Z-Shield: A Lightweight Hybrid Browser-Based Intrusion Detection Framework Using Hybrid Machine Learning

Oluwasegun Godwin Bamisaye[1], Osichinaka Chiedu Ubadike[2], Adeniran Kolade Ademuwagun[3], Obunike Arinze Ubadike[4], Samaila Musa Abdullahi[5] and Freeman Bitrus[6]

[1,3,5,6]Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

[2]Department of Aerospace Engineering, Faculty of Air Engineering, Air Force Institute of Technology, Kaduna, Nigeria

[4]Department of Computer Science, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

DOI: https://doi.org/10.51584/IJRIAS.2026.11070014

Received: 08 July 2026; Accepted: 13 July 2026; Published: 25 July 2026

ABSTRACT

Client-side attacks such as cross-site scripting, SQL injection, and command-and-control traffic increasingly slip past server-centric defences because the browser itself is where the damage happens. This paper introduces Z-Shield, a real-time detection framework built as a Google Chrome extension and organised around a three-tier architecture. At its core is a hybrid engine: a supervised Random Forest handles known attack categories, while an unsupervised Isolation Forest watches for the zero-day cases no labelled dataset could have anticipated. We evaluate the system on the BCCC-CSE-CIC-IDS2018 dataset using five behavioural flow metrics: Flow Duration, Total Forward Packets, Total Backward Packets, Mean Packet Length, and Flow Inter-Arrival Time Mean. The supervised layer reaches 99.13% accuracy, 99.90% precision, 99.27% recall, and a 99.58% F1-score, and the unsupervised layer peaks at 97.09% isolation accuracy, ahead of the 93% benchmark reported by prior work. Importantly, the full pipeline resolves in under 100 milliseconds end-to-end, which keeps it usable for everyday browsing rather than just the lab.

Keywords: Browser Extension, Hybrid Machine Learning, Isolation Forest, Random Forest, Web Security, Zero-Day Detection

INTRODUCTION

Web browsers have quietly become one of the most attractive targets for attackers. Cross-site scripting, SQL injection, phishing, command-and-control channels, and denial-of-service attempts all now happen at the client side, often in ways that never touch the parts of the network a server-side monitor can actually see [1, 5]. That blind spot is structural: network-layer detection systems were not built to look inside a browser tab, so they simply cannot observe what happens there [2, 9].

Signature-based detection compounds the problem. It depends on a library of known threat fingerprints, which by definition cannot cover an attack nobody has catalogued yet [1, 6]. Machine learning offers a way around this limitation. Isolation Forest, for instance, works from the premise that anomalies are rare and structurally different from normal traffic [3, 11], and needs no prior knowledge of what an attack looks like. Supervised classifiers like Random Forest complement this by recognising known attack families with high precision, using ensembles of decision trees to vote on a classification.

This paper presents Z-Shield, a hybrid detection framework that runs as a Chrome extension and combines supervised and unsupervised learning inside a three-tier client-backend-persistence architecture. Three contributions follow from this design: (1) a three-tier hybrid detection architecture that embeds both Isolation Forest and Random Forest directly in a browser-native extension; (2) a five-feature behavioural flow pipeline

tuned for sub-100ms inference; and (3) empirical results, namely 99.13% supervised accuracy and 97.09% unsupervised isolation accuracy on the BCCC-CSE-CIC-IDS2018 benchmark with sub-100ms end-to-end latency, that demonstrate the approach holds up under realistic conditions.

LITERATURE REVIEW

Behavioural Anomaly Detection

Chua et al. [3] applied Isolation Forest to e-commerce web traffic and reported 93% accuracy and 95% precision, though with a 10% false-negative rate and a scope limited to application-level logs. Popova [16] found that Isolation Forest performs well precisely when labelled data is scarce, but noted that pushing recall too high can erode accuracy once traffic spikes. Lakshmi et al. [11] confirmed the algorithm's value in software-defined networking contexts, and Djidjev [4] proposed siForest, a structured variant built on binary tree partitioning that is effective, but with recall that drops off against fast-evolving threats.

Hybrid Intrusion Detection

Almuhanna and Dardouri [1] showed that weighted ensembles improve detection of rare attack patterns in network intrusion detection systems, at the cost of computational overhead that introduces latency spikes in production. Arnob et al. [2] reviewed the IDS literature systematically and concluded that continuous real-time monitoring paired with careful feature engineering is not optional but a baseline requirement for deployment. Mohale and Obagbuwa [12] identified a persistent tension between model explainability and computational efficiency, one that Z-Shield addresses through a lightweight, Joblib-serialised model architecture rather than a heavier, harder-to-interpret ensemble.

Browser-Level Detection

Prasad et al. [15] built a Chrome extension for phishing detection and flagged browser performance degradation as a key deployment risk. Holdbrook et al. [5] argued that feature selection quality, more than model choice, determines real-time detection accuracy in browser execution monitoring. Islam [7] proposed ML-based URL detection inside a Chrome extension, though it only handles a limited set of URL formats. King et al. [10] introduced PP3D, an in-browser, vision-based defence against web manipulation attacks, one more signal that browser-native security is an active research direction. Hozouri et al. [6] surveyed the current IDS landscape and pointed to the absence of end-to-end, browser-native solutions as a genuine gap.

Dataset

We selected the BCCC-CSE-CIC-IDS2018 dataset [13] for its breadth: over 5.6 million flow records spanning Brute Force, Botnet, DoS, DDoS, XSS, and SQL Injection categories. Ji et al. [9] showed that metadata-driven flow analysis can detect threats effectively without inspecting payload content, which validates the approach Z-Shield takes: relying on statistical flow features rather than HTTPS payload inspection, and preserving user privacy in the process.

METHODOLOGY

System Architecture

Z-Shield follows a three-tier hybrid detection architecture. Tier 1, client-side data capture, is a Chrome extension written in JavaScript that watches browser network activity continuously, extracts five core behavioural flow metrics, and packages them as structured JSON payloads for the backend. Tier 2, backend intelligence, is a Django 5.2.13 REST API that hosts the hybrid AI engine, runs preprocessing, and executes inference through all three detection layers in sequence. Tier 3, persistence and visualisation, logs every detected anomaly to a PostgreSQL database via the AnomalyLog model, surfaced through an administrative dashboard for centralised monitoring and human-in-the-loop feedback.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII June 2026

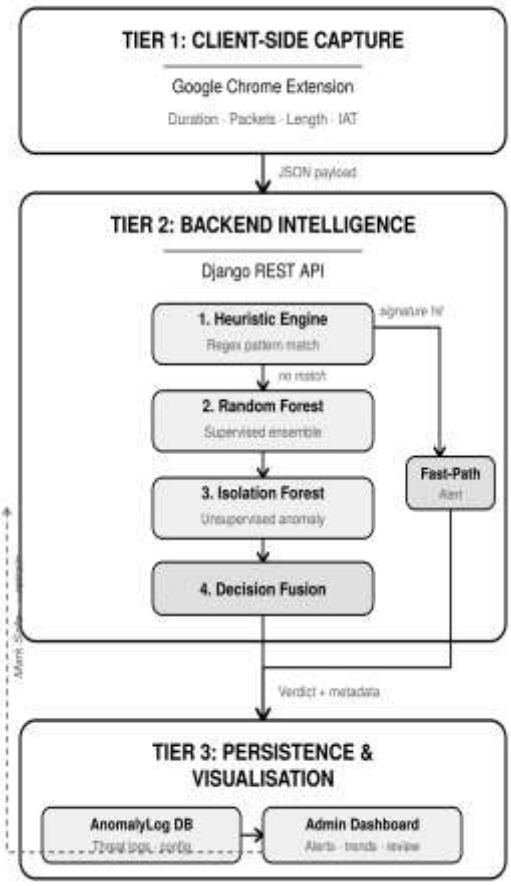

Figure 1: Z-Shield Three-Tier Hybrid Detection Architecture (Author, 2026)

Feature Engineering and Preprocessing

The raw BCCC-CSE-CIC-IDS2018 feature space exceeds 80 attributes. We narrowed this to five behavioural flow metrics chosen for their discriminative power at the browser level: (1) Flow Duration, which separates automated probes from genuine user interaction; (2) Total Forward Packets, sensitive to scanning and injection activity; (3) Total Backward Packets, which reflects response anomalies; (4) Mean Packet Length, which flags the payload-size irregularities typical of injection attacks; and (5) Flow IAT Mean, which captures the timing rhythm characteristic of different connection types [9].

Preprocessing ran in three stages. First, we removed null and infinite values, which are common artefacts of packet-rate calculations, and pruned duplicate records so the model would not overfit to noise. Second, we aggregated the five selected metrics statistically per session. Third, because Mean Packet Length follows a power-law distribution typical of web traffic, we applied a log1p transform before scaling:

$$x' = \ln(1 + x) \quad (1)$$

where x is the raw Mean Packet Length value and x' is the transformed feature. This compresses the long tail of large packet sizes without distorting the smaller, more common values. We then applied StandardScaler normalisation to give every feature zero mean and unit variance, as shown in Eq. (2):

$$z = (x - \mu) / \sigma \quad (2)$$

where μ and σ are the feature's mean and standard deviation computed on the training partition. This step ensures no single feature dominates the anomaly score simply because of its native numeric scale.

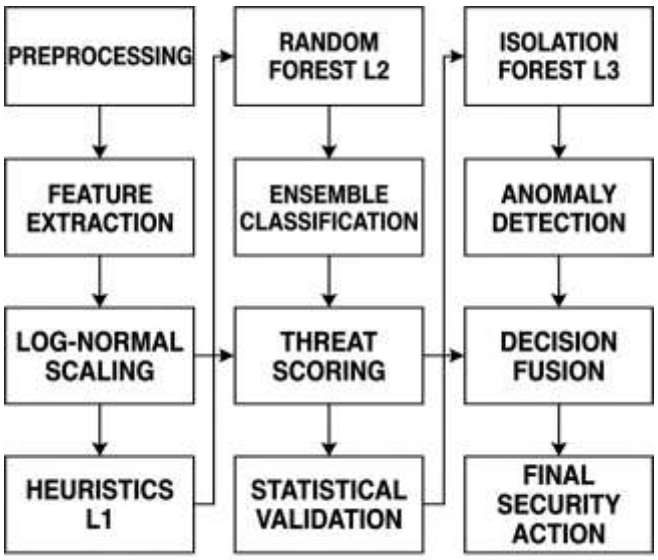

Figure 2: Block Diagram of Model Implementation (Author, 2026)

Hybrid Detection Pipeline

Three layers make up the Z-Shield pipeline. Layer 1, the context-aware heuristic engine, applies regular-expression pattern matching to URL parameters and request headers, catching SQL injection sequences (UNION SELECT, OR 1=1) and XSS markers in sub-millisecond time. A signature match triggers an immediate alert through this Fast-Path, bypassing the AI layers entirely, since there is no reason to wait on a model when the signature is already unambiguous.

Layer 2, the supervised Random Forest, trains 100 decision-tree estimators on BCCC-CSE-CIC-IDS2018 with cross-validation. Each tree casts a vote, and the ensemble prediction is the majority outcome:

$$\hat{y} = \text{mode}\{ h_1(x), h_2(x), ..., h\_T(x) \} \quad (3)$$

where h_t(x) is the prediction of the t-th tree and T = 100. Individual trees split on the feature and threshold that most reduce Gini impurity at each node, given in Eq. (4):

$$G = 1 - \Sigma\, p_i^2 \quad (4)$$

where p_i is the proportion of samples belonging to class i at that node.

Layer 3, the unsupervised Isolation Forest, is configured with a 3% contamination ratio to reflect realistic, sparse-anomaly network conditions [3], and needs no labelled attack data to flag zero-day behavioural deviations. Isolation Forest scores each point by how quickly it can be separated from the rest of the data through random partitioning; the anomaly score is:

$$s(x, n) = 2^{(- E[h(x)] / c(n))} \quad (5)$$

where E[h(x)] is the average path length to isolate point x across all trees, and c(n) is the expected path length for an unsuccessful search in a binary search tree of n points, used to normalise the score. Scores close to 1 indicate a strong anomaly; scores near 0.5 indicate a normal point.

Decision Fusion combines the three layers' outputs: a high-confidence benign classification from Random Forest can override a marginal Isolation Forest anomaly signal, which keeps the combined false-positive rate down. We express this fusion rule as:

$$\text{Alert} = \text{HeuristicMatch} \vee (\text{RF} = \text{malicious}) \vee (\text{IF\_score} > \tau \wedge \text{RF\_confidence} < \kappa) \quad (6)$$

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII June 2026

where τ is the Isolation Forest anomaly threshold and κ is the confidence threshold below which a Random Forest benign classification is not trusted enough to suppress an anomaly flag. Trained models are serialised with Joblib, which keeps backend load times under a millisecond.

Evaluation Methodology

We evaluated the system on the standard confusion-matrix metrics: accuracy, precision, recall, and F1-score, defined in Eqs. (7) to (10).

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (7)$$

$$\text{Precision} = TP / (TP + FP) \quad (8)$$

$$\text{Recall} = TP / (TP + FN) \quad (9)$$

$$F1 = 2 \cdot (\text{Precision} \cdot \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (10)$$

The unsupervised component was evaluated separately under the same 3% contamination ratio used by Chua et al. [3], to keep the comparison fair. End-to-end latency was measured from the moment the Chrome extension captures a feature to the moment a risk score is returned, under both normal browsing conditions and simulated high-volume traffic bursts.

Performance Evaluation

Z-Shield was deployed on Pop!_OS Linux with Python 3.10.12, Django 5.2.13, Scikit-learn 1.7.2, Pandas 2.3.3, and Joblib 1.5.3. We evaluated it on a withheld validation subset of BCCC-CSE-CIC-IDS2018 covering XSS, SQL Injection, Brute Force, Botnet, and DoS attack vectors across 37,556 instances. Latency was measured under normal operating conditions and again under simulated high-volume traffic bursts to check real-time suitability held up under stress.

RESULTS AND DISCUSSION

Figures 3 and 4 show Z-Shield running live. Figure 3 captures the Chrome extension flagging active SQL Injection and XSS attempts, listing the offending URL strings, timestamps, and per-alert action controls. Figure 4 shows the administrative dashboard with an abnormal traffic flow flagged, illustrating the full pipeline from client-side capture through to centralised oversight.

Figure 3: Z-Shield Browser Extension in Alert Mode Showing Live SQL Injection and XSS Detections

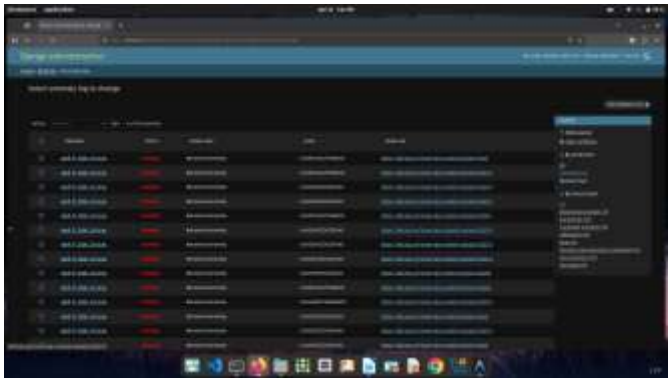

Figure 4: Administrative Dashboard Flagging an Abnormal Traffic Flow

Supervised Layer Performance

Table 1: Aggregated System Performance Metrics (Supervised Random Forest)

Metric	Result	Significance
Accuracy	99.13%	Overall correctness across all traffic classes
Precision	99.90%	Reliability of alerts; minimal false alarms
Recall	99.27%	992+ per 1,000 attacks detected
F1-Score	99.58%	Harmonic balance of precision and recall

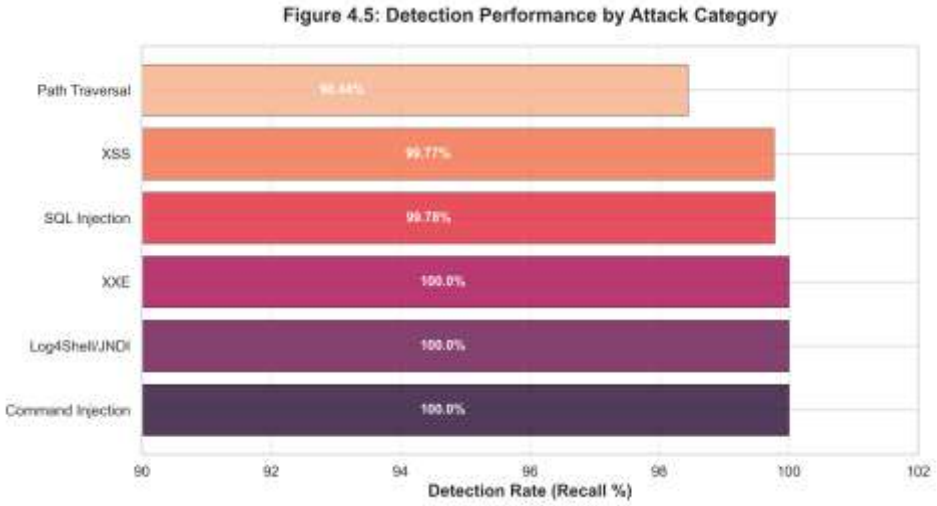

Figure 5: Supervised Layer Performance Metrics

The 99.13% global accuracy shows the model reliably tells benign and malicious flows apart across attack categories. The 99.90% precision means alert noise is negligible, an important property if analysts are expected to act on every alert. The confusion matrix in Figure 6 reports 581 true negatives, 36,668 true positives, 37 false positives, and 270 false negatives across the 37,556 validation instances, which lines up internally with the headline metrics above.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII June 2026

Figure 6: Supervised Detection Confusion Matrix (n = 37,556)

Figure 7 breaks recall down by attack category. Path Traversal came in lowest at 98.44% recall, while XXE, Log4Shell/JNDI, and Command Injection each hit 100%. XSS and SQL Injection, the two categories most central to this study, reached 99.77% and 99.78% recall respectively, which is a reassuring result given that client-side threats were the whole motivation for building Z-Shield [5, 6].

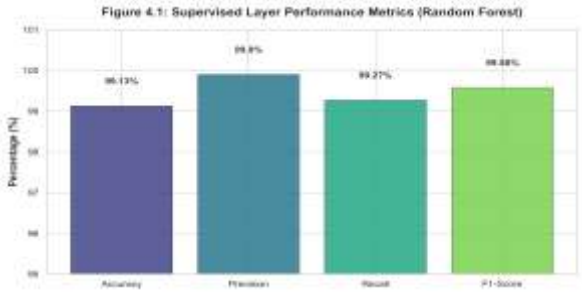

Figure 7: Detection Performance by Attack Category (Recall)

Unsupervised Layer Performance

Table 2: Comparative Performance: Z-Shield vs. Chua et al. [3] Baseline

Metric	**Chua et al. [3]**	**Z-Shield**
Accuracy	93.00%	97.09%
Recall	90.00%	94.76%
Precision	95.00%	99.64%

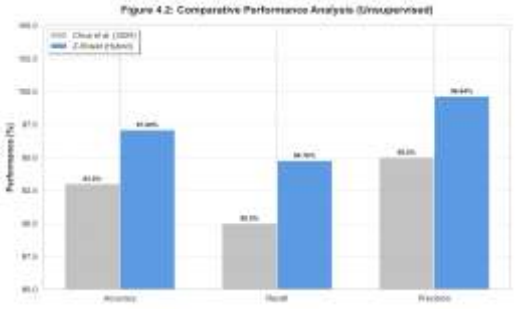

Figure 8: Comparative Performance: Z-Shield vs. Baseline

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII June 2026

Z-Shield's peak isolation accuracy of 97.09% beats the Chua et al. [3] baseline of 93% by 4.09 percentage points, with a 4.76-point gain in recall. Most of that improvement traces back to the log1p normalisation applied to Mean Packet Length (Eq. (1)): compressing the power-law tail of packet sizes lets Isolation Forest draw cleaner partitioning boundaries instead of letting a handful of extreme values dominate the splits. A second factor is that Z-Shield works from browser-level flow features, which expose the timing rhythm of an attack, something application-level text logs, the basis of the baseline study, simply do not capture.

Operational Realism and False Positive Analysis

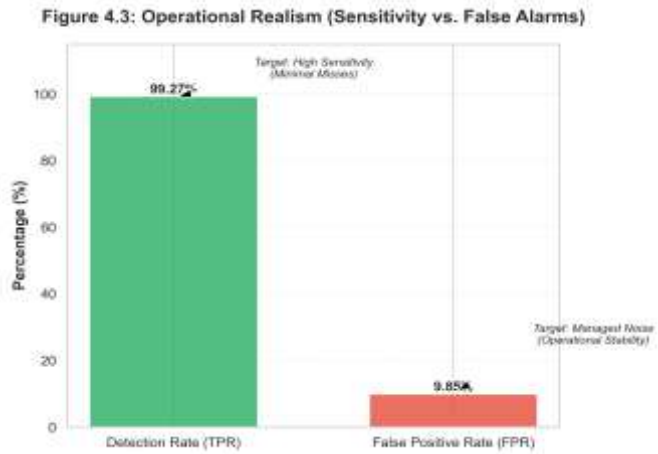

Figure 9: Detection Sensitivity (99.27%) vs. False Alarm Rate (9.85%)

Under combined hybrid testing, Z-Shield's false positive rate came to 9.85%, computed as in Eq. (11):

$$FPR = FP / (FP + TN) \quad (11)$$

This number reflects a familiar trade-off in intrusion detection: push sensitivity too high and analysts drown in alert fatigue; push specificity too high and zero-day attacks slip through. A 9.85% FPR is a reasonable middle ground: sensitive enough to catch novel threats, manageable enough that a human analyst can still act on it. It also speaks directly to the instability Almuhanna and Dardouri [1] observed in their heavier ensemble-based NIDS under production-scale data. A human-in-the-loop mechanism lets users flag legitimate traffic as safe from the dashboard, and those annotations feed periodic retraining, gradually recalibrating thresholds and pushing the FPR down over time [12].

End-to-End Latency

The complete Z-Shield pipeline, from Chrome extension feature capture, through JSON transmission and backend preprocessing, to hybrid model inference, stayed under 100 milliseconds end-to-end across every test condition, including high-volume stress tests. Joblib serialisation kept model loading to sub-millisecond time, and the context-aware heuristic engine finished its regex matching in well under a millisecond, reserving the costlier AI inference only for traffic that makes it past the Fast-Path filter. Together these results support the case that Z-Shield is fast enough for live browser deployment [15].

Comparative Analysis

Against Chua et al. [3], Z-Shield improves unsupervised accuracy by 4.09% and recall by 4.76%, which we attribute to browser-level flow features exposing attack timing that static log content cannot show. Against Almuhanna and Dardouri [1], Z-Shield reaches comparable global accuracy (99.13%) while holding sub-10ms inference latency, a sharp contrast to the latency instability their heavier ensemble architecture exhibited. Running detection client-side also takes load off the central server, addressing a scalability limitation common to server-centric NIDS approaches [2].

LIMITATIONS AND FUTURE WORK

While Z-Shield demonstrates strong detection performance, several limitations should be acknowledged, and each points to a concrete direction for follow-up work.

Dataset scope: Evaluation was conducted on a single benchmark, BCCC-CSE-CIC-IDS2018, which, despite its scale and category diversity, cannot fully capture the variability of live, real-world browser traffic or the pace at which attack patterns evolve. Validating Z-Shield against additional public datasets and traffic collected from real browsing environments would strengthen confidence in its generalisability.

Browser coverage: The current implementation targets Google Chrome specifically. Extending the extension layer to Firefox, Microsoft Edge, and Safari would broaden its practical deployment footprint, though this requires re-implementing extension logic against each platform's own APIs rather than a simple repackaging.

Resource and scalability analysis: This study did not include a detailed computational resource analysis, CPU utilisation, memory overhead, or scalability under enterprise-scale, high-volume traffic conditions. Such profiling is necessary before large-scale deployment claims can be made with confidence, and is planned as a dedicated evaluation in future work.

Baseline comparison: The comparative evaluation was limited to a small set of baseline methods. Benchmarking Z-Shield against deep learning, transformer-based, and graph neural network intrusion detection approaches would offer a more complete picture of its relative standing.

Robustness, explainability, and privacy: This work did not examine adversarial robustness, model explainability, or the privacy implications of a browser-native detection pipeline in depth. These are important directions given the sensitivity of browsing data, and are the focus of ongoing follow-up work. Federated learning for cross-user threat intelligence sharing and adaptive, online threshold recalibration also remain promising extensions to reduce false positives without compromising user privacy.

CONCLUSION

This paper presented Z-Shield, a real-time malicious web behaviour detection framework deployed as a Google Chrome extension and built around a hybrid machine learning architecture. By combining a supervised Random Forest with an unsupervised Isolation Forest inside a three-tier client-backend-persistence architecture, the system reached 99.13% supervised accuracy, 99.90% precision, 99.27% recall, a 99.58% F1-score, and 97.09% Isolation Forest detection accuracy, 4.09 percentage points above the 93% baseline set by prior foundational work, while keeping end-to-end latency under 100 milliseconds throughout testing.

Future work will focus on extending Z-Shield to additional browsers, validating the framework on diverse real-world datasets, improving model explainability and robustness against adversarial attacks, and investigating adaptive online learning techniques for continuous threat detection. Advanced sequence-learning models, such as LSTMs or transformer-based architectures, may also be explored to improve the detection of complex multi-stage attacks.

REFERENCES

1. R. Almuhanna and S. Dardouri, "A machine learning approach for anomaly based network intrusion detection," Frontiers in Artificial Intelligence, vol. 8, p. 1625891, 2025. https://doi.org/10.3389/frai.2025.1625891
2. A. K. B. Arnob et al., "A comprehensive systematic review of intrusion detection systems," Journal of Edge Computing, vol. 4, no. 1, pp. 73-104, 2025. https://doi.org/10.55056/jec.885
3. W. Chua et al., "Web traffic anomaly detection using isolation forest," Informatics, vol. 11, no. 4, p. 83, 2024. https://doi.org/10.3390/informatics11040083
4. C. Djidjev, "siForest: Detecting network anomalies with set structured isolation forest," IEEE Trans. Network Science Eng., vol. 11, no. 2, pp. 292-305, 2024. https://arxiv.org/abs/2412.06015

5. R. K. B. Holdbrook, A. Kwubeghari, and N. G. Ezeji, "An analysis of key tools for detecting cross site scripting attacks on web based systems," Lecture Notes ICST, vol. 54, pp. 187-202, 2024. https://www.researchgate.net/publication/377873080
6. A. Hozouri, A. Mirzaei, and M. Effatparvar, "A comprehensive survey on intrusion detection systems with advances in machine learning," Discover Artificial Intelligence, vol. 5, p. 314, 2025. https://doi.org/10.1007/s44163-025-00578-1
7. S. Sriram, V. Kumar, and T. Keerthivasan, "Advanced malware detecting Chrome extension using machine learning," Daffodil International University Archive, 2025. https://www.researchgate.net/publication/391218113
8. I. H. Ji et al., "AI based anomaly detection over encrypted traffic: A systematic literature review," Sensors, vol. 24, no. 3, p. 898, 2024. https://doi.org/10.3390/s24030898
9. Z. Okonkwo et al., "A graph representation framework for encrypted network traffic classification," Computers & Security, vol. 148, p. 104134, 2024. https://doi.org/10.1016/j.cose.2024.104134
10. S. King et al., "PP3D: An in-browser vision-based defense against web behavior manipulation attacks," arXiv preprint, 2025. https://arxiv.org/abs/2510.18465
11. M. Lakshmi et al., "Evaluating isolation forest for anomaly detection in SDN security," J. Electrical Systems, vol. 19, no. 4, pp. 279-297, 2023. https://www.researchgate.net/publication/387517103
12. V. Z. Mohale and I. C. Obagbuwa, "Evaluating ML based intrusion detection systems: Enhancing transparency," Frontiers in Computer Science, vol. 7, p. 1520741, 2025. https://doi.org/10.3389/fcomp.2025.1520741
13. A. Mohamed, "CSE-CIC-IDS2018 Dataset," Mendeley Data, V1, 2024. https://doi.org/10.17632/29hdbdzx2r.1
14. H. Park et al., "Unsupervised ML methods for anomaly detection in network packets," Electronics, vol. 14, no. 14, p. 2779, 2025. https://doi.org/10.3390/electronics14142779
15. D. Prasad et al., "Enhancing internet security: A ML based browser extension to prevent phishing attacks," 2024 Intl. Conf. on Communication, 2024. https://doi.org/10.1109/IC3SE62002.2024.10593201
16. A. S. Kechedzhiev and O. L. Tsvetkova, "Anomaly detection research using isolation forest in machine learning," StudNet, vol. 3, no. 12, pp. 1460-1470, 2020. https://www.researchgate.net/publication/379984675