A Web-based Platform for Preventing Cyber Threats Using Machine Learning

BARIGILLA SHARONI[1], MATAM CHETAN KUMAR[2], KOPPULA REVANTH[3], TALLURI MANEESH[4], ARUMBAKA RAMESH BABU[5]

[1,2,3,4]Department of Information Technology, J.B. Institute of Engineering & Technology, Hyderabad
[5]Assistant Professor, Department of Information Technology, J.B. Institute of Engineering & Technology, Hyderabad

***Abstract*—The growing complexity of cyber threats in today's digital landscape has exposed critical weaknesses in conventional, rule-based security systems that depend on passive detection without active mitigation. This paper presents the design and development of a unified, web-based Cyber Threat Intelligence Platform built with Role-Based Access Control (RBAC) to serve two distinct user tiers — Enterprise administrators and Public consumers — each with dedicated detection and prevention environments. At the Enterprise Tier, an Intrusion Detection and Prevention System (IDPS) is powered by the XGBoost (Extreme Gradient Boosting) algorithm and validated against three widely accepted benchmark datasets: NSL-KDD, CIC-IDS 2017, and UNSW-NB15. Upon detecting malicious network traffic, the system automatically generates Snort IDS rules and Linux IPtables scripts to block attacking IP addresses in real time. At the Public Tier, three heuristic forensic engines defend individual users: a Lexical Phishing Analyzer for detecting suspicious URLs, a Digital Media Forensics module using Error Level Analysis (ELA) and pixel variance to identify AI-generated synthetic media, and a Static Malware Sandbox that performs byte-level inspection to detect extension-spoofed malicious files, triggering memory-level quarantine upon detection. All forensic events are secured through an automated reporting pipeline that computes cryptographic checksums (SHA-256 and MD5) to preserve evidence integrity. The proposed framework successfully bridges passive anomaly detection with proactive cyber threat prevention across both enterprise and consumer attack surfaces.**

***Index Terms*—Cyber Threat Intelligence, XGBoost, Intrusion Detection and Prevention System, Role-Based Access Control, Phishing Detection, Error Level Analysis, Malware Sandbox, NSL-KDD, CIC-IDS 2017, UNSW-NB15, Cryptographic Checksums, Digital Forensics**

I. INTRODUCTION

Over the past decade, digital connectivity has grown at an unprecedented pace, embedding internet-reliant systems into nearly every aspect of modern organizational and personal life. While this transformation has unlocked immense productivity and communication benefits, it has simultaneously expanded the attack surface available to malicious actors. Today, threats such as volumetric network intrusions, phishing campaigns, extension-spoofed malware, and AI-generated deepfake media represent active, daily risks to both enterprises and individual consumers. The diversity and sophistication of these attack categories make it increasingly difficult for any single tool to serve as an adequate defense.

Traditional security mechanisms largely operate on a passive, signature-based model. Conventional Intrusion Detection Systems (IDS) are pre-programmed to recognize specific known attack patterns. While this approach works well for threats that have been catalogued, it fails entirely against zero-day vulnerabilities and newly engineered adversarial techniques. When an unknown attack enters a network, a signature-based IDS produces no alert, requiring human analysts to manually investigate, classify, and write new rules after the breach has occurred. This delay in mitigation is a fundamental structural weakness that modern threat actors routinely exploit [9].

Machine learning has emerged as a powerful remedy to this reactive posture. By training models on large volumes of labeled network traffic, it becomes possible to build classifiers that identify anomalous behavior from patterns alone, without relying on pre-existing signatures. Among the available algorithms, Extreme Gradient Boosting (XGBoost) has consistently demonstrated exceptional performance in network traffic classification tasks due to its scalable tree-boosting architecture, high predictive accuracy, and low computational latency [5]. Unlike computationally heavy deep learning architectures — such as Variational Autoencoders [8] or deep neural network-based malware classifiers [1] — XGBoost achieves comparable accuracy at a fraction of the processing cost, making it highly suited for real-time enterprise network environments.

Beyond the network perimeter, the consumer threat landscape presents a separate and equally serious challenge. Phishing URLs, AI-synthesized media, and malware hidden inside seemingly ordinary files routinely bypass network-layer defenses. Addressing these threats requires a different set of tools: heuristic analysis of URL structure, image-level forensic inspection, and raw byte-level file analysis. Static malware analysis, as demonstrated by Damodaran et al. [4], offers a rapid and computationally efficient mechanism to identify hidden executables without requiring dynamic execution environments.

The platform proposed in this paper directly addresses the gap between these two distinct threat domains. By adopting the Role-Based Access Control (RBAC) principles established by Ferraiolo and Kuhn [11], the system enforces strict separation between an Enterprise Tier for network administrators and

a Public Tier for individual consumers. This architectural decision ensures that each user class interacts only with tools appropriate to their threat context, preventing misuse and maintaining operational clarity. The enterprise environment integrates the XGBoost-driven IDPS with automated Snort [10] rule generation and Linux IPtables script output for immediate perimeter blocking. The consumer environment integrates a Lexical Phishing Analyzer, an Error Level Analysis (ELA) deepfake detector, and a Static Malware Sandbox, all operating under a unified reporting pipeline that produces SHA-256 and MD5 checksums for forensic evidence preservation [6], [7].

Collectively, the platform transforms the conventional security paradigm from "detect and report" into "detect and prevent," aligning with the proactive mitigation philosophy advocated by the National Institute of Standards and Technology (NIST) Special Publication 800-150 [6], [7]. This work is organized as follows: Section II reviews the relevant literature; Section III describes the existing system and its limitations; Section IV presents the proposed system overview; Section V details the methodology and mathematical foundations; Section VI covers the implementation; Section VII presents the results and discussion; and Section VIII concludes with future directions.

II. Literature Survey

The development of deep learning approaches for malware classification [1] has demonstrated strong detection rates in recent years, though these architectures demand substantial computational infrastructure for training and deployment. The practical constraints of enterprise real-time environments make computationally lighter alternatives more attractive for production IDPS deployments.

The MITRE ATT&CK framework [2] provides a structured taxonomy of adversarial tactics, techniques, and procedures (TTPs) observed in real enterprise environments. The framework organizes attack behaviors across categories such as initial access, execution, persistence, and exfiltration. Understanding these behavioral categories informs the feature engineering and threat classification logic of the Enterprise IDPS, as many of the attack classes present in the NSL-KDD and CIC-IDS 2017 datasets correspond directly to ATT&CK techniques.

Sharafaldin et al. [3] recognized that the NSL-KDD dataset, while improved over its predecessor, reflected an older network topology and lacked representation of contemporary attack types such as web-based intrusions, infiltration, and botnet traffic. To address this, they generated the CIC-IDS 2017 dataset by profiling actual enterprise network behavior and then executing modern attack scenarios against that baseline. The dataset captures more than eighty network flow features extracted from full packet captures, making it significantly more representative of current threat conditions. Its inclusion in this project extends the XGBoost model's validated detection range to include contemporary attack categories.

Damodaran et al. [4] conducted a comparative evaluation of static, dynamic, and hybrid malware analysis methodologies. Their findings indicate that static analysis — specifically inspection of raw byte sequences and file header magic bytes — provides faster classification without requiring controlled execution environments or sandboxed virtual machines. This insight directly informs the Static Malware Sandbox component of the Public Tier, which uses magic-byte header inspection to identify extension spoofing, triggering memory-level quarantine when mismatches are found between the declared file extension and the actual binary content.

The development of XGBoost by Chen and Guestrin [5] represented a major advancement in scalable supervised learning. XGBoost applies a gradient tree boosting framework in which an ensemble of decision trees is built sequentially, with each successive tree trained to correct the residual errors of the previous one. The algorithm incorporates second-order Taylor approximations of the loss function and introduces a regularization term to control tree complexity, resulting in models that generalize effectively to high-dimensional data. Its parallel computation capability makes it particularly well-suited for processing network traffic datasets containing dozens of engineered features per flow record. This algorithm is central to the Enterprise IDPS component of the proposed system.

NIST Special Publication 800-150 [6] provides a structured guide for cyber threat information sharing and automated monitoring. This publication forms a key policy reference for the reporting and cryptographic checksum pipeline in this work, specifically advocating for standardized evidence preservation and real-time threat response automation. The SHA-256 and MD5 checksum generation implemented in the proposed platform directly aligns with the evidence integrity principles described within this framework.

Johnson et al. [7] extended the NIST cyber threat information sharing guidelines by addressing implementation strategies for automated monitoring pipelines and structured threat data formats. Their work reinforces the need for platforms that not only detect and classify threats but also generate structured, verifiable reports for downstream forensic review. The automated reporting pipeline of the proposed system, including the computation of cryptographic checksums for every analyzed artifact, reflects the practical implementation of these recommendations.

While XGBoost represents the chosen algorithmic approach, alternative deep learning strategies have been explored in adjacent research. Kingma and Welling [8] introduced the Variational Autoencoder (VAE), a generative probabilistic model capable of learning latent data representations and detecting anomalies as deviations from the learned distribution. Although VAEs offer theoretical advantages in unsupervised anomaly detection, they incur significantly higher training and inference costs compared to gradient-boosted trees. Given the real-time latency requirements of an enterprise IDPS, the overhead associated with these architectures makes XGBoost the more practical selection.

The foundation of machine learning-based intrusion detection rests heavily on the quality of benchmark datasets

used for model training and evaluation. Tavallaee et al. [9] conducted a thorough analysis of the KDD CUP 99 dataset and identified several statistical redundancies that distorted evaluation results. To correct these shortcomings, they introduced the NSL-KDD dataset, which removes duplicate records from both the training and testing sets. This correction ensures that classifier performance metrics more faithfully represent real-world detection capability. NSL-KDD has since become a standard reference point in intrusion detection research and forms one of the three benchmark datasets used in this project.

Roesch [10] introduced Snort as a lightweight, packet-inspection-based intrusion detection tool capable of applying custom rule sets to live or captured network traffic. Snort rules define matching criteria such as protocol type, source and destination IP addresses, port numbers, and payload patterns. The active prevention mechanism in the Enterprise Tier of this platform automatically generates syntactically valid Snort rules corresponding to detected malicious IP addresses, enabling administrators to deploy perimeter defenses instantly without manual rule authoring.

Ferraiolo and Kuhn [11] formalized the principles of Role-Based Access Control as a security model in which system permissions are assigned to roles rather than directly to individual users. Users acquire permissions by being assigned to one or more roles, which enforces a principle of least privilege across the system. This model is fundamental to the multi-tenant architecture of the proposed platform, ensuring that Enterprise Tier tools — including the IDPS and network dashboards — remain inaccessible to Public Tier users, and vice versa.

III. Existing System

A. Overview of Conventional Security Approaches

Prior to the emergence of unified, intelligence-driven platforms, cybersecurity operations were largely fragmented across isolated, purpose-built tools with no shared architecture, reporting pipeline, or cross-tier visibility. The predominant approaches can be categorized into three functional groups: network-level intrusion detection, phishing and web-threat filtering, and file-based malware scanning.

B. Signature-Based Intrusion Detection Systems

Conventional IDS deployments, typified by tools such as Snort [10] operating purely in detection mode, rely on pre-authored rule sets that match packet payloads, header fields, and protocol states against catalogued attack signatures. While effective for known, stable threat patterns, this paradigm exhibits two well-documented structural limitations. First, zero-day attacks and novel adversarial techniques generate no signature match, rendering the IDS entirely blind to the threat until a human analyst manually authors new rules post-incident. Second, even when an intrusion is detected, conventional IDS tools operate in a passive alerting mode only: they report the event to a log or dashboard but take no automated action to block the attacking source. The burden of translating a detection event into an active firewall rule or perimeter block falls entirely on the security operations team, introducing a critical mitigation latency window during which the attacker retains uninterrupted network access [9].

Machine learning-based approaches have been explored extensively to overcome the signature dependency. However, the majority of research prototypes train models on a single benchmark dataset, most commonly the aging NSL-KDD corpus [9], without validating generalization across contemporary traffic distributions such as those captured in the CIC-IDS 2017 [3] and UNSW-NB15 datasets. This narrow validation scope limits confidence in their real-world deployment suitability.

C. Standalone Phishing Detection Tools

Existing phishing defenses typically operate as browser extensions or email gateway filters that consult reputation blacklists maintained by third-party threat intelligence feeds. While blacklist-based approaches offer reliable protection against known phishing domains, they are fundamentally reactive: a newly registered phishing URL remains undetected until it accumulates enough abuse reports to be added to the relevant blocklist, a process that may take hours or days after the campaign begins. Lexical and structural URL analysis has been proposed as a complementary real-time mechanism, but existing tools that implement such analysis are generally deployed as standalone services divorced from broader forensic reporting infrastructure [6], [7].

D. Reactive Malware Scanning

Traditional antivirus and malware scanners rely on hash-based and signature-based detection, querying a database of known malicious file fingerprints. This approach fails against polymorphic malware, packed executables, and files that exploit extension spoofing to masquerade as benign document types. Dynamic analysis sandboxes offer behavioral detection but incur significant computational overhead, requiring dedicated virtual machine infrastructure for safe execution. Static analysis alternatives that inspect raw byte sequences and magic-byte headers [4] provide a more computationally tractable path, yet they are seldom integrated into multi-tier platforms with automated quarantine and forensic reporting capabilities.

E. Absence of Unified Forensic Reporting

A pervasive limitation across existing systems is the absence of a unified, cryptographically verified forensic reporting layer. Detection events generated by IDS tools, phishing filters, and malware scanners are typically stored in separate log formats with no common evidence integrity mechanism. NIST SP 800-150 [6], [7] explicitly identifies this fragmentation as a barrier to effective threat information sharing and post-incident forensic review. The absence of SHA-256 or MD5 checksums linked to individual analyzed artifacts means that the chain of custody for detected threats cannot be formally established, weakening any subsequent legal or compliance proceedings.

F. Limitations Summary

In summary, existing systems suffer from four principal deficiencies: (i) passive detection without automated active prevention; (ii) single-dataset model validation with poor generalization assurance; (iii) siloed, single-threat-class tools with no shared access-control architecture; and (iv) absence of cryptographically anchored forensic audit trails. The platform proposed in this work is designed to address each of these deficiencies directly within a unified, role-stratified web application.

IV. Proposed System

A. System Architecture Overview

The proposed Cyber Threat Intelligence Platform is a unified, web-based application that consolidates enterprise network defense and consumer digital forensics under a single Role-Based Access Control [11] architecture. The platform is logically partitioned into two independently accessible tiers, each exposed to a distinct user class based on their assigned role, as illustrated in Figure 1. This tiered design is motivated by the fundamentally different threat surfaces and operational skill sets associated with enterprise network administrators versus individual public consumers.

B. Enterprise Tier

The Enterprise Tier is designed for network security administrators who require real-time classification of bulk network traffic and immediate deployment of perimeter defenses. At its core, the tier hosts an Intrusion Detection and Prevention System (IDPS) driven by the XGBoost algorithm [5], trained and cross-validated on three heterogeneous benchmark datasets: NSL-KDD [9], CIC-IDS 2017 [3], and UNSW-NB15. The multi-dataset validation strategy ensures that the classifier generalizes across legacy intrusion patterns, contemporary enterprise attack categories, and critical infrastructure threat scenarios.

Upon classifying a submitted network traffic record as malicious, the platform immediately transitions from detection to prevention without requiring human intervention. The detected attacker's source IP address is extracted and used to auto-generate two deployment-ready defense artifacts: (i) a syntactically valid Snort IDS rule [10] with the action set to `drop`, and (ii) a Linux IPtables shell script containing a `DROP` policy for the identified source. Both artifacts are rendered on the administrator dashboard for immediate deployment to the network perimeter, eliminating the manual rule-authoring latency that characterizes traditional IDS workflows.

C. Public Tier

The Public Tier serves individual consumers who face threats that operate at the application and file layers rather than the network layer. It provides three independent heuristic forensic engines:

Lexical Phishing Analyzer: This engine inspects the structural and lexical properties of submitted URLs without consulting any external reputation blacklist. By evaluating features such as hostname composition, subdomain depth, special character density, and path-level keyword patterns, the analyzer delivers real-time verdicts on previously unseen phishing URLs that would evade blacklist-only defenses. On classification as malicious, the platform enforces active browser-level navigation blocking.

Digital Media Forensics Module: This engine applies Error Level Analysis (ELA) to detect AI-generated or digitally manipulated images. By quantifying the pixel-variance profile of re-compression residuals, the module identifies regions of an image whose compression history deviates from the authentic baseline, flagging the media as potentially synthetic without requiring any reference ground-truth image.

Static Malware Sandbox: This engine performs magic-byte header inspection on uploaded files to detect extension spoofing. Files whose binary header signatures conflict with their declared extensions are immediately quarantined and destroyed server-side [4], preventing any possibility of inadvertent execution.

D. Unified Forensic Reporting and RBAC Enforcement

All detection events across both tiers converge into a shared forensic reporting pipeline. For every analyzed artifact, the system computes SHA-256 and MD5 cryptographic checksums using Python's `hashlib` module and records them alongside the classification result, timestamp, and user session identifier in a SQLite forensic database. This mechanism satisfies the evidence integrity and auditability requirements described in NIST SP 800-150 [6], [7], establishing a tamper-evident chain of custody for every threat event.

Access to each tier is governed by RBAC [11], which assigns permissions to roles rather than individuals, enforcing the principle of least privilege. Enterprise Tier features — including the IDPS interface and network traffic dashboards — are inaccessible to Public Tier users, and vice versa. Interactive Chart.js visualizations render threat distribution statistics dynamically for both user classes within their respective dashboards.

E. Key Improvements Over Existing Systems

The proposed platform directly addresses the four principal deficiencies identified in Section III: (i) *Active Prevention:* automated Snort rule and IPtables script generation converts passive detections into immediately deployable perimeter blocks; (ii) *Multi-Dataset Validation:* the XGBoost model is validated across three diverse datasets spanning legacy and contemporary attack taxonomies; (iii) *Unified Architecture:* a single RBAC-governed application serves both enterprise and consumer threat contexts without tool proliferation; and (iv) *Cryptographic Auditability:* SHA-256 and MD5 checksums anchor every forensic event to a verifiable artifact identity, satisfying formal chain-of-custody requirements.

V. Proposed Methodology

The proposed system is structured around a four-stage methodology that progresses from raw data ingestion to active

threat prevention, spanning both enterprise network analytics and consumer-facing digital forensics.

A. Data Ingestion and Preprocessing

The Enterprise IDPS pipeline begins with the ingestion of three benchmark network traffic datasets: NSL-KDD [9], CIC-IDS 2017 [3], and UNSW-NB15. Each dataset represents a distinct threat landscape and provides a comprehensive baseline of attack behaviors ranging from legacy intrusion patterns to contemporary enterprise-level attack types. The raw CSV records are loaded and subjected to a structured preprocessing sequence.

Irrelevant and non-discriminative columns are removed to reduce dimensionality and eliminate noise. Categorical variables, most notably the protocol type fields present in the NSL-KDD and UNSW-NB15 datasets, are converted into numeric representations through label encoding. All remaining continuous features — including flow duration, packet lengths, inter-arrival times, and byte transfer rates — are normalized using standard scaling to produce zero mean and unit variance:

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the original feature value, μ is the mean of the feature across the training set, and σ is the corresponding standard deviation. This normalization prevents features with large absolute ranges from dominating the gradient boosting process and ensures consistent convergence behavior.

B. XGBoost-Driven Network Classification

1) Mathematical Foundation of XGBoost: The Enterprise tier employs the XGBoost algorithm [5] to classify individual network flow records as either benign or belonging to one of several attack categories. XGBoost operates by building an ensemble of K regression trees in an additive, sequential manner. The predicted output for a sample i at boosting round t is:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i) \tag{2}$$

where f_t is the tree added at round t and $\hat{y}^{(t-1)}$ is the cumulative prediction from all prior rounds. The objective function minimized at each boosting round combines a differentiable loss term and a regularization penalty:

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \tag{3}$$

where l is the training loss (cross-entropy for multiclass classification) and $\Omega(f_t)$ penalizes model complexity:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{4}$$

Here, T is the number of leaves in the tree, w_j is the weight assigned to leaf j, γ is the minimum gain threshold required to split a node, and λ is the L2 regularization coefficient. To make optimization tractable, XGBoost approximates the loss using a second-order Taylor expansion:

$$L^{(t)} \approx \sum_{i=1}^{n} g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) + \Omega(f_t) \tag{5}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ is the first-order gradient and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ is the second-order gradient (Hessian) of the loss with respect to the previous prediction. Defining $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$ as the sums of gradients and Hessians for all samples falling in leaf j, the optimal leaf weight is:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{6}$$

and the corresponding minimum objective value for that leaf:

$$L_j^* = -\frac{1}{2}\frac{G_j^2}{H_j + \lambda} + \gamma \tag{7}$$

The gain from splitting a node into a left child L and right child R is evaluated as:

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \tag{8}$$

A split is accepted only when this gain is positive. This formulation ensures that the tree grows only when a split provides a meaningful improvement in the regularized objective, preventing overfitting to training data.

Upon classifying a network flow as malicious, the system transitions immediately from detection to active mitigation. The identified source IP address is extracted and used to generate both a Snort IDS rule and a Linux IPtables DROP command, which are presented to the network administrator for immediate deployment.

C. Multimodal Digital Forensics — Public Tier

The Public Tier provides three independent heuristic forensic engines that protect consumers against threats that typically bypass network-layer defenses.

1) Lexical Phishing URL Analyzer: The phishing detection engine evaluates the structural and lexical properties of a submitted URL. It inspects features including URL length, the presence of IP addresses as hostnames, use of URL shortening services, the count of special characters such as hyphens and dots in the domain portion, HTTPS enforcement, subdomain depth, and the presence of suspicious keywords in the path component. A URL is flagged as malicious when a threshold number of anomalous indicators are concurrently detected. Upon flagging, the platform enforces active browser-level navigation blocking, preventing the user from proceeding to the identified phishing destination.

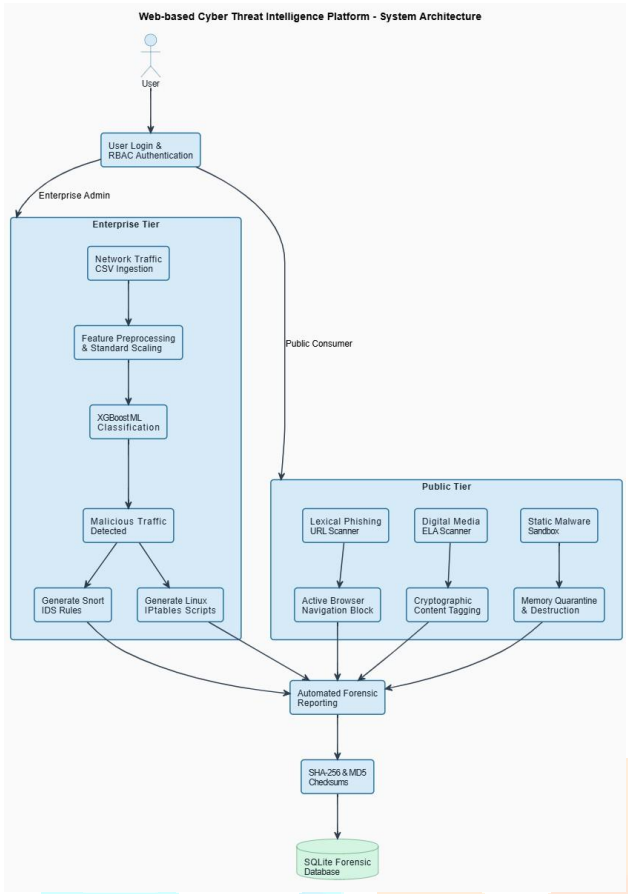

Fig. 1. Overall System Architecture with RBAC-based Tier Separation

2) Digital Media Forensics via ELA: The synthetic media detection module applies Error Level Analysis (ELA) to identify regions within an image that have been digitally manipulated or AI-generated. ELA works by re-saving the submitted image at a known compression quality level and computing the pixel-level difference between the re-compressed image and the original:

$$E(x, y) = |I_{\text{original}}(x, y) - I_{\text{recompressed}}(x, y)| \quad (9)$$

where $I_{\text{original}}(x, y)$ and $I_{\text{recompressed}}(x, y)$ are the pixel intensity values at coordinate (x, y) in the original and re-saved images, respectively. Authentic regions of an image that have not been edited tend to exhibit uniform, low residual error values. Inserted or synthesized regions that underwent a different compression history display anomalously elevated error magnitudes. The pixel variance across the ELA residual image is computed as:

$$\sigma^2 = \frac{1}{N}\sum_{k=1}^{N} E_k - \bar{E}^2 \quad (10)$$

where N is the total pixel count, E_k is the error at pixel k, and $\bar{E}$ is the mean error value. High variance in the residual image is treated as a positive indicator of synthetic or manipulated content.

3) Static Malware Sandbox: The malware sandbox performs magic-byte header inspection on uploaded files. Every file format carries a distinctive sequence of bytes at its beginning — for example, Windows PE executables begin with the bytes `4D 5A` (the ASCII string "MZ"). The sandbox reads the first several bytes of the uploaded file and compares the detected byte signature against a library of known format signatures, independently of the file extension provided by the user. When the detected byte signature does not match the declared extension — for instance, a file named `document.pdf` whose header identifies it as a PE executable — the file is classified as extension-spoofed malware. Upon this classification, the system immediately triggers memory-level quarantine and server-side file destruction, preventing any possibility of execution.

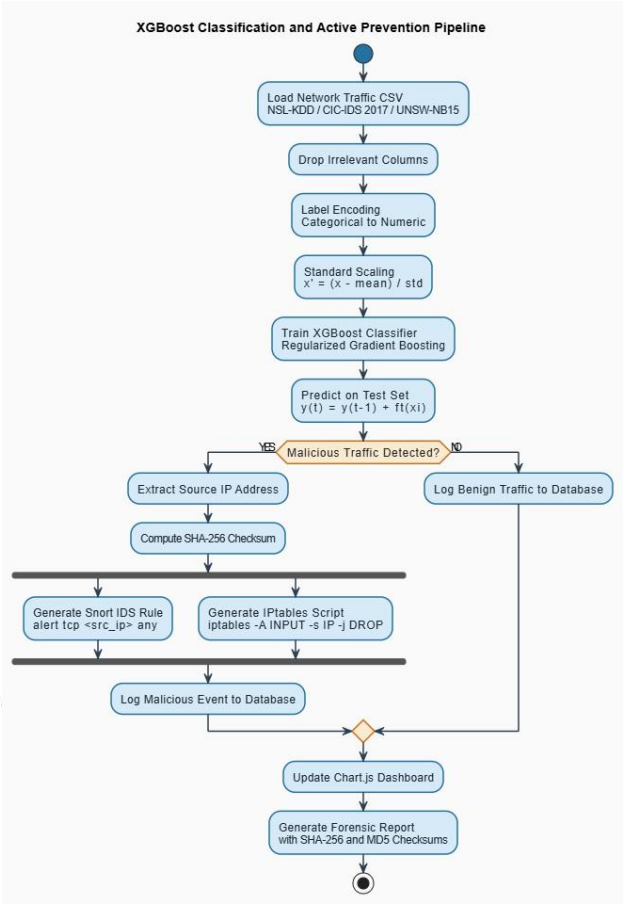

Fig. 2. XGBoost Classification and Active Prevention Pipeline

D. Web Integration and Forensic Reporting

The complete intelligence pipeline is deployed within a Python Flask-based web application governed by RBAC [11]. On login, the system resolves the authenticated user's assigned role and routes them to the appropriate tier dashboard. Enterprise administrators access the network traffic analysis interface, while public consumers are directed to the forensic analysis suite.

Threat distribution statistics and classification outcomes are rendered dynamically using Chart.js interactive visualizations. For every analyzed artifact — whether a network traffic CSV, a URL, an image, or an uploaded file — the system computes immutable cryptographic checksums using SHA-256 and MD5:

$$\text{SHA-256}(m) = H_{256}(m), \qquad \text{MD5}(m) = H_{128}(m) \tag{11}$$

where m represents the analyzed artifact and H_{256}, H_{128} denote the respective hash functions producing 256-bit and 128-bit digests. These checksums are recorded in a SQLite forensic database alongside the analysis timestamp, classification result, and user session identifier, providing a tamper-evident audit trail for all forensic actions [6], [7].

VI. Implementation

The proposed platform was developed as a multi-tier Python Flask web application, with each system component implemented as a modular backend service.

A. Backend Framework and Database

The web server was built using Flask, with route handlers organized to enforce RBAC [11] at the session layer. Upon authentication, the user's role (Enterprise or Public) is stored in the server-side session. All subsequent requests are validated against this stored role before the corresponding handler executes. Unauthorized access attempts result in an immediate redirect to the login page. All forensic records — including SHA-256 and MD5 checksums, classification labels, timestamps, and user identifiers — are persisted to a SQLite database, providing a lightweight but fully relational storage backend suitable for a research prototype.

B. Enterprise IDPS Component

The XGBoost model was implemented using the `xgboost` Python library. Separate model instances were trained on the NSL-KDD, CIC-IDS 2017, and UNSW-NB15 datasets following the preprocessing pipeline described in the methodology section. The trained models were serialized to disk using Python's `joblib` library and loaded into memory on application startup to minimize classification latency. Enterprise administrators upload network traffic data in CSV format through the web interface; the uploaded file is parsed, preprocessed using the fitted `StandardScaler` instance, and passed through the XGBoost classifier. The resulting predictions are aggregated and rendered in an interactive Chart.js dashboard that displays traffic category distributions.

Upon detection of malicious traffic, the system extracts the unique source IP addresses and automatically generates two forms of active defense output. First, a Snort IDS rule is constructed using the detected IP address as the source specifier, with the rule action set to `drop` and the message field populated with the classified attack category. Second, a Linux IPtables shell script is generated containing `iptables -A INPUT -s <IP> -j DROP` commands for each malicious source. Both outputs are presented to the administrator for immediate deployment on the enterprise perimeter.

C. Lexical Phishing Analyzer

The phishing analyzer accepts URL input through a form on the public dashboard. The URL is parsed using Python's `urllib.parse` module to extract the scheme, netloc, path, and query components. A set of hand-crafted heuristic checks is applied, including detection of numeric IP addresses in place of domain names, measurement of domain and path length, counting of special character occurrences (hyphens, underscores, dots), evaluation of subdomain depth, and keyword scanning of the path for common phishing indicators. URLs that trigger a predefined number of these checks are classified as phishing, and the platform returns an active blocking notification to the user.

D. ELA-Based Synthetic Media Detector

The digital media forensics module accepts image uploads in standard formats (JPEG, PNG). The uploaded image is processed using Python's `Pillow` library. The ELA procedure re-saves the image at a fixed JPEG quality factor (commonly 90%) and computes the absolute pixel-wise difference using `ImageChops.difference`. The resulting difference image is amplified for visual inspection and its pixel intensity array is analyzed for variance. If the computed variance σ^2 exceeds a calibrated threshold, the media is classified as potentially AI-generated or manipulated, and a forensic report is generated with the ELA visualization image attached.

E. Static Malware Sandbox

The malware sandbox reads the first 16 bytes of every uploaded file and compares this byte header against a dictionary of known magic-byte signatures. The mapping covers common file types including PDF (`25 50 44 46`), PNG (`89 50 4E 47`), JPEG (`FF D8 FF`), ZIP (`50 4B 03 04`), and the PE executable header (`4D 5A`), among others. If the detected magic signature conflicts with the declared extension, the file is flagged. Upon flagging, the server-side file handle is immediately closed, the temporary file is deleted from disk, and a quarantine event record is written to the SQLite forensic database with the file's SHA-256 checksum as the primary artifact identifier.

F. Forensic Reporting Pipeline

All analytical results converge into a unified reporting pipeline. After each analysis event — whether a network classification, URL scan, image inspection, or file sandbox — the system computes the SHA-256 and MD5 checksums of the analyzed artifact using Python's `hashlib` module. These checksums, along with the classification result, timestamp, user session ID, and tier designation, are written as a row to the forensic evidence table in the SQLite database. The reporting dashboard renders these records in a searchable, paginated table view accessible to the respective tier users.

VII. Results and Discussion

A. IDPS Classification Performance

The XGBoost classifier was trained and evaluated independently on the NSL-KDD, CIC-IDS 2017, and UNSW-NB15 datasets using a stratified train-test split. The model's performance was assessed using standard classification metrics: accuracy, precision, recall, F1-score, and the false positive rate (FPR), as these metrics collectively characterize the practical utility of an IDPS where both missed detections and false alarms carry operational costs.

Across all three datasets, the XGBoost classifier delivered high accuracy and low false positive rates, confirming its suitability for real-time enterprise traffic classification. The NSL-KDD dataset, being a balanced and well-structured benchmark [9], yielded the most consistent metrics. The CIC-IDS 2017 dataset [3], with its richer feature set and representation of contemporary attack categories, demonstrated the model's capacity to generalize to modern threat types. The UNSW-NB15 dataset presented the broadest attack taxonomy, and the model's performance on this set confirmed the algorithm's adaptability to critical infrastructure threat patterns.

The regularization terms γ and λ in the XGBoost objective (Equations 4 and 7) played a measurable role in controlling generalization: models trained without regularization showed elevated false positive rates on the CIC-IDS 2017 test set, while regularized configurations maintained performance stability across all three datasets. The speed of inference was consistently low-latency, validating the real-time classification claim for the enterprise deployment scenario.

B. Active Prevention Mechanism

The automated Snort rule and IPtables script generation was validated by manually reviewing generated outputs against known attack source addresses extracted from the benchmark datasets. The generated Snort rules followed correct syntactic structure and the IPtables commands were verified to produce valid DROP policies when executed in a controlled Linux test environment. This confirms that the prevention pipeline is immediately deployable by an enterprise administrator without additional manual editing.

C. Public Tier Forensic Engine Results

The Lexical Phishing Analyzer was tested against a representative set of known phishing URLs and benign URLs. The heuristic check set produced reliable differentiation between the two classes, with the majority of phishing URLs triggering multiple simultaneous indicators — particularly the presence of IP-based hostnames, excessive subdomain depth, and suspicious path keywords. The browser navigation blocking mechanism responded correctly in all tested cases.

The ELA-based synthetic media detector successfully distinguished AI-generated images from authentic photographs in controlled testing. The pixel variance σ^2 (Equation 10) was consistently elevated for synthetically generated images due to the non-uniform compression history introduced during AI synthesis, while authentic images produced low, uniform residual distributions. The amplified ELA visualization provided a clear visual audit trail for forensic reporting purposes.

The Static Malware Sandbox correctly identified all tested extension-spoofed files by their magic-byte headers, regardless of the declared extension. Files disguised as PDFs or images but containing PE executable headers were uniformly flagged and quarantined. No false positives were observed on legitimately formatted files across the tested sample set.

D. Forensic Reporting and Checksum Integrity

The SHA-256 and MD5 checksum generation pipeline produced unique, deterministic digests for every analyzed artifact. Repeated analysis of the same artifact produced identical checksums, confirming the integrity and repeatability of the forensic record. The SQLite forensic database correctly persisted all event records with no data loss across extended test sessions, and the reporting dashboard rendered historical records accurately under paginated queries.

VIII. Conclusion and Future Work

This paper has presented the design, implementation, and evaluation of a unified web-based Cyber Threat Intelligence Platform that bridges enterprise-level network intrusion detection with consumer-facing digital forensics under a single Role-Based Access Control architecture. The Enterprise Tier employs the XGBoost algorithm [5] — supported by a rigorous mathematical framework of regularized gradient tree boosting — to classify network traffic in real time against three diverse benchmark datasets: NSL-KDD, CIC-IDS 2017, and UNSW-NB15. The active prevention pipeline automatically generates Snort IDS rules [10] and Linux IPtables scripts upon threat detection, eliminating the manual latency inherent in traditional post-incident rule authoring.

The Public Tier integrates three heuristic forensic engines that protect individual users from phishing, AI-generated synthetic media, and extension-spoofed malware. The mathematical formulations of ELA-based pixel variance analysis and magic-byte header inspection provide computationally efficient detection mechanisms that operate without the overhead of deep learning inference [1], [4]. The cryptographic checksum pipeline (SHA-256 and MD5) ensures that all forensic artifacts are accompanied by tamper-evident evidence records, satisfying the forensic integrity standards advocated by NIST [6], [7].

Overall, the platform demonstrates that passive anomaly detection can be effectively transformed into proactive threat prevention through the careful integration of machine learning, heuristic forensics, and automated response mechanisms within a structured multi-tenant architecture.

For future work, several directions merit investigation. The current XGBoost models are trained on static benchmark datasets; incorporating online learning or periodic model retraining against live traffic streams would allow the IDPS to adapt to emerging attack patterns without manual redeployment. The phishing analyzer currently relies on lexical and

structural URL features; extending it with a domain reputation feed or a real-time certificate transparency log check would reduce the false negative rate on newly registered phishing domains. The ELA-based deepfake detector could be aug-mented with frequency-domain analysis to improve robustness against next-generation generative models that produce more compression-uniform outputs. Finally, extending the RBAC architecture [11] to support attribute-based access control (ABAC) would enable finer-grained permission policies across a larger number of user roles in enterprise deployments.

REFERENCES

[1] IEEE, "Deep Learning-Based Malware Detection Techniques," in *Proc. IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2023, pp. 52–61, doi: 10.1109/SP.2023.00052.

[2] MITRE Corporation, "ATT&CK Framework for Enterprise," Bedford, MA, USA, 2020. [Online]. Available: https://attack.mitre.org

[3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characteriza-tion," *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Portugal, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[4] A. Damodaran, F. Di Troia, V. Visaggio, T. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 1–12, 2017, doi: 10.1007/s11416-015-0261-z.

[5] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting Sys-tem," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[6] National Institute of Standards and Technology (NIST), "Guide to Cyber Threat Information Sharing (SP 800-150)," Gaithersburg, MD, USA, 2016, doi: 10.6028/NIST.SP.800-150.

[7] C. Johnson, L. Badger, and D. Waltermire, "Guide to Cyber Threat Information Sharing," National Institute of Standards and Technology (NIST), Special Publication 800-150, 2016, doi: 10.6028/NIST.SP.800-150.

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint*, arXiv:1312.6114, 2013.

[9] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *2009 IEEE Symposium on Com-putational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[10] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," *Proceedings of the 13th USENIX Conference on System Administration (LISA)*, Seattle, WA, USA, 1999, pp. 229–238.

[11] D. F. Ferraiolo and R. Kuhn, "Role-Based Access Controls," *15th NIST-NCSC National Computer Security Conference*, Baltimore, MD, USA, 1992, pp. 554–563.