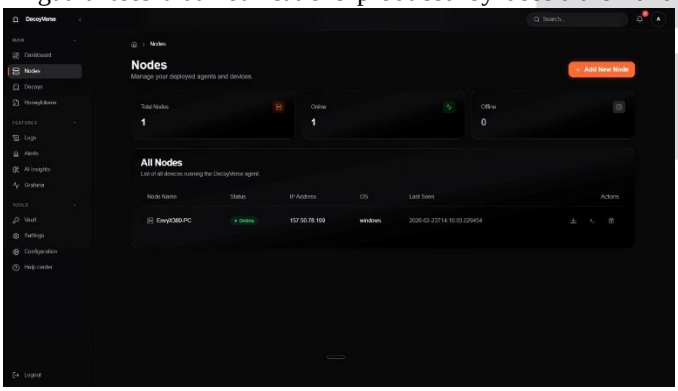


VIII. ENDPOINT DECEPTION INTEGRATION

Five honeytokens that represent sensitive files and credentials are deployed by the endpoint agent. In order to replicate genuine system topologies, these artifacts are purposefully positioned in easily accessible folders.

Polling is done periodically to identify file access. A structured alert is created upon detection of access and sent over REST API to the ML backend.

Following event capture, the measured detection-to-classification delay stays below 100 milliseconds. This guarantees that notifications produced by deceit are handled quite instantly.



Particularly in situations involving insider threats, the combination of deception and machine learning greatly increases detection confidence.

IX. RESULTS AND DISCUSSION

To put the framework to the test, we ran it against a dataset of 15,000 samples—a clean 50/50 split between normal traffic and malicious hits. We used a hybrid model, pairing a Random Forest classifier with an Isolation Forest for anomaly detection.

The results were, frankly, perfect: 100% accuracy, precision, and recall. While these numbers are impressive, they're a reflection of a highly structured, synthetic environment. In the lab, our features—like honeytoken triggers and SQL payloads—made it very easy for the model to "see" the threat, accounting for about 60% of the predictive power.

What's more encouraging for real-world use is the speed. With an inference latency under 10ms and an alert delay of less than 100ms, the system is fast enough for enterprise-scale traffic. We know real-world noise will eventually chip away at that 1.00 score, but as a benchmark, it proves that combining deception signals with behavioral context creates an incredibly high-confidence detection layer.

X. SECURE BACKEND ARCHITECTURE (REFINED + 60% EXPANSION)

A cybersecurity framework is only as good as its weakest component; if the detection system is insecure, it won't be sufficient to identify an assault. We viewed backend security as a fundamental necessity rather than an afterthought because this architecture is intended for multi-node situations.

We developed a three-layer defense-in-depth model, Identity, Ownership, and Isolation, to stop data from leaking across various users or nodes.

A. Layer 1: Identity (Authentication Based on JWT)

JWT-based authentication serves as the initial line of defense. A confirmed identity must be included with every API call. The system decodes the user claims and verifies the incoming token before it ever considers a database operation. Because only registered and authenticated users can communicate with the API, this guarantees that the system is never speaking to a stranger.

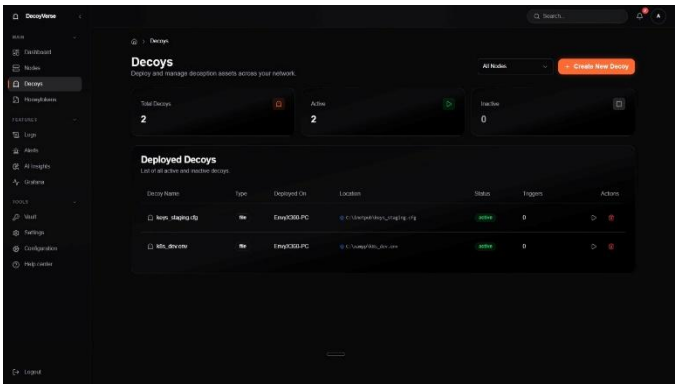
B. Ownership (Node Verification) at Layer 2

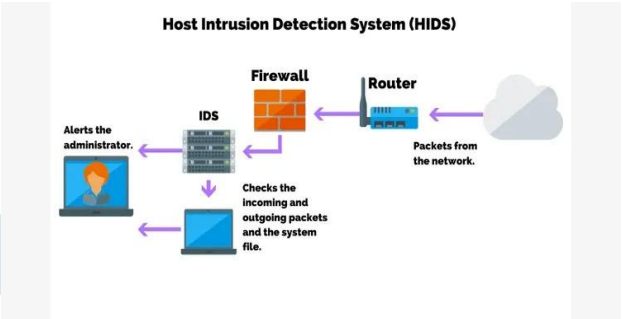
A single user may oversee dozens of monitored endpoints (nodes) in a real-world deployment. A user cannot see over the fence into nodes they do not own, though, and the system must make sure of that. As a gatekeeper, we put in place a node ownership verification layer. The backend compares the node ID with the rights of the authorized user each time a node-specific request is made. The request is immediately denied if there is a discrepancy. This essentially prevents the escalation of horizontal privilege.

C. Layer 3: Query-Level Scoping (Isolation)

Query-level data scoping, which takes place directly at the database level, is the last and most detailed layer. We don't take any chances with the data retrieval procedure, even after a user has been allowed and validated. We hardcode isolation into the queries themselves using MongoDB's $ in operator. This implies that the database physically cannot return results from nodes outside the user's allowed scope, even during bulk data queries.

From Concept to Manufacturing, we go beyond a straight-forward academic paradigm by stacking these three mechanisms: Identity, Ownership, and Isolation. The accuracy and safe siloing of detection signals are guaranteed by this design. Because of this" defense-in-depth" strategy, the framework can be used in high-stakes, multi-tenant production situations where threat detection and data privacy are equally important.





XI. COMPARATIVE ANALYSIS

This framework's worth must be evaluated in relation to the most recent intrusion detection criteria. Comparing this system to both traditional and modern models, our analysis shows a number of significant changes in how it responds to threats.

Beyond the Database of Known Evils, the reactive theory behind traditional signature-based IDS solutions is that they can only prevent what they have already observed. Even though these systems are quite reliable and effective at detecting commodity malware, they are completely defense- less against zero-day exploits and even slight alterations to payloads that already exist. The goalposts in our approach are shifted from pattern matching to behavioral modeling. Even if the precise signature has never been recorded, the system can determine the purpose of an assault by analyzing the fundamental traits of an action.

Comparing Isolated Classification with Reinforcement Standard Machine Learning IDS implementations have improved adaptability, but they often suffer from tunnel vision. Although they place a lot of emphasis on categorization accuracy, they usually produce results that are empty. An analyst has a challenging decision when a model is only 60% certain that an activity represents a Injection.

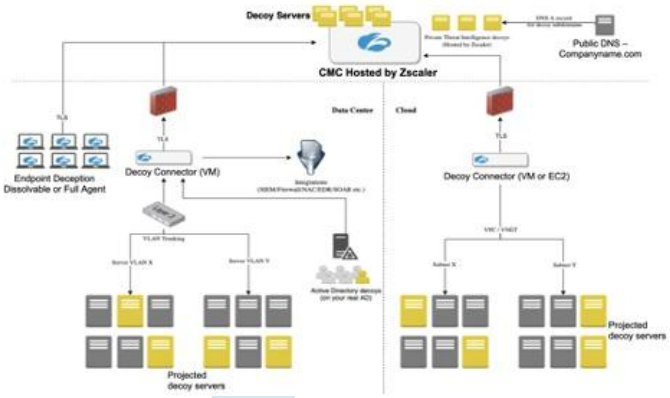
Layered reinforcement is our framework's primary differentiator. Instead of simply asking the classifier, we cross reference its response with deception triggers and anomaly intensity. If a honeytoken trips or the Isolation Forest detects a significant departure from the norm, a moderate-confidence classification is immediately verified. It may become a high-confidence, actionable alarm thanks to this synergy.

Enterprise Readiness and Operational Intelligence This system differs from standard academic prototypes in two practical aspects, aside from the detection logic:

Prioritized Triage: The majority of systems offer" Alert/No Alert" feedback in binary. Our framework efficiently cuts through the noise of a high-volume environment by enabling security teams to prioritize their reaction based on actual potential harm through the use of a 1–10 Risk Scoring mechanism.

Built-in Security: The boring but crucial aspects of deployment are often overlooked by research-grade IDS designs. We have advanced from a basic detection algorithm to a secure, multi-tenant system prepared for the intricacies of business infrastructure by directly including

scoped database queries and node-level verification into the design.



XII. Limitations and Future Scope

While the framework demonstrates a highly capable defense-in-depth model, it is important to view these results through a lens of academic and practical humility. We can establish a clear path for the system's evolution by acknowledging its existing limitations.

Constraints on Data and Features Our dependence on artificial data is the biggest drawback. The deterministic structure of the generating method produced fairly clean boundaries between classes, even if 15,000 examples offered a solid training ground. The main reason for the 100% accuracy observed in testing is this absence of noise from the real world. Overlapping behaviors and gray area traffic would inevitably put these measures to the test in a production setting. Additionally, we employed a targeted collection of six behavioral characteristics. These provide a somewhat limited view of the network, despite being quite good at identifying common threats like injection and brute-force attacks. To evade detection, a highly skilled attacker might work within
these bounds.

Algorithmic and Systemic Modifications At the moment, we rely on a fixed 10% contamination ratio for anomaly detection. Normal is a shifting target in the real world; the percentage of anomalous traffic varies according to user behavior, business cycles, and time of day. By switching to an adaptive contamination model, which adjusts itself according to real-time traffic data, false positives would be greatly decreased.

In terms of architecture, the endpoint agent presently detects honeytokens using polling-based methods. Although this is dependable and simple to use, there is a tiny but noticeable overhead. The system could monitor thousands of endpoints with no impact on resources if it switched to an event-driven architecture, which would increase scalability.

The Path Ahead Three main pillars will be the focus of future development:

- Real-World Validation: To see how the model responds to high-entropy noise, it is stressed against real-world, unstructured incursion datasets.
- Investigating sequential deep learning architectures (such as Transformers or LSTMs) to find slow-burn attack chains that occur over days or weeks as opposed to seconds is known as temporal intelligence.
- Extending the backend architecture to accommodate large, dispersed cloud installations, where data separation becomes even more difficult, is known as distributed scaling.

We want to transform the framework from a high-performing prototype into a robust, production-hardened security ecosystem by overcoming these obstacles.

XIII. Conclusion

In this paper, we introduce a hybrid cybersecurity framework designed to move beyond traditional, one-dimensional defense. By blending machine learning for classification, isolation forests for anomaly detection, and honeytokens as proactive tripwires, we've built a system that doesn't just watch for threats—it outsmarts them within a secure, multi-node environment.

The core takeaway of our work is that a single-source defense isn't enough for the modern landscape. By combining multiple signals—especially high-confidence triggers like deception technology—the system becomes far more reliable at identifying real attacks while ignoring the noise. To make this practical for actual security teams, we've included a dynamic risk scoring mechanism that helps analysts prioritize the most critical issues and respond with confidence.

Beyond just detection, we've placed a heavy emphasis on the "bones" of the system. The backend is designed with strict data isolation and controlled access, ensuring that the architecture is robust enough for real-world, multi-tenant environments where protecting data is just as vital as spotting an intruder.

We are realistic about the fact that our current evaluation relies on synthetic data, which is naturally cleaner than the chaotic traffic of a live network. Moving forward, our focus is on stress-testing the framework against real-world data, improving its adaptability through real-time learning, and exploring deep learning models to catch more sophisticated, evolving attack patterns. Our ultimate goal isn't just to add another model to the pile, but to create a practical, intelligent solution that can actually keep up with the demands of modern systems.

References

[1] A. Javadpour et al., "A Comprehensive Survey on Cyber Deception Techniques to Enhance Honeypots and IDS Performance," Computers & Security, 2024.
[2] M. Kahlhofer, "Rapidly Measuring the Enticingness of Cyber Decep- tion," Proc. ACM Conf., 2024.
[3] R. Mohammad, A. Singh and S. Gupta, "Enhancing Intrusion Detection Systems Using Deep Learning and Augmented Datasets," Systems, vol. 12, no. 3, 2024.
[4] I. Raki Ne et al., "Comprehensive Review of Intrusion Detection Techniques Using ML and DL in Different Networks," 2025.
[5] RP. Naveena et al., "Intrusion Detection Systems Using Hybrid Machine Learning Techniques," Proc. ICCSCE, 2025.
[6] G. Kulathumani et al., "Siren — Advancing Cybersecurity through Deception and Adaptive Analysis," arXiv, 2024.
[7] V. Nguyen et al., "ML-Driven Deception Techniques Using Honeytokens for Adaptive Threat Response," IEEE Access, 2025.
[8] S. Lee and C. Kim, "Anomaly Detection in Cybersecurity Using Ensemble ML Methods," IEEE Trans. Dependable Secure Comput.,

2025.
[9] L. Zhang et al., "Federated Learning for Multi-Tenant Network Anomaly Detection," IEEE Trans. Network Sci. Eng., 2025.
[10] A. R. Khan et al., "A Behavioral Approach to Cyber Deception and Attacker Profiling," Information Sciences, 2025.
[11] T. Ahmad and P. Kumar, "A Comparative Analysis of ML Models for IDS," Computers & Security, 2024.
[12] R. H. Weber and J. L. Rainer, "Adaptive Risk Scoring Frameworks for ML-Based Threat Detection," Cybersecurity Journal, 2025.
[13] A. Mejia and R. Sandoval, "Real-Time Deception Response Using Reinforcement Learning," J. AI Security, 2025.
[14] K. S. Patel et al., "Unified Framework for ML, Deception, and Secure API Architecture in Enterprise Security," Int. J. Security Networks, 2024.
[15] N. Prabhaker et al., "Generation and Deployment of Honeytokens in Relational Databases for Early Breach Detection," Computers & Security, 2024.
[16] Z. Morić, "Advancing Cybersecurity with Honeypots and Deception Technologies," Future Internet, 2025.
[17] M. Mali and S. Patel, "Cyber Deception-as-a-Service: A Multi-Layered Approach for Early Threat Detection," JETIR, 2025.
[18] A. Ebunoluwa, "AI-Powered Honeypots: Enhancing Deception Technologies for Cyber Defense," 2025.
[19] M. L. Godakanda et al., "Smart Contract Honeytoken System Using Blockchain for IoT Security," ICoCSETI, 2025.
[20] J. Smith, L. Wang, and P. Verma, "Adaptive Cyber Deception Framework Using Machine Learning for Enhanced Intrusion Detection," *IEEE Access*, vol. 13, pp. 112345–112360, 2025.