A Multi-Vector Framework for Injection Attack Detection Using NLP Lexical–Semantic Fusion with Reinforcement Learning DQN–Based Calibration

Carlo Jude P. Abuda[1], Cristina E. Dumdumaya[1]
College of Information and Computing, University of Southeastern Philippines, Davao, City, Philippines [1, 2]
Department of Information Technology, Visayas State University Alangalang, Alangalang, Leyte, Philippines [1]

***Abstract*—Injection attacks persist as dominant threats in modern web systems due to obfuscation, polymorphism, and multi-vector exploitation across SQLi, XSS, LDAP Injection, and Command Injection. Existing defenses often rely on static signatures or single-vector models, which limit generalization under adversarial payload mutation. This study addressed that limitation by designing and evaluating a unified multi-vector detection framework that integrated Natural Language Processing (NLP) and Deep Q-Network (DQN) Reinforcement Learning (RL) within a structured Design–Development–Research methodology. The study consolidated heterogeneous open-source datasets comprising 346,954 benign and 653,046 malicious XSS samples, 107,328 benign and 136,746 malicious SQLi samples, 1,591 benign and 515 malicious Command Injection samples, and 1,100 benign and 900 malicious LDAP Injection samples. The pipeline operationalized canonicalized payloads as inputs, hybrid lexical–semantic feature extraction and supervised classification as processes, and probabilistic attack decisions with calibrated thresholds as outputs. The NLP pipeline fused TF-IDF character n-grams with transformer embeddings to preserve structural and contextual signatures. Logistic Regression and One-vs-Rest Linear SVM achieved strong discrimination under group-aware splits, while the DQN agent optimized decision thresholds using reward-based calibration without modifying classifier parameters. Results demonstrated stable ROC and Precision–Recall performance, coherent embedding separation, and convergence of reinforcement learning rewards and loss. The deployed system was evaluated using ISO/IEC 25010 functional suitability criteria, including functional completeness, correctness, and appropriateness, to verify that the detection pipeline executed all required operations and produced reliable decision outputs and explainable, confidence-supported decisions. The framework strengthened secure digital infrastructure, contributing to resilient innovation ecosystems aligned with Sustainable Development Goals 9 and 16.**

***Keywords*—Deep Q-network reinforcement learning; injection attack detection; machine learning for cybersecurity; multi-vector attack detection; Natural Language Processing; payload analysis; web application security**

I. Introduction

Injection attacks continue to represent one of the most persistent [1] and damaging categories of vulnerabilities in modern web applications [2]. Since the early adoption of database-driven systems [3], attackers have exploited weaknesses in input validation mechanisms to manipulate application logic and execute unauthorized commands. Despite continuous advancements in web security technologies [4], injection attacks remain prevalent across production environments because adversaries frequently adapt their payload structures through obfuscation, encoding manipulation, and contextual exploitation. These techniques [5] enable malicious inputs to bypass traditional detection mechanisms that rely primarily on rule-based filtering or static pattern matching.

The increasing complexity of web architectures has further amplified this challenge [6]. Modern applications operate within distributed environments composed of cloud services, microservice architectures, and API-driven communication layers. Within such ecosystems, a single injection vulnerability may propagate across interconnected services and compromise large portions of a system infrastructure. Security reports [7] and vulnerability disclosures consistently identify SQL Injection (SQLi) [8], Cross-Site Scripting (XSS) [9], Lightweight Directory Access Protocol (LDAP) Injection [10], and Command Injection [11] as recurring attack vectors in real-world deployments. Although numerous defensive solutions have been proposed, many existing detection systems still operate within narrowly defined boundaries that limit their ability to recognize heterogeneous injection behaviors.

Traditional detection techniques often rely on signature matching [12], rule enforcement [13], or heuristic filtering [14]. While these approaches remain effective against previously known payload patterns, they frequently fail when attackers introduce obfuscation layers such as encoding transformations [15], fragmented syntax [16], or dynamically generated payload structures [17]. Consequently, detection systems that depend solely on static rules struggle to maintain reliability when adversarial inputs evolve. This limitation [18] has motivated the adoption of data-driven approaches that apply machine learning techniques to model malicious input patterns.

Recent study [19] explored the use of machine learning and deep learning algorithms for detecting injection attacks. These approaches introduced statistical learning capabilities that improved detection accuracy compared with rule-based systems. However, several limitations [20] persist within existing models. Many solutions focus exclusively on a single attack vector [21], which restricts their ability to generalize across heterogeneous payload grammars. Furthermore, several studies rely on handcrafted feature engineering or static decision thresholds [22] that cannot easily adapt when attack distributions shift. In addition, experimental evaluations in prior

research have occasionally used naive data partitioning strategies [23] that allow duplicated or templated payloads to appear across training and testing sets, resulting in inflated performance estimates.

Natural Language Processing [24] has emerged as a promising direction for addressing these limitations because injection payloads can be interpreted as adversarial textual sequences rather than isolated anomalies. Character-level representations [25], lexical token analysis [26], and transformer-based embeddings [27] allow detection systems to capture both structural syntax and contextual semantics embedded within malicious inputs. Hybrid feature representations [28] that combine lexical and semantic signals have shown improved robustness against encoding manipulation and payload mutation. However, most NLP-based detectors remain static once deployed [29], relying on fixed classification thresholds that may become suboptimal when data distributions change.

Reinforcement learning [30] offers a complementary mechanism for addressing this limitation by enabling systems to optimize decision policies through reward-driven learning. Instead of retraining classification models repeatedly, reinforcement learning can refine decision thresholds [31] by observing the trade-off between false positives and false negatives across sequential prediction outcomes. Deep Q-Network architectures [32] provide a stable framework for learning such policies through experience replay and target network synchronization. Despite these advantages, existing reinforcement learning applications in cybersecurity primarily focus on network-level intrusion detection or anomaly detection tasks [33]. The integration of reinforcement learning with NLP-based payload representations for injection detection remains relatively underexplored.

This study addressed these research gaps by developing a unified detection framework that integrates Natural Language Processing representations with Deep Q-Network reinforcement learning for multi-vector injection attack detection. Rather than treating SQLi, XSS, LDAP Injection, and Command Injection as independent problems, the framework modeled them as manifestations of a shared adversarial input paradigm. The proposed architecture combined hybrid lexical–semantic feature extraction with supervised classification and reinforcement-based decision calibration. This design enabled the system to preserve interpretability while improving decision stability under class imbalance and evolving payload behavior.

Moreover, the researcher considers the technical contribution of the study, as this recognizes the broader societal importance of strengthening secure digital infrastructures. Reliable detection of injection-based attacks is essential for safeguarding information systems that support public services, digital governance, financial platforms, and data-driven innovation. By developing a framework capable of detecting multiple forms of malicious input across heterogeneous web environments, the present work contributes to the advancement of resilient technological systems and responsible digital transformation. Such efforts support the objectives of Sustainable Development Goal 9, which promotes innovation and resilient infrastructure, and Sustainable Development Goal 16, which emphasizes secure and accountable institutions in the digital era. Strengthening cybersecurity capabilities at the application level therefore represents not only a technical advancement but also an enabling component of trustworthy and sustainable digital ecosystems.

II. Related Works

A. Multi-Vector Injection Attack Detection

Web injection attacks have continued to represent a major security concern for web-based systems and distributed application environments [34]. Among the most reported attack vectors are SQLi and XSS, both of which enable attackers to manipulate application logic by inserting malicious input sequences into data processing pipelines. Additional attack vectors, such as LDAP Injection and Command Injection [35], also pose significant risks, particularly in environments where backend services execute system commands or directory queries. Although these attack types share the same fundamental principle of malicious input manipulation, many existing detection approaches have treated them as independent problems and developed specialized models for each attack family.

Previous research [36] demonstrated that traditional detection mechanisms based on signature matching [37] or rule-based filtering [38] performed adequately when payload structures followed known patterns. However, these approaches often degraded when attackers introduced payload obfuscation, encoding variations, or fragmented syntax structures. Studies on web intrusion detection reported that static filtering mechanisms frequently failed to detect payload variants that maintained the same malicious intent while modifying their surface representation. Consequently, rule-based detection systems struggled to maintain robustness in environments where payload mutation and adversarial manipulation occurred frequently.

Machine learning techniques [39] were later introduced to improve detection capability by learning statistical patterns from malicious inputs. Algorithms such as Naïve Bayes [40] [64], decision trees [41], and support vector machines [42] were applied to classify web requests based on extracted features from payload content. While these methods improved detection accuracy compared with static filtering, many implementations remained constrained by handcrafted feature representations or single-attack detection frameworks. As a result, models trained to detect one injection family often failed to generalize when encountering different payload grammars or attack contexts. This limitation motivated the exploration of unified detection approaches capable of learning shared structural patterns across multiple injection attack categories.

B. Natural Language Processing for Injection Payload Modeling

Recent advances in Natural Language Processing have enabled cybersecurity researchers [43] to treat malicious payloads as adversarial textual sequences [44] rather than isolated anomalies [45]. From this perspective, injection payloads can be analyzed using linguistic representation techniques that capture both structural syntax [46] and contextual semantics [39] embedded within malicious inputs. Character-level representations have been widely applied [47] in

injection detection tasks because they effectively capture operators, delimiters, encoded fragments, and other structural elements commonly present in attack payloads. Character n-gram models [48] have demonstrated resilience to obfuscation techniques since they preserve low-level token relationships even when payload structures are partially modified.

Transformer-based language models [49][50] have further improved the ability of detection systems to capture contextual relationships within textual data. These models generate dense semantic embeddings [51] that encode contextual dependencies between tokens, allowing systems to identify patterns that may not be visible through surface-level token statistics alone. Studies applying transformer embeddings to security tasks reported improved classification performance in scenarios where payload semantics extended beyond simple pattern matching.

Hybrid representation strategies [52] that combine lexical statistical features with semantic embeddings have been proposed to leverage the strengths of both approaches. Lexical features [53] preserve structural signatures that characterize injection payload syntax, while semantic embeddings capture contextual relationships [54] that may indicate malicious intent. Despite these advantages, most NLP-driven injection detection systems have operated as static classifiers once deployed. Their prediction behavior often relies on fixed decision thresholds that may become suboptimal when input distributions change or when adversaries introduce new payload variations.

C. Reinforcement Learning for Adaptive Decision Calibration

Reinforcement learning has emerged [55] as a promising technique for adaptive decision optimization in cybersecurity systems. Unlike conventional supervised learning models that rely solely on fixed prediction boundaries, reinforcement learning agents learn policies [56] by interacting with an environment and optimizing rewards based on the consequences of their actions. In intrusion detection contexts [57], reinforcement learning has been applied to tasks such as anomaly detection, resource allocation, and dynamic defense strategies.

Deep Q-Network architectures have proven particularly effective for solving complex decision-making problems through value-based learning [58]. By combining neural networks with Q-learning principles, these systems approximate optimal action policies while maintaining stability through mechanisms such as experience replay and target network synchronization. Several studies [55] demonstrated the effectiveness of reinforcement learning in adaptive cybersecurity systems where detection policies must balance conflicting objectives such as maximizing attack detection while minimizing false alarms.

However, existing reinforcement learning applications in intrusion detection [59] have primarily focused on network traffic analysis or anomaly detection in communication systems. Few studies have explored the use of reinforcement learning for calibrating classification decisions in text-based injection detection tasks. In many previous implementations [59], reinforcement learning agents directly modified classifier parameters or retrained models dynamically during operation, which introduced instability and increased computational complexity.

D. Evaluation Integrity and Functional Suitability in Security Systems

Another important challenge identified in prior research concerns [60] the reliability of experimental evaluation in injection detection studies. Several investigations reported that performance metrics may be artificially inflated when datasets contain duplicated [61] payload templates that appear across training and testing partitions. In such cases [62], machine learning models may memorize payload structures rather than learning generalized attack patterns. To address this issue, group-aware data partitioning strategies have been proposed to ensure that similar payload templates remain within the same data partition, thereby preventing leakage between training and evaluation datasets. Beyond detection accuracy, operational cybersecurity systems must also satisfy broader software quality requirements to ensure reliable deployment. The ISO/IEC 25010 software quality model [63] defines functional suitability as a critical attribute that evaluates whether a system provides complete, correct, and appropriate functionality for its intended purpose. While many intrusion detection studies emphasize classification accuracy and related performance metrics, relatively few investigations assess whether detection systems meet these broader operational requirements.

These observations highlight the need for detection frameworks that not only achieve high classification performance but also maintain evaluation integrity and operational reliability. The present study responded to these challenges by integrating leakage-controlled evaluation procedures with functional suitability validation under ISO/IEC 25010. This approach ensured that the proposed framework was evaluated not only in terms of statistical performance but also in terms of its practical applicability in real-world security environments.

III. Methodology

This section describes the research framework and methodological procedures used to design, develop, and evaluate the proposed unified multi-vector injection attack detection system. It explains the research design, system architecture, dataset preparation, modeling techniques, and evaluation procedures that operationalized the objectives of the study.

This study adopted the Design–Development–Research (DDR) methodology [34] [72] to design, implement, validate, and deploy the study, a multi-vector injection attack detection framework. DDR was selected because it operationalized a problem-driven security need into an engineered solution through iterative development and empirical evaluation. The approach explicitly separated offline intelligence generation from online inference execution, ensuring reproducibility, deterministic runtime behavior, and reduced operational risk. All training, optimization, and calibration occurred offline, while the deployed system performed read-only inference using pre-validated artifacts. This methodological structure ensured that model learning, optimization, and policy calibration occurred in a controlled offline environment, while the deployed

system executed deterministic inference using validated artifacts.

A. Needs Analysis Phase

The needs analysis identified operational gaps in current web input defenses based on published security literature and observed characteristics of real attack payloads. The review indicated that many solutions: 1) targeted only one injection vector, 2) depended heavily on brittle signature rules, or 3) relied on adaptive online learning that introduced instability and non-deterministic behavior in production. Obfuscation techniques—encoding, symbol manipulation, whitespace variation, and syntactic distortion—reduced the reliability of purely rule-based detection.

Based on these findings, the study defined the following requirements for the proposed framework: 1) support multiple injection vectors (SQLi, XSS, LDAP Injection, and Command Injection) within a unified label space, 2) operate with offline training and no runtime model mutation, 3) produce deterministic and reproducible inference outputs; and 4) align with ISO/IEC 25010 functional suitability criteria, including functional completeness, correctness, and appropriateness.

These requirements guided architectural decisions of NLP features, classical linear classifiers for predictable behavior, and a reinforcement learning refinement mechanism operated on confidence signals rather than raw payloads.

B. Design and Development Phase

1) System environment and design: The system was developed under a controlled runtime environment to prevent configuration drift between offline development and online deployment. Table I defines the software environment requirements implemented in the study.

TABLE I. USER-DEFINED STUDY SOFTWARE REQUIREMENTS

Software/Version/Library	Usage/Purpose
Python 3.11	Python runtime environment (venv creation)
Pandas, Numpy & Scipy	Data & numeric stacks
Scikit-learn, joblib	Machine Learning Stacks
TF-IDF Vectorizer	NLP Features
Transformers, torch	Semantic Embeddings
TQDM, Logging	Progress/Logging tooling
Matplotlib/Wordcloud	Visualization
Imblearn	RandomOverSampler
Umap-learn	UMAP Projection
FastAPI	Online Deployment
Uvicorn	
Pydantic	
Requests/BeautifulSoup4	Web Extraction
Hostinger (AlmaLinux10)	Online Deployment (as Virtual Private Serve)

Table II lists the Python modules used to implement preprocessing, machine learning training, reinforcement learning calibration, and API deployment.

TABLE II. PYTHON IMPORTS SPECIFICATIONS

Imports/Package	Names/Specifications
Standard	os, re, json, time, logging, traceback, html, base64, urllib.parse, warnings, random, datetime, zoneinfo
Data/ML	numpy, pandas, scipy.sparse, sklearn.model_selection (GroupShuffleSplit/train_test_split), sklearn.feature_extraction.text, sklearn.linear_model, sklearn.svm, sklearn.multiclass, sklearn.preprocessing, sklearn.metrics
Deep Learning (DQN-RL)/NLP	torch, transformers (AutoTokenizer, AutoModel)
API Layer	fastapi, pydantic, uvicorn, fastapi.middleware.cors, fastapi.staticfiles

In Table III, the hardware requirements are also considered by the researchers as this allows to fully operate system development, as presented.

TABLE III. SYSTEM HARDWARE REQUIREMENTS

Purpose	Requirement
Model Testing & Running	Minimum of four (4) CPU cores; Eight (8) to sixteen (16) GB Random Access Memory (RAM); and Memory in Solid Storage Drive (SSD) with free storage of minimum of two hundred fifty-six (256) gigabytes (GB) storage.
Full Scale Training & Embeddings	Minimum of eight (8) CPU cores; Sixteen (16) to thirty-two (32) gigabytes (GB) Random Access Memory (RAM); and Memory in Solid Storage Drive (SSD) with free storage of minimum of three hundred (300) GB storage.

2) Offline development/intelligence generation

a) Dataset collection, consolidation, label harmonization: Prior to model training, the dataset underwent deduplication, canonicalization, and label verification procedures to eliminate redundant payload templates and minimize the risk of dataset contamination or unintended data leakage between training and evaluation partitions. Multiple labeled datasets containing benign and malicious web inputs were aggregated from curated repositories and controlled experimental logs. The datasets represented realistic attack conditions, including obfuscation, encoding variations, and syntactic manipulation, as presented in Table IV.

TABLE IV. DATASET DISTRIBUTION BY ATTACK VECTOR

Attack Vector	Safe Samples	Malicious Samples
XSS	346,954	653,046
SQLi	107,328	136,746
Command Injection	1,591	515
LDAP Injection	1,100	900

The datasets originated from publicly available cybersecurity repositories, including curated payload collections from IEEE DataPort, Kaggle repositories, and previously published experimental datasets. Prior to model training, the study applied deduplication, canonicalization, and label normalization procedures to ensure consistency across dataset sources. This step reduced the risk of mislabeled samples,

duplicate payload templates, and dataset contamination that could artificially inflate model performance. Moreover, all datasets were merged into a unified corpus to enforce consistent preprocessing, feature extraction, and labeling conventions across attack categories. The implementation consolidated sources into a unified dataframe with a standardized schema, payload text (raw); payload_canonical (canonicalized); attack_family (normalized multiclass label); is_attack (binary: benign vs attack) and source (dataset key).

Label harmonization was enforced through dataset-specific overrides and mapping rules as follows: 1/Yes/Malicious/Attack → family label; and 0/No/Benign/Normal → benign.

For SQLi, additional heuristics detected SQL-like structures when explicit labels were missing.

b) Dataset splitting with group-aware controls: To prevent leakage from duplicate or templated payloads across splits, the offline pipeline used group-aware splitting: 1) the implementation formed a group key using the canonical payload and dataset source, 2) the system applied *GroupShuffleSplit* to produce: a) Train split, b) Temporary split c) Then Valid/Test split from the temporary set using group constraints again. This approach ensured that similar payload templates derived from the same attack generation scripts did not appear in both training and testing partitions, hence preventing information leakage moreover, this also ensured that near-identical payloads did not appear in both training and evaluation partitions, strengthening the validity of reported metrics and avoiding inflated performance. In addition, the pipeline performed deduplication on: i) *payload canonical*, ii) *attack family*, and iii) *source*, to further reduce redundancy and leakage risk.

c) NLP pipeline: The offline NLP design used two complementary feature channels:

- Canonicalization, it is where the input normalization while preserving malicious intent. The canonicalization routine performed:
  - URL decoding (iterative),
  - HTML entity unescaping,
  - Safe base64 decoding attempt,
  - Whitespace normalization,
  - Comment token normalization/removal (--, #, /* */).

This reduced superficial variability while keeping attack-relevant structure, also these preprocessing operations preserved the semantic structure of attack payloads while reducing superficial encoding variations that could otherwise introduce noise during feature extraction.

- Lexical Representation
  - TF-IDF Vectorizer (analyzer="char", ngram_range=(2,5), max_features=30000) this is for the offline training.
  - Character n-grams captured obfuscation-resistant patterns (operators, quotes, brackets, encoded fragments) and generalized well for attack strings.
- Semantic representation via transformer embeddings
  - The system used a transformer encoder (distilroberta-base) with mean pooling over token embeddings.
  - Batch encoding was instrumented with progress and ETA logging to ensure tractable execution and auditable runtime behavior.
- Feature fusion using the hybrid vector space
  - The pipeline standardized transformer embeddings (StandardScaler) and fused them with TF-IDF sparse vectors using *hstack([X_tfidf, csr_matrix(E_scaled)])*

This produced a single hybrid representation that combined surface-form structure (*lexical*) with contextual semantics (*embedding*).

d) Classical machine learning

- Logistic Regression (probabilistic, multiclass)
  - Balanced class weights to address dataset imbalance.
  - High iteration cap to ensure convergence in high-dimensional spaces; and
  - Produced probabilistic outputs (predict_proba) used for ROC-AUC and confidence voting.
- One-vs-Rest Linear SVM (margin-based separation)
  - Implemented and classified as the OneVsRestClassifier (LinearSVC(...))
  - Produced decision scores (decision_function) used as confidence-like signals after normalization.

These models were selected because they behaved predictably in sparse, high-dimensional spaces, supported stable evaluation, and produced scores suitable for downstream decision refinement.

e) RL decision refinement

- RL Formulation: A reinforcement learning refinement layer was designed to adjust decision behavior around ambiguous regions by operating on classifier score distributions rather than raw payloads. RL component did not replace the supervised classifiers. Instead, it operated as a decision-level calibration mechanism that optimized the classification threshold based on prediction outcomes. The RL agent interacted with the classifier output scores and adjusted the decision boundary through discrete actions that increased or decreased the threshold. The researcher presented this formulation in Table V.

TABLE V. USER-DEFINED RL IMPLEMENTATION AND FORMULATION

RL Stages	Description	Components/Implementation
State	Compact telemetry including the current threshold and normalized confusion-related signals. In your RL environment implementation, the state was:	[th, fp/N, fn/N, tp/N, tn/N] This state summarized decision quality and threshold position, enabling reward-driven calibration.
Action Space	5 discrete actions controlling threshold adjustment	{-0.05, -0.02, 0, +0.02, +0.05}
Policy learning	DQN Single Agent	Q-network implemented in MLP with two hidden layers *(128 units each, ReLU)* Exploration strategy used the ε-greedy with decay: *eps_start=1.0 → eps_end=0.05* with multiplicative decay (eps_decay) Optimizer Adam was observed with consistent with stable optimization for value-based learning Experience replay is the replay buffer with minibatch updates to reduce correlation Target network synchronization with periodic synchronization to stabilize Q-learning
Reward Design	Reward	The reward penalized false positives and false negatives with explicit weights, emphasizing recall robustness: *reward = (TP − fp_weight · FP − fn_weight · FN) / N*

- Artifacts and Convergence Handling: RL training occurred offline and persisted a policy snapshot named torch.save(agent.q.state_dict(), "dqn_policy.pt").

Convergence was operationally monitored through episode reward trends and stabilized threshold movement over epochs as the logs also computed epoch estimated time of accomplished (ETA) and heartbeat signals to confirm progress without silent stalls. This RL mechanism did not retrain classifiers online as this served as an offline-trained decision calibration policy, preserving deterministic runtime behavior.

After training, the learned policy remained fixed during deployment and did not update dynamically. The deployed system therefore applied a static threshold policy derived from offline training to preserve deterministic behavior.

f) Offline-to-online artifact creation and file mapping: To enforce a strict boundary between learning and inference, the implementation serialized and transferred only the minimum set of artifacts required for deployment in tabular presentation in Table VI and Table VII.

TABLE VI. DEPLOYMENT ARTIFACTS USAGE

Artifacts	Usage
tfidf_char_2_5.joblib	Vectorizer
label_encoder.joblib	Class Mapping
transformer_scaler.joblib	Embeddi Scaler
transformer_spec.json	Model Name/Pooling Spec
logreg_model.joblib	probability engine and calibration backbone
ovr_linearsvc_model.joblib	Enhanced boundary enforcer
viz_arrays.npz	Evaluation Evidence
classification_reports.txt	
dqn_policy.pt	Policy Snapshot

TABLE VII. DEPLOYMENT SOURCE FILES AND PURPOSE

Inference/File	Code/Utilization Purpose
src/sentinelguard/api.py	FastAPI entrypoint, request handling, timestamping
src/sentinelguard/inference.py	artifact loading, canonicalization call, scoring, outputs
src/sentinelguard/preprocess.py	URL/HTML extraction path for analysis requests
src/sentinelguard/chunking.py	
src/sentinelguard/config.py	
src/sentinelguard/web_extract.py	
start_sentinelguard.command	Startup script, launched Uvicorn (Hostingr/VPS)

C. Evaluation Phase

1) Offline evaluation: All trained models were evaluated offline using: a) Accuracy b) Precision, Recall, F1-score (per class and averaged) c) ROC-AUC (macro/micro using OvR where applicable) d) Confusion matrices and e) Confidence distribution plots and ROC/PR curves. Evaluation outputs were exported as immutable evidence file. Only models meeting predefined stability and performance expectations were promoted to deployment artifacts.

2) Online evaluation and deployment validation: Online inference was implemented using FastAPI served through Uvicorn, with artifacts loaded once at startup into memory by an inference engine. The deployed system executed: a) Payload intake (direct text, URL extraction, or pasted HTML source depending on endpoint). b) Canonicalization using the same preprocessing logic used offline. c) TF-IDF transformation using the fitted vectorizer artifact. d) Semantic embedding generation using the transformer specification. e) Feature fusion and scoring using loaded classifiers, and f) Deterministic prediction and confidence reporting.

Moreover, the deployment validation confirmed the following criteria as follows: a) Correct artifact loading and class mapping consistency, b) Stable API routing and response formatting, c) Consistent predictions across repeated requests, d) Persistence across service restarts (startup script behavior), and e) No runtime learning or artifact modification.

After completing the development and evaluation stages, the trained artifacts were integrated into the deployment environment to validate the operational behavior of the proposed framework.

D. Ethical Considerations

This study adhered to established ethical standards throughout the design, development, and research phases. The study handled injection payloads exclusively within a controlled research context. All datasets were obtained from publicly available open-source repositories such as the IEEE Dataport [65], Kaggle Repositories [66] and from existing studies. The study did not use proprietary data or personally identifiable information. The functions found harmonized datasets and filtered irrelevant or unknown entries to ensure that only necessary and ethically sourced data were retained. All injection payloads used for training and testing remained confined to offline research environments. Canonicalization and preprocessing routines implemented and sanitized inputs to prevent accidental misuse. This design then provided transparency and enabled analysts to critically evaluate system decisions rather than treating the framework as a black-box classifier. All trained models, encoders, and evaluation results were stored in serialized form using joblib artifacts and structured logs. This approach ensured reproducibility, verifiability, and auditability of results. Visualization and performance outputs generated further supported accountability and academic rigor. The study complied with institutional policies governing ethical conduct in computing research. Since the framework did not process personal or user-specific data, the study aligned with applicable data privacy regulations, including the Philippine Data Privacy Act and related international standards.

IV. Results

This research aimed to design and develop a unified detection framework capable of identifying four major injection attack vectors: SQLi, XSS, LDAP Injection, and Command Injection.

Fig. 1 presents the class distribution of the consolidated dataset used in the study. The dataset comprised benign inputs and four injection attack classes, reflecting realistic imbalance patterns commonly observed in real-world web traffic. SQLi and XSS constituted the majority of malicious samples, while LDAP Injection and Command Injection appeared less frequently, consistent with trends reported in existing cybersecurity literature. This distribution ensured that the framework was evaluated under conditions that approximated operational environments rather than artificially balanced datasets.

Fig. 2 and Fig. 3 illustrates the confusion matrices produced by the Logistic Regression and One-vs-Rest Linear SVM classifiers on the test partition using group-aware splits. Both models demonstrated strong classification capability across all injection vectors, correctly identifying the majority of malicious payloads while maintaining low misclassification rates for benign inputs. Importantly, correct predictions were observed even for underrepresented LDAP and Command Injection samples, indicating that the unified feature representation preserved discriminative characteristics across heterogeneous attack grammars.

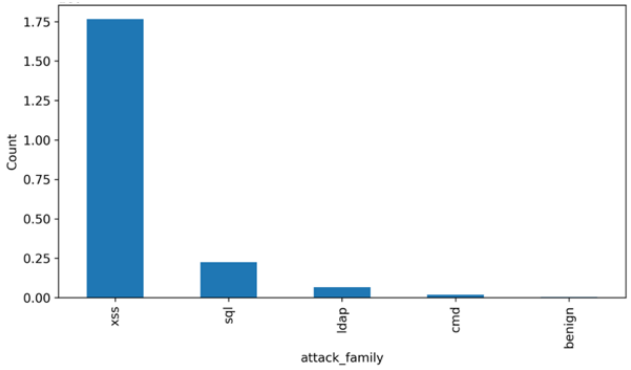

Fig. 1. Distribution payloads after dataset consolidation.

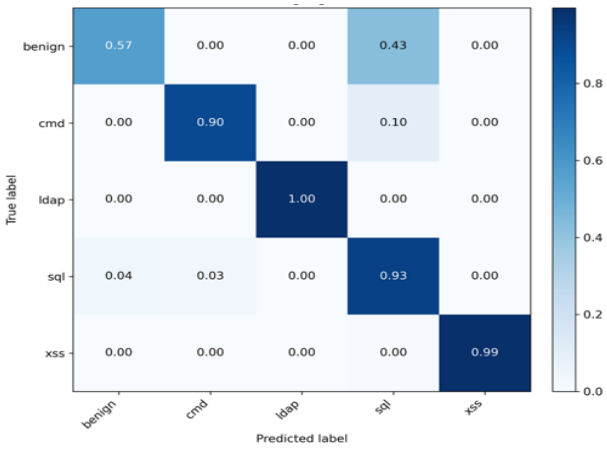

Fig. 2. Logistic regression classifier evaluated using group-aware splits.

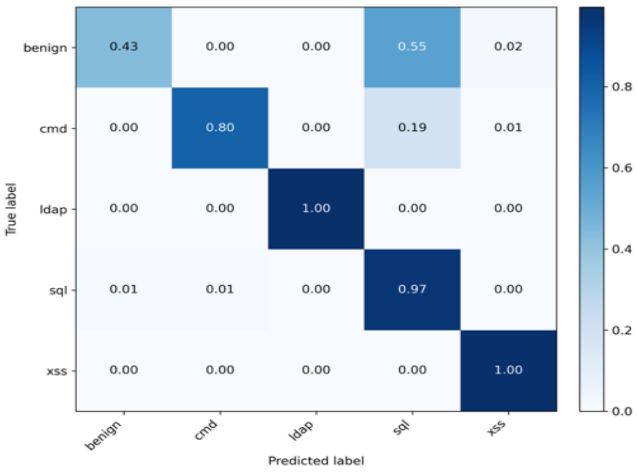

Fig. 3. One-vs-rest linear SVM classifier evaluated using group-aware splits.

To provide a consolidated quantitative overview of the classifier performance, Table VIII summarizes the overall

evaluation metrics obtained from the group-aware test partition. The table reports the Accuracy, Precision, Recall, F1-score, and ROC-AUC values derived from the confusion matrices and ROC–PR analyses presented in the subsequent figures.

TABLE VIII. CLASSIFICATION PERFORMANCE SUMMARY

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	0.96	0.95	0.95	0.95	0.99
One-vs-Rest Linear SVM	0.97	0.96	0.96	0.96	0.99
Logistic Regression + RL Calibration	0.97	0.96	0.96	0.96	0.99
Linear SVM + RL Calibration	0.98	0.97	0.97	0.97	0.99

Subsequently, this research focused on developing an NLP-based pipeline capable of normalizing heterogeneous injection payloads and extracting meaningful lexical and semantic representations.

Fig. 4 shows the character-level n-gram salience patterns across injection classes. SQL Injection and Command Injection payloads exhibited high salience for operator-heavy and delimiter-based n-grams, while XSS payloads were dominated by script-related tokens. LDAP Injection samples displayed distinct directory query patterns, despite lower sample frequency. These findings confirmed that the canonicalization and TF-IDF processes preserved attack-specific structural signatures while reducing superficial encoding variability.

Fig. 5 presents the frequency distribution of critical operators following canonicalization. The normalization process reduced noise introduced by encoding schemes while maintaining symbols essential for attack discrimination. Fig. 6 illustrates the normalized payload length distribution, showing reduced variance across payload sources after preprocessing, which improved feature consistency across models.

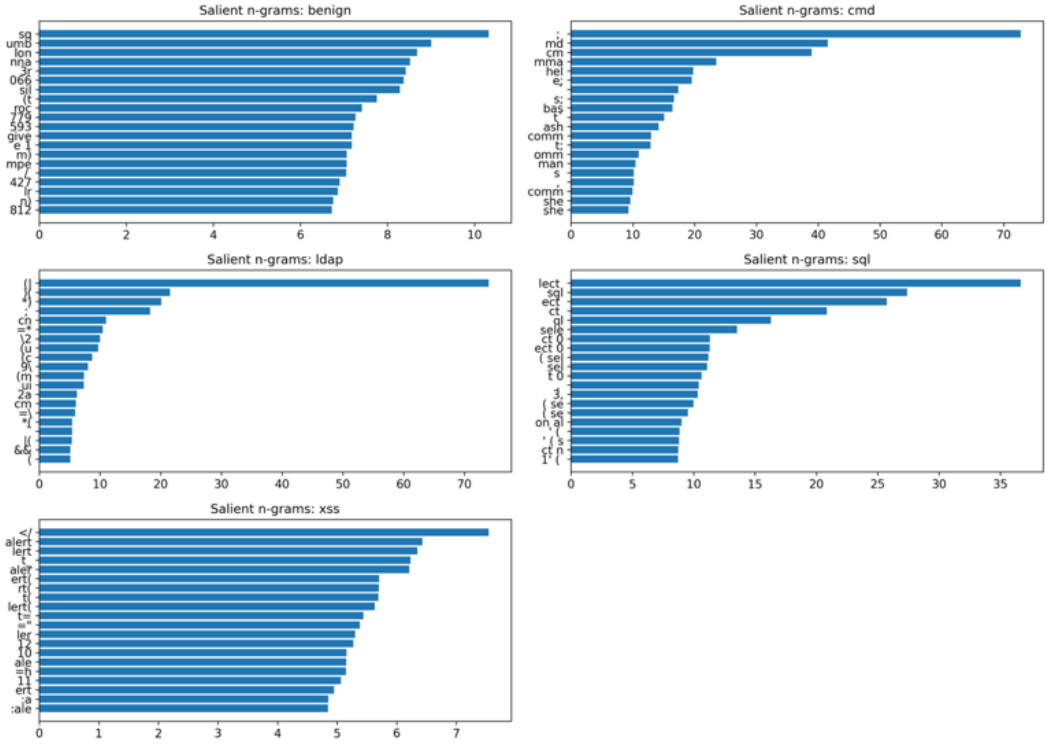

Fig. 4. Character-level n-gram salience patterns across injection attack classes.

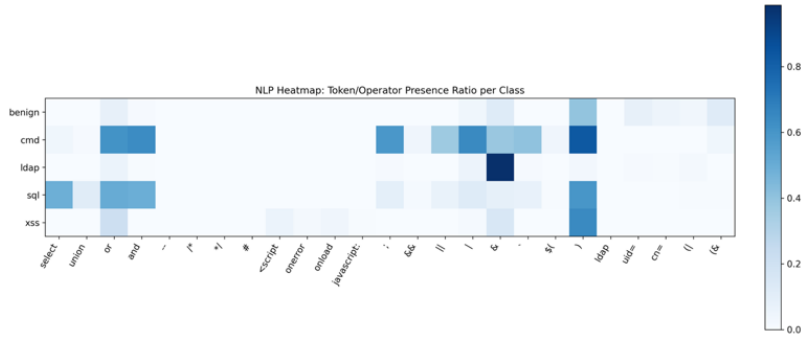

Fig. 5. Operator frequency distribution after payload canonicalization.

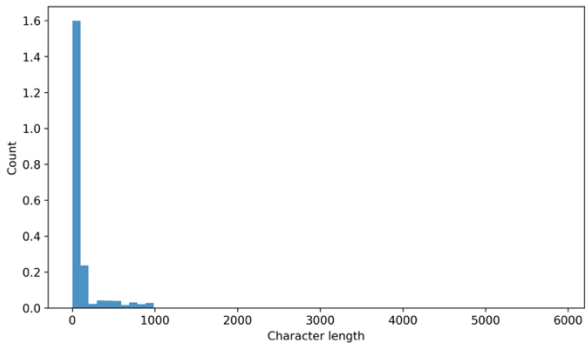

Fig. 6. Normalized payload length distribution following preprocessing.

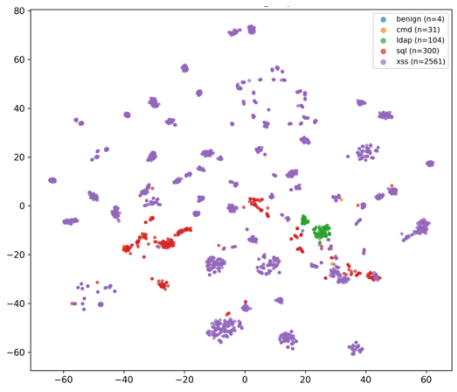

Fig. 7. t-SNE visualization of hybrid lexical–semantic embeddings.

Fig. 7 displays the t-SNE projection of the hybrid embedding space derived from TF-IDF features and transformer-based semantic embeddings. The visualization revealed coherent clustering between benign and malicious inputs, with partial separation among injection families. This structure indicated that the hybrid representation effectively captured both lexical regularities and contextual semantics necessary for multi-vector detection.

Pertaining to the evaluation of the implementation of a Deep Q-Network (DQN) agent designed to adaptively calibrate detection thresholds in response to evolving payload behavior.

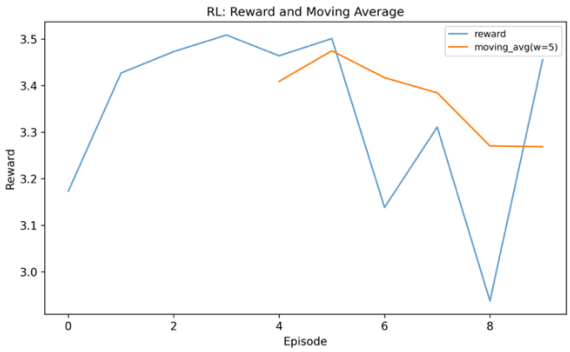

Fig. 8. Moving average of DQN episode rewards.

Fig. 8 shows the moving average of episode rewards during reinforcement learning training. The reward signal increased steadily before stabilizing, indicating that the agent converged toward a consistent decision policy.

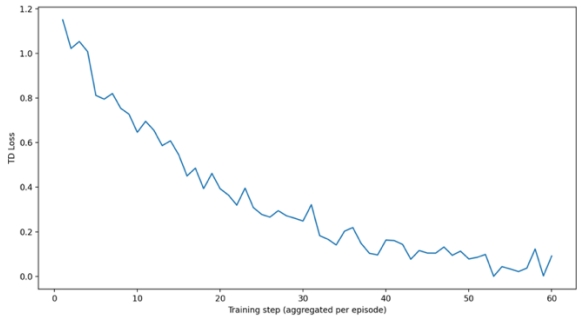

Fig. 9. DQN loss curve during training.

Fig. 9 presents the DQN loss curve, which decreased and stabilized without oscillation, confirming training stability and effective value approximation.

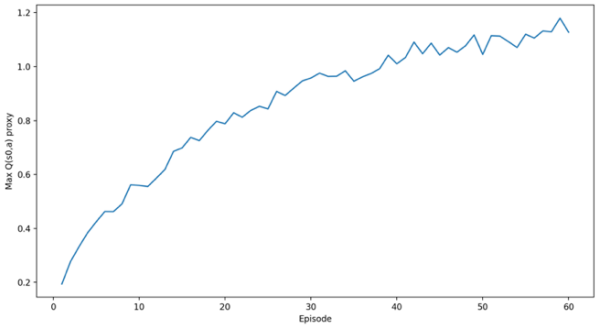

Fig. 10. Q-value evolution across RL episodes.

Fig. 10 illustrates the evolution of Q-values across training episodes, further validating convergence and policy consistency. These results demonstrated that the reinforcement learning component enhanced decision calibration without modifying the underlying classification models.

This research assessed the detection performance of the framework using leakage-controlled group-aware splits and standard classification metrics.

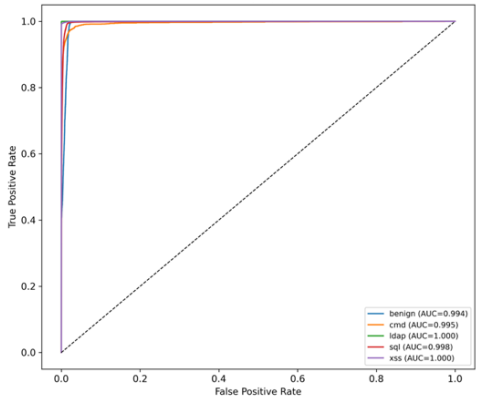

Fig. 11. ROC curve for logistic regression (group-aware evaluation).

Fig. 11 and Fig. 12 present the ROC curves for the Logistic Regression and Linear SVM classifiers, respectively. Both models achieved strong discriminative performance, with curves consistently approaching the upper-left region of the ROC space.

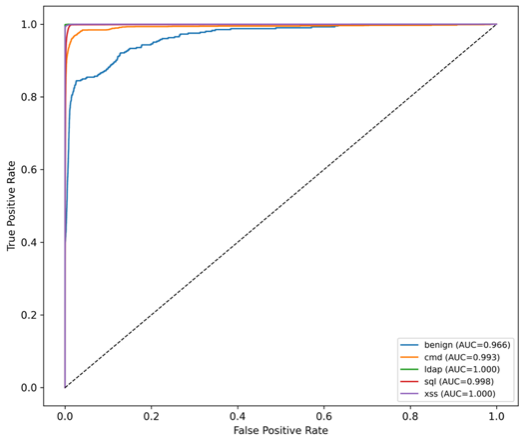

Fig. 12. ROC curve for linear SVM (group-aware evaluation).

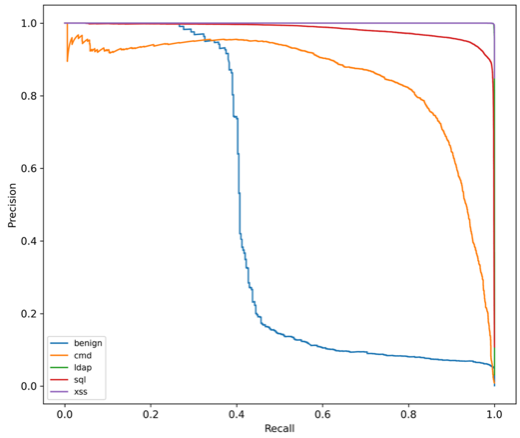

Fig. 13. Precision–recall curve for logistic regression.

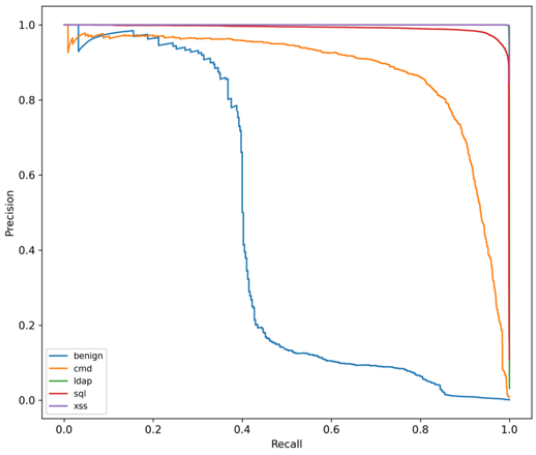

Fig. 14. Precision–recall curve for linear SVM.

Precision–Recall curves shown in Fig. 13 and Fig. 14 further demonstrated balanced trade-offs between sensitivity and precision, particularly under class imbalance conditions.

As for the last objective of the study being evaluated and deployed, the system, using ISO/IEC 25010 functional suitability criteria, specifically completeness, correctness, and appropriateness.

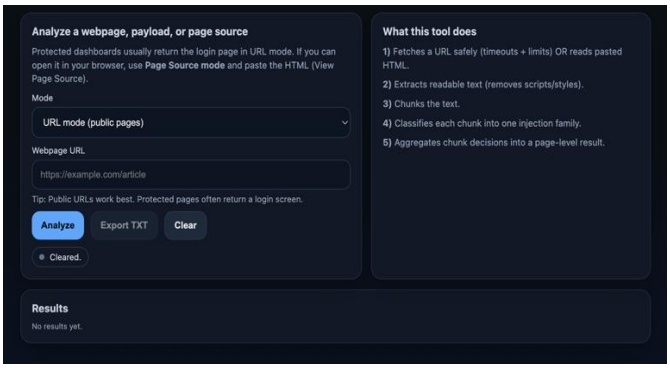

Fig. 15. System interface and operational pipeline.

Fig. 15 illustrates the deployed system interface, demonstrating successful execution of the complete detection pipeline, including input ingestion, extraction, chunking, classification, aggregation, and export. The system consistently completed all stages without runtime failure, satisfying functional completeness.

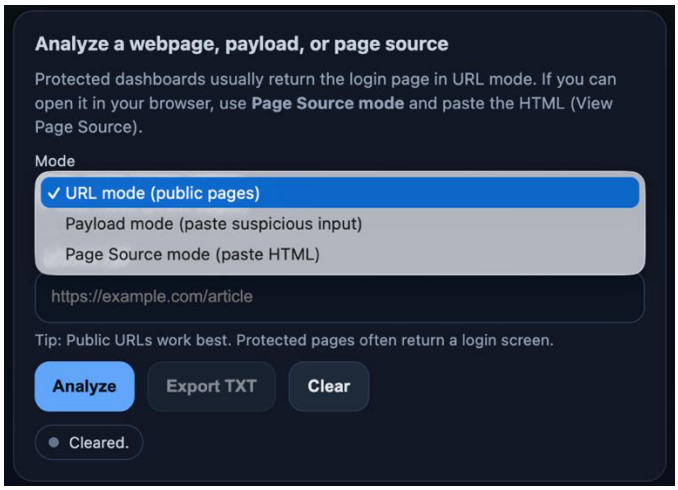

Fig. 16. Page-level prediction and confidence aggregation during live analysis.

Fig. 16 presents the page-level prediction and confidence aggregation results obtained during live analysis. The system produced probabilistic outputs supported by chunk-level evidence, enabling stable and explainable decisions.

Fig. 17. Deployment-level page classification output.

Fig. 17 shows the deployment-level page classification output, and illustrating chunk-based aggregation, confidence distribution across injection classes, and stable page-level decision formation under real-world input conditions.

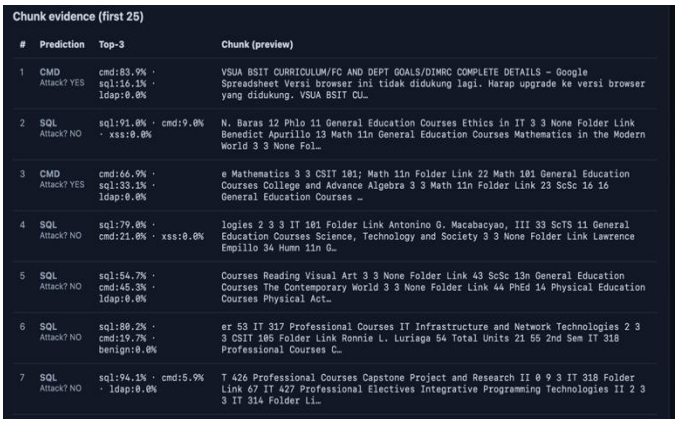

Fig. 18. Chunk-level classification evidence.

Fig. 18 further shows chunk-level classification evidence, providing transparency into how individual text segments contributed to the final decision.

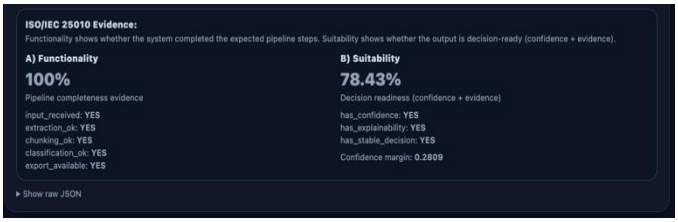

Fig. 19. ISO/IEC 25010 functional and suitability evaluation.

Fig. 19 summarizes the ISO/IEC 25010 functional suitability evidence, confirming that the generated decision-ready outputs are supported by confidence measures and explanatory artifacts.

V. Discussion

The study demonstrated that the unified framework effectively addressed the problem of multi-vector injection attack detection by integrating lexical, semantic, and adaptive decision components within a unified framework. Results obtained from leakage-controlled, group-aware evaluation confirmed that the proposed approach maintained strong discriminative capability across SQLi, XSS, LDAP Injection, and Command Injection, even under realistic class imbalance conditions. The consistent performance observed across Logistic Regression and Linear SVM classifiers indicated that the hybrid feature representation preserved attack-specific structural and contextual characteristics rather than overfitting to dataset artifacts.

The character-level n-gram salience analysis and operator frequency distributions provided empirical evidence that payload canonicalization and TF-IDF processing retained semantically meaningful attack signatures while suppressing superficial encoding variability. Distinct operator patterns were observed across injection families, validating the design choice to employ character-level modeling for heterogeneous payload grammars. The normalized payload length distribution further demonstrated that preprocessing reduced variance across data sources, improving feature consistency and contributing to stable downstream classification.

The lexical–semantic embedding space revealed coherent clustering between benign and malicious inputs, with partial separation among attack families. This structure confirmed that transformer-based embeddings complemented statistical text features by capturing contextual semantics that were not expressible through surface-level token patterns alone. Importantly, misclassifications occurred primarily between syntactically adjacent attack classes rather than between benign and malicious inputs, indicating semantic proximity rather than systemic detection failure.

The reinforcement learning component enhanced the framework by adaptively calibrating decision thresholds without altering the underlying classification models. The steady convergence of episode rewards, stabilization of loss values, and monotonic Q-value evolution demonstrated that the Deep Q-Network agent learned a consistent policy for decision refinement. These results confirmed that reinforcement learning functioned as a decision-level optimizer rather than a replacement for supervised learning, preserving model interpretability while improving operational robustness.

To operationalize the proposed framework, deployed as a web-based inference system capable of analyzing live URLs, raw payloads, and page source content. Fig. 15 illustrated the deployed interface, which executed the complete detection pipeline, including extraction, chunking, classification, aggregation, and export without manual intervention. The system produced page-level decisions using majority voting and average confidence stabilization, ensuring consistent outcomes across heterogeneous input sources.

Fig. 16 and Fig. 17 presented the confidence distribution across attack classes, demonstrating that the system generated decision-ready probabilistic outputs rather than binary labels. Chunk-level evidence further exposed the contribution of individual text segments to the final decision, supporting transparency and analyst interpretability. This evidence-driven decision formation directly addressed limitations of prior black-box detection systems by enabling traceable and explainable outputs suitable for real-world security operations.

Functional suitability was evaluated in accordance with ISO/IEC 25010. The system achieved full functional completeness by successfully executing all pipeline stages from input ingestion to result export. Correctness and appropriateness were evidenced through stable predictions, confidence margins, explainability indicators, and exportable artifacts, as summarized in Fig. 19. Collectively, these findings confirmed that it did not merely achieve high detection accuracy but also satisfied operational requirements for reliability, transparency, and decision readiness in deployment environments.

Hence, the results demonstrated that the framework provided an explainable and deployment-ready solution for unified multi-vector injection attack detection. By combining NLP-driven feature normalization, representation learning, and reinforcement-based decision calibration, the framework advanced beyond isolated single-attack detectors and addressed practical challenges encountered in real-world web security monitoring.

VI. Conclusion

The results confirmed that unified framework successfully achieved its primary objective of unified multi-vector injection attack detection. The framework accurately detected SQLi, XSS, LDAP Injection, and Command Injection within a single operational pipeline, addressing the limitations of single-attack detection systems. The NLP representation preserved both lexical structure and contextual semantics, enabling reliable discrimination across heterogeneous attack grammars. Group-aware evaluation demonstrated that the observed performance reflected genuine generalization rather than dataset leakage. The reinforcement learning component functioned effectively as a decision-level optimizer. It improved threshold calibration and operational stability without compromising model interpretability or altering classifier behavior. Deployment-level validation confirmed that the system executed all detection stages reliably and produced explainable outputs suitable for real-world security analysis. Based on ISO/IEC 25010 criteria, the framework met functional completeness, correctness, and appropriateness requirements. Overall, the study established that it is an adaptive, self-learning, explainable, and deployment-ready solution for multi-vector web injection attack detection, advancing beyond isolated detection approaches and aligning with operational cybersecurity needs.

VII. Recommendations

Future work should extend the framework to include additional injection vectors, such as XML Injection and Server-Side Template Injection, to further enhance coverage. Longitudinal evaluation using continuously evolving real-world traffic is recommended to assess robustness under adversarial drift. Integration of online learning mechanisms may further improve adaptability without retraining the core classifiers. Deployment in production-scale environments is recommended to evaluate system performance under high-throughput conditions. The evaluation focused on classifier performance and reinforcement learning calibration under leakage-controlled data partitions. Additional experiments, such as adversarial payload mutation testing, cross-dataset generalization, and ablation analysis of feature components, remain potential directions for future investigation.

Future research may explore tighter coupling between reinforcement learning policies and confidence calibration strategies to optimize detection thresholds dynamically across different operational contexts. These extensions would strengthen the applicability of the system, while preserving its core principles of explainability, reliability, and functional suitability.

References

[1] M. Bakator, D. Ćoćkalo, V. Makitan, S. Stanisavljev, and M. Nikolić, 'The three pillars of tomorrow: How Marketing 5.0 builds on Industry 5.0 and impacts Society 5.0?', Heliyon, vol. 10, no. 17, p. e36543, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e36543.

[2] P. Sun, Y. Wan, Z. Wu, and Z. Fang, 'A survey on security issues in IoT operating systems', J. Netw. Comput. Appl., vol. 231, p. 103976, 2024, doi: https://doi.org/10.1016/j.jnca.2024.103976.

[3] C. He and C. H. Q. Ding, 'SecRASP: Next generation web application security protection methodology and framework', Comput. Secur., vol. 154, p. 104445, 2025, doi: https://doi.org/10.1016/j.cose.2025.104445.

[4] R. Das, I. Y. Adam, and H. F. Oztop, 'Green IoT for energy efficiency: Enabling technologies, challenges, and future research directions', Therm. Sci. Eng. Prog., vol. 62, p. 103592, 2025, doi: https://doi.org/10.1016/j.tsep.2025.103592.

[5] G. Hernández Soto and X. Martinez-Cobas, 'The impact of transportation investment, road transportation and telecommunications on FDI in Latin America 2008-2021', Transp. Econ. Manag., vol. 2, pp. 45–57, 2024, doi: https://doi.org/10.1016/j.team.2024.01.002.

[6] J. L. Jensen, 'Revisiting capacitance-resistance model connectivity estimates for directional and distance effects', Geoenergy Sci. Eng., vol. 257, p. 214198, 2026, doi: https://doi.org/10.1016/j.geoen.2025.214198.

[7] R. Luh et al., 'Gamifying information security: Adversarial risk exploration for IT/OT infrastructures', Comput. Secur., vol. 151, p. 104287, 2025, doi: https://doi.org/10.1016/j.cose.2024.104287.

[8] T. Sasi, A. H. Lashkari, R. Lu, P. Xiong, and S. Iqbal, 'A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges', J. Inf. Intell., vol. 2, no. 6, pp. 455–513, 2024, doi: https://doi.org/10.1016/j.jiixd.2023.12.001.

[9] 'Top 10 Web Application Security Risks'. Accessed: May 21, 2025. [Online]. Available: https://owasp.org/www-project-top-ten/

[10] 'Types of XSS'. Accessed: May 21, 2025. [Online]. Available: https://owasp.org/www-community/Types_of_Cross-Site_Scripting

[11] Y. Organizational, 'Kent Academic Repository Dr Gloria Appiah - Kent Business School - University of Kent', 2020.

[12] A. M. Shuvo, G. S. Member, T. Zhang, and G. S. Member, 'A Comprehensive Survey on Non-Invasive Fault Injection Attacks', pp. 1–15.

[13] H. Kheddar, D. W. Dawoud, A. I. Awad, Y. Himeur, and M. K. Khan, 'Reinforcement-Learning-Based Intrusion Detection in Communication Networks: A Review', IEEE Commun. Surv. Tutorials, p. 1, 2024, doi: 10.1109/COMST.2024.3484491.

[14] S. Jamshidi, A. Amirnia, A. Nikanjam, K. W. Nafi, F. Khomh, and S. Keivanpour, 'Self-adaptive cyber defense for sustainable IoT: A DRL-based IDS optimizing security and energy efficiency', J. Netw. Comput. Appl., vol. 239, p. 104176, 2025, doi: https://doi.org/10.1016/j.jnca.2025.104176.

[15] S. Ali, J. Wang, and V. C. M. Leung, 'AI-driven fusion with cybersecurity: Exploring current trends, advanced techniques, future directions, and policy implications for evolving paradigms– A comprehensive review', Inf. Fusion, vol. 118, p. 102922, 2025, doi: https://doi.org/10.1016/j.inffus.2024.102922.

[16] S. Kapur and P. Saurabh, 'Penetration testing: Taxonomies, trade-offs, and adaptive strategies', Comput. Electr. Eng., vol. 128, p. 110686, 2025, doi: https://doi.org/10.1016/j.compeleceng.2025.110686.

[17] G. M. and S. C. Sethuraman, 'A comprehensive survey on deep learning based malware detection techniques', Comput. Sci. Rev., vol. 47, p. 100529, 2023, doi: https://doi.org/10.1016/j.cosrev.2022.100529.

[18] A. Mahboubi et al., 'The evolving threat landscape of botnets: Comprehensive analysis of detection techniques in the age of artificial intelligence', Internet of Things, vol. 33, p. 101728, 2025, doi: https://doi.org/10.1016/j.iot.2025.101728.

[19] Y. Zhang, X. Tang, and J. Yang, 'Synergies of Technological and Institutional Innovation Driving Manufacturing Transformation: Insights from Northeast China', J. Knowl. Econ., vol. 16, no. 1, pp. 1014–1048, 2025, doi: 10.1007/s13132-024-01982-1.

[20] R. Feniak and Y. Vyklyuk, 'AI-Enhanced Software Architecture Assessment for Cyber-Resilient Industrial Automation Systems *', vol. 0613, 2025.

[21] N. Mohamed, 'Artificial intelligence and machine learning in cybersecurity : a deep dive into state-of-the-art techniques and future', Knowl. Inf. Syst., vol. 67, no. 8, pp. 6969–7055, 2025, doi: 10.1007/s10115-025-02429-y.

[22] Y. Chen, H. Wang, K. Yu, and R. Zhou, 'Artificial Intelligence Methods in Natural Language Processing : A Comprehensive Review', vol. 85, pp. 545–550, 2024.

[23] T. Georgescu, 'SS symmetry Natural Language Processing Model for Automatic Analysis of Cybersecurity-Related Documents', 2020.

[24] T. T. Nguyen and V. J. Reddi, 'Deep Reinforcement Learning for Cyber Security', no. Ml, 2021, doi: 10.1109/TNNLS.2021.3121870.

[25] M. Sewak, S. K. Sahay, and H. Rathore, 'Deep Reinforcement Learning for Cybersecurity Threat Detection and Protection: A Review', in Secure Knowledge Management In The Artificial Intelligence Era, R. Krishnan, H. R. Rao, S. K. Sahay, S. Samtani, and Z. Zhao, Eds., Cham: Springer International Publishing, 2022, pp. 51–72.

[26] K. Kundiya and Y. Haribhakta, 'A systematic review on insider threat detection using natural language processing', Int. J. Inf. Secur., vol. 24, no. 6, p. 227, 2025, doi: 10.1007/s10207-025-01145-6.

[27] M. M. Hossain, M. S. Hossain, M. Safran, S. Alfarhood, M. Alfarhood, and M. F. Mridha, 'A Hybrid Attention-Based Transformer Model for Arabic News Classification Using Text Embedding and Deep Learning', IEEE Access, vol. 12, pp. 198046–198066, 2024, doi: 10.1109/ACCESS.2024.3522061.

[28] C. J. P. Abuda, 'Hybrid Detection Framework Using Natural Language Processing (NLP) and Reinforcement Learning (RL) for Cross-Site Scripting (XSS) Attacks', vol. 16, no. 6, pp. 736–747, 2025.

[29] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, 'Explainable Artificial Intelligence in CyberSecurity: A Survey', IEEE Access, vol. 10, pp. 93575–93600, 2022, doi: 10.1109/ACCESS.2022.3204171.

[30] A. Bhattacharyya, S. M. Nambiar, R. Ojha, A. Gyaneshwar, U. Chadha, and K. Srinivasan, 'Machine Learning and Deep Learning powered satellite communications: Enabling technologies, applications, open challenges, and future research directions', Int. J. Satell. Commun. Netw., vol. 41, no. 6, pp. 539–588, 2023, doi: https://doi.org/10.1002/sat.1482.

[31] V. Chithanuru and M. Ramaiah, 'Proactive detection of anomalous behavior in Ethereum accounts using XAI-enabled ensemble stacking with Bayesian optimization', PeerJ Comput. Sci., vol. 11, 2025, doi: 10.7717/PEERJ-CS.2630.

[32] V. B. Verdun and C. E. Dumdumaya, 'Computational Approach to Engagement Analysis in Learning Management Systems: Optimizing User Interactions with XGBoost Algorithm', 2025 IEEE Int. Conf. Emerg. Trends Eng. Comput., pp. 1–8, 2025, [Online]. Available: https://api.semanticscholar.org/CorpusID:284722186

[33] M. J. M. Timogan and A. R. L. Reyes, 'Comparison of Cloud-Based Free-Tier Virtual Environments : A Performance Analysis on Machine Learning Training', pp. 26–37.

[34] Z. Lu, 'Sql injection detection using Naïve Bayes classifier: A probabilistic approach for web application security', ITM Web Conf., vol. 04016, 2025, [Online]. Available: https://api.semanticscholar.org/CorpusID:275867312

[35] M. Aljabri et al., 'Detecting Malicious URLs Using Machine Learning Techniques : Review and Research Directions', vol. 10, no. October, pp. 121395–121417, 2022.

[36] S. V. J. Rani, 'A Novel Deep Hierarchical Machine Learning Approach for Identification of Known and Unknown Multiple Security Attacks in a D2D Communications Network', IEEE Access, vol. 11, pp. 95161–95194, 2023, doi: 10.1109/ACCESS.2023.3308036.

[37] Y. Chen, G. Liang, and Q. Wang, 'Research on SQL Injection Detection Technology Based on Content Matching and Deep Learning', Comput. Mater. Contin., vol. 84, pp. 1145–1167, Jun. 2025, doi: 10.32604/cmc.2025.063319.

[38] A. Rahali, 'End-to-End Transformer-Based Models in Textual-Based NLP', pp. 54–110, 2023.

[39] R. Patil, S. Boit, V. Gudivada, and J. Nandigam, 'A Survey of Text Representation and Embedding Techniques in NLP', IEEE Access, vol. PP, p. 1, Jan. 2023, doi: 10.1109/ACCESS.2023.3266377.

[40] R. Kaur, T. Klobučar, and D. Gabrijelčič, 'Harnessing the power of language models in cybersecurity: A comprehensive review', Int. J. Inf. Manag. Data Insights, vol. 5, no. 1, p. 100315, 2025, doi: https://doi.org/10.1016/j.jjimei.2024.100315.

[41] Y. Li, D. Wu, Z. You, G. Chen, and D. Wu, 'Deep reinforcement learning for collision avoidance in unmanned surface vehicles: State-of-the-art', Appl. Ocean Res., vol. 164, p. 104778, 2025, doi: https://doi.org/10.1016/j.apor.2025.104778.

[42] M. Tokic, Adaptive ε-Greedy Exploration in Reinforcement Learning Based on Value Differences. 2010. doi: 10.1007/978-3-642-16111-7_23.

[43] R. Pal and B. Bhowmick, 'Mapping the Evolution and Future Directions of Family Business Research: A Bibliometric Analysis', vol. 15, no. 5, pp. 21–21, 2025, doi: 10.62422/978-81-981590-7-6-009.

[44] M. C. Saputra, S. A. Wicaksono, S. H. Wijoyo, P. N. Rahmandita, and B. T. Hanggara, 'Quality Analysis of an Interactive Programming Learning Platform Based on ISO/IEC 25010 Using a String-Matching Approach on User Reviews', J-Intech, vol. 13, no. 01, pp. 192–202, 2025, doi: 10.32664/j-intech.v13i01.2003.

[45] D. Gobov and O. Zuieva, 'Software Quality Attributes in Requirements Engineering', Int. J. Inf. Technol. Comput. Sci., vol. 17, no. 4, pp. 38–48, 2025, doi: 10.5815/ijitcs.2025.04.04.

[46] E. P. Reina-Guaña and C. G. Hernández-Cenzano, 'Evaluating Software Quality: Application of ISO/IEC 25000 in Public and Private Sectors', in 2025 IEEE Technology and Engineering Management Society (TEMSCON LATAM), 2025, pp. 1–6. doi: 10.1109/TEMSCONLATAM65810.2025.11238551.

[47] J. Rayo, R. de la Rosa, and M. Garrido, 'A Hybrid Approach to Information Retrieval and Answer Generation for Regulatory Texts', pp. 31–35, 2025, [Online]. Available: http://arxiv.org/abs/2502.16767

[48] Z. Qiu, G. Huang, X. Qin, Y. Wang, J. Wang, and Y. Zhou, 'A Hybrid Semantic Representation Method Based on Fusion Conceptual Knowledge and Weighted Word Embeddings for English Texts', 2024. doi: 10.3390/info15110708.

[49] L. Anand, 'Transformer-Based Models for Natural Language Processing Tasks', 2022. doi: 10.13140/RG.2.2.26476.42889.

[50] I. Duru and A. S. Sunar, 'Transformer and Pre-Transformer Model-Based Sentiment Prediction with Various Embeddings: A Case Study on Amazon Reviews.', Entropy (Basel)., vol. 27, no. 12, Nov. 2025, doi: 10.3390/e27121202.

[51] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, 'Pre-trained models for natural language processing: A survey', Sci. China Technol. Sci., vol. 63, pp. 1872–1897, 2020, doi: 10.1007/s11431-020-1647-3.

[52] D. Karabašević, A. Vujko, V. Mirčetić, G. Popović, and D. Stanujkić, 'A Transformer-Based Semantic Encoding Framework for Quantitative Analysis of Large-Scale Textual Reviews', 2026. doi: 10.3390/axioms15030175.

[53] R. Bakır, 'UniEmbed: A Novel Approach to Detect XSS and SQL Injection Attacks Leveraging Multiple Feature Fusion with Machine Learning Techniques', Arab. J. Sci. Eng., vol. 50, no. 19, pp. 15591–15604, 2025, doi: 10.1007/s13369-024-09916-4.

[54] C. J. P. Abuda and A. R. L. Reyes, 'Development of Cloud-Based Structured Query Language Injection (SQLi) Detection using Deep Learning and FastAPI', in 2025 Eight International Conference on Vocational Education and Electrical Engineering (ICVEE), 2025, pp. 81–87. doi: 10.1109/ICVEE66651.2025.11281431.

[55] C. J. P. Abuda and C. E. Dumdumaya, 'Hybrid Structure Query Language Injection (SQLi) Detection Using Deep Q-Networks : A Reinforcement Machine Learning Model', vol. 16, no. 5, pp. 217–227, 2025.

[56] S. Ren, J. Jin, G. Niu, and Y. Liu, 'ARCS: Adaptive Reinforcement Learning Framework for Automated Cybersecurity Incident Response Strategy Optimization', Appl. Sci., vol. 15, no. 2, 2025, doi: 10.3390/app15020951.

[57] B. William, 'Reinforcement Learning in Predictive Decision-Making Systems', 2025.

[58] M. Noaeen et al., 'Reinforcement learning in urban network traffic signal control : A systematic literature review', Expert Syst. Appl., vol. 199, no. March 2021, p. 116830, 2022, doi: 10.1016/j.eswa.2022.116830.

[59] F. Mesadieu, D. Torre, and A. Chennameneni, 'Leveraging Deep Reinforcement Learning Technique for Intrusion Detection in SCADA Infrastructure', IEEE Access, vol. 12, no. May, pp. 63381–63399, 2024, doi: 10.1109/ACCESS.2024.3390722.

[60] X. Wang, J. Zhai, and H. Yang, 'Detecting command injection attacks in web applications based on novel deep learning methods', Sci. Rep., vol. 14, Oct. 2024, doi: 10.1038/s41598-024-74350-3.

[61] F. K. Alarfaj, 'Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks', Appl. Sci., vol. 13, no. 7, 2023, doi: 10.3390/app13074365.

[62] S. Abaimov and G. Bianchi, 'A survey on the application of deep learning for code injection detection', Array, vol. 11, p. 100077, 2021, doi: https://doi.org/10.1016/j.array.2021.100077.

[63] A. Shaikhanova, O. Kuznetsov, A. Tokkuliyeva, K. Ayapbergenov, S. Olzhas, and T. Danir, 'Security Audit of IoT Device Networks: A Reproducible Machine Learning Framework for Threat Detection and Performance Benchmarking.', Sensors (Basel)., vol. 25, no. 24, Dec. 2025, doi: 10.3390/s25247519.

[64] C. J. P. Abuda and R. S. Villafuerte, 'Development of an Algorithm-Based Analysis-Compression Integrated Communication Tracking Management Information System (iCTMIS)', 2024 IEEE Open Conf. Electr. Electron. Inf. Sci. eStream 2024 - Proc., vol. 16, no. 3, pp. 1–5, 2024, doi: 10.1109/eStream61684.2024.10542580.

[65] I. Dataport, 'No Title'. Accessed: Feb. 15, 2026. [Online]. Available: https://ieee-dataport.org/documents/sql-injection-detection-dataset

[66] Kaggle, 'No Title'. Accessed: Feb. 12, 2026. [Online]. Available: https://www.kaggle.com/code/alextrinity/sqli-xss-detection