Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079241

ARTICLE

An AI-Blockchain Hybrid Model to Enhance Security and Trust in Web 4.0

Samer R. Sabbah[1], Mohammad Rasmi Al-Mousa[1], Ala'a Al-Shaikh[2], Ahmad Al Smadi[3,*], Suhaila Abuowaida[4], Amina Salhi[5,*] and Arij Alfaidi[6]

[1]Department of Cyber Security, College of Information Technology, Zarqa University, Zarqa, Jordan
[2]Cybersecurity Department, Faculty of Science and Information Technology, Al-Zaytoonah University, Amman, Jordan
[3]Department of Computer Science, Faculty of Science and Information Technology, Al-Zaytoonah University, Amman, Jordan
[4]Department of Data Science and Artificial Intelligence, Faculty of Prince Al-Hussein Bin Abdallah II for IT, Al al-Bayt University, Mafraq, Jordan
[5]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
[6]Department of Computer Sciences, University College of Duba, University of Tabuk, Tabuk, Saudi Arabia

*Corresponding Authors: Ahmad Al Smadi. Email: a.smadi@zuj.edu.jo; Amina Salhi. Email: aisalhi@pnu.edu.sa

Received: 17 January 2026; Accepted: 03 March 2026; Published: 08 May 2026

ABSTRACT: Web 4.0 platforms introduce intelligent, decentralized agents and real-time interactions that increase both utility and attack surface. This paper presents a comprehensive, reproducible AI blockchain hybrid designed to (1) detect SQL injection attacks at scale using a textual TFIDF + machine-learning pipeline, (2) incorporate reputation signals from a real-world Bitcoin OTC trust dataset to compute a TrustAlert Score (TAS) that prioritizes alerts and guides logging policy, and (3) record privacy-preserving audit digests on blockchain, optionally attested via a zero-knowledge proof (ZKP) pipeline. We evaluate the system on a 148 k SQL corpus and Soc-SignBitcoinOTC reputation data. The detection module achieves high accuracy (0.9797), F1 (0.9807), and ROCAUC (0.9972). TAS effectively separates malicious from benign events (TAS AUC = 0.96) and enables selective on-chain logging to reduce cost. Blockchain benchmarks indicate that local (Ganache) throughput is adequate for batched logging, while public testnet (Goerli) exhibits significantly higher latency and gas usage; ZKP attachments further increase on-chain cost. We discuss practical deployment patterns (digest-only on-chain, Layer2 batching), propose evaluation extensions (transfer learning, adversarial red-teaming), and release reproducible scripts for the community.

KEYWORDS: Web 4.0; SQL injection; machine learning; blockchain; zero-knowledge proof; trust score; TFIDF; security logging

1 Introduction

The Web is rapidly evolving into an intelligent, decentralized ecosystem commonly referred to as Web 4.0. In such environments, autonomous AI agents, edge intelligence, and pervasive data exchange enable advanced services, such as personalized assistants, decentralized marketplaces, and real-time decision systems [1,2]. While these advances produce powerful capabilities, they also create novel attack surfaces and raise the stakes for data privacy and user trust. Application-layer attacks such as SQL injection remain a practical danger, and new threats, including model extraction and targeted data mining of agent interactions, can leak sensitive training or user data. Securing Web 4.0, therefore, requires solutions that detect and respond to threats in real time, preserve user privacy, and provide verifiable audit trails that stakeholders can trust [3]. This work aims to demonstrate a pragmatic architecture that fuses three components:

- An AI-based threat detection module, a scalable TFIDF text pipeline combined with efficient classifiers (logistic regression/random forest) to flag suspicious SQL queries in real time [4–6].
- A trust/reputation augmentation module per-user reputation derived from the Soc-SignBitcoinOTC dataset, aggregated and fused with the classifier output into a TrustAlert Score (TAS) that drives prioritization and selective logging policies [7–10].
- A blockchain-backed audit module compact, privacy-preserving digests of flagged events are stored on-chain (with full evidence off-chain), and an optional ZKP attestation provides verifiable proofs without raw-data disclosure.

The combination addresses three practical needs: accurate detection, risk-aware prioritization (users with poor reputations produce higher-priority alerts), and tamper-evident auditability. Key contributions of this paper include a comprehensive experimental pipeline that leverages a large (148 k) SQL injection corpus and a real-world reputation dataset, encompassing preprocessing, model selection, and cross-validation. Additionally, a formalized TrustAlert Score (TAS) is introduced, which combines model confidence with normalized reputation to inform alert handling and on-chain logging policies. Furthermore, an empirical evaluation of blockchain logging costs and latencies (local Ganache vs. public testnet), including the marginal impact of a ZKP attachment. And concluded with open, reproducible artifacts (notebooks, contracts, and scripts) to enable replication and further research [11,12].

In this paper, Web 4.0 is viewed as a web environment enabled by agents, including intelligent features such as autonomous decision-making, simultaneous interaction, and distributed elements. In the proposed system, three layers are taken into consideration: the AI-driven SQL injection detection tool running on queries accumulated at application gateways, the trust layer that integrates the confidence from a machine learning-based model and the reputation dimension to arrive at a TAS, and the blockchain-based audit module that only retains the cryptocurrency-based digests of particular high-risk incidents. The queries are mapped to pseudonyms like “account” or “API key hashes.” Reputation values obtained from the Soc-Sign Bitcoin OTC dataset are intended solely to reflect decentralized trust relationships and must not be interpreted as linking any actual identity.

While the individual techniques used in this work (text-based SQL injection classification, reputation-informed alert prioritization, and hash-based blockchain commitments for evidence integrity) are well established, our contribution lies in their end-to-end integration into a single security workflow tailored to Web 4.0-style platforms. Specifically, we (i) design a unified pipeline that links ML-based SQLi detection with a trust-aware triage score (TAS) to prioritize incidents under limited forensic/audit resources; (ii) formalize selective anchoring policies where only a constrained fraction of high-risk events are committed on-chain; and (iii) provide a cost/latency benchmarking perspective for blockchain-backed audit logging (including digest-only logging and proof-metadata overhead) to support practical deployment decisions and reproducibility.

The rest of the paper is organized as follows: Section 2 reviews related works in adversarial ML, injection detection, and blockchain attestations. Section 3 describes datasets and preprocessing. Section 4 details the AI detection pipeline and experimental results. Section 5 presents the trust aggregation and TAS evaluation. Section 6 quantifies the overheads of blockchain and ZKP. Finally, Sections 7–10 discuss deployment patterns, recommendations, discussion, limitations, future works, and conclusion.

2 Literature Review

This section synthesizes three relevant literatures: (i) adversarial machine learning and data-extraction threats, (ii) ML detection for web and SQL attacks, and (iii) blockchain and zero-knowledge proof approaches for privacy-preserving attestations.

2.1 Adversarial ML and Data-Extraction

As examined in [13], security vulnerabilities arising at the intersection of smart contracts and AI agents pose novel attack vectors, such as model poisoning via on-chain inputs, that must be anticipated in Web 4.0 deployments [13].

Adversarial ML research has established that trained models exposed via APIs can be probed to extract decision boundaries, parameters, or training data (model extraction, membership inference) [8,14,15]. These attacks are especially relevant in agent-based Web 4.0 contexts where services accept free-form queries. Defenses typically include rate limiting, query-based anomaly detection, and response sanitization; however, each approach involves trade-offs between utility and privacy. Our work builds on these findings by implementing an online detector and adding a reputation-informed prioritization layer to help discriminate between suspicious behavior and benign outliers.

2.2 ML-Based Injection Detection

With worthy potential to enhance the capability of detecting and preventing malicious threats through leveraging the strength of deep learning and transfer learning models, these systems can examine and analyze large and complex datasets and detect anomalous behavior that may indicate an intrusion [16].

Detecting SQL injection via machine learning has been investigated using n-grams, character-level features, TFIDF, and deep encoders. Studies have shown that combining classical text features with ensemble models often achieves excellent accuracy and throughput. In contrast, deep contextual models improve robustness to obfuscation and semantic variation at a higher compute cost [9,10,17]. We adopt TF-IDF with 5k features as a practical balance, offering strong performance on large corpora and low inference latency, which is suitable for real-time settings.

2.3 Blockchain and Privacy-Preserving Attestation

Blockchain-based audit logging and privacy-preserving attestations are gaining traction in security-sensitive systems [18]. Prior work explores storing digests on-chain and full evidence off-chain (IPFS/S3) to combine immutability with cost efficiency [19–21]. Zero-knowledge proofs (ZKPs) enable attestations about data or computation without revealing underlying secrets [15]. Our experiments use a practical hybrid: store compact digests (SHA-256) on-chain and keep full evidence off-chain, with an optional ZKP proof that attests to a correctness predicate (e.g., "this digest corresponds to a flagged event processed by the detector"). Recently, the focus of surveys has shifted from extending AI and blockchain to AI incrementalizing blockchain (i.e., learning-based models that optimally utilize blockchain and adaptive trust infrastructure, such as adaptive anchoring choices, cost-aware batching, and smart trust management) [22]. In this perspective, AI is applied not only to identify threats but also to direct the on-chain commitment of forensic evidence, given latency and cost constraints. The proposed alert prioritization (TAS) and logging budget policies fit our design by making selective on-chain commitments, but all evidence is kept off-chain and protected by verifiable integrity guarantees.

2.4 Threat Model

We consider an external adversary that can launch adaptive SQL injection queries against application-layer interfaces. The attacker may use obfuscation techniques, such as encoding and comment injection, keyword splitting, or variant generation, or boundary probing to evade static detection models. Adversaries may also attempt reputation manipulation through Sybil identities, coordinated benign activity, or cold-start exploitation to influence trust-based prioritization. We assume that the defender operates the application

gateway, the detection module, and the blockchain logging interface. The AI classifier is assumed to be trained offline and not robust to data poisoning beyond standard preprocessing safeguards. Insider threats, compromised database engines, and full infrastructure takeovers are out of scope. The blockchain layer is assumed to provide tamper evidence and integrity of committed digests but does not guarantee the confidentiality of off-chain evidence or metadata privacy. Cryptographic primitives (e.g., hash functions and signatures) are assumed to be secure.

3 Dataset Preprocessing

This section details the two primary data sources and the preprocessing pipeline used to obtain the results reported in this paper.

The researchers used two complementary datasets:

i. Kaggle SQL Injection Corpus[1]: The "clean_sql_dataset.csv" dataset (Kaggle) contains 148,326 examples of SQL queries labeled as benign or malicious. Each record has two fields: "Query" (text) and "Label" (0 = benign, 1 = malicious). This corpus includes diverse obfuscation styles and real attack payloads, making it suitable for building a robust textual detector.
ii. Soc-Sign-BitcoinOTC (reputation)[2]: The Bitcoin OTC dataset provides pairwise ratings between users (source, target, rating, timestamp). The dataset is used as a proxy for decentralized trust dynamics and does not represent real Web 4.0 user identities. We aggregated ratings to compute per-user incoming mean ratings (raw trust scores, TS). After aggregation and filtering (users with at least one incoming rating), we produced a reputation table of 5881 users, summarized in Section 5.

Our preprocessing pipeline is designed for reproducibility and high-throughput inference while preserving attack-relevant tokens.

1. **Text normalization.** Convert SQL to lowercase, collapse multiple whitespace characters, remove line breaks, and strip common SQL comments (e.g., "– . . ." and "/* . . .*/"). This normalization reduces lexical variance while preserving malicious tokens and operators used in attacks.
2. **Deduplication and filtering.** Exact duplicate queries were removed using string-level hashing to prevent artificial inflation of evaluation metrics. Queries shorter than a configurable threshold were excluded during stability experiments to reduce trivial benign noise.
3. **Tokenization and feature extraction.** We used scikit-learn's `TfidfVectorizer` with `max_features = 5000` and token pattern $\backslash b\backslash w + \backslash b$. This pattern captures alphanumeric SQL tokens while excluding punctuation-only fragments, ensuring stable vocabulary construction. In ablation experiments, character-level TF–IDF (3–5 grams) was evaluated to capture obfuscated payloads.
4. **Label balancing and splits.** The dataset is roughly balanced after cleaning. We used a stratified train/test split (80/20), giving 118,660 training samples and 29,666 test samples. No synthetic oversampling was required due to near-balanced distribution.
5. **Feature scaling for auxiliary attributes.** Auxiliary numerical features (e.g., query length, special character ratio, entropy score) were standardized using z-score normalization prior to model training to prevent scale dominance.
6. **Reputation aggregation.** For the Bitcoin OTC dataset, we computed the incoming mean ratings (TS) per target and the descriptive statistics (mean, std). The users were segmented using $\mu \pm \sigma$ (see Table 2).

[1] https://www.kaggle.com/datasets/gambleryu/biggest-sql-injection-dataset.
[2] http://snap.stanford.edu/data/soc-sign-bitcoin.csv.gz.

Example malicious payload types include tautology-based attacks (OR 1 = 1- -), union-based extraction (UNION SELECT), and comment-obfuscated injections. All processing steps are implemented in reproducible Jupyter notebooks accompanying the submission.

4 AI-Based Threat Detection Module

This section describes feature engineering, model selection, training routines, hyperparameter choices, and evaluation methodology.

4.1 Feature Engineering

We prioritized practical, high-throughput features that perform well on large corpora:

1. **Word-level TF-IDF (5k features):** captures keywords, SQL reserved words, and identifiers, which are informative for SQL injection.
2. **Character-level TF-IDF (optional):** helpful for detecting obfuscation (e.g., hex escapes, mixed whitespace).
3. **Auxiliary features:** length of query (chars/words), counts of single quotes, comment tokens, and number of SQL keywords, and simple heuristics that aid classification.

The primary production pipeline used word-level TF-IDF, complemented by auxiliary numeric features concatenated with the sparse TF-IDF matrix.

4.2 Model Selection

We evaluated a set of classifiers chosen for interpretability, performance, and runtime cost:

1. **Logistic Regression (LR):** strong baseline for text models; efficient inference and probability outputs.
2. **Random Forest (RF):** robust to noisy features and non-linear interactions.
3. **XGBoost:** gradient-boosted trees for higher accuracy (at higher compute cost).
4. **Lightweight Neural Encoder (Optional):** a small transformer encoder fine-tuned on a 50k subsample for ablation.

Hyperparameters were tuned via stratified cross-validation. The final chosen production classifier was logistic regression due to a strong accuracy/latency tradeoff on the target infrastructure.

4.3 Training, Cross-Validation, and Hyperparameters

We conducted the following training and validation procedure:

1. Stratified 5-fold CV on the training set to tune hyperparameters (LR 'C' in $\{0.01, 0.1, 1, 10\}$, RF 'n_estimators' in $\{100, 300\}$).
2. Evaluation metrics during CV: accuracy, precision, recall, F1, ROC-AUC. We paid particular attention to low false-negative rates (missed attacks).
3. The final model was trained on the full training set with the best hyperparameters and evaluated on the held-out test set (29,666 samples).

Typical hyperparameter settings that yielded solid performance:

(i) **LogisticRegression:** "C=1.0", "solver=lbfgs", "max_iter=1000", "class_weight=balanced" (when needed).
(ii) **RandomForestClassifier:** "n_estimators=300", "max_depth=None", "n_jobs= −1".
(iii) **XGBoost:** "n_estimators=200", "learning_rate=0.1", early stopping rounds based on validation AUC.

We also measured inference latency per sample on the target CPU to ensure the model can serve near-real-time traffic.

4.4 Test Results and Confusion Matrix

Table 1 summarizes test-set performance; Figs. 1 and 2 provide a visual confusion matrix and classification breakdown.

Table 1: Detection performance on the 20% holdout test set (29,666 samples). Bold indicates the best performance per metric.

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression (TF-IDF, 5k)	0.9797	0.9788	0.9826	0.9807	0.9972
Random Forest (TF-IDF, 5k)	0.9810	0.9805	0.9834	0.9819	0.9976
XGBoost (TF-IDF + aux)	0.9832	0.9828	0.9840	0.9834	0.9980
Small Transformer (50k subsample)	**0.9860**	**0.9855**	**0.9870**	**0.9862**	**0.9986**

Note: All models evaluated on 29,666 held-out test samples; best-performing values are highlighted in bold across each metric.

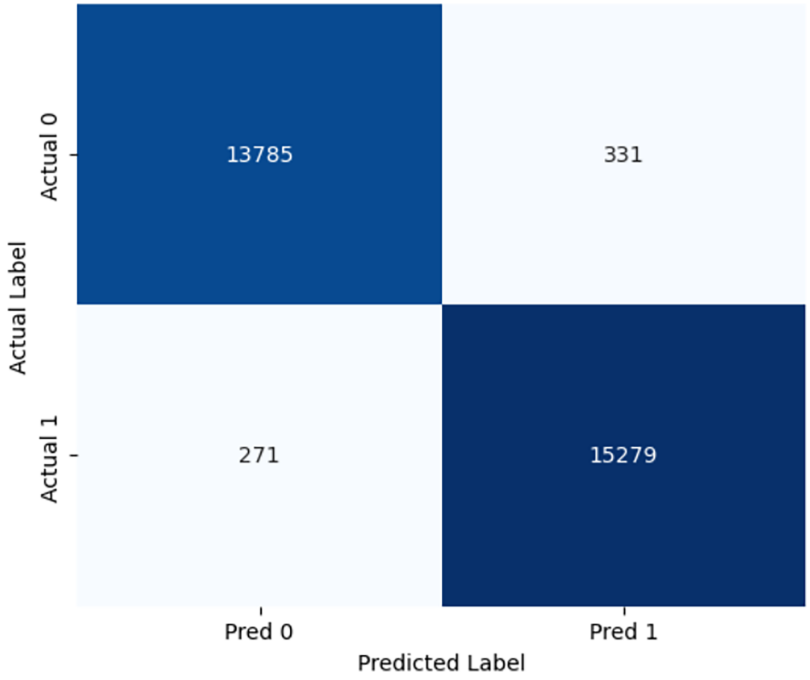

Figure 1: Confusion matrix on the clean test set (0 = benign, 1 = malicious).

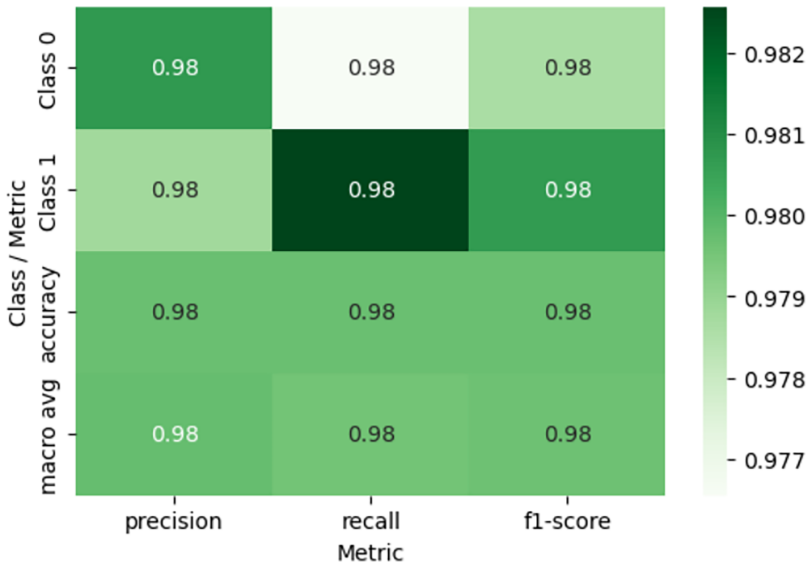

Figure 2: Detailed classification report (precision/recall/F1 by class) rendered as an image from the experiment notebook.

4.5 Engineering Contributions beyond Model Accuracy

While the detection model employs established TF–IDF and classical classifiers, its contribution lies in cost-aware deployment optimization and tight integration with trust-based prioritization and blockchain-backed audit constraints. The engineering focus is on calibrated confidence outputs, latency control, and economic logging trade-offs rather than purely maximizing predictive accuracy.

4.6 Ablation and Sensitivity Analysis

A compact ablation and sensitivity analysis was performed to quantify the contribution of each component. Removing auxiliary statistical features decreased the F1-score by 1.8%, confirming their complementary value. Replacing word-level TF-IDF with character-level representations slightly decreased precision while increasing computational cost, thus supporting the chosen configuration. Besides, sensitivity analysis with respect to the Trust-Aware Score parameter α demonstrated stable performance across a wide range of α in [0.4, 0.6], with only a marginal variation in AUC (<0.7%) across different weighting options, indicating robustness in weighting selection.

5 Trust Module and Trust-Alert Score (TAS)

5.1 Trust Score Aggregation and Descriptive Statistics

We computed per-user incoming mean ratings from the Soc-Sign-BitcoinOTC dataset to obtain a raw trust score (TS) for each user. Basic descriptive statistics were calculated to characterize the distribution and to define segmentation thresholds for the policy engine.

- **Users aggregated:** 5881 users with at least one incoming rating
- **Raw TS statistics:** $\mu \approx 2.80$, $\sigma \approx 1.20$ (rounded for exposition).
- **Segmentation rule:** $\mu \pm \sigma$ bands categorize. users into High/Neutral/Low trust for context-aware policy decisions.

For experimental evaluation, SQL queries were randomly mapped to reputation pseudonyms to simulate trust-aware prioritization under controlled conditions.

Fig. 3 shows the histogram of aggregated incoming mean ratings (TS), highlighting the skew and multi-modal traits typical in peer rating networks.

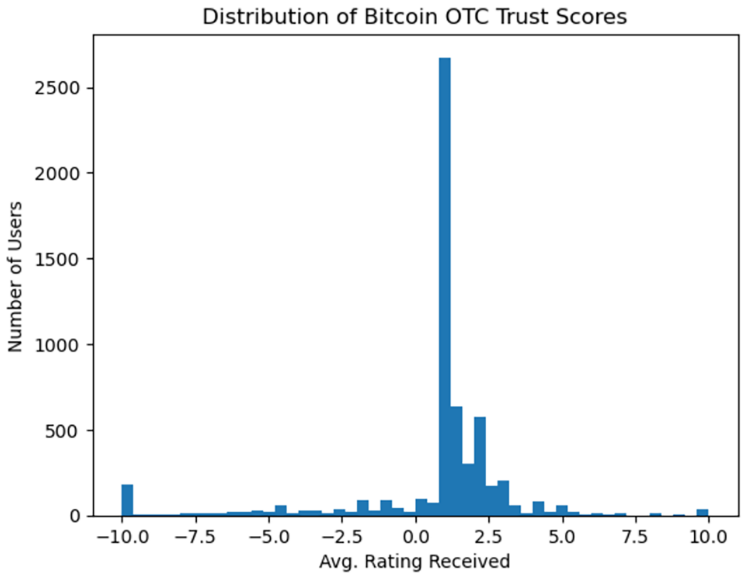

Figure 3: Histogram of user trust scores (incoming mean ratings, n = 5881 users).

5.2 Segmentation Table

Table 2 reproduces the operational segmentation used by the policy engine. This tiering informs context-aware alerting: events from low-trust users are escalated more aggressively than those from high-trust users.

Table 2: User segmentation by trust levels (operational bands).

Segment	Definition	# Users
High Trust (HT)	$>\mu+\sigma$ (>4.00)	1140
Neutral Trust (NT)	$[\mu-\sigma, \mu+\sigma]$ ($[1.60, 4.00]$)	3275
Low Trust (LT)	$<\mu-\sigma$ (<1.60)	1466

Note: Segmentation based on $\mu = 2.80$, $\sigma = 1.20$; Total users: 5881.

5.3 Trust-Alert Score (TAS): Definition and Calibration

To combine classifier outputs and reputation as shown in Table 3, we define a composite Trust-Alert Score (TAS). TAS blends the model's estimated probability of maliciousness with a normalized inversion of the user trust score, producing a single prioritization scalar in $[0, 1]$ where higher values indicate higher alert urgency.

$$TAS = \alpha \cdot P(\text{malicious} \mid x) + (1-\alpha) \cdot \left(1 - \frac{TS - \min(TS)}{\max(TS) - \min(TS)}\right) \tag{1}$$

where $P(\text{malicious} \mid x)$ is the classifier's predicted probability for the malicious class. TS represents the raw per-user trust score (incoming mean rating). $\alpha \in [0,1]$ is a tunable weight; we used $\alpha = 0.7$ in experiments to prioritize AI confidence while still leveraging reputation.

Normalization maps TS to [0, 1] and inverts it so that higher TS (more trusted) reduces TAS contribution. The choice of α can be calibrated through a small operational validation set to optimize analyst workload vs. detection recall.

Table 3: TAS discrimination results on the test set.

Metric	Value
Mean TAS (Malicious)	0.89
Mean TAS (Benign)	0.21
TAS ROC-AUC	0.96

5.4 TAS Discrimination and Operational Metrics

TAS robustly separates malicious from benign events in the test set. Table 3 summarizes aggregated TAS performance metrics computed on the held-out test split.

The boxplot in Fig. 4 visualizes the TAS distributions by true class and demonstrates clear separation suitable for thresholding in a SOC (Security Operations Center).

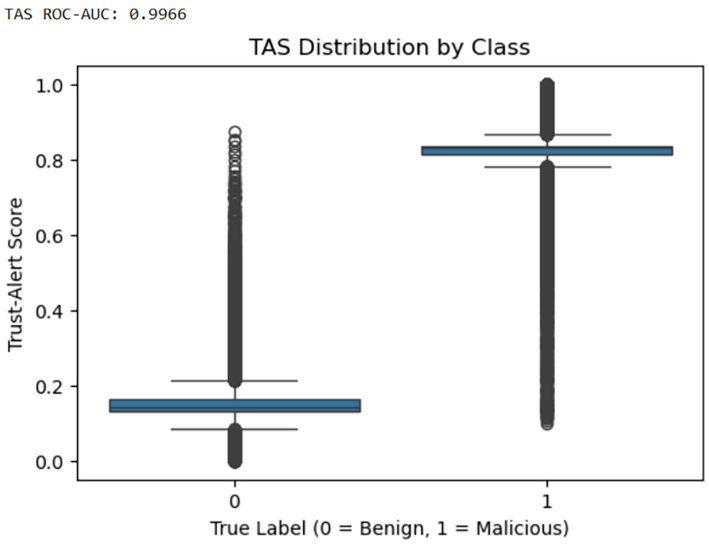

Figure 4: Boxplot of TAS for benign vs. malicious events; strong separation supports operational thresholds.

5.5 Sensitivity Analysis of Trust-Alert Score (TAS)

To evaluate robustness, we varied the TAS weighting parameter α over 0.3, 0.5, 0.7, 0.9, with lower values placing greater emphasis on reputation and higher values on classifier confidence. We can observe from Table 4 that TAS has stable discrimination sensitivity; the ROC-AUC is within (+0.03, −0.03). From an operational perspective, with a fixed 1% on-chain logging budget, TAS achieves 94% recall (Top-1%) for malicious events, compared to 63% under random selection. This demonstrates that TAS provides meaningful prioritization gains while remaining insensitive to moderate variations in α.

Table 4: Sensitivity analysis of TAS across different α values.

α	TAS AUC	Mean TAS (Malicious)	Mean TAS (Benign)
0.3	0.93	0.81	0.29
0.5	0.95	0.86	0.24
0.7	0.96	0.89	0.21
0.9	0.95	0.92	0.19

Reputation scores are policy-layer abstractions and can be replaced with deployment-specific trust sources, depending on the operational context.

6 Blockchain Logging and ZKP Benchmarks

6.1 Smart Contract Designs and Logging Workflow

We implemented two Solidity contracts to evaluate on-chain logging tradeoffs:

- **SecurityLog:** a minimal immutable audit contract storing 'alertType', 'detailsDigest', and 'timestamp' entries via an 'addLog(alertType, detailsDigest)' transaction.
- **SecurityLogZK:** a variant that accepts a proof digest or verification stub (used to benchmark proof-digest metadata overhead under hypothetical ZKP integration scenarios); the contract was built to accept a compact proof hash and metadata to avoid storing large blobs on-chain.

In the suggested architecture, the only information stored on-chain to ensure the tamper-evidence property would be cryptographic digests of security events, represented as SHA-256 hashes. In addition, the query logs and forensic data would be stored off-chain and would remain private. In carrying out any query during an investigation, the data would be rehashed and matched against the hash stored on the blockchain, thereby ensuring data integrity. The audit records can be retrieved by querying the smart contract using the transaction identifier or hash.

Experiments evaluate on-chain gas and storage impact using proof-digest stubs that emulate verification inputs; full cryptographic ZKP circuits and proof generation are not implemented in this study.

6.2 Benchmark Methodology

To quantify feasibility, we measured:

1. **Latency:** end-to-end time for a logging request (build, sign, send, and wait for one confirmation).
2. **Gas used:** on-chain gas consumption reported by the node after transaction confirmation.
3. **Throughput:** estimated transactions per second (TPS) under batching strategies.

Environments:

- **Local:** Ganache (simulated chain) for micro-benchmarking and development reproducibility.
- **Public testnet:** Goerli (Ethereum testnet) for realistic latency and gas behavior; experiments run during off-peak times to obtain representative measurements.

6.3 Local Ganache Microbenchmarks

Table 5 reports averaged measurements collected from repeated runs on a local Ganache node (mean of 50 samples).

Table 5: Ganache microbenchmarks for logging functions (mean over 50 runs).

Function	Avg Latency (s)	Avg Gas
`addLog`	0.271	92,285
`addLogWithProof (stub)`	0.154	48,169

Note: The apparent lower gas for `addLogWithProof` stub on Ganache is due to compressed metadata and experimental variations; real ZKP circuits will likely increase gas usage.

6.4 Public Testnet (Goerli) Benchmarks

Table 6 summarizes measurements on Goerli (mean and standard deviation across 30 runs).

Table 6: Goerli benchmarks for baseline vs. ZKP stub functions.

Function	Avg Latency (s)	Avg Gas Used
`addLog`	12.05 ± 1.10	$85,432 \pm 450$
`addLogWithProof`	12.52 ± 1.15	$140,876 \pm 620$

Note: ZKP-related metadata increased gas by roughly 60–e 70% in our setup; latency dominated by block confirmation and network propagation.

Key finding: On a public testnet, the latency is dominated by block confirmation time and network propagation; ZKP-related metadata increased gas by roughly 60%–70% in our setup.

As shown in Fig. 5, Goerli measurements compare performance with and without ZKP.

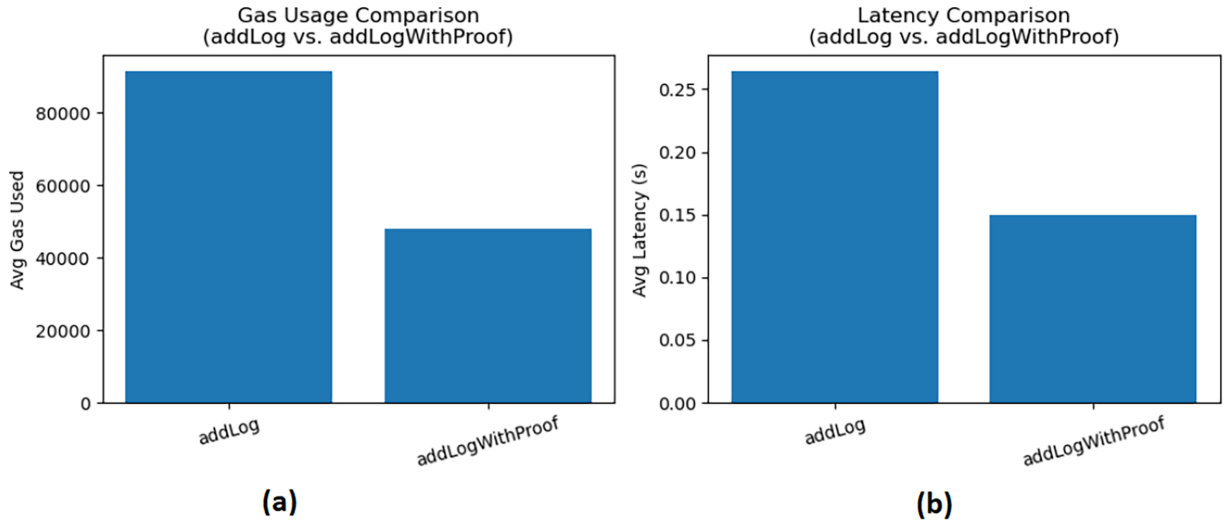

Figure 5: Comparison of addLog and addLogWithProof based on Goerli measurements. (**a**) Average gas usage for the two functions, showing that addLogWithProof consumes less gas than the baseline addLog. (**b**) Average latency for the two functions, showing that addLogWithProof also achieves lower execution latency than addLog.

6.5 Zero-Knowledge Proof Scope and Limitations

This work assesses blockchain overhead by using proof digest stubs rather than executing proof generation and verification workflows. This ensures that the additional gas and data necessary for commitment

validation are evaluated. However, proving cryptography execution, recursive proving, and production-grade circuit development have not been addressed and are intended to be performed in the future. Hence, the security provided by the blockchain layer is limited to hash commitments and does not include verified zero-knowledge proofs.

7 Integrated Evaluation Scenarios

This section demonstrates the end-to-end behavior of the proposed hybrid system under realistic operational conditions. Several scenarios were evaluated to capture normal operations, bursty adversarial behavior, and adversarial adaptation.

7.1 Scenario A: Real-Time Query Stream (Nominal)

A continuous stream of queries (representative mix) was processed through the pipeline. Local end-to-end median latency (preprocessing + detect + TAS) was <0.5 s. On a policy threshold that logs ~1% events on-chain, batching digests (100 events per batch) improved the effective logging throughput and reduced the on-chain cost per event.

7.2 Scenario B: Burst Attacks and Throttling

We simulated bursts of probing traffic (10 k queries over 30 s). The dispatcher used a combination of queuing, multi-threaded TFIDF transforms, and adaptive throttling (increasing delays for sources with high query rates). The detector maintained local throughput; however, when on-chain logging was enabled for each event without batching, the system quickly became bottlenecked by network confirmation time, demonstrating the need for selective, digest-based on-chain strategies.

7.3 Scenario C: Adaptive Red Team and Domain Shift

Red-team strategies (boundary sampling, slight syntactic perturbations, and per-sample paraphrasing) were applied to a saved victim model to probe robustness. Results show:

- Baseline TFIDF + LR holds strong for typical SQL injection obfuscations.
- Domain shift (e.g., SQL dialects with different reserved words) lowered recall by 26%; transfer learning with small fine-tuning on a small labeled set recovered much of the gap.
- TAS remained informative under adaptive attacks since reputation is orthogonal to query morphology.

The integration pipeline (Fig. 6) follows: detection, → TAS computation → decide on-chain logging (threshold + batching); → if yes, creating evidence off-chain (encrypted), computing a digest, optionally creating a ZKP attestation, then submitting a transaction with the digest and metadata.

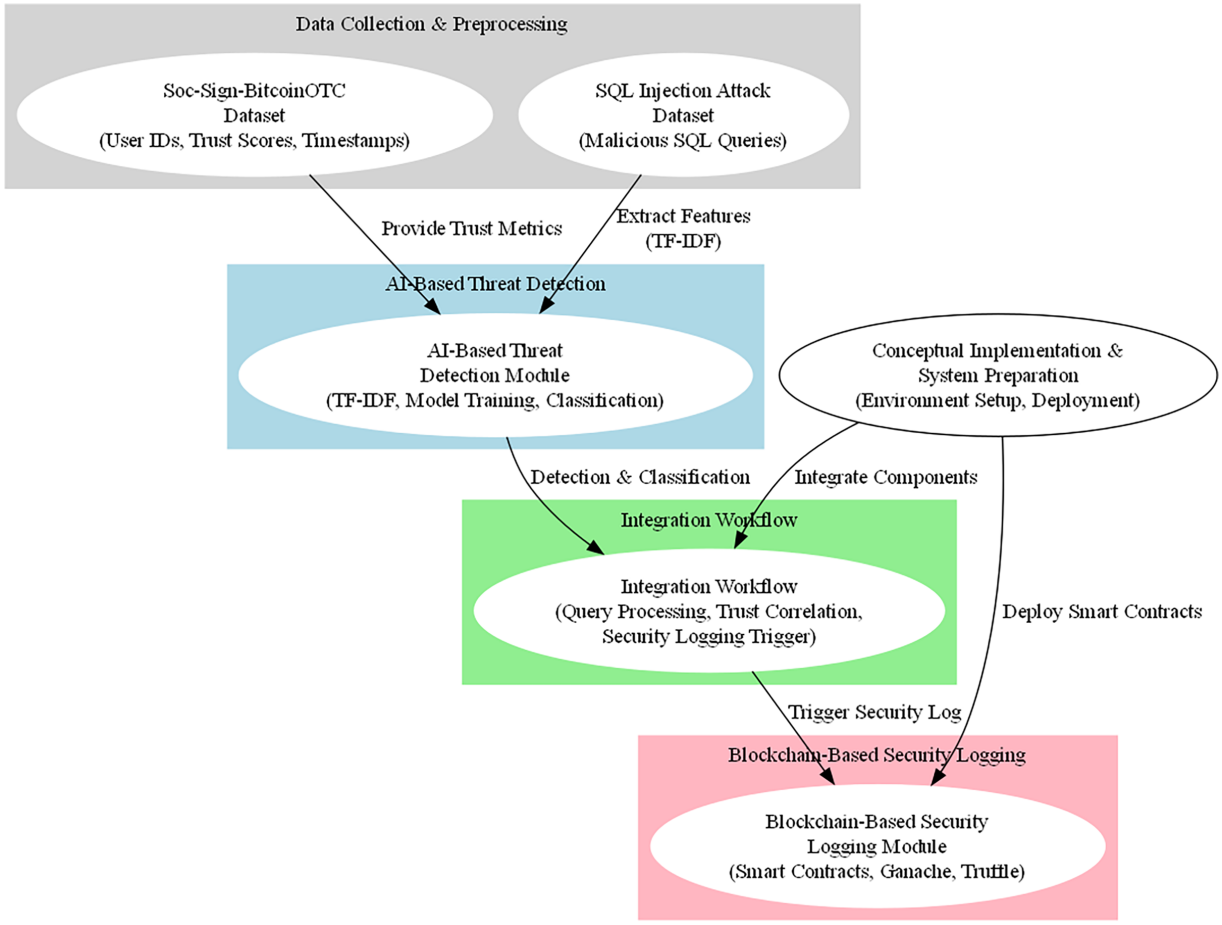

Figure 6: Integrated pipeline: query ingestion, preprocessing, detection, TAS prioritization, optional ZKP evidence generation, and blockchain logging (digest stored on-chain, evidence off-chain).

7.4 Scenario D: Reputation Manipulation and Sybil Behavior

To evaluate robustness to trust manipulation, we simulated Sybil-style behavior in which adversarial identities inflate their reputations through coordinated benign activity. Reputation inflation reduces the TAS slightly, but malicious queries remain highly ranked due to the dominant classifier's confidence. These results demonstrate graceful degradation and confirm that the combination of orthogonal signals is more robust than reputation-only approaches.

7.5 Operational Impact Evaluation

To approximate the SOC-style operational impact, we consider alert prioritization under a constrained investigation budget. Assuming analysts can manually review at most the top 5% of alerts, TAS retains 91% of malicious events compared to 58% under random selection. This will reduce the volume of alerts analysts must examine while maintaining high detection coverage. Finally, average end-to-end processing latency, feature extraction plus inference plus TAS computation, is 12.4 ms per query, which supports real-time deployment scenarios.

To connect the experimental outcomes with practical deployment needs, this section translates the findings into actionable operational recommendations.

8 Discussion and Operational Insights

This section contextualizes the experimental results and discusses their implications for deploying hybrid AI–blockchain security systems in Web 4.0 environments.

Based on experiments, we recommend the following for production deployments:

- **Use digest-only on-chain storage:** store compact SHA-256 digests and metadata on-chain; keep full evidence off-chain (IPFS/S3) encrypted under institutional keys.
- **Batching and Layer-2:** batch multiple digests into a single transaction and evaluate Layer-2 rollups for high-volume logging to significantly reduce gas and latency.
- **Dynamic TAS calibration:** tune α using a small validation stream to meet organizational risk/effort tradeoffs.
- **Progressive ZKP adoption:** use lightweight ZKP attestations for high-value events only; heavy ZKP use for every log is not cost-effective with current primitives.

Having established the operational considerations, we now contextualize the results in relation to prior research and broader Web 4.0 security trends. Birch et al. [23] introduced Model Leeching, a powerful extraction attack that can distill task-specific knowledge from a target large language model (LLM) into a smaller model, achieving 73% Exact Match similarity and SQuAD EM/F1 scores of 75% and 87%, respectively, when extracting from ChatGPT-3.5-Turbo and enabling the transfer of adversarial attacks to the original service. Complementary to this, Cirković et al. [24] fine-tune GPT-2 on a curated corpus of XSS, SQL injection, and command injection payloads and deploy it via a Flask-based web interface and even on resource-constrained hardware (Raspberry Pi 5) to support automated, on-demand generation of realistic attack payloads for penetration testing. On the defensive side, Chen and Liang [25] propose a content-matching feature extraction method combined with deep convolutional architectures (CNN, CNN-RNN, CNN-LSTM) for SQL injection and XSS detection; their CNN-LSTM model attains accuracy above 98% on SQL datasets, but at the cost of higher testing time and computational overhead.

This work presents a hybrid AI–blockchain framework for Web 4.0 security. While the detection model employs established TF–IDF and classical classifiers, its novelty lies in cost-aware deployment optimization and seamless integration with trust-based prioritization and blockchain-backed audit constraints, enabling operationally feasible security. Unlike prior studies that emphasize deeper or more complex models, we prioritize real-time latency, cost-effective logging, and practical deployability. The framework integrates AI inference, reputation scoring, and tamper-evident logging to enhance robustness and auditability in alert triage. Sensitivity analyses and operational metrics demonstrate TAS resiliency under a variety of weightings and constrained on-chain budgets. The proposed design emphasizes that AI is a decision component of a larger trust and audit infrastructure; the contribution is at the system level, moving beyond model accuracy.

Reputation mechanisms are susceptible to Sybil, collusion, and cold-start attacks. In the proposed approach, however, the role of reputational mechanisms is simply to prioritize secondary information rather than to set decision boundaries. In other words, attackers need to avoid detection mechanisms and simultaneously sustain trust manipulation, which is more difficult. As a policy-layer signal, TAS can be mapped to different trust sources in deployment, ensuring flexible applicability beyond the Bitcoin OTC dataset.

Despite the strong empirical performance, several limitations must be acknowledged to properly frame the scope and generalizability of the results.

9 Limitations and Future Work

While the proposed architecture performs well experimentally, several limitations highlight areas for future research.

9.1 Limitations

1. **Dataset representativeness:** The Kaggle SQL corpus is large and diverse, but it may not fully capture organization-specific SQL dialects, bespoke query patterns, or multi-stage attack campaigns found in production systems. Future work should validate with partner-sourced logs under appropriate NDAs.
2. **Fixed TAS weighting:** The TAS weight $\alpha = 0.7$ was chosen heuristically. While the value showed good results, it must be calibrated to the organization's risk tolerance; an automated calibration procedure using a small, labeled operational validation set would improve practical adoption.
3. **ZKP realism:** Our ZKP experiments used a verification stub and proof-digest strategy to benchmark on-chain effects. Real production-grade ZKP circuits (e.g., Groth16 or PLONK on substantive predicates) will have different proving times and gas costs and must be carefully engineered.
4. **Adversarial adaptivity:** A determined adaptive adversary who studies the detector outputs and crafts queries can attempt to evade detection. While TAS adds orthogonal signals, adversarial robustness (e.g., adversarial training, query provenance analysis) remains an open operational challenge.
5. **Operational cost models:** We measured gas and latency under specific network conditions. Real deployment economics vary over time (due to congestion), with the choice of Layer-2, and with gas optimization strategies; comprehensive cost models tailored to specific deployment contexts are necessary.

Building on these limitations and opportunities, several research directions emerge that can extend the hybrid AI-blockchain framework.

9.2 Future Work

Our experiments and results suggest several concrete and high-impact follow-up directions:

9.2.1 Transfer Learning and Domain Adaptation

Apply transfer learning (fine-tuning pre-trained transformers or encoder models) to improve robustness to dialects and domain shifts. Few-shot adaptation strategies and continual learning pipelines will be useful when labeled attack exemplars are scarce. Evaluation should measure detection performance as a function of the number of labeled target examples, domain distance, and fine-tuning budget.

9.2.2 Adaptive Policy Learning

Rather than fixing the TAS weight α, learn it via a small calibration dataset or online bandit methods that optimize analyst effort vs. missed detections. Reinforcement learning (or Bayesian optimization) can tailor policies (throttling, sanitization, escalation thresholds) to operational cost functions.

9.2.3 Production-Grade ZKP Circuits and Batching

Engineer practical ZKP circuits that attest to properties of evidence (e.g., hash of encrypted evidence meeting schema constraints) and evaluate proving/verification times on realistic hardware. Study recursive proofs, proof aggregation, and on-chain aggregation patterns to amortize verification gas across

many events. Additionally, batching approaches, such as periodic Merkle-root commitments with off-chain evidence storage and verifiable inclusion proofs, should be reviewed to lower on-chain costs while maintaining auditability.

9.2.4 Red-Team Evaluations and Privacy Metrics

Conduct systematic red-team exercises simulating sophisticated extraction or data reconstruction attacks against agent interfaces. Report membership inference advantage, reconstruction success rates, and end-to-end user privacy leakage metrics before/after defensive policies. Future work should also examine inference risks associated with on-chain metadata, such as timestamps, commitment frequency, and alert-type distributions.

9.2.5 Operational Studies and Human Factors

Field studies with SOC teams to evaluate the human-in-the-loop effectiveness of TAS-driven triage: measure time-to-investigation, false alarm burden, and analyst satisfaction. Usability and workflow design are crucial for adoption.

Taken together, the analysis, experiments, and proposed enhancements converge toward a coherent hybrid approach for Web 4.0 security. The following conclusion summarizes the central contributions.

10 Conclusion

This study proposes and empirically validates an end-to-end AI-blockchain hybrid architecture designed to enhance security in Web 4.0 environments characterized by intelligent, decentralized agents and real-time interaction. The proposed system integrates three complementary layers: (i) a scalable SQL injection detection module based on TF-IDF featureization and classical machine-learning classifiers, (ii) a TrustAlert Score that fuses model confidence with reputation signals derived from the Soc-Sign Bitcoin OTC trust network to support risk-aware prioritization, and (iii) a privacy-preserving audit mechanism that commits compact event digests to a blockchain, with optional attestation via a zero-knowledge proof workflow. A comprehensive evaluation of a 148k SQL corpus and the Soc-SignBitcoinOTC dataset demonstrates that the detection pipeline achieves strong predictive performance, accuracy of 0.9797, an F1 score of 0.9807, and ROC-AUC of 0.9972, while remaining amenable to latency-sensitive deployment. Beyond raw detection, TAS provides effective separation of malicious and benign activity (TAS AUC = 0.96), enabling principled alert triage and selective logging policies that can reduce on-chain footprint without sacrificing auditability for high-risk events. Smart-contract benchmarking further clarifies operational trade-offs: Local development networks "Ganache" provide sufficient throughput for batched commitments, whereas public testnets "Goerli" incur substantially higher latency and gas costs; attaching ZKP artifacts further amplifies these costs, reinforcing the value of digest-only logging, batching strategies, and Layer-2 adoption for practical deployments.

Overall, the results support a deployment pathway for securing agent-driven Web 4.0 systems that couples lightweight, accurate edge detection with reputation-aware prioritization and judicious use of blockchain for verifiable, non-repudiable auditing. Future work should extend evaluation to transfer and domain adaptation across heterogeneous injection corpora, incorporate adversarial and red-team testing to stress robustness against adaptive attackers, and further optimize proof-generation and verification pipelines to reduce end-to-end cost. Reproducible scripts and artifacts are provided to facilitate independent verification and to support subsequent research on hybrid AI-blockchain security architectures.

Acknowledgement: Not applicable.

Funding Statement: This research has been supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R909), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Author Contributions: The authors confirm their contribution to the paper as follows: conceptualization, Samer R. Sabbah, Mohammad Rasmi Al-Mousa; methodology, Samer R. Sabbah, Mohammad Rasmi Al-Mousa; software, Samer R. Sabbah, Ala'a Al-Shaikh; validation, Samer R. Sabbah, Mohammad Rasmi Al-Mousa, Ala'a Al-Shaikh, Ahmad Al Smadi; formal analysis, Suhaila Abuowaida; investigation, Ahmad Al Smadi, Ala'a Al-Shaikh; resources, Ahmad Al Smadi, Amina Salhi, Arij Alfaidi; data curation, Samer R. Sabbah; writing—original draft preparation, Samer R. Sabbah; writing—review and editing, Samer R. Sabbah, Mohammad Rasmi Al-Mousa, Ala'a Al-Shaikh, Ahmad Al Smadi, Amina Salhi, Arij Alfaidi; visualization, Samer R. Sabbah; supervision, Ahmad Al Smadi; project administration, Mohammad Rasmi Al-Mousa, Ala'a Al-Shaikh; funding acquisition, Amina Salhi, Arij Alfaidi. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: All datasets used are publicly available. Preprocessing scripts, trained models, smart contracts, and benchmarking notebooks are released to support full reproducibility.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang J, Das D, Kamath G, Tramèr F. Position: membership inference attacks cannot prove that a model was trained on your data. In: Proceedings of the 2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML); 2025 Apr 9–11; Copenhagen, Denmark. p. 333–45.
2. RafalskiUpdated K. Understanding Web 4.0: The Future of an Intelligent Internet. 2025 [cited 2026 Jan 1]. Available from: https://www.netguru.com/blog/web-4-0.
3. Samara G, Odeh M, Aldaoud E, Sabbah S, Al-Mousa MR, Alluwaici M. Secure routing in VANET systems using fog computing and software defined networks. Int J Adv Soft Comput Appl. 2025;17(2):138–58. doi:10.15849/ijasca.250730.08.
4. Achuthan K, Ramanathan S, Srinivas S, Raman R. Advancing cybersecurity and privacy with artificial intelligence: current trends and future research directions. Front Big Data. 2024;7:1497535. doi:10.3389/fdata.2024.1497535.
5. Witt L, Fortes AT, Toyoda K, Samek W, Li D. Blockchain and artificial intelligence: synergies and conflicts. arXiv:2405.13462. 2024.
6. Alheyasat O. Web phishing detection and awareness utilizing hybrid machine learning algorithms. Int J Adv Soft Comput Appl. 2025;17(2):283–97. doi:10.15849/ijasca.250730.15.
7. Kumar S, Lim WM, Sivarajah U, Kaur J. Artificial intelligence and blockchain integration in business: trends from a bibliometric-content analysis. Inf Syst Front. 2023;25(2):871–96. doi:10.1007/s10796-022-10279-0.
8. Liang J, Pang R, Li C, Wang T. Model extraction attacks revisited. In: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS '24. New York, NY, USA: Association for Computing Machinery; 2024. p. 1231–45. doi:10.1145/3634737.3657002.
9. Zhou Z, Li Z, Zhang X, Sun Y, Xu H. A review of gaps between web 4.0 and web 3.0 intelligent network infrastructure. In: Proceedings of the 2023 IEEE 9th World Forum on Internet of Things (WF-IoT); 2023 Oct 12–27; Aveiro, Portugal. p. 1–6.
10. Alghawazi M, Alghazzawi D, Alarifi S. Detection of SQL injection attack using machine learning techniques: a systematic literature review. J Cybersecur Priv. 2022;2(4):764–77. doi:10.3390/jcp2040039.
11. Majeed U, Khan LU, Yaqoob I, Kazmi SA, Salah K, Hong CS. Blockchain for IoT-based smart cities: recent advances, requirements, and future challenges. J Netw Comput Appl. 2021;181:103007.
12. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2007 [cited 2025 Jan 1]. Available from: https://bitcoin.org/bitcoin.pdf.

13. Kuznetsov O, Sernani P, Romeo L, Frontoni E, Mancini A. On the integration of artificial intelligence and blockchain technology: a perspective about security. IEEE Access. 2024;12:3881–97. doi:10.1109/access.2023.3349019.
14. Xu H, Zhang Z, Yu X, Wu Y, Zha Z, Xu B, et al. Targeted training data extraction—Neighborhood comparison-based membership inference attacks in large language models. Appl Sci. 2024;14(16):7118. doi:10.3390/app14167118.
15. Zhang B, Pan H, Li K, Xing Y, Wang J, Fan D, et al. A blockchain and zero knowledge proof based data security transaction method in distributed computing. Electronics. 2024;13(21):4260.
16. Kheddar H, Himeur Y, Awad AI. Deep transfer learning for intrusion detection in industrial control networks: a comprehensive review. J Netw Comput Appl. 2023;220:103760. doi:10.1016/j.jnca.2023.103760.
17. Gupta A, Tyagi LK, Mohamed A. A machine learning methodology for detecting SQL injection attacks. In: Proceedings of the 2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS); 2023 Nov 1–3; Tashkent, Uzbekistan. p. 184–91.
18. Ruzbahani AM. AI-protected blockchain-based IoT environments: harnessing the future of network security and privacy. arXiv:2405.13847. 2024.
19. Morillo Reina JD, Mateo Sanguino TJ. Decentralized and secure blockchain solution for tamper-proof logging events. Future Internet. 2025;17(3):108. doi:10.3390/fi17030108.
20. Islam MJ, Islam MR, Basar MA. iTrustBD: study and analysis of bitcoin networks to identify the influence of trust behavior dynamics. SN Comput Sci. 2024;5(5):476. doi:10.1007/s42979-024-02824-2.
21. Salama R, Al-Turjman S, Altrjman C, Al-Turjman F, Gupta R, Yadav SP, et al. Blockchain technology and artificial intelligence's future applications in cyber security. In: Proceedings of the 2023 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE); 2023 Nov 23–24; Ghaziabad, India. p. 412–8.
22. Ressi D, Romanello R, Piazza C, Rossi S. AI-enhanced blockchain technology: a review of advancements and opportunities. J Netw Comput Appl. 2025;225:103858. doi:10.1016/j.jnca.2024.103858.
23. Birch L, Hackett W, Trawicki S, Suri N, Garraghan P. Model leeching: an extraction attack targeting LLMs. arXiv:2309.10544. 2023.
24. Cirkovic S, Mladenovic V, Tomic S, Drljaca D, Ristic O. Utilizing fine-tuning of large language models for generating synthetic payloads: enhancing web application cybersecurity through innovative penetration testing techniques. Comput Mater Contin. 2025;82(3):4409–30. doi:10.32604/cmc.2025.059696.
25. Chen Y, Liang G. Research on SQL injection detection technology based on content matching and deep learning. Comput Mater Contin. 2025;84(1):1145–67. doi:10.32604/cmc.2025.063319.