AMERICAN JOURNALS
Open Access, Peer Reviewed Journals

American Journal of Technology and Applied Sciences
ISSN (E): 2832-1766 Volume 49, June - 2026

MACHINE LEARNING-BASED DETECTION OF MALICIOUS WEB REQUESTS USING HTTP TRAFFIC FEATURES

Sabina Norbekova Tolib qizi
Tashkent University of Information Technologies
named after Muhammad al-Khwarizmi, Tashkent, Uzbekistan

ABSTRACT

Web applications have become essential components of modern digital infrastructure, supporting online banking, e-commerce, education platforms, healthcare systems and public administration services. However, their continuous exposure to the Internet makes them frequent targets of cyberattacks such as injection attacks, cross-site scripting, path traversal, authentication abuse and automated vulnerability scanning. Traditional rule-based security mechanisms remain useful for detecting known attack signatures, but they are often limited against modified, encoded or previously unseen malicious requests. This study investigates machine learning-based detection of malicious web requests using HTTP traffic features. The experiment uses the CSIC 2010 HTTP dataset, which contains normal and anomalous web requests generated for testing web attack protection systems. Nineteen request-level features were extracted from URL, request method and content fields, including URL length, character composition, encoded characters, path depth, request method indicators and content-related measures. Four machine learning models, namely Logistic Regression, Decision Tree, Random Forest and Linear Support Vector Machine, were trained and evaluated using accuracy, precision, recall, F1-score and confusion matrix analysis. The experimental results show that Random Forest achieved the best overall performance, with 92.27% accuracy, 92.47% precision, 88.37% recall and 90.37% F1-score. Feature importance analysis showed that URL length, URL letters, URL digits, content letters and content length were the most influential predictors. The findings indicate that machine learning can serve as an additional analytical layer for web application security, although its reliability depends on feature engineering, dataset representativeness, class balance and continuous monitoring.

KEYWORDS

Machine learning; web application security; malicious request detection; HTTP traffic; cybersecurity; anomaly detection; Random Forest; feature engineering.

Introduction

Web applications are now central to digital transformation. They are used in banking, education, public administration, e-commerce, healthcare and many other areas where users interact with digital services through browsers, mobile applications and application programming interfaces. As global digital dependency increases, web applications have become one of the most exposed components of modern information systems. The International Telecommunication Union reported that approximately 5.5 billion people were online in 2024, representing 68% of the global population [1]. This level of connectivity creates significant opportunities for digital development, but it also increases the potential attack surface for cybercriminals.

Web application attacks remain a major cybersecurity concern because web systems often process sensitive data, user credentials, payment information and business-critical operations. The OWASP Top 10 is widely used as a reference standard for critical web application security risks and highlights categories such as broken access control, cryptographic failures, injection, insecure design and security misconfiguration [2]. These weaknesses may allow attackers to access unauthorized data, manipulate application behaviour, inject malicious commands or disrupt service availability.

The financial impact of cybersecurity incidents is substantial. IBM reported that the global average cost of a data breach reached USD 4.88 million in 2024, increasing from USD 4.45 million in 2023 [3]. Although not all data breaches originate from web applications, exposed web interfaces, insecure APIs, weak authentication mechanisms and misconfigured services often become entry points in cyberattack chains.

Traditional security mechanisms such as signature-based intrusion detection systems, web application firewalls and manually written rules remain important. However, these methods may fail when attackers modify payloads, encode malicious strings, imitate normal user behaviour or use automated tools to generate large volumes of requests. Rule-based systems are especially limited when attacks do not exactly match known signatures.

Machine learning provides a complementary approach to web application security. Instead of relying only on predefined rules, ML models can learn patterns from historical request data and classify new requests as normal or malicious. This makes machine learning useful for detecting abnormal HTTP requests, suspicious parameter structures, unusual payload lengths and malicious keyword patterns. However, the success of ML-based detection depends on feature quality, dataset representativeness, evaluation methodology and operational integration.

The aim of this study is to evaluate machine learning models for detecting malicious web requests using HTTP traffic features. The study focuses on request-level feature extraction and comparative analysis of commonly used ML classifiers. The main research question is: How effectively can machine learning models detect malicious web requests using HTTP traffic-based features?

The contribution of this study is threefold:

1. It extracts practical HTTP request-level features for malicious request detection.
2. It compares Logistic Regression, Decision Tree, Random Forest and Linear SVM using standard cybersecurity evaluation metrics.
3. It analyses feature importance and discusses the practical value and limitations of ML-based web request detection for web application security.

2. Related Work

Machine learning has been widely investigated in cybersecurity, particularly in intrusion detection, malware classification, anomaly detection and phishing detection. Earlier intrusion detection systems were largely based on signatures and predefined rules. These approaches can be effective against known attacks, but they are less suitable for detecting novel or modified attacks.

Sommer and Paxson argued that machine learning has potential in intrusion detection but warned that operational deployment is difficult because real-world network environments are complex, dynamic and often different from laboratory datasets [4]. Their work highlights the importance of realistic data, meaningful features and careful evaluation when applying machine learning to security problems.

Buczak and Guven provided a broad survey of machine learning and data mining methods for cybersecurity intrusion detection [5]. Their review shows that supervised models such as Decision Tree, Support Vector Machine, Random Forest and neural networks have been widely applied to security classification problems. At the same time, unsupervised methods are useful when labelled attack data are not available.

In the context of web application security, machine learning can be applied to HTTP request classification, anomaly detection, bot detection, malicious URL identification and API abuse detection. Unlike general network intrusion detection, web request detection focuses on application-layer features such as URL structure, query parameters, HTTP methods, user-agent values, payload length and suspicious strings. This makes feature engineering particularly important.

The CSIC 2010 HTTP dataset has been used in studies on web attack detection because it contains normal and anomalous HTTP requests. The dataset was developed for testing web attack protection systems and includes automatically generated web requests [6]. Although it is older than modern production traffic, it remains useful for educational and experimental research because it directly represents HTTP request-level web application traffic.

Recent security research also shows increasing interest in representation learning and feature-based detection for HTTP traffic. For example, HTTP request embedding approaches have been evaluated on CSIC2010 and other datasets for anomalous traffic detection [7]. Nevertheless, feature-engineered classical ML models remain valuable because they are easier to interpret, less computationally expensive and suitable for environments where explainability is important.

3. Materials and Methods

3.1 Dataset Description

This study uses the CSIC 2010 HTTP dataset, a publicly available dataset created for web application attack detection research. The dataset contains automatically generated HTTP requests representing both normal and anomalous behaviour. It was developed by the Information Security Institute of CSIC, the Spanish Research National Council, and is intended for testing web attack protection systems [6].

The experimental dataset used in this study contains 61,065 HTTP requests, including 36,000 normal requests and 25,065 malicious or anomalous requests. The classification task is formulated as a binary classification problem in which normal HTTP requests are labelled as 0 and malicious or anomalous HTTP requests are labelled as 1.

Table 1. Class distribution of the CSIC 2010 HTTP dataset

Class	Meaning	Number of records
0	Normal HTTP request	36,000
1	Malicious or anomalous HTTP request	25,065
Total	All HTTP requests	61,065

3.2 Data Preprocessing

Raw HTTP request data cannot be directly used by most machine learning algorithms. Therefore, preprocessing was required to transform semi-structured request fields into structured numerical features. Missing values in textual fields were replaced with empty strings, the request method was encoded into binary indicators, and content-length values were converted into numerical format when possible.

The preprocessing stage included request parsing, label encoding, text normalization, missing-value handling, feature construction and train-test splitting. The dataset was divided into training and testing subsets using an 80:20 stratified split. As a result, 48,852 records were used for training and 12,213 records were used for testing. Stratification was applied to preserve the class distribution in both subsets.

3.3 Feature Engineering

Feature engineering is a critical part of this study because HTTP requests are semi-structured text data. Instead of using raw request text only, this research extracts interpretable request-level features that may indicate malicious behaviour. Nineteen features were extracted from URL, content and request method fields.

Table 2. Extracted HTTP traffic features

Feature	Description	Security relevance
url_length	Length of the requested URL	Very long URLs may contain encoded payloads or abnormal paths
url_special_chars	Number of special characters in the URL	Attacks often contain symbols such as %, <, >, quotes and slashes
url_digits	Number of digits in the URL	Useful for detecting abnormal parameter values
url_letters	Number of alphabetic characters in the URL	Captures structural composition of the URL
url_encoded_chars	Number of URL-encoded character patterns	Encoded sequences may hide malicious payloads
url_suspicious_keywords	Count of risky keywords in URL	Detects terms such as select, union, script and traversal patterns
url_path_depth	Depth of URL path	Deep or unusual paths may indicate traversal attempts
url_num_parameters	Number of query parameters	Abnormal parameter count may indicate manipulation
url_extension_length	Length of file extension in URL	Reflects requested resource pattern
content_length	Length of request body content	Long content may indicate payload-based attacks

content_special_chars	Number of special characters in content	Payloads often contain special symbols
content_digits	Number of digits in content	Captures numeric parameter patterns
content_letters	Number of alphabetic characters in content	Captures payload structure
content_encoded_chars	Number of encoded characters in content	Encoded payloads may indicate evasion
content_suspicious_keywords	Count of risky keywords in content	Captures injection and scripting indicators
declared_content_length	Declared content length header	Represents request body length when available
method_get	Binary indicator for GET method	Encodes HTTP method
method_post	Binary indicator for POST method	Encodes HTTP method
method_put	Binary indicator for PUT method	Encodes HTTP method

Examples of suspicious keywords included select, union, insert, delete, drop, script, alert, traversal patterns such as ../ and ../../, passwd, or 1=1, eval, sleep, benchmark, cmd and exec. These features were selected because they are interpretable and relevant to common web attack patterns.

3.4 Machine Learning Models

Four machine learning algorithms were compared: Logistic Regression, Decision Tree, Random Forest and Linear Support Vector Machine. Logistic Regression was used as a baseline model because it is simple, interpretable and suitable for binary classification. Decision Tree was included because it can model rule-like relationships and is easy to interpret, although it may overfit if not properly controlled. Random Forest was selected as an ensemble method that combines multiple decision trees and can provide feature importance scores [8]. It is robust for structured data and can capture nonlinear interactions among features. Linear SVM was included as a high-dimensional linear classifier based on the support vector machine principle [9]. The implementation used scikit-learn, a widely used Python library for machine learning [10].

3.5 Evaluation Metrics

In cybersecurity classification tasks, accuracy alone is not sufficient because datasets may be imbalanced. A model may show high accuracy while still failing to detect many attacks. Therefore, this study evaluates model performance using accuracy, precision, recall, F1-score and confusion matrix analysis.

Table 3. Evaluation metrics used in the experiment

Metric	Meaning	Importance
Accuracy	Overall proportion of correctly classified samples	General performance indicator
Precision	Proportion of predicted attacks that were truly malicious	Helps reduce false alerts
Recall	Proportion of real malicious samples correctly detected	Helps reduce missed attacks
F1-score	Harmonic mean of precision and recall	Balances detection and false-alert reduction
Confusion matrix	Distribution of true positives, false positives, true negatives and false negatives	Supports detailed error analysis

Recall is especially important in security contexts because missed attacks can cause serious damage. However, precision is also important because excessive false positives can overwhelm security analysts. Therefore, F1-score was used as the primary model-ranking metric in this study.

4. Experimental Workflow

The experimental workflow consisted of seven stages: dataset loading, request parsing, feature extraction, preprocessing and encoding, train-test split, model training and model evaluation. This workflow transformed raw HTTP request data into numerical features suitable for machine learning classification.

Table 4. Experimental workflow

Stage	Description
1. Dataset loading	Load CSIC 2010 HTTP request records
2. Request parsing	Extract URL, method and content-related fields
3. Feature extraction	Generate 19 numerical request-level features
4. Preprocessing and encoding	Handle missing values and encode method indicators
5. Train-test split	Apply 80:20 stratified split
6. Model training	Train Logistic Regression, Decision Tree, Random Forest and Linear SVM
7. Model evaluation	Compare accuracy, precision, recall, F1-score and confusion matrix

5. Results

5.1 Model Performance Comparison

The experiment was conducted on 61,065 HTTP requests from the CSIC 2010 dataset, including 36,000 normal requests and 25,065 malicious or anomalous requests. After preprocessing, 19 HTTP traffic-based features were extracted. The dataset was divided into training and testing subsets using an 80:20 stratified split. As a result, 48,852 samples were used for training and 12,213 samples were used for testing.

Four machine learning models were trained and evaluated: Logistic Regression, Decision Tree, Random Forest and Linear Support Vector Machine. The experimental results are presented in Table 5.

Table 5. Performance comparison of machine learning models

Model	Accuracy	Precision	Recall	F1-score
Random Forest	0.922705	0.924650	0.883702	0.903713
Decision Tree	0.905674	0.930627	0.832236	0.878686
Linear SVM	0.754033	0.798160	0.536405	0.641613
Logistic Regression	0.747318	0.774111	0.542789	0.638133

Random Forest achieved the best overall performance, with an accuracy of 92.27%, precision of 92.47%, recall of 88.37% and F1-score of 90.37%. Decision Tree was the second-best model, achieving an accuracy of 90.57% and F1-score of 87.87%. In contrast, Logistic Regression and Linear SVM produced lower recall values, indicating that these linear models were less effective in identifying malicious web requests.

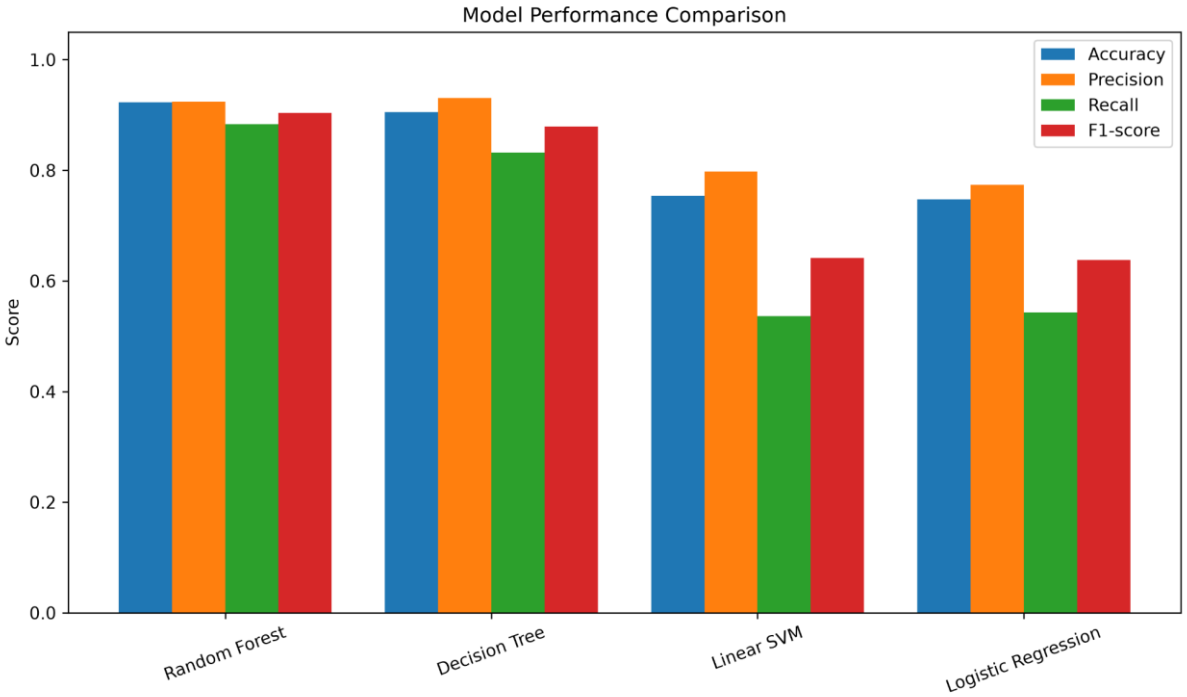

Figure 1. Model performance comparison of evaluated machine learning models.

5.2 Confusion Matrix Analysis

The confusion matrix of the best-performing Random Forest model is shown in Figure 2. Out of 7,200 normal requests in the test set, 6,839 were correctly classified as normal and 361 were incorrectly classified as malicious. Out of 5,013 malicious requests, 4,430 were correctly detected as malicious and 583 were misclassified as normal.

In web application security, false negatives are especially critical because they represent malicious requests that pass undetected. The Random Forest model reduced false negatives compared with the linear models, but the presence of 583 missed malicious requests shows that ML-based detection should be used as an additional layer rather than a standalone protection mechanism.

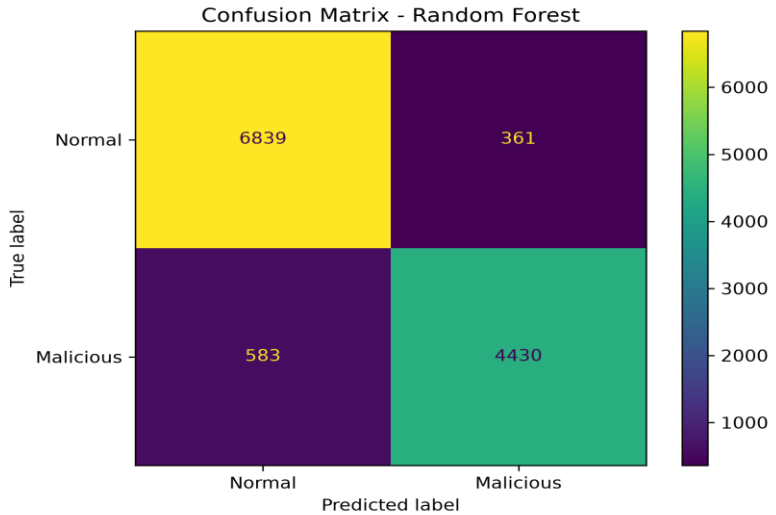

Figure 2. Confusion matrix of the Random Forest model.

5.3 Feature Importance Analysis

Feature importance analysis was performed using the Random Forest model in order to identify which HTTP request characteristics contributed most to malicious request detection. The most influential features are presented in Table 6 and Figure 3.

Table 6. Top HTTP traffic features according to Random Forest importance

Rank	Feature	Importance
1	url_length	0.161979
2	url_letters	0.148721
3	url_digits	0.122856
4	content_letters	0.085850
5	content_length	0.081084
6	url_special_chars	0.061774
7	content_encoded_chars	0.060385
8	url_encoded_chars	0.054729
9	content_digits	0.042827
10	url_extension_length	0.036487

The feature importance results indicate that URL-based features played a major role in detecting malicious requests. The most important feature was URL length, followed by URL letters and URL digits. This suggests that malicious requests often differ from normal requests in terms of URL structure and character composition. Content-related features such as content length, content letters and content encoded characters were also important, especially for POST requests where malicious payloads may be placed in the request body.

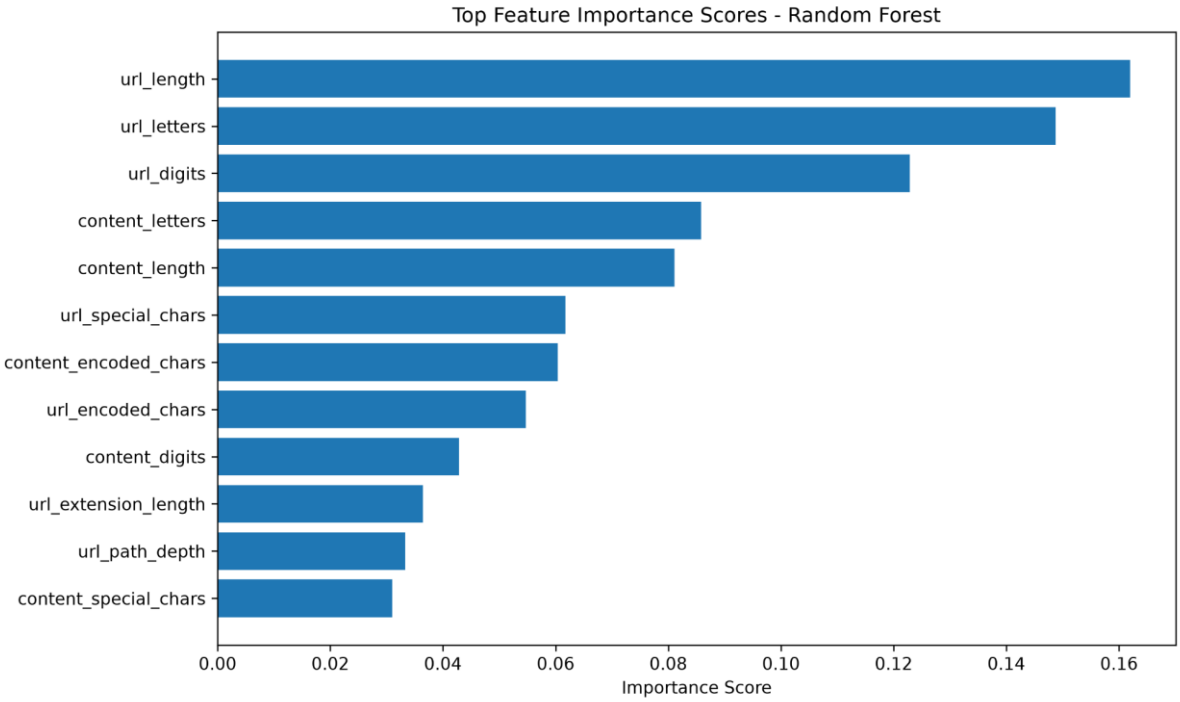

Figure 3. Feature importance scores generated by the Random Forest classifier.

6. Discussion

The experimental findings demonstrate that machine learning can effectively support malicious web request detection when meaningful HTTP traffic features are extracted. Among the evaluated models, Random Forest achieved the strongest overall performance with an F1-score of 90.37%. This result suggests that ensemble-based models are well suited for web request classification tasks because they can learn nonlinear relationships between request-level indicators and malicious behaviour.

The comparison between Random Forest and Decision Tree is also important. Although both are tree-based methods, Random Forest outperformed a single Decision Tree in accuracy, recall and F1-score. This shows the advantage of ensemble learning, where multiple decision trees reduce overfitting and improve generalization. Decision Tree achieved slightly higher precision than Random Forest, but its recall was lower. In cybersecurity contexts, recall is particularly important because false negatives represent malicious requests that remain undetected.

Logistic Regression and Linear SVM showed significantly lower recall values. This indicates that linear decision boundaries were not sufficient for distinguishing malicious and normal HTTP requests based on the extracted features. Web attacks often involve complex combinations of suspicious characters, encoded payloads, URL length, parameter manipulation and request body structure. Therefore, models capable of handling nonlinear interactions are more appropriate for this classification task.

The feature importance analysis further supports this interpretation. URL length, URL letters, URL digits and content length were among the most influential features. These findings suggest that malicious HTTP requests often differ structurally from normal requests. For example, injection attacks and path traversal attempts may introduce unusual character sequences, encoded payloads or longer request paths. Therefore, feature engineering plays a central role in ML-based web request detection.

However, the results should be interpreted with caution. Although the CSIC 2010 dataset is useful for controlled experiments, it may not fully represent modern production web environments. Contemporary web applications often use REST APIs, JSON payloads, cloud-based infrastructure, microservices and JavaScript-heavy frontends. Therefore, high performance on this dataset does not automatically guarantee the same performance in real-world deployment. Future work should evaluate the proposed approach on more recent datasets and real web server logs.

Overall, the findings indicate that ML-based malicious request detection can serve as a valuable additional layer in web application security. It should not replace secure coding, authentication controls, vulnerability testing or web application firewalls. Instead, it can support security monitoring by identifying suspicious requests and prioritizing alerts for further analysis.

7. Limitations

This study has several limitations. First, it uses a public benchmark dataset rather than live production web server logs. Although the CSIC 2010 dataset is suitable for web attack detection experiments, it may not fully represent modern web application traffic. Second, the study focuses on binary classification and does not separately classify each attack type. Third, the extracted features are manually designed and may not capture all semantic characteristics of malicious HTTP requests. Fourth, deep learning models were not included because the study prioritizes interpretable and lightweight machine learning models.

Another limitation is that the dataset may contain patterns that are easier to separate than real-world traffic. As a result, the obtained performance should be treated as experimental evidence rather than a guarantee of deployment performance. Future research may extend this work by using larger and more recent datasets, applying deep learning models such as CNN or LSTM, performing multiclass attack classification and testing the proposed approach on real web server logs.

8. Conclusion

This study investigated machine learning-based detection of malicious web requests using HTTP traffic features. The research used the CSIC 2010 HTTP dataset and extracted 19 interpretable request-level features from URL, content and method fields. Several machine learning models, including Logistic Regression, Decision Tree, Random Forest and Linear Support Vector Machine, were compared using accuracy, precision, recall and F1-score.

The experimental results showed that Random Forest achieved the best overall performance, with 92.27% accuracy, 92.47% precision, 88.37% recall and 90.37% F1-score. Feature importance analysis indicated that URL length, URL letters, URL digits, content letters and content length were the most influential predictors. These findings demonstrate that structural properties of HTTP requests can provide useful signals for detecting malicious behaviour.

Machine learning should be viewed as a complementary analytical layer within a broader defence-in-depth strategy. It can support web application firewalls, intrusion detection systems and security analysts, but it cannot replace secure software development, vulnerability management and expert review. Future research should focus on deep learning-based payload analysis, real-time deployment, adversarial robustness and evaluation on modern web application traffic.

References

[1] International Telecommunication Union, “Facts and Figures 2024: Internet Use,” 2024. [Online]. Available: https://www.itu.int/itu-d/reports/statistics/2024/11/10/ff24-internet-use/

[2] OWASP Foundation, “OWASP Top Ten Web Application Security Risks,” 2025. [Online]. Available: https://owasp.org/www-project-top-ten/

[3] IBM Security, “Cost of a Data Breach Report 2024,” 2024. [Online]. Available: https://www.ibm.com/reports/data-breach

[4] R. Sommer and V. Paxson, “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection,” in Proceedings of the IEEE Symposium on Security and Privacy, 2010, pp. 305-316.

[5] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, 2016.

[6] IMPACT Cyber Trust, “HTTP Dataset CSIC 2010,” 2010. [Online]. Available: https://www.impactcybertrust.org/dataset_view?idDataset=940

[7] M. Gniewkowski, H. Maciejewski, T. R. Surmacz, and W. Walentynowicz, “HTTP2vec: Embedding of HTTP Requests for Detection of Anomalous Traffic,” arXiv preprint arXiv:2108.01763, 2021.

[8] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[9] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.
[10] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[11] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 160, 2021.
[12] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
[13] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," in Military Communications and Information Systems Conference, 2015.
[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proceedings of the 4th International Conference on Information Systems Security and Privacy, 2018, pp. 108-116.
[15] A. Thakkar and R. Lohiya, "A Review of the Advancement in Intrusion Detection Datasets," Procedia Computer Science, vol. 167, pp. 636-645, 2020.