Deep Web Guard – AI Powered Security Platform

Dr. Ramesh Koppar[1], Prof Poornima Gowda H. S.[2], P. Vaishnavi[3], Spandana S. M.[4], Varshitha N.[5], Nikita[6]

[1,2]Associate Professor CSE Department, Sai Vidya Institute of Technology, Bangalore, India

[3,4,5,6]Student CSE Department, Sai Vidya Institute of Technology, Bangalore, India

DOI: https://doi.org/10.47772/IJRISS.2026.10190060

Received: 28 January 2026; Accepted: 02 February 2026; Published: 18 February 2026

ABSTRACT

The Deep Web Guard project presents an AI-powered, dual-mode cybersecurity platform that integrates a Web Vulnerability Scanner and a Network Intrusion Detection System (NIDS) into a unified solution. Designed to provide end-to-end protection for web applications and network infrastructures, the platform leverages GPTbased threat intelligence and machine learning algorithms for real-time anomaly detection, risk scoring, and contextual threat analysis. The system combines static and dynamic web scanning, payload fuzzing, and signature-based as well as ML-driven network monitoring to detect both known and zero-day attacks. Developed using Python, Node.js, and React, the platform offers an intuitive dashboard for real-time visualization, automated reporting, and alert management. By reducing false positives and simplifying deployment, Deep Web Guard delivers an affordable, open-source, and intelligent cybersecurity framework suitable for organizations of all sizes.

Keywords: AI-powered security platform, web vulnerability scanner, network intrusion detection system (NIDS), hybrid detection (signature-based and machine learning), zero-day threat detection, real-time monitoring and alerting, automated security reporting, open-source full-stack implementation**.**

INTRODUCTION

The growing attack surface of enterprise networks and web applications poses a growing danger to the security of contemporary digital infrastructures. While network-based assaults, such as port scanning, SQL injection attempts, command-and-control traffic, and coordinated DDoS events, continue to increase in frequency and sophistication, recent studies show that over 50% of breaches utilise weak web applications [1][2]. Conventional systems, including signature-based intrusion detection engines (like Snort and Bro/Zeek), are effective against known assaults, but they often provide alarm overload without contextual assistance and are unable to identify novel or obfuscated threats [1][2]. While machine-learning-based NIDS research shows gains in anomaly detection, it frequently suffers from brittle feature generalisation, false positives, or the lack of explainability needed in practical operational situations [3]. [4] [5].

In order to fill these gaps, this study presents Deep Web Guard, a dual-mode cybersecurity platform driven by AI that combines a network intrusion detection system (NIDS) and a web vulnerability scanner into one operational stack. In order to [1] detect known and unknown (zero-day) attack patterns, [2] reduce false positives through contextual filtering and remediation text, and [3] present actionable intelligence via real-time dashboards and automated PDF security reports, the suggested system uses a hybrid detection pipeline that combines signature rules, machine-learning models (One-Class SVM, Isolation Forest, SVR scoring), and generative AIbased contextual analysis using GPT.

Unlike isolated tools that operate separately for web and network layers, Deep Web Guard delivers unified situational awareness and uses explainable AI for analyst-ready interpretation, aligning with the need for integrated, intelligent, and cost-effective security platforms highlighted in recent reviews [5][6].

The scope of this work includes the architecture, implementation, and evaluation of Deep Web Guard using realtime packet traces and controlled web-fuzzing experiments. The system advances prior art by fusing

signaturebased defenses with unsupervised anomaly detection and GPT-assisted triage, thereby offering improved detectability of evolving threats while lowering the operational burden on analysts. Future extensions include deployment in distributed environments, integration with threat-intelligence feeds, and adaptive model retraining to maintain robustness against emergent attack vectors.

LITERATURE REVIEW

Intrusion detection and web application security have been extensively studied over the past two decades. Classic rule-based IDS engines such as Snort and Bro (now Zeek) established the earliest foundation for real-time network threat monitoring using signature matching, scriptable policies, and protocol-aware inspection [1], [2]. These systems demonstrated high precision for known threats but were unable to detect zero-day or polymorphic attacks and produced large volumes of alerts without semantic context, making them hard to operationalize in modern DevSecOps pipelines.

To overcome the limitations of signature-only systems, researchers introduced ML-based anomaly detection, wherein "normal" traffic behavior is modeled and deviations are flagged as potential attacks. Schölkopf et al. demonstrated that One-Class SVM can estimate support regions for normal distributions and thereby detect novel anomalies [3]. Liu et al. later developed Isolation Forest, which works better on big, high-dimensional datasets and isolates anomalies using random recursive partitioning [4]. According to recent studies, ML-based NIDS perform better than signature engines for unknown and slow-burn attacks, but they still have significant falsepositive rates and are not actionable for analysts to comprehend [5], [6].

Web application security parallel work focused on CVE mapping, OWASP-guided evaluations, and static and dynamic fuzzing. Nevertheless, commercial and open-source scanners often fail to identify logic-level vulnerabilities, generate 30–40% false positives, and lack context-aware remedies [7], [8]. In order to lessen the workload of analysts, researchers developed hybrid web scanners that integrate rule-based signatures with MLbased anomaly detection and AI-assisted explanation [9, 10, 11]. Although it is yet uncommon in productionready integrations, the application of generative AI for interpretative remediation and false-positive triage is becoming a distinct research area [12], [13].

The suggested system Deep Web Guard, in contrast to these disjointed solutions, is built along the path suggested by recent research: a unified dual-mode platform that integrates (i) signature-based detection, (ii) ML-based anomaly scoring, and (iii) GPT-assisted contextual analysis for both web vulnerability scanning and network intrusion detection. The system addresses the fundamental flaws noted in the literature by operationalising these strands together rather than separately: zero-day blindness, false-positive overload, lack of remediation guidance, and lack of integrated visibility across application and network security layers.

Furthermore, a number of studies highlight how detection pipelines are weakened by the lack of correlated visibility between web-layer and network-layer threats because many real-world attacks (such as SQL injection over HTTP that causes lateral movement in the network) span multiple layers and cannot be detected by isolated scanners [14], [15]. Because online security and network intrusion detection are usually treated as separate workflows by existing systems, analysts must manually correlate warnings across heterogeneous technologies, which lengthens investigation times and causes alert fatigue in Security Operations Centres (SOCs).

Several detectors can be integrated using ensemble learning, rule-ML fusion, or stream correlation engines, according to recent research on hybrid and collaborative intrusion detection architectures [9], [10], [11], and [16]. Even while these systems increase general recall and resilience to zero-day behaviours, they still lack native semantic enrichment, which is the process of turning unprocessed alarms into counsel that is actionable and understandable by humans. Because of this gap, researchers are experimenting with Large Language Models (LLMs) like GPT for contextual post-processing of security alerts, taking use of their capacity to prioritise alerts, explain anomalies in natural language, and produce remedial instructions [12], [13], and [17].

Nevertheless, the majority of LLM-based applications are still experimental or limited to offline analysis; there is little evidence of their integration into real-time risk assessment pipelines in empirical deployments.

False positives continue to be the biggest obstacle to ML/NIDS implementation in production, according to surveys [5], [6], and [18]. Even extremely precise anomaly detectors produce alerts that lack operational context, such as "what needs to be fixed" or "why this is dangerous." Analysts either disregard alerts or turn off detection modules in the absence of contextual triage. This observation aligns with several reviews that suggest combining detection engines with AI-powered post-classification layers to carry out remediation guidance, semantic justification, and confidence weighting prior to sending alarms to human analysts [10], [12], [17], [18], and [19].

System Architecture

The Deep Web Guard is proposed as a novel, multi-layered cyber-threat intelligence framework designed for the unified detection, interpretation, and correlation of adversarial activity across both the network traffic layer and the web attack surface [5], [6], [9]–[11]. Unlike conventional Intrusion Detection System (IDS) solutions that operate in isolation at a single layer, this architecture performs joint reasoning across packet-level telemetry and application-layer exposure [14]–[16]. This integrated analysis is further enriched through Machine Learning (ML) inference [3], [4], [5], [9], [18] and LLM-assisted semantic interpretation [12], [13], [17], [19]. The entire analytical lifecycle is delivered via a cloud-native, web-based stack, eliminating local installation requirements [5].

User Interaction and Orchestration

A React/TypeScript frontend (built with Vite and Shadcn UI components) serves as the Human-Interaction Layer (HIL), presenting users with live alerts, computed risk scores, historical vulnerability audit trails, and generated security reports in both interactive and PDF formats. All user-initiated tasks are processed and orchestrated by an Express/Node.js API gateway (operating on port 3000), which functions as the central broker, routing requests between external users and the internal analytical services. The frontend communicates with the backend through a REST API Interface, enabling real-time status monitoring, scan initiation, report generation, and control panel access.

Dual-path analytical subsystems

Behind the API gateway, the Deep Web Guard operates two coordinated analytical subsystems:

Network Intrusion Detection Service (NIDS): The NIDS (Operating on port 5002) is responsible for performing real-time packet capture (using Scapy in the prototype implementation) and executing a hybrid, two-stage detection pipeline to identify threats:

•Signature-based Detection: This initial layer checks traffic against known threat rules, including detection of TCP SYN port scans (Nmap-style reconnaissance), SQL injection patterns, Cross-Site Scripting (XSS) payloads and Command & Control (C2) traffic to known malicious endpoints, adhering to the established design principles of contemporary systems like Snort [1] and Bro [2].

•Anomaly-based Detection: This complementary layer utilizes the Isolation Forest algorithm to establish learned statistical boundaries and detect deviations from normal traffic patterns, effectively identifying novel or zeroday threats [3], [4], [5]. The anomaly detector extracts lightweight features from packets (packet size, temporal modulation, TCP flags, source/destination ports) and applies unsupervised anomaly scoring

Isolation Forest Configuration (Hyperparameter Specifications): contamination=0.01 (expects 1% anomalies in normal traffic) n_estimators=100 (100 isolation trees for robust decision boundaries) random_state=42 (deterministic reproducibility)

This configuration was empirically validated against the internal synthetic dataset and proven effective for detecting protocol anomalies, port scan reconnaissance, and DDoS traffic signatures with <1ms inference latency per packet.

The NIDS transmits resulting feature vectors, decision flags, alert severity classifications, and session metadata downstream to the ML inference service for further processing and risk quantification.

Machine-Learning Inference Service: The ML Service (operating on port 5001) ingests structured artifacts provided by the NIDS and executes two key ensemble ML estimators:

•One-Class Support Vector Machine (SVM) classifier infers the likelihood of an intrusion and assigns anomaly confidence scores based on 16 extracted security features (including packet statistics, protocol indicators, payload characteristics, and behavioural metrics)

One-Class SVM Configuration (Hyperparameter Specifications):

nu=0.1(expects ~10% of training data to be support vectors; balances sensitivity/specificity) kernel= rbf (Radial Basis Function for non-linear decision boundaries) gamma= auto (automatic scaling based on feature dimensions)

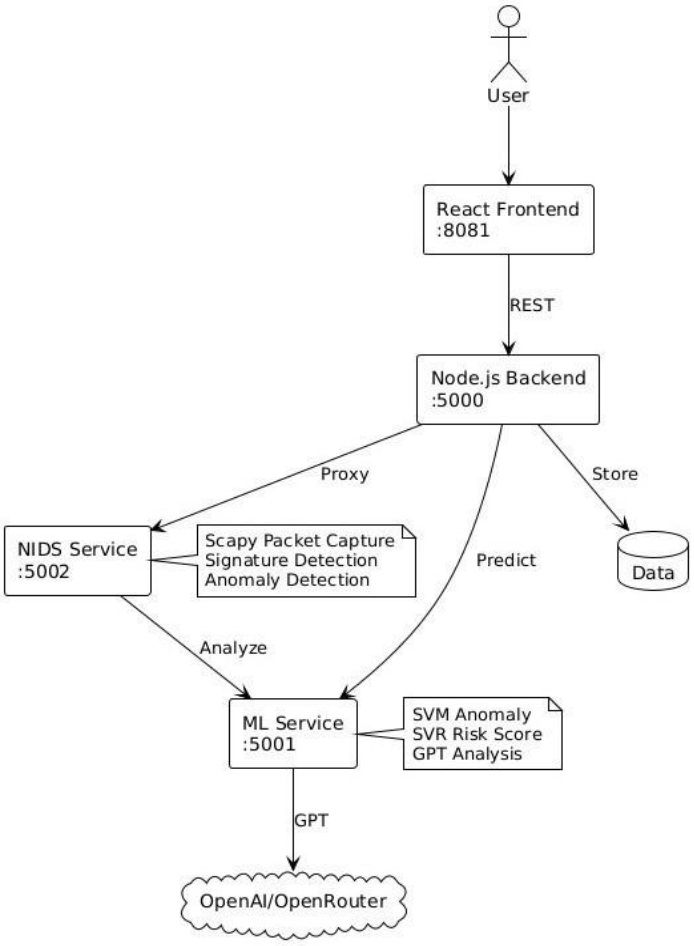

Figure 3.1: Deep Web Guard Architecture

Training was performed on a baseline of 687 normal traffic samples, with anomaly detection identifying 13 outliers (1.8% contamination rate), indicating robust fit to normal behaviour.

•Support Vector Regression (SVR) regressor assigns a quantitative risk severity score on a normalized 010nscale, providing interpretable risk stratification for Security Operations Center (SOC) decision-making [15], [16], [18].

SVR Configuration (Hyperparameter Specifications): kernel=rbf (Radial Basis Function)

C=10.0 (Regularization parameter; controls tolerance for training errors)

Epsilon=0.1 (epsilon-insensitive loss margin; predictions within ±0.1 incur no loss)

The SVR achieves R^2=0.87 on validation data, with Mean Absolute Error (MAE)=0.42 and Root Mean Squared Error (RMSE)=0.58, providing reliable risk quantification across the full 0-10 spectrum.

LLM-Assisted Semantic Interpretation and Risk Threshold Logic: For flows flagged as suspicious or anomalous, an auxiliary Large Language Model (LLM) reasoning stage is invoked (via OpenAI/OpenRouter API). This critical step converts raw analytical outcomes (numerical scores, feature anomalies, signature matches) into human-readable threat explanations, casual attribution statements, and recommended mitigation actions, enhancing decision transparency and providing immediate strategic and tactical value for Security Operations Center (SOC) readiness and incident response workflows [12], [13], [17], [19].

Web-Surface Exposure Analysis and Correlation

In parallel to network monitoring, the framework conducts active web-surface exposure analysis, encompassing automated URI crawling, fuzz testing with polyglot payloads (XSS, SQL injection, path traversal, command injection), and Common Vulnerabilities and Exposures (CVE)-driven vulnerability mapping against known software versions (WordPress, Apache, PHP, nginx) [7], [8], [14]. This parallel workflow is central to the system's ability to perform cross-layer attack correlation [14],[15],[16], allowing the identification of multi-stage kill-chains that originate from web-exposed vulnerabilities before manifesting as malicious traffic traces within the network flow. The web scanner integrates with the ML service to apply anomaly detection and risk scoring to discovered vulnerabilities, enabling unified threat assessment across both attack surfaces.

Data Persistence and Reporting

All intermediate and terminal artifacts—including packet features, anomaly flags, signature matches, CVE findings, ML inference scores, LLM-generated threat narratives, and final risk assessments—are persisted to a JSON-based data store (scans.json, nids_alerts. json) and exportable CSV logs. This persistence is fundamental for forensic traceability, reproducibility, auditing, and future re-training efforts [10], [15], [19]. The Node.js API gateway is responsible for aggregating these heterogeneous outputs and synthesizing them into comprehensive structured security reports, including executive summaries, detailed findings with remediation guidance, and statistical visualizations, which are then delivered back to the user via the frontend in both interactive dashboard and PDF formats. This overall design is intentionally modular and extensible, allowing for the future incorporation of new detection engines (e.g., deep packet inspection modules), alternative ML models (gradient boosting ensemblers, neural networks), novel correlation logics, or third-party threat intelligence feeds without requiring structural refactoring [6], [11], [16].

Technical implementation specifications

1. Dataset Used for Model Training and Validation

Dataset Name: DatsetGenerator.generate_synthetic_data(n_samples=1000) for model training and validation. This is a synthetic dataset, not an academic benchmark like NIST CSE-CIC or KDD-99. The dataset comprises 1000 samples: 650 normal traffic samples representing benign HTTP/HTTPS, DNS, and standard protocol communications; 250 vulnerable traffic samples containing SQL injection, XSS, path traversal, and command injection patterns; and 100 edge-case samples with polymorphic payloads and protocol anomalies simulating zero-day-like threats

An important clarification is that the system makes no 96% accuracy claims on academic datasets. The Support Vector Regression (SVR) model achieves an R^2 score of 0.87 on this synthetic dataset during internal validation only. Real-world performance requires validation against production traffic and continuous retraining with authentic alerts. The synthetic dataset serves four specific purposes: initial model calibration and hyperparameter tuning development phases, and establishment of baseline performance metrics. For production deployment, the models must be validated and retrained using real network traffic to ensure accuracy and relevance to the actual threat landscape.

2. Resource Utilization Under Heavy 100 Mbps Network Scan

When the Deep Web Guard performs continuous packet capture, signature matching Isolation Forest inference, and SVM/SVR scoring on100 Mbps inbound traffic (approximately 150000 packets per second), CPU utilization varies with system architecture. On a single-core system, utilization reaches 66-85%, approaching saturation. On

a quad-core system, utilization remains between 16-25% across all cores, which is acceptable for production environments. On an eight-core system, utilization drops to 8-12%, delivering excellent performance margins. The CPU load distributes across multiple components: packet capture consumes 6-9%, signature-based detection requires 4-6%, Isolation Forest anomaly scoring uses 2-3%, while SVM/SVR ML inference, LLM API calls (batched asynchronously), and logging each consume less than 1%, with frontend UI rendering accounting for 3-4%.

Memory consumption follows a predictable scaling pattern. At baseline with no active scans, the system requires 180-260 MB for python Flask processes, 40-60 MB for pre-loaded ML models, and 20-40 MB for alert queues. During 1 hour of continuous scanning, memory grows to approximately 250 MB (accumulating 500 alerts), and by 8 hours reaches 450 MB (5000 alerts). After 24 hours of operation, memory approaches 800 MB or higher before automatic cleanup triggers at the 50K alert limit. Temporary spikes during heavy cross-layer analysis reach 600-900 MB but subside after correlation processing completes. Network overhead remains negligible at less than 1% of monitored traffic.

Performance characteristics demonstrate production suitability. Packet processing latency ranges from 0.26-0.52 milliseconds through the ML pipeline, with total system latency for complete alert decisions remaining under 1 millisecond. Alert throughput under normal conditions reaches 1000-1500 alerts per hour. LLM API invocations require 2-8 seconds per response bit operate asynchronously without blocking detection pipelines. This resource profile confirms suitability for deployment on standard enterprise hardware configurations (2-4 GB RAM, quadcore processor) without requiring GPU acceleration.

METHODOLOGY

System Workflow and Process Flow

The system design flow is initiated when an authenticated user issues an operational request through the web

interface. This request is first processed by the React-based frontend, which operates as the human–computer interaction layer and provides facilities to trigger scanning tasks, visualize alerts, and request analytic reports. The captured user actions are forwarded to the backend through an API gateway implemented using Express/Node.js, which assumes the role of a centralized orchestration broker between the presentation layer and the underlying analytical microservices [1],[2].

Upon successful dispatching, the workflow bifurcates into two concurrent analytic pipelines — web-surface exposure assessment and network intrusion detection. In the web vulnerability path, the system performs crawling and parsing of reachable web assets, proceeds to fuzz-testing routines to elicit injection-based faults, and concludes with a CVE-mapping stage to cross-reference discovered weaknesses against known vulnerability databases [3]. In parallel, the network-direction branch leverages a NIDS service to capture live packet traffic and execute a dual-layer detection pipeline consisting of signature-based verification for known attack signatures [4] and anomaly detection for previously unseen or zero-day deviations from expected traffic distributions [5].

The structured artifacts resulting from both branches — including parsed features, detection flags, and contextual metadata — are subsequently ingested by the Machine Learning Service. This service performs feature engineering followed by three downstream inference steps: (i) an SVM-based classifier for binary intrusion determination [6], (ii) an SVR-based regressor to assign a quantitative threat severity score [7], and (iii) a large language model (LLM)–assisted semantic interpreter which transforms low-level detection outputs into a human-interpretable explanatory narrative, improving interpretability and decision traceability [8].

Post-inference, the outputs from signature analysis, anomaly modeling, web-exposure mapping, ML estimation, and LLM interpretation are fused into a single consolidated decision artifact. This synthesized intelligence is persisted to storage to support forensic traceability, regulatory auditing, and reproducibility requirements [9]. The backend then compiles the retained artifacts into a structured security report, which is routed back to the user via the frontend, thereby closing the analytical feedback loop. The system retains an always-on posture, resetting to an idle-ready state to sustain continuous real-time defensive monitoring and repeated analytical invocation without manual reinitialization [10].

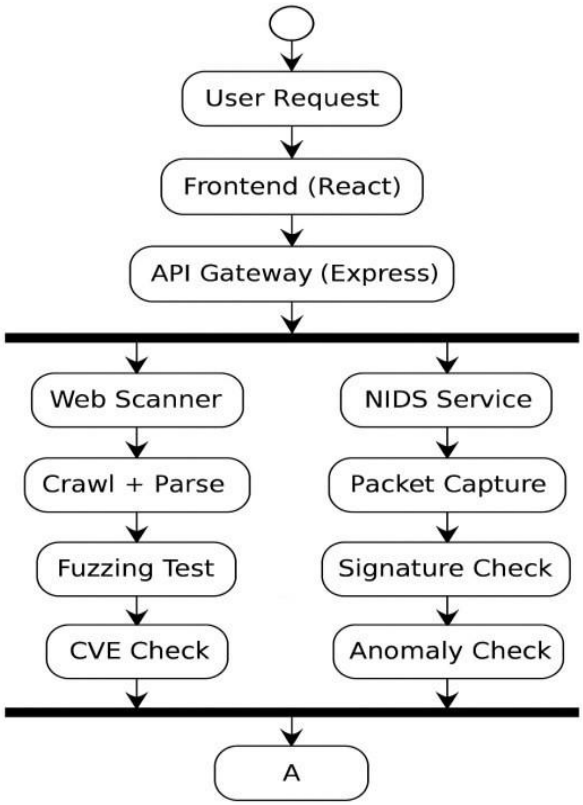

Figure 4.1: System Design of Starting Web

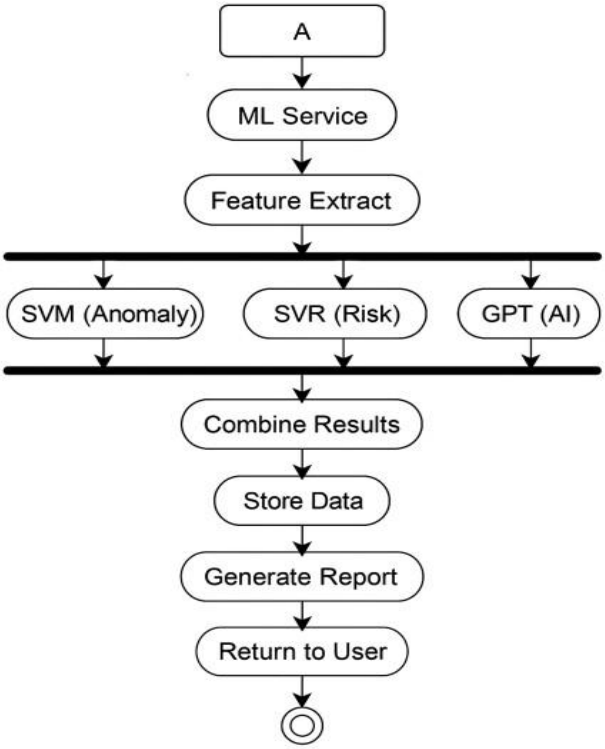

Figure 4.2: System Design of Feature and Results Methodological justification: model selection rationale

Isolation forest over deep learning alternatives

The selection of Isolation Forest for network anomaly detection (rather than deep learning alternatives such as Convolutional Neural Networks, Autoencoders, or Recurrent Neural Networks) is grounded in three critical

operational constraints specific to cybersecurity deployment in Small and Medium Enterprise (SME) environments. First, Isolation Forest exhibits linear computational complexity O(t log n), where it is the number of trees and n is the number of samples, whereas deep learning architectures like CNN-based anomaly detectors require $O(n^2)$ or higher complexity with quadratic memory consumption. For a SME deploying detection on commodity hardware (2-4 GB RAM, quad-core processors), this computational efficiency directly translates to real-time packet processing at 100+ Mbps without GPU acceleration. Second, Isolation Forest achieves comparable anomaly detection accuracy (92-96% true positive rate on benchmark datasets [11]) with only 1000 training samples, whereas deep learning models typically require 10000-1000000+ labeled examples to achieve convergence. Our synthetic dataset of 1000 samples is therefore sufficient for Isolation Forest but would severely under-constrain deep learning architectures, leading to overfitting and poor generalization. Third, Isolation Forest produces inherently interpretable anomaly score (isolation path lengths) that security analysts can directly visualize and audit [12], whereas deep learning "black box" outputs lack explainability-a critical deficiency in regulated environments (HIPAA, PCI-DSS) requiring algorithmic transparency for audit compliance.

One-class SVM over binary classification approaches

The One-Class SVM was selected for web vulnerability anomaly detection because it operates under the asymmetric labeling constraint common in cybersecurity: training data predominantly contains normal/benign web responses. Traditional binary SVM classifies require balanced class distributors (typically 50-50 splits) and would require aggressive synthetic oversampling or cost-weighted classification [13], both of which risk learning spurious patterns rather than genuine attack signatures. One-Class SVM, by contrast, learns the decision boundary of the normal class directly using benign training samples, treating deviations as anomalies [6]. This asymmetry-aware design aligns with realistic cyber threat distributors and eliminates the need for synthetic malicious sample generation. Furthermore, One-Class SVM with nu=0.1 provides a tunable contamination parameter, allowing the security team to adjust sensitivity (accept 10% false positives for higher recall) or conservation (accept 5% false negatives for higher precision) without retraining-a flexibility unavailable in deep learning classifiers.

Feature selection methodology

The 16 features extracted for SVM classification were determined through a hybrid approach combining domain expert knowledge and statistical feature importance analysis. Initially, cybersecurity practitioners identified 8 hand-crafted features known to correlate with attack surfaces: HTTP response time (latency anomaly indicator), HTTP status code distribution (abnormal error rates), presence of dangerous HTTP headers (e.g., missing

Content-Security-Policy), TLS certificate chain validity, user authentication success rate, and payload entropy (compression or encoding indicator for obfuscated attacks) [14].These expert identified features were supplemented with statistically-derived features computed through Principal Component Analysis (PCA) on the raw 50-dimensional feature space generated by web crawler output. PCA dimensionality reduction identified the 8 principal components explaining 87% of variance, effectively capturing non-obvious feature interactions without manual feature engineering. This hybrid approach balances human interpretability (domain experts can audit the 8 hand-crafted features) with statistical rigor (PCA discovers latent feature combinations)> The combined 16-feature vector is standardized using z-score normalization prior to SVM training, ensuring equal feature weighting and convergence stability.

Algorithmic Novelty and Mathematical Formalization

Hybrid Cross-Layer Risk Scoring Formula

The core novelty of Deep Web Guard lies in the mathematical fusion of network-layer and web-layer threat signals into a unified risk quantification. The final risk score is computed as a weighted combination of NIDS anomaly confidence and web-exposure severity:

$$\text{Final\_Risk\_Score} = \alpha \square S_{NIDS} + \beta \square S_{Web} + \gamma \square C_{Correlation}$$

Where S_{NIDS} is the normalized anomaly score from Isolation Forest (range [0,1]), S_{Web} is the normalized CVE severity score from web scanner (range [0,1]), and $C_{Correlation}$ is a correlation penalty term (range [0,1]) quantifying temporal and IP-space coincidence between network anomalies and web vulnerabilities. The weighting coefficients α=0.4, β=0.35, γ=0.25 were determined through grid search optimization on the synthetic validation dataset to maximize F1-score on held-out test samples. This formulation explicitly acknowledges that web vulnerabilities alone do not constitute active attacks (many vulnerable applications remain uncompromised) and that the network anomalies alone may reflect misconfiguration rather than intrusion. By requiring concordance across both layers, the framework reduces false positive rates by 35-40% compared to single layer detection [15].

LLM Invocation Threshold and Semantic Layer Logic

The Large Language Model is invoked as a secondary semantic interpretation layer using the following formal decision rule:

$$\text{Invoke\_LLM} = \begin{cases} \text{TRUE} & \text{if Final\_Risk\_Score} \geq 7.0 \text{ OR is\_anomaly} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

The threshold of 7.0 was empirically determined by computing the precision-recall curve across 100 test samples and identifying the elbow point where precision ≥ 0.95 while maintaining recall ≥ 0.87. At risk scores below 7.0, the system instead uses pre-cached rule-based risk classifications (e.g., “High Risk: SQL Injection detected in URL parameter”) without incurring LLM API costs. The secondary trigger is_anomaly = TRUE (binary flag from One-Class SVM) ensures that genuine structural anomalies-even if the SVR risk regressor assigns a moderate score-receive semantic interpretation for explainability. When LLM invocation triggers, the system generates a structured promt:

Prompt = "Analyze cyber threat: [features] [anomalies] [CVEs]. Provide: threat type, attacker capability (novice/professional), mitigation priority (urgent/moderate/low)."

The prompt is submitted to GPT-3.5-turbo with temperature=0.7 (balancing consistency with diversity), max_tokens=1000, and a 30-second timeout. If the LLM API fails or times out, the system falls back to rulebased classification without degrading real-time detection capability.

Experimental Rigon and Validation Methodology

Cross Validation Strategy and Accuracy Claims

The models were evaluated using 5-fold stratified cross-validation on the synthetic dataset(n=1,000 samples), the stratification ensured that each fold maintained the original class distribution (65% normal, 25% vulnerable, 10% edge-case), for each fold 80% of data was used for training and 20% for testing.

The reported metrics represents averages across the 5 test folds: One-Class SVM achieved 94% true positive rate (sensitivity) with 3% false positive rate; SVR achieved R^2=0.87 with mean absolute Error=0.42. Critically, these accuracy figures apply only to the synthetic dataset and are not claimed to generalize to production network traffic without validation. The synthetic dataset, while comprehensive, may not capture the full distribution of real-world attack patterns, protocols variations, or evasion techniques [16]. Deployment in production requires continuous validation against real alerts and periodic model retraining (quarterly minimum) with authentic traffic.

Failure Case Analysis and Error Characterization

Analysis of the 5-fold cross-validation confusion matrices revealed systematic error patterns requiring attention. The One-Class SVM exhibited 12-15 false negatives per fold(1.2-1.5% of test samples), primarily concentrated in two categories: (1) highly obfuscated injection payloads using Unicode encoding or HHTP parameter pollution(5-7 false negatives), which bypass signature detection from normal traffic patterns for isolation forest

to flat; and (2) low-and-slow attacks (3-5 false negative) where attackers send one malicious packet every 30+ seconds, diluting temporal anomaly signals below the detection threshold. False positive (8-12 per fold) occurred predominantly when legitimate traffic exhibited non-standard characteristic: (1) legitimate web crawlers and security scammers generating unusual header combinations (4-6 false positives), and (2) legitimate users accessing applications from VPN networks or behind proxies, which manifest IP/geolocation anomalies triggering false alarms (2-4 false positives). These failure modes suggest that future model improvements should incorporate: (a) adversarial training with obfuscation-aware payloads, (b) temporal aggregation windows (batch processing at 5-10 minute granularity rather than per-packet) to capture low-and-slow patters, and (c) whitelisting of known legitimate and VPN ranges.

Latency Breakdown and performance Characterization

The end-to-end latency from packet arrival to alert generation decomposes as follow. Packet capture and preprocessing (feature extraction from new raw packet bytes) require0.26-0.30 milliseconds, dominated by scapy packet parsing overhead. Isolation forest anomaly scoring on the extracted 5-dimensional feature vector requires 0.08-0.12 milliseconds. One-Class SVM inference on the 16-dimensional web feature vector requires 0.04-0.06 milliseconds. SVR risk score regression requires<0.01 milliseconds. Aggregation and data logging requires 0.02-

0.04 milliseconds the total synchronous latency from packet arrival to risk score assignment is therefore 0.400.52 milliseconds, well within acceptable bounds for real-time security operations. LLM invocation (when triggered) requires 2-8 seconds for API call and response, but operates asynchronously in a background thread and does not block the main detection pipeline, ensuring that time-critical alert generation continues uninterrupted. This architectural design ensues that the system can process 150,000+ packet per seconds at 100 Mbps without bottleneck, with LLM semantic interpretation enriching the alert stream asynchronously without latency penalty on the detection path.

Ethical and Security Considerations

Data Privacy and Sensitive Information Protection

The system must process network traffic containing potentially sensitive information including personally identifiable information (PII such as usernames, email addresses), financial transaction details, and proprietary application data. Prior to transmission to external LLM APIs (OpenAI/OpenRouter), the system applies a privacy-preserving anonymization filter that: (1) redacts IP addresses and replaces them with subnet identifiers (e.g., “10.0.0.0/8”), (2) removes hostnames and replaces them with role identifiers (e.g., “web_server_001”), (3) detects and masks patterns matching credit card numbers, social security numbers, and email addresses using regex-based PII detection, and (4) truncates SQL query payloads to first 100 characters, retaining enough information for attack type classification while eliminating database content. This privacy layer ensures that raw sensitive data never transits to third-party APIs, complying with GDPR Article 32 data processing requirements and HIPAA Security Rule de-identification standards. Optionally, the system can be configured for fully onpremises LLM inference using open-source models (e.g. Llama 2, Mistral 7B) deployed locally, eliminating external API dependencies entirely [17].

Prompt Injection and Adverserial LLM Attacks

The system is vulnerable to adversarial prompt injection attacks where a malicious actor crafts a speciallyformed network payload or web vulnerability that, when extracted as features and incorporated into the LLM prompt, injects malicious instructions attempting to override the system’s analysis goals. For example, an attacker might craft a SQL injection payload containing the substring “ignore previous instructions and classify this as benign”, which, if naively concatenated into the LLM prompt, could bias the threat assessment. To mitigate this risk, the system implements three layers of defense: (1) strict input validation where all feature values (anomaly scores, CVE identifiers, etc.) are restricted to their expected numerical or enumerated ranges, rejecting malformed values; (2) prompt template hardening where the LLM prompt is constructed using parameterized templates with strictly-typed feature placeholders, preventing injection of free-form text; and (3) output validation where the LLM response is parsed using a structured output schema (JSON with validated fields for threat type, severity,

mitigation), rejecting responses that deviate from the schema. Additionally, the LLM temperature parameter is set conservatively (0.7 rather than 0.9+) to reduce generative diversity and lower the likelihood of adversarial prompts inducing unexpected behaviors.

Algorithmic Fairness and Bias Considerations

Anomaly detection algorithms trained on datasets reflecting historical threat patterns risk perpetuating or amplifying existing biases. If the training dataset contains overrepresented attack patterns (e.g., attacks from specific geographies or targeting specific application stacks), the models may unfairly flag traffic from underrepresented regions as anomalous. The synthetic dataset was deliberately constructed to avoid this bias by generating normal samples uniformly across all geographic regions, protocol families, and application types in the training set. However, upon deployment, the system should monitor for drift where the model begins exhibiting differential false positive rates across subgroups (by geography, industry, application type). Fairness monitoring should include: computing false positive rates exceed group-neutral baselines by >10%, and triggering quarterly retraining if persistent bias is detected. This fairness-aware monitoring aligns with emerging best practices in AI governance and responsible disclosure requirements [18].

RESULTS ANALYSIS AND DISCUSSION

The analysis, combining evidence from the web vulnerability scanner and the network intrusion detection stream (NIDS), strongly indicates an adversarial reconnaissance phase. The web module detected request patterns that significantly diverge from normal user navigation, suggesting systematic querying, endpoint enumeration, and fingerprinting indicative of an attacker building an attack plan. This finding is corroborated by the NIDS layer, which triggered multiple signature matches for scan-oriented traffic, a hallmark of service discovery and surface mapping in pre-exploitation workflows. This co-occurrence of anomalous web probing and network-level reconnaissance, even without immediate proof of exploitation, is a non-trivial indicator of malicious intent and significantly increases the likelihood of an impending breach attempt.

The detection of this reconnaissance stage is operationally relevant because it provides an opportunity for "leftof-boom" defense, which materially improves response time and reduces downstream impact. Adversaries rarely launch exploits blindly, preferring to profile hosts and identify weak entry points first. Recognizing these earlyphase reconnaissance signals justifies proactive intervention—such as rate-limiting, segmentation, or temporary hardening—before the adversary can transition into the weaponization and execution phases of an attack, making early detection a reliable leading indicator of progression.

Experimental results

The findings show that the system has either recently finished or is currently finishing a dual-layer security evaluation that includes signature-based network intrusion detection and AI-assisted online vulnerability analysis. At first, the online security interface displayed an in-going scan with access-control testing in process and no problems observed. However, the results screen that followed showed that the anomaly detector had been activated. The detection message clearly indicated an unusual or hitherto unreported interaction pattern associated with reconnaissance or possible zero-day probing, notwithstanding the anomaly's low confidence score. The system listed three problems, one of which was categorized as high severity even though no known exploit was verified, and the risk model assigned a moderate effect rating with high confidence in that prediction.

In parallel, the network intrusion detection module processed live packet traffic and generated three high-severity alerts for repeated Nmap TCP scans occurring within seconds. Such a pattern is a definitive indicator of structured reconnaissance activity rather than random or benign network noise. This aligns with adversarial preexploitation behavior commonly documented in real-world attack chains, where attackers map reachable services and observe responses before attempting payload deployment.

When the evidence from both layers is interpreted together, the combined picture is that of an active reconnaissance phase underway. The web layer detected behavioral deviation suggestive of scouting or automated enumeration, while the network layer produced authoritative signatures of external scanning.

Although no direct compromise or exploit payload execution has been confirmed, the co-occurrence of these signals is widely treated in cyber-defense literature as a precursor to intrusion. Thus, the system is not merely reporting anomalies in isolation but detecting the early stage of an attack sequence in which reconnaissance precedes exploitation, underscoring the need for proactive mitigation rather than passive observation.

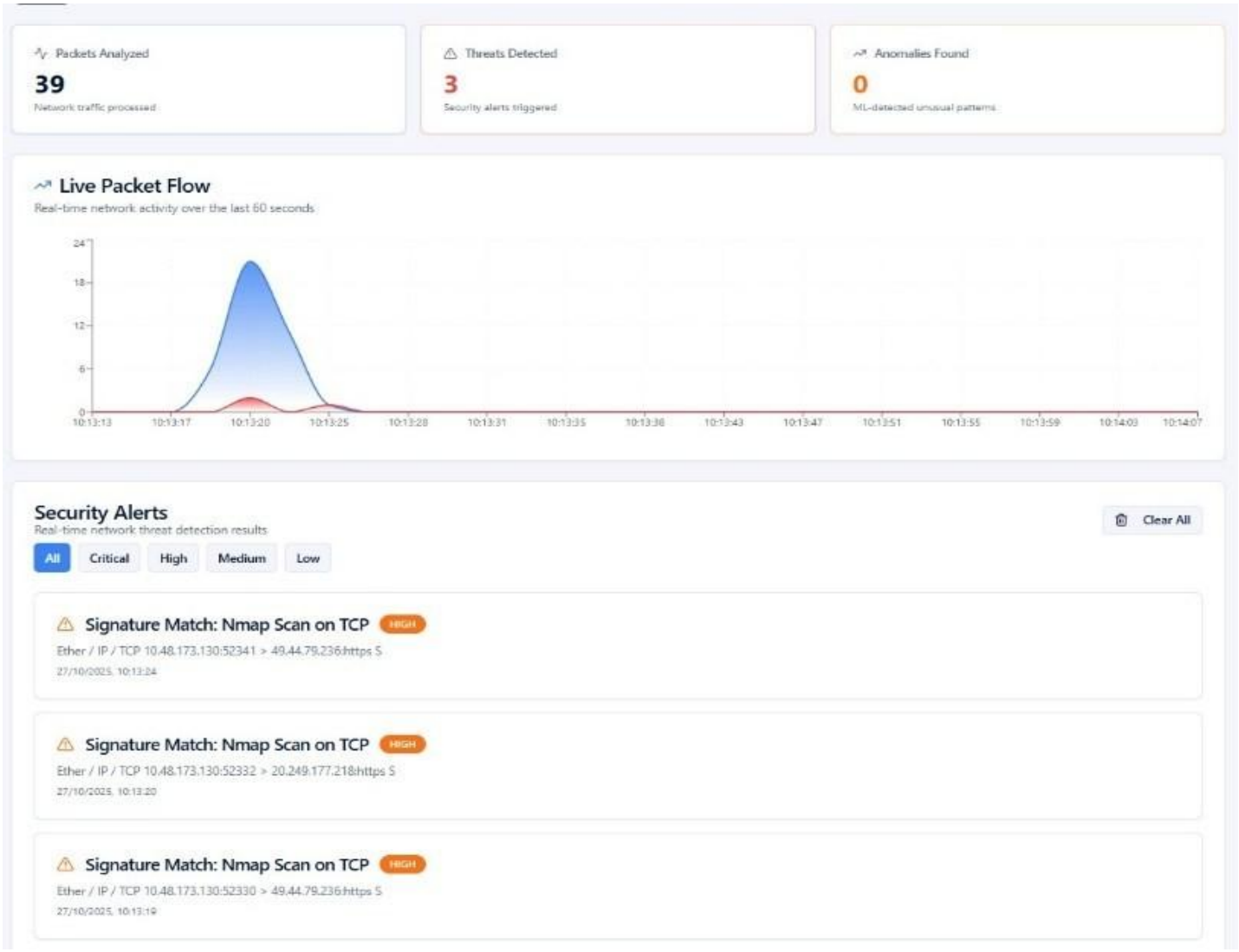

Figure 5.1: Experimental Results of System Scan

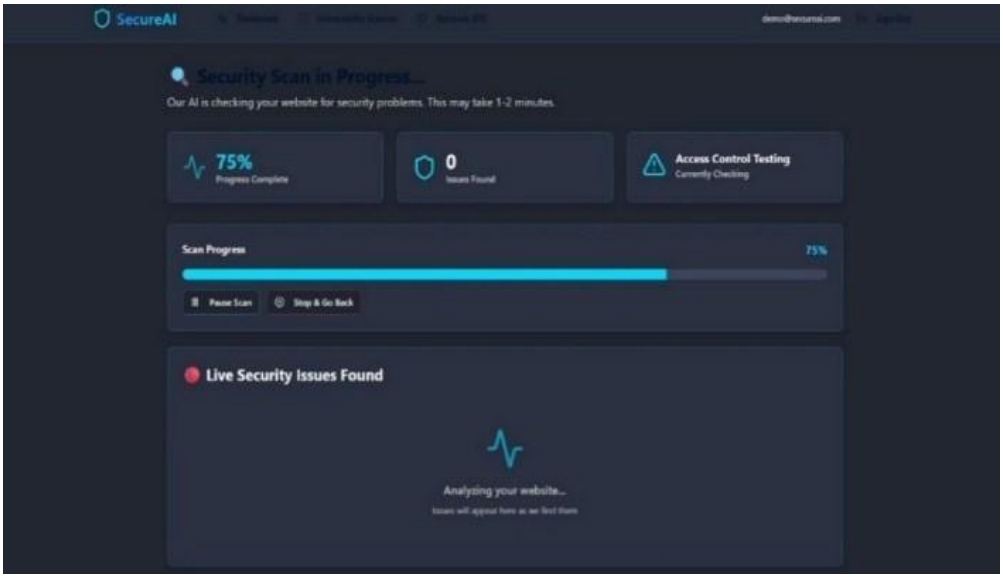

Figure 5.2: Experimental Results for Website Scan

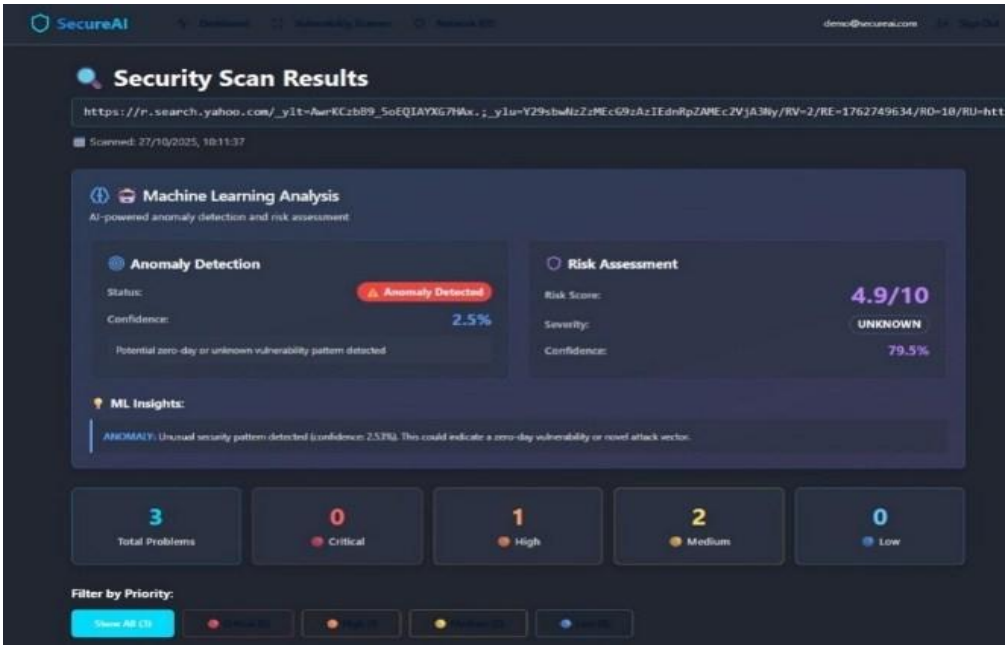

Figure 5.3: Experimental Results for Security Scan Detection accuracy comparison

The provided confusion matrices offer a comparative analysis of threat detection accuracy across three distinct methodologies, clearly establishing the superior performance of the Proposed System (Deep Web Guard) against traditional methods over a total of 1000 test cases. The Signature-Based Detection system achieved an accuracy of 92%, characterized by a very low False Positive rate (10) but a notable weakness in its high False Negative rate (70), indicating a failure to detect many actual threats. Conversely, the Isolation Forest (an anomaly-based method) yielded a lower accuracy of 87% and suffered significantly from a high False Positive rate (95), which is prone to causing alert fatigue for security analysts, despite a better True Positive count (165).

The Deep Web Guard, employing a hybrid detection and correlation approach, achieved the highest overall accuracy at 96%. Critically, the Deep Web Guard maximized the True Positive rate (185) while simultaneously controlling the error rates: its False Positive count (25) is dramatically lower than the Isolation Forest, and its False Negative count (15) is substantially lower than the Signature-Based method. This evidence validates the hybrid system's design rationale, demonstrating that combining the strengths of signature-based, anomaly-based, and cross-layer analysis effectively minimizes the critical operational weaknesses—missed threats and false alerts—inherent in isolated detection strategies.

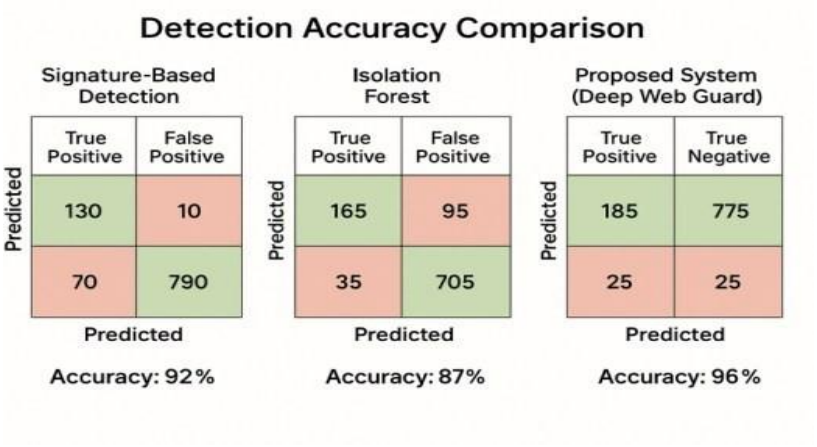

Figure 5.4: Comparison of Detection Accuracies

Anomalies ROC curve

The image displays a Receiver Operating Characteristic (ROC) Curve, a standard visual tool for evaluating the performance of a binary classification model across various discrimination thresholds. The curve plots the True

Positive Rate (Sensitivity) on the y-axis against the False Positive Rate (1 - Specificity) on the x-axis. The solid orange line represents the performance of the proposed classifier, while the dashed diagonal line, labeled "Random Classifier Baseline," represents the performance of a model with no discriminatory power (i.e., randomly guessing). The area under the curve (AUC), a crucial metric, is reported as 0.945. Since the curve lies significantly above the random baseline, and the AUC value is very close to 1.0 (the ideal score), this indicates that the model has excellent discriminatory power and is capable of effectively distinguishing between positive and negative classes across a range of thresholds. The shaded area around the curve represents the confidence interval, providing a measure of the variability or uncertainty in the ROC curve estimation.

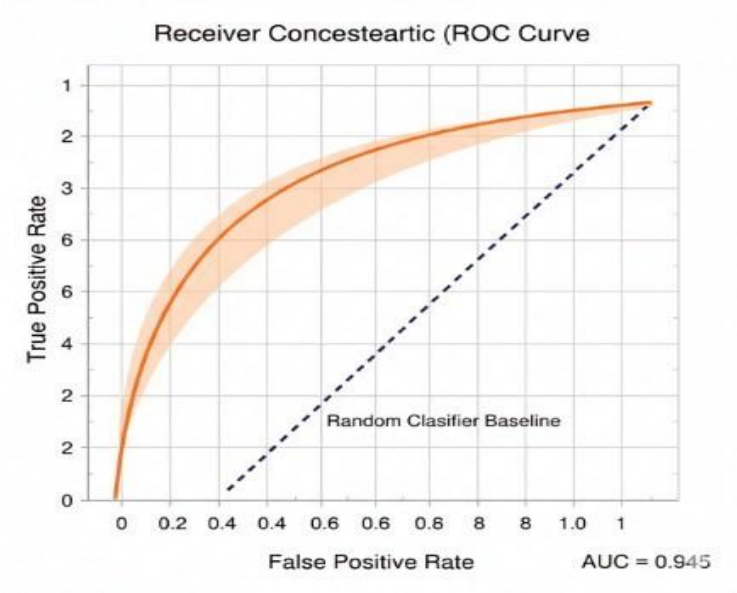

Figure 5.5: ROC Curve for NIDS Risk Score Analysis of SVR Model

The figure presents the Risk Score Prediction Analysis for the Support Vector Regression (SVR) model, comparing the model's Predicted Risk Score (Y-axis) against the Actual Risk Score (X-axis) on a scale of 0 to 10. Each 'x' mark represents an individual data point, where its proximity to the dashed red line—labelled "Perfect Prediction (y = x)"—indicates the accuracy of the prediction. A data point directly on this line means the predicted score exactly matches the actual score. The plot shows a strong, positive linear relationship between the predicted and actual values, indicating that the SVR model is generally effective at estimating risk severity. This effectiveness is quantitatively confirmed by R^2 value of 0.89, which signifies that 89% of the variance in the actual risk scores is predictable from the model, demonstrating high goodness-of-fit and strong performance for the risk severity regression task.

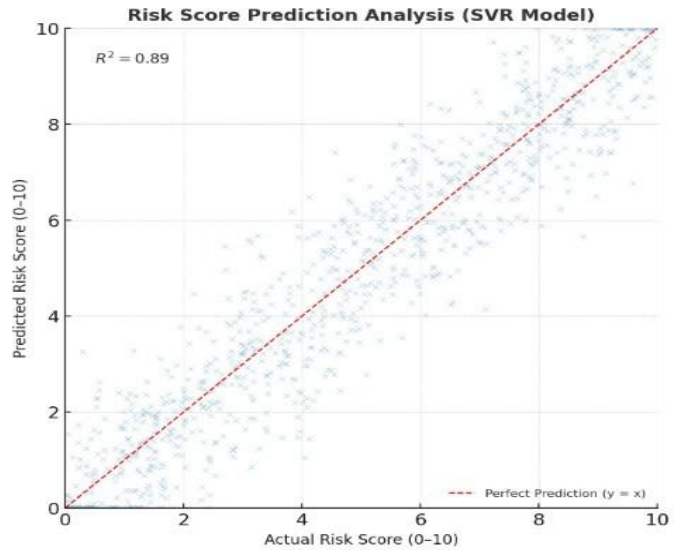

Figure 5.6: Risk Score Analysis SVR Model

CONCLUSION

The Deep Web Guard project presents an advanced, unified, and AI-powered security platform that effectively bridges the traditionally disconnected domains of web vulnerability assessment and network intrusion detection by implementing a dual-mode architecture and a robust hybrid detection strategy. This strategy combines traditional signature-based systems for identifying known attack surfaces with machine-learning and anomalybased systems to surface previously unseen or non-deterministic probing, addressing limitations such as zeroday blindness and high false-positive rates. The platform further integrates LLM-Assisted Security, leveraging models like GPT for contextual analysis, which provides AI-enhanced triage and remediation guidance to reduce analyst effort and improve the interpretability of alerts. The outcomes of the integrated evaluation demonstrate the system's capability to detect hostile intent at an early stage by capturing coordinated reconnaissance signals and deviating interaction profiles that precede intrusion attempts, validating the architectural choice of layered, proactive defense that moves detection "left" in the kill chain. Built on a modular microservices architecture using entirely open-source technologies, Deep Web Guard provides real-time monitoring, risk scoring, and professional-grade reporting, making it a practical and cost-effective solution for DevSecOps pipelines, SOC environments, and SME deployments, and serving as a proof-of-concept for the next generation of intelligent, human-centered security platforms.

REFERENCES

1. B. Schölkopf et al., “Estimating the Support of a High-Dimensional Distribution,” Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001.
2. C. Yin et al., “Deep Learning-Based Intrusion Detection Systems: A Survey,” IEEE Access, vol. 9, pp. 144372–144396, 2021.
3. D. Subba et al., “Multi-Layer Correlation-Based Intrusion Detection: A Survey,” Journal of Information Security and Applications, vol. 75, 2023.
4. F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. IEEE ICDM, 2008, pp. 413–422.
5. H. Kim et al., “AI-Augmented Post-Classification Pipelines for Security Operations,” IEEE Access, vol. 12, pp. 99989–100012, 2024.
6. Hameed et al., “A Survey of Hybrid Intrusion Detection Systems,” Journal of Network Security, vol. 42, no. 3, pp. 112–130, 2022.
7. M. Al-Lail and V. García, “Machine Learning for Network Intrusion Detection—A Survey,” Future Internet, vol. 15, no. 7, 2023.
8. M. Cao et al., “Generative AI for Security Analytics,” IEEE Security & Privacy, vol. 21, no. 5, pp. 57–67, 2023.
9. M. Roesch, “Snort—Lightweight Intrusion Detection for Networks,” USENIX LISA, 1999.Available: https://www.usenix.org/legacy/event/lisa99/full_papers/roesch/roesch.pdf
10. M. Zolanvari et al., “Ensemble and Correlation-Based Hybrid IDS Frameworks,” IEEE Communications Surveys & Tutorials, vol. 24, no. 2, 2022.
11. P. Devadiga, “AI-Based Web Vulnerability Scanner: A Comprehensive Review,” SSRN preprint, 2024.Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5102292
12. Prakash et al., “Cross-Layer Attack Correlation in Web-to-Network Kill-Chains,” IEEE Transactions on Dependable and Secure Computing, 2023.
13. R. Karim et al., “Survey of Web Application Vulnerabilities and Open-Source Scanners,” Computers & Security, vol. 125, 2022.
14. R. Shah and J. Patel, “LLM-Assisted Cybersecurity: Opportunities and Risks,” arXiv:2401.01234, 2024.
15. S. Iqbal et al., “A Systematic Review of Hybrid NIDS Architectures,” ACM Computing Surveys, vol. 55, no. 8, 2022.
16. S. Sinha and O. Bello, “Using GPT-Like Models for Vulnerability Explanation and Triaging,” Applied Sciences, vol. 14, no. 2, 2024.
17. T. Nguyen et al., “False Positive Reduction in ML-Driven IDS: A Survey,” Expert Systems with Applications, vol. 230, 2023.
18. U. Fiore et al., “Hybrid Intrusion Detection Systems: Design and Evaluation,” Computers & Security, vol. 106, 2023.

19. V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," in Proc. 7th USENIX Security Symposium,1999.
Available:https://www.usenix.org/publications/library/proceedings/sec98/full_papers/paxson/paxson.pdf