UDC 004.85
IRSTI 06.81.29

https://doi.org/10.55452/1998-6688-2026-23-3-347-361

[1]***Norliza Katuk,**
PhD, ORCID ID: 0000-0001-8805-2574,
*e-mail: k.norliza@uum.edu.my
[1]**Hikmah Nurdin,**
B. Comp. Science, ORCID ID: 0009-0001-3766-5482
e-mail: hikmahnurdin750@gmail.com
[2]**M. Ali Fauzi,**
PhD., ORCID ID: 0000-0002-9696-2807
e-mail: moch.ali.fauzi@ub.ac.id
[3]**Mhd. Furqan**
PhD., ORCID ID: 0000-0002-7255-4454
e-mail: mfurqan@uinsu.ac.id

[1]School of Computing, Universiti Utara Malaysia, Malaysia
[2]Faculty of Computer Science, Universitas Brawijaya, Indonesia
[3]Faculty of Science and Technology, Universitas Islam Negeri Sumatera, Indonesia

AQRACE: MACHINE-LEARNING-BASED DETECTION OF MALICIOUS URLS IN QR CODES FOR REAL-TIME SECURITY ASSESSMENT

Abstract

The widespread use of Quick Response (QR) codes has increased exposure to QR-based phishing, malicious redirection, and malware distribution because users cannot readily inspect the destination before scanning. This study develops aQRace, a machine-learning-based QR code security application that analyses embedded Universal Resource Locators (URLs) before users access their destinations. The study combines URL-based feature extraction, comparative machine-learning evaluation, and operational system integration. URL characteristics covering structural, linguistic, technical, and advanced properties were extracted for classification. Four supervised machine-learning algorithms–XGBoost, Random Forest, Decision Tree, and Logistic Regression–were trained and evaluated using a balanced dataset of 14,000 URLs comprising 7,000 safe and 7,000 malicious samples. Model performance was compared using accuracy, precision, recall, and F1-score. XGBoost achieved the highest accuracy of 84% and precision of 0.80, whereas Random Forest achieved the highest recall of 0.95. Both models achieved an F1 Score of 0.84. XGBoost was selected for deployment based on its overall performance profile and integrated into a client-server application comprising a Flutter mobile frontend and Flask cloud backend. The application supports QR code image uploads and camera-based scanning, and performs URL processing, feature extraction, classification, and user warnings before destination access. Functional testing with representative safe and malicious QR codes confirmed the operation of the complete detection workflow. The findings demonstrate that conventional machine learning using URL-based features can support pre-access QR code security assessment while providing an operational alternative to increasingly complex deep learning approaches.

Keywords: QR code security, quishing, malicious URL detection, machine learning, XGBoost, phishing detection, cybersecurity

Received: September 3, 2026; accepted: September 10, 2026.

Introduction

Quick Response (QR) codes provide a convenient mechanism for connecting physical and printed materials to digital content. Their ease of use and compatibility with mobile devices have

encouraged adoption in digital payments, marketing, authentication, information sharing, and other online services. This convenience also introduces security risks because the destination encoded within a QR code is not readily visible to users [1]. Attackers can exploit this characteristic by distributing malicious QR codes or replacing legitimate codes with fraudulent ones that redirect users to phishing websites or harmful content [2]. The use of QR codes in phishing is commonly known as "quishing" [3]. In a quishing attack, a malicious Uniform Resource Locator (URL) is encoded within a QR code and presented as legitimate content. Once scanned, the code may redirect users to a fraudulent login page, phishing website, malware source, or another malicious destination [4]. Unlike conventional phishing messages, in which a suspicious URL can be inspected before opening, a QR code hides the destination in its encoded image. Users consequently have less information available when deciding whether a code is safe to scan. The security problem, therefore, extends beyond QR code decoding. A conventional scanner can retrieve an embedded URL, but successful decoding does not establish whether the destination is legitimate. Without security assessment before access, users may proceed to a malicious destination before the risk is identified. This limitation is particularly relevant in environments where QR codes are scanned rapidly, and users have little opportunity to inspect decoded addresses. A secure scanning mechanism should therefore analyse the embedded URL before access to the destination and provide an immediate warning when malicious characteristics are detected.

Machine learning offers one approach to this problem because safe and malicious URLs may exhibit different structural, lexical, and technical characteristics. Al-Zahrani et al. [5] examined malicious QR code detection using lexical characteristics extracted from embedded URLs. Five machine-learning classifiers were evaluated using a balanced dataset of 50,000 safe and 50,000 malicious URLs. Their findings demonstrated that URL characteristics can support malicious-URL classification. The evaluation, however, was primarily dataset-based and did not demonstrate the operational performance of the resulting classifier in a deployed QR code scanning process. Sarkhi and Mishra [6] extended URL-based detection towards a lightweight mobile-oriented approach incorporating URL feature extraction, risk scoring, and Decision Tree classification. Lexical and host-based characteristics allowed the system to operate with relatively low computational requirements. Although such an approach is suitable for resource-constrained applications, reliance on a single classifier may restrict the representation of complex relationships among URL characteristics. The effectiveness of URL-based detection may therefore depend not only on the extracted features but also on the classification algorithm used to learn from them.

Comparative machine-learning studies further demonstrate the importance of classifier selection. Herlina and Soeparno [7] evaluated Random Forest, Support Vector Machine, Logistic Regression, Naive Bayes, k-Nearest Neighbours, and XGBoost for detecting phishing URLs associated with QR codes. XGBoost achieved the highest reported accuracy of 92%. The result supports the potential of ensemble learning for identifying patterns associated with phishing URLs. Their evaluation, however, remained at the dataset level, without examining how the classifier would operate once integrated into a QR code scanning application. Other research has focused more directly on operational detection. Vaithilingam and Shankar [4] examined QR-related threats, including quishing, QRLjacking, and malware distribution, and considered artificial intelligence and machine learning for providing early security warnings. Pawar et al. [8], similarly proposed an AI-based QR code scanner intended to detect and block malicious links in real time. These studies highlight the importance of moving security assessment closer to the point at which users interact with QR codes. More broadly, contemporary classification research has increasingly moved from conventional machine-learning models towards architectures capable of learning richer representations from multiple sources of information. Attention mechanisms and multimodal fusion are particularly useful when a classification decision depends on complementary data modalities. For example, Furqan et al. [9], developed an attention-based Vision Transformer for multiclass skin lesion classification that integrates dermoscopic images with structured clinical metadata via a mutual attention fusion mechanism. The model achieved 93.4% accuracy, an F1-score of 92.2%, and an AUC of 0.95, illustrating the potential of combining heterogeneous representations within a unified classification framework. A similar progression

towards richer feature representations is evident in QR code security. Alsulami et al. [10] proposed a malicious QR code detection system that combines AlexNet-based feature extraction, principal component analysis (PCA), and recurrent neural networks. Their GRU configuration achieved accuracies of 99.81% and 99.59% on two datasets, with an average computation time of 7.433 ms per sample. A real-time prototype was also developed, demonstrating the feasibility of deploying the model in an operational environment.

Tian and Zhu [11] further advanced QR code phishing detection by combining QR code decoding, multimodal deep learning, and incremental learning. Their approach transforms decoded QR content into a URL-semantic classification problem and employs BERT, Convolutional Neural Network (CNN), and BiLSTM components to capture local and global URL patterns. Incremental learning enables the model to adapt to emerging phishing samples without repeatedly retraining the complete architecture. The progression from lexical analysis to deep and multimodal learning demonstrates increasing sophistication in QR code threat detection. It also introduces greater complexity in modelling and implementation. Conventional machine-learning approaches therefore remain relevant where direct URL analysis and straightforward integration into an operational scanning workflow are priorities. Two related issues motivate the present study. First, classification performance depends on the feature representation, dataset, and learning algorithm employed. Comparison using a common dataset and feature representation is necessary to assess how different classifiers perform in terms of accuracy, precision, recall, and F1-score. This is important in security classification because the consequences of false positives and false negatives differ. A false negative may expose a user to a malicious destination, whereas a false positive may unnecessarily restrict access to a legitimate URL. Second, predictive performance on an offline dataset does not, by itself, demonstrate the operational feasibility of a QR code security mechanism. Deployment requires QR acquisition, decoding, URL preprocessing, feature extraction, classification, presentation of the security result, and restriction of destination access when a threat is identified. There remains practical value in determining whether conventional machine-learning classifiers operating on URL characteristics can support this complete workflow without relying on a more elaborate deep-learning architecture.

To address these issues, this study develops aQRace, a machine-learning-based QR code security application that analyses embedded URLs before users access their destinations. URLs are characterised by structural, linguistic, technical, and advanced features. Four supervised machine learning algorithms—XGBoost, Random Forest, Decision Tree, and Logistic Regression–are evaluated on the same balanced dataset and feature representation. The selected classifier is subsequently integrated into a client-server QR code scanning application that supports both static image uploads and camera-based scanning.

The study addresses the following research questions:

RQ1: What URL-based features can be extracted to support the machine-learning classification of safe and malicious URLs embedded in QR codes?

RQ2: Which machine-learning algorithm provides the most effective performance for classifying QR code-embedded URLs as safe or malicious?

RQ3: How can the selected machine-learning classifier be integrated into a real-time QR code scanning system for detecting malicious URLs and providing security warnings to users?

Accordingly, the objectives of the study are: (1) to extract structural, linguistic, technical, and advanced URL features for the classification of safe and malicious URLs embedded in QR codes; (2) to evaluate and compare XGBoost, Random Forest, Decision Tree, and Logistic Regression using accuracy, precision, recall, and F1-score; and (3) to integrate the selected classifier into a QR code scanning application and evaluate its functional operation using safe and malicious QR code samples. The contribution of this study lies in linking comparative machine-learning evaluation to operational QR code security. Rather than treating malicious URL classification as an isolated prediction task, aQRace incorporates URL processing, feature extraction, classification, and user warnings into the QR code scanning workflow. The objective is not to outperform computationally sophisticated deep learning architectures solely on predictive accuracy, but to examine the performance and practical integration of conventional machine learning with a compact URL-based representation.

Table 1 – Comparison of related studies on malicious QR code and URL detection

Study	Detection approach	Data/features	Main contribution	Limitation relevant to the present study
Al-Zahrani et al. [5]	ML classification of QR code URLs	50,000 safe and 50,000 malicious URLs; lexical features	Demonstrated malicious-URL classification using URL characteristics	Primarily offline dataset evaluation
Pawar et al. [8]	AI-based QR code scanner	Benign and malicious QR code data	Real-time detection and warning of malicious links	Limited reporting of classification and deployment performance
Herlina and Soeparno [7]	Comparative ML classification	6,000 phishing and 6,000 legitimate URLs	Compared six classifiers; XGBoost achieved 92% accuracy	Dataset-level evaluation without operational deployment
Sarkhi and Mishra [6]	Lightweight URL analysis and Decision Tree	Lexical and host-based URL features	Mobile-oriented detection with low computational requirements	Reliance on a single classifier and predefined features
Vaithilingam and Shankar [4]	AI/ML analysis of QR-related threats	Quishing, QRLjacking, and malware scenarios	Emphasised proactive detection and user warning	Limited comparative classifier evaluation
Alsulami et al. [10]	AlexNet, PCA, and recurrent neural networks	Image-derived QR code features	GRU achieved 99.81% and 99.59% accuracy; real-time prototype	More complex image-based deep-learning pipeline
Tian and Zhu [11]	QR decoding, multimodal deep learning, and incremental learning	URL semantic and structural representations	Combines BERT, CNN, and BiLSTM with incremental adaptation	Greater modelling and training complexity
Present study (aQRace)	Comparative URL-based ML integrated with QR scanning	Balanced safe/ malicious URL dataset; structural, linguistic, technical, and advanced features	Compares four classifiers under the same representation and deploys the selected model	Functional deployment evaluation limited to representative QR code samples

Materials and methods

The study adopted a system development and experimental evaluation approach comprising four stages: dataset preparation, URL feature extraction, machine-learning model development and evaluation, and system implementation and functional testing. The first three stages established the malicious-URL classification mechanism, while the final stage integrated the selected classifier into the aQRace application. The detection process begins when a QR code is submitted as an image or captured using a device camera. The QR code is decoded to retrieve the embedded URL. The URL is then normalised and processed before the required features are extracted. These features are passed to the trained classifier, which assigns the URL to either the safe or malicious class. The result is subsequently returned to the application. A malicious classification generates a security warning and restricts direct access to the detected destination. The original dataset contained URLs labelled as benign, phishing, malware, and defacement. For binary classification, benign URLs were assigned to the safe class, whereas phishing, malware, and defacement URLs were consolidated into the malicious class. URL preprocessing was performed to improve consistency before feature extraction. A URL scheme was added when absent, with HTTP as the default. Hostnames were converted to lowercase, leading www. prefixes were removed, fragments and trailing slashes were eliminated, per cent-encoded characters were decoded, and duplicate records were removed. Query parameters and

path structures were retained because they may contain structural and lexical information relevant to malicious-URL detection. Preliminary experiments used more than 300,000 pre-processed URLs. Although high classification accuracy was initially achieved, subsequent QR code testing showed a tendency towards malicious classifications, suggesting poor generalisation under the intended deployment conditions. Different dataset sizes and class distributions were therefore examined. A balanced dataset of 14,000 URLs was selected for final model development, comprising 7,000 safe and 7,000 malicious URLs. A fixed random seed was applied during sampling to support reproducibility. Although this configuration produced slightly lower accuracy than the preliminary large-scale experiments, it resulted in more stable classification behaviour during subsequent system testing.

Each pre-processed URL was transformed into a numerical feature vector. The feature extraction procedure used four categories of characteristics: structural features, linguistic features, technical identifiers, and advanced metrics. Structural features describe the composition of the URL and include URL length, domain length, path length, and number of subdomains. These variables provide information about the structural complexity of the destination. Linguistic characteristics capture character-level and lexical properties of the URL. They include the number of digits, number of special characters, and presence of suspicious keywords. These features indicate unusual lexical patterns that may occur in malicious addresses. Technical identifiers include using an IP address as the hostname, the presence of HTTPS, and the use of homoglyph characters. Homoglyphs are visually similar characters that can be used to imitate legitimate domain names. Advanced metrics incorporate Shannon entropy and domain frequency analysis. Shannon entropy measures randomness in the URL string and may indicate obfuscated or automatically generated addresses, while domain frequency provides additional information about the occurrence of hostnames.

Table 2 – URL features used for QR code URL classification

Feature category	Extracted characteristics
Structural	URL length; domain length; path length; number of subdomains; top-level domain length
Linguistic	Number of digits; number of special characters; presence of suspicious keywords
Technical identifiers	IP-based hostname; HTTPS; homoglyph characters
Advanced metrics	Shannon entropy; domain frequency analysis

The same feature representation was supplied to all four classifiers so that differences in performance could be attributed to the learning algorithms rather than differences in input representation. Four supervised machine-learning algorithms were evaluated: XGBoost, Random Forest, Decision Tree, and Logistic Regression. These algorithms represent ensemble tree-based learning, single-tree classification, and linear classification strategies. The balanced dataset was divided into training and testing subsets at an 80:20 ratio using stratified sampling to preserve class balance. All models were trained using the same URL feature representation. A probability threshold of 0.45 was applied to probabilistic classification. Performance was evaluated using accuracy, precision, recall, and F1-score. Accuracy measures the proportion of all instances classified correctly. Precision measures the proportion of predicted malicious instances that are actually malicious. Recall represents the proportion of malicious instances correctly identified, while F1-score provides the harmonic mean of precision and recall. Multiple measures were considered because accuracy alone does not fully characterise a security classifier. High recall reduces the number of malicious URLs the system misses, whereas high precision reduces unnecessary warnings for legitimate destinations.

aQRace was implemented using a client-server architecture comprising a Flutter mobile frontend and a cloud-based Flask backend. The frontend was developed using Flutter and Dart and supports two QR code input mechanisms: uploading a static PNG or JPG image and scanning a QR code with the device camera. Communication with the backend is performed through RESTful HTTP requests. The backend was implemented in Python using Flask and deployed on the Render platform. It manages QR code decoding, URL processing, feature extraction, and machine-learning classification. QR codes

are decoded using Pyzbar together with the Python Imaging Library. The decoded URL subsequently undergoes normalisation, short-link expansion via HTTP redirection, and checks against a predefined list of trusted domains. These procedures provide additional rule-based processing around the machine-learning component. URLs requiring further analysis are transformed into the required feature representation and passed to the trained XGBoost classifier. Scikit-learn and XGBoost were used for model development, while Joblib was used to store and load the trained model during runtime. When a URL is classified as malicious, the backend returns a warning response to the Flutter application, which restricts direct access to the destination. Cross-Origin Resource Sharing was enabled using Flask-CORS to support communication between system components. Development followed an iterative Agile process. The frontend and backend were initially tested in a local environment before the completed backend was deployed to the cloud environment.

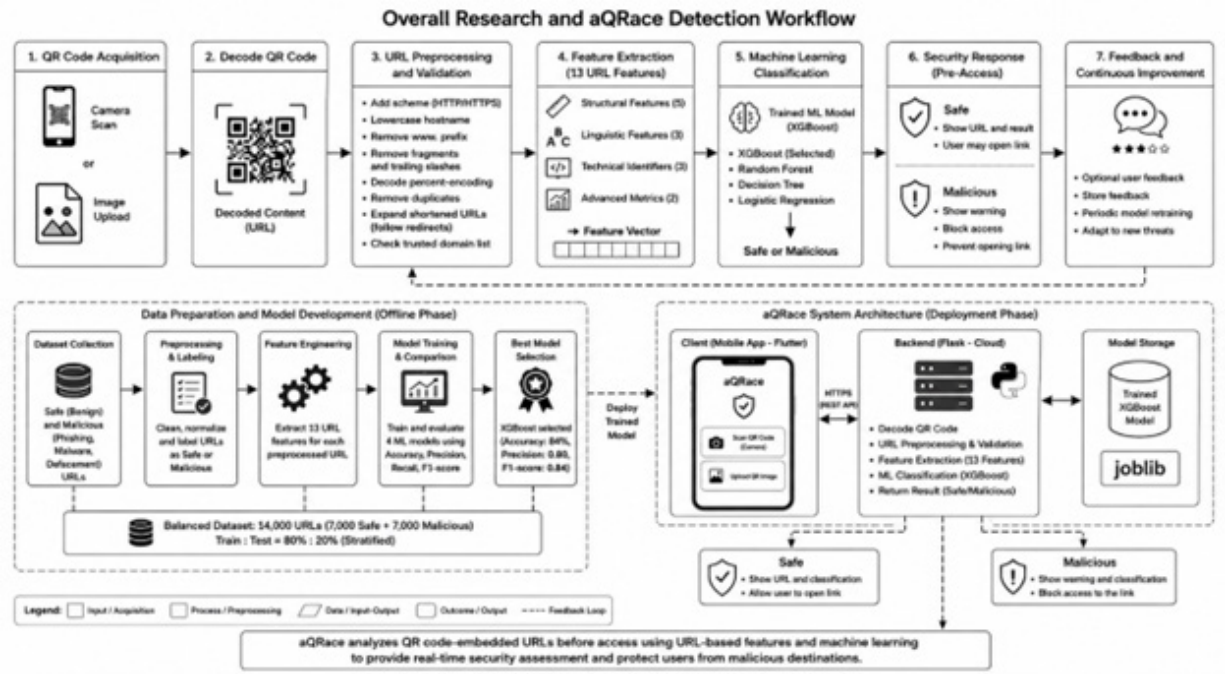

Figure 1 – Overall research and aQRace detection workflow

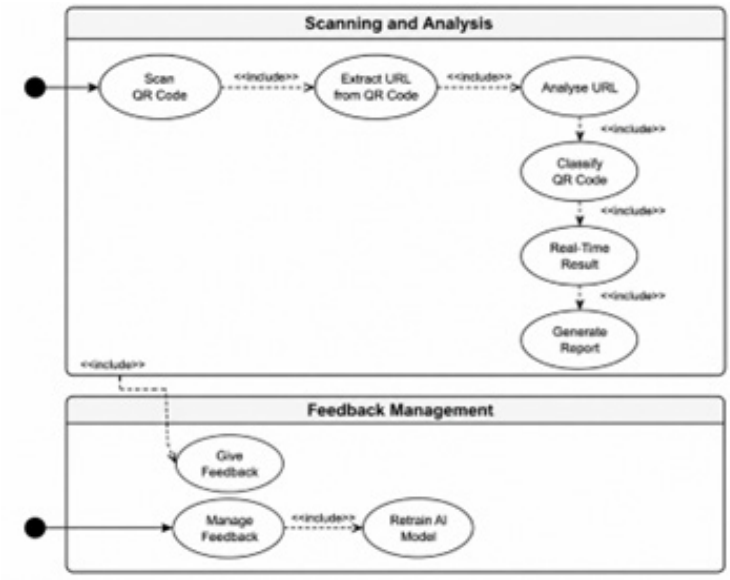

Figure 2 – Use case diagram of the aQRace system

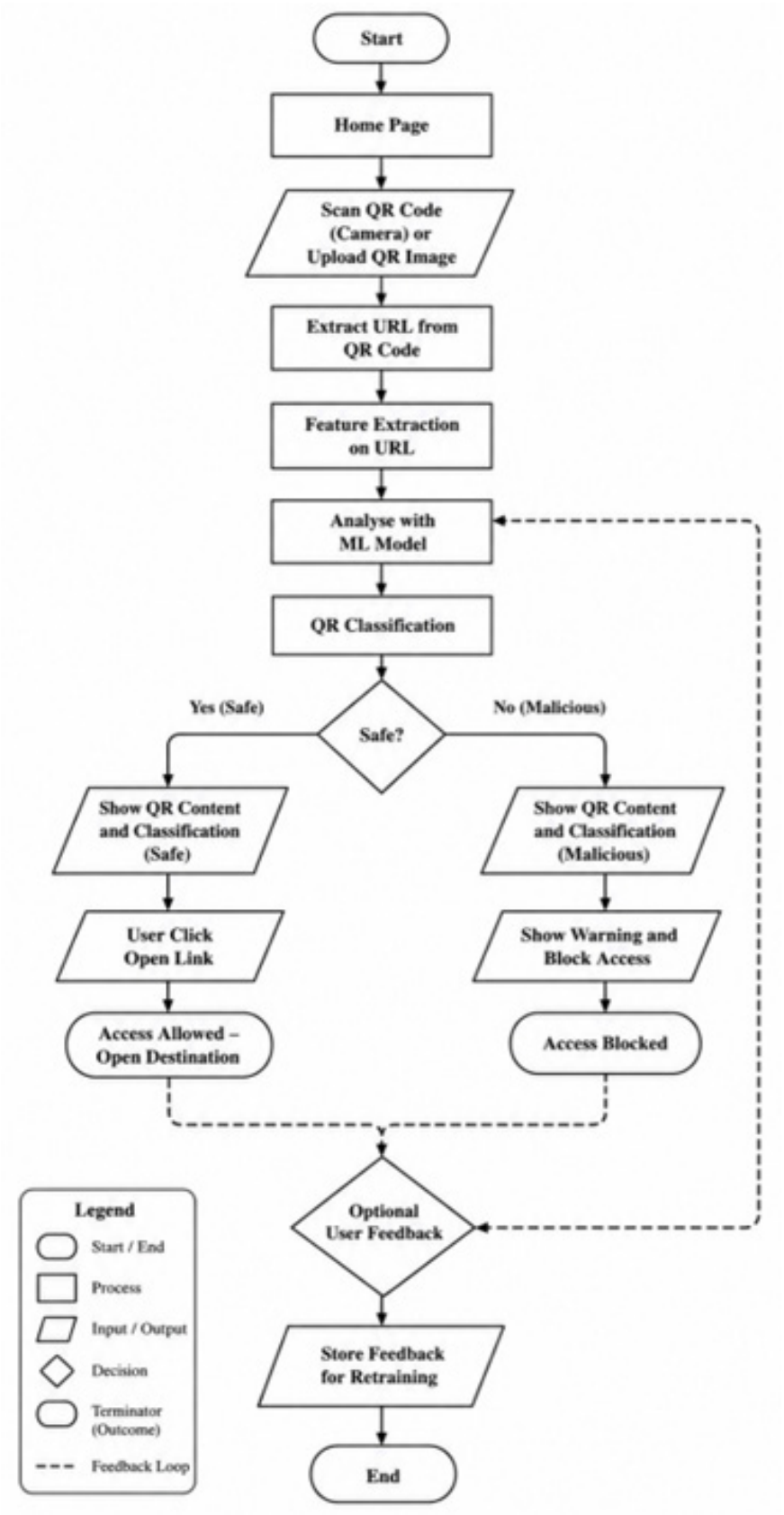

Figure 3 – Operational flow of the aQRace QR code security and malicious URL detection system

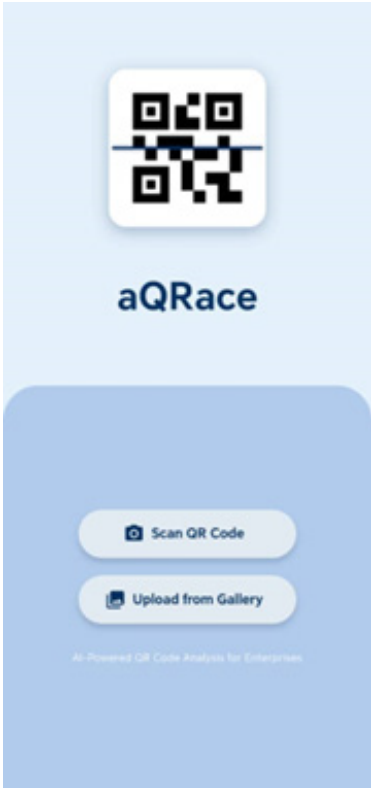

Figure 4 – Home interface showing QR code input options

Figure 5 – Camera-based QR code scanning and classification interface

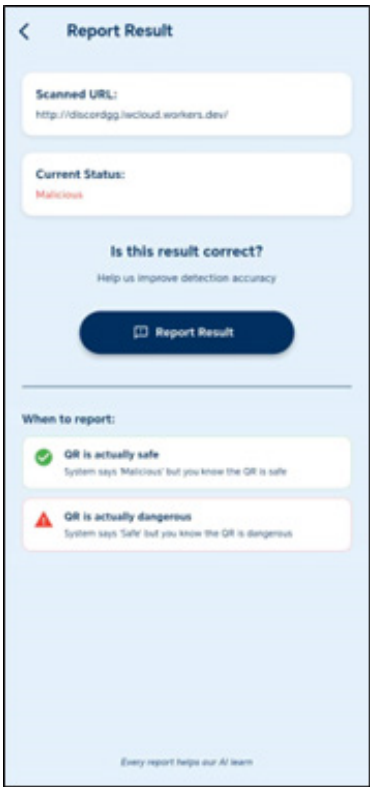

Figure 6 – Malicious URL warning and access restriction

Functional testing was conducted after deployment to determine whether the integrated application could complete the detection workflow using QR codes from both classes. Six representative QR codes were selected, comprising three safe and three malicious samples. The safe examples represented the Federal Reserve, Expedia, and MyFitnessPal domains, while the malicious examples were drawn from malicious samples used during system evaluation. Each sample passed through QR acquisition, decoding, URL processing, feature extraction, classification, and presentation of the security result. QR codes were tested through the supported input mechanisms, including image upload and camera-based scanning. The predicted class was then compared with the known class of each sample. This procedure was intended as functional deployment testing rather than an independent measurement of classification accuracy. Quantitative model performance was evaluated separately using the test dataset mentioned above.

Results

The four classifiers produced different performance profiles across the evaluation measures. Table 3 presents the results obtained from the test dataset.

Table 3 – Classification performance of the machine-learning models

Model	Accuracy (%)	Precision	Recall	F1-score
XGBoost	84	0.80	0.89	0.84
Random Forest	81	0.75	0.95	0.84
Decision Tree	79	0.73	0.92	0.82
Logistic Regression	71	0.65	0.93	0.76

XGBoost achieved the highest classification accuracy at 84%, followed by Random Forest at 81%, Decision Tree at 79%, and Logistic Regression at 71%. XGBoost also achieved the highest precision, 0.80, indicating it produced the highest proportion of correct malicious classifications among instances predicted as malicious. The ranking differed for recall. Random Forest achieved the highest recall at 0.95, followed by Logistic Regression at 0.93 and Decision Tree at 0.92. XGBoost recorded a recall of 0.89. Random Forest therefore detected the largest proportion of malicious URLs within the test set. XGBoost and Random Forest both achieved the highest F1-score of 0.84. Decision Tree recorded an F1-score of 0.82, while Logistic Regression obtained 0.76. The equal F1-scores of XGBoost and Random Forest resulted from different precision-recall profiles: XGBoost produced higher precision, whereas Random Forest produced higher recall. Considering the four measures jointly, XGBoost provided the strongest overall balance for deployment. It achieved the highest accuracy and precision and shared the highest F1-score with Random Forest. Random Forest was superior when recall alone was considered. The selection of XGBoost was therefore based on its overall performance profile rather than on superiority across all evaluation measures.

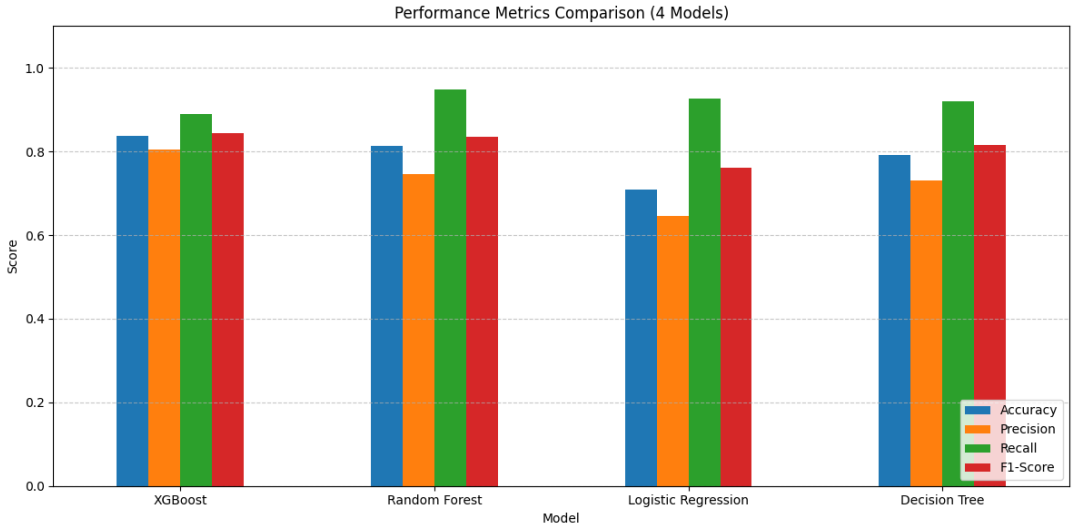

Figure 7 – Comparative performance of XGBoost, Random Forest, Decision Tree, and Logistic Regression

The selected XGBoost classifier was successfully integrated with the Flutter frontend and Flask backend. The deployed application completed the processing sequence from QR code acquisition to URL classification without requiring users to access the decoded destination before security assessment. Both supported input modes were operational. Static QR code images could be uploaded from the device, while the camera interface allowed direct QR code scanning. Following decoding, the URL was submitted to the backend, processed, transformed into the required feature representation, and classified. When the backend returned a malicious classification, the application displayed a security warning and restricted direct access to the detected URL. The classifier therefore operated within the QR code interaction rather than as an independent machine-learning component. The implementation demonstrates that QR acquisition, decoding, URL processing, feature extraction, classification, and user notification can operate as a connected workflow within the deployed application. The deployed system was tested using six representative QR codes comprising three safe and three malicious samples. Table 4 presents the known class and the classification returned by aQRace.

Table 4 – Functional testing of representative QR code samples

Sample	Known class	aQRace classification	Outcome
Safe QR 1	Safe	Safe	Correct
Safe QR 2	Safe	Safe	Correct
Safe QR 3	Safe	Safe	Correct
Malicious QR 1	Malicious	Malicious	Correct
Malicious QR 2	Malicious	Malicious	Correct
Malicious QR 3	Malicious	Malicious	Correct

All three safe samples were classified as safe, while the three malicious samples were classified as malicious. The classification outcomes were therefore consistent with the known labels of the six samples. The test also confirmed that the complete processing pipeline operated after cloud deployment. The application acquired and decoded each QR code, processed the resulting URL, generated the required feature representation, obtained a classification from the deployed model, and returned the corresponding result to the user interface. These six cases were used to verify system functionality and should not be interpreted as an independent estimate of classification performance. The quantitative performance of the classifier is represented by the test-set results reported earlier in this section.

Discussion

The findings demonstrate the feasibility of representing QR code destinations using URL-derived characteristics prior to destination access. The feature representation combines structural, linguistic, technical, and advanced information and can be generated after a QR code is decoded, without requiring the destination webpage to be opened. This approach is consistent with previous research showing that lexical and host-based URL characteristics can aid in detecting malicious QR codes. Its practical relevance lies in the placement of classification between QR decoding and destination access. Instead of requiring users to manually verify the legitimacy of an unfamiliar decoded URL, the URL is used as input for an automated security assessment. The present findings should not, however, be interpreted as evidence that every extracted characteristic contributes equally to classification. The experiment evaluates the feature set as a whole and does not measure the independent predictive contribution of individual characteristics. Feature-importance analysis, ablation testing, or explainability techniques would be necessary to establish which characteristics contribute most strongly to classification. The model comparison demonstrates that classifier selection affects the balance between accuracy, precision, and recall. XGBoost achieved the highest accuracy of 84% and precision of 0.80, whereas Random Forest achieved the highest recall of 0.95. Both obtained an F1-score of 0.84.

This distinction is particularly important in cybersecurity. A false negative may expose a user to a malicious destination because the system fails to issue a warning. From this perspective, Random Forest's higher recall is advantageous. Conversely, excessive false-positive warnings may reduce usability and eventually encourage users to disregard security alerts. XGBoost's higher precision provides an advantage in this respect. The equal F1-scores further illustrate this trade-off. XGBoost and Random Forest achieved the same F1 score via different precision-recall profiles. XGBoost was selected for aQRace because it achieved higher accuracy and precision while retaining a recall of 0.89. This result does not establish XGBoost as universally superior. A deployment that places substantially greater cost on missed malicious URLs could reasonably favour Random Forest.The stronger performance of the tree-based models relative to Logistic Regression also suggests that relationships among the extracted characteristics may not be adequately represented through a simple linear decision boundary. XGBoost, Random Forest, and Decision Tree achieved accuracies of 84%, 81%, and 79%, respectively, compared with 71% for Logistic Regression. Tree-based methods can capture non-linear interactions among structural, linguistic, and technical characteristics without explicitly specifying those relationships. The results broadly correspond to those of Herlina and

Soeparno [7], who also identified XGBoost as their highest-accuracy classifier, reporting 92%. The lower 84% accuracy obtained in the present study should not be interpreted as a direct performance deficit, as the studies use different datasets, preprocessing procedures, feature representations, and experimental settings. Classification results obtained under different evaluation conditions cannot be compared as if they represent performance on the same task and data.

Recent deep-learning research reports substantially higher predictive performance. Alsulami et al. [10], for example, achieved accuracies of 99.81% and 99.59% using a GRU-based architecture with AlexNet-derived QR image features reduced via PCA. Their study also demonstrated five-fold cross-validation, an average computation time of 7.433 ms per sample, and a real-time prototype. The two approaches nevertheless represent different design strategies. Alsulami et al. [10] derive high-dimensional representations from QR code images before dimensionality reduction and recurrent classification. aQRace decodes the QR code and performs classification based on the destination URL's characteristics. Direct comparison of the reported accuracy values would therefore be inappropriate. The contribution of aQRace is better understood as a URL-based approach that can be integrated into a pre-access scanning workflow, rather than as an attempt to outperform recent deep-learning architectures solely on predictive accuracy. The implementation results address an important distinction between machine-learning classification and operational QR code security. Dataset evaluation establishes predictive performance, but it does not determine whether a model can function effectively when users encounter a potential threat. In aQRace, classification occurs after QR acquisition and decoding but before destination access. The decoded URL is processed, converted into the required feature representation, classified, and returned to the frontend. When the classification is malicious, the application presents a warning and restricts direct access.

This architecture extends the classifier from an experimental model to a security component in user interaction. The approach is consistent with earlier work emphasising proactive warnings and real-time detection. Model architecture should nevertheless be considered in relation to the application's operational requirements rather than to predictive performance alone. Contemporary AI systems increasingly employ specialised architectures and representations to address the requirements of particular deployment contexts. For example, Fauzi et al. [12] employed a hybrid GCN-LSTM architecture for human-pose-based fall detection in an elderly monitoring system designed around privacy-preserving pose representations. This illustrates how model design can be shaped not only by classification performance but also by the characteristics of the data and operational requirements of the intended system. In QR code security, Tian and Zhu [11] similarly demonstrate a more sophisticated design strategy by formulating QR phishing detection as a URL semantic-classification problem and combining BERT, CNN, and BiLSTM representations with incremental learning. Their approach is designed to adapt to emerging phishing patterns while reducing the need for repeated full retraining. Compared with these architectures, aQRace adopts a more straightforward conventional machine-learning strategy centred on URL characteristics. This simplicity should not be interpreted as evidence of superior computational efficiency because processing time and resource consumption were not experimentally compared. It does, however, demonstrate that pre-access QR security can be implemented without requiring a multimodal or deep-learning architecture.

The aQRace workflow also incorporates rule-based URL processing, including normalisation, shortened-link expansion, and trusted-domain checking, around the machine-learning component. The classifier should therefore be regarded as one component of the deployed security workflow rather than the complete security mechanism. Functional testing confirmed the operation of this integrated process. However, the correct classification of six representative samples should not be described as 100% system accuracy. These samples establish only functional operation; quantitative performance evidence remains the test-set evaluation reported in the Results section. Two practical implications emerge from the findings. First, conventional machine-learning methods remain relevant to QR code security despite advances in deep and multimodal learning. When the principal security concern is the destination encoded in a QR code, URL-based classification enables analysis to focus directly on the content characteristics associated with the potential threat. Second, model selection should reflect the intended security objective. The highest-accuracy model is not automatically the

best classifier for every environment. Random Forest may be preferable when minimising missed malicious URLs is the dominant requirement because it achieved a recall of 0.95. XGBoost offers a different trade-off through higher accuracy and precision. This distinction is important for user-facing security applications. Frequent incorrect warnings may reduce user confidence, whereas undetected malicious destinations undermine the system's protective function. The balance between precision and recall is consequently both a machine-learning consideration and a system-design decision. The broader development of AI classification systems also indicates that increasingly complex models are most useful when their complexity addresses a specific characteristic of the problem. Multimodal fusion, for example, can be advantageous when complementary information is distributed across heterogeneous inputs, as demonstrated by Furqan, Katuk, and Hartama [9]. Likewise, hybrid spatial-temporal architectures can be appropriate when the underlying problem involves relationships across both space and time, as illustrated by Fauzi et al. [12]. The design of aQRace follows the same principle at a different level of complexity: its model and feature representation are centred specifically on characteristics of the decoded URL because the destination constitutes the primary object of security assessment.

Several limitations should be considered when interpreting the findings. First, the final models were developed using a balanced dataset of 14,000 URLs. Equal representation facilitates model comparison, but the proportion of safe and malicious URLs encountered in actual use is unlikely to be equal. Future studies should therefore evaluate the classifiers on independent, naturally imbalanced datasets. Second, the study relies on a predefined set of URL features. The experiment does not determine the independent contribution of each characteristic. Feature-importance analysis, SHAP-based explanations, and ablation experiments could identify the most informative characteristics and determine whether redundant features can be removed. Third, the study compares four conventional machine-learning classifiers but does not experimentally compare them with deep-learning architectures on the same dataset. Recent approaches have reported substantially higher accuracy under their respective experimental conditions. A controlled comparison using identical training and testing data would provide stronger evidence regarding the relative predictive, computational, and deployment characteristics of conventional and deep learning approaches. Fourth, deployment testing involved only six representative QR codes. This confirms system operation but does not constitute large-scale real-world validation. Future testing should include substantially larger collections of previously unseen QR codes, shortened URLs, redirection chains, obfuscated domains, homograph attacks, and QR codes captured under varying environmental conditions.

Finally, response time, resource consumption, network latency, scalability, usability, and warning effectiveness were not experimentally evaluated. These factors are important for a real-time user-facing security application. Future evaluation should therefore extend beyond classification metrics to end-to-end processing time, resource utilisation, user experience, and behavioural responses to security warnings. Future development could also examine ensemble approaches that combine the precision of XGBoost with the recall of Random Forest. Explainable machine learning could be incorporated to make warning decisions more transparent to users and system administrators. Continual model updating and richer semantic representations may further improve adaptation to evolving malicious URLs. More advanced multimodal fusion strategies, similar in principle to those used in attention-based classification systems, could also be investigated by combining URL characteristics with QR-image or contextual information.

Conclusion

This study developed and evaluated aQRace, a machine-learning-based QR code security application designed to analyse embedded URLs before users access their destinations. The research addressed three interrelated aspects of malicious QR code detection: URL feature representation, comparative machine-learning classification, and operational integration of the selected classifier. A URL-based feature representation that encompasses structural, linguistic, technical, and advanced characteristics was used to classify destinations as safe or malicious. Four classifiers were evaluated

using a balanced dataset of 14,000 URLs. XGBoost achieved the highest accuracy of 84% and precision of 0.80, whereas Random Forest achieved the highest recall of 0.95. Both models achieved an F1 Score of 0.84. The results demonstrate that model selection involves a meaningful trade-off between precision and recall rather than a single universally superior performance measure.

XGBoost was selected for deployment based on its overall performance profile and integrated into a client-server architecture comprising a Flutter frontend and Flask backend. The application supports QR code image uploads and camera-based scanning, and performs URL processing, feature extraction, classification, and user notification before access to the destination. Functional testing with representative safe and malicious QR codes confirmed that the complete workflow operated after deployment. The principal contribution of the study therefore extends beyond classifier comparison. It demonstrates how URL-based machine learning can be incorporated directly into the QR code scanning process to provide a pre-access security assessment. The approach offers a conventional machine-learning alternative to increasingly sophisticated image-based and multimodal deep-learning architectures, without claiming superiority over them in predictive accuracy or computational efficiency. The findings should nevertheless be interpreted within the study's limitations. The models were developed using a balanced dataset, the predictive contribution of individual URL features was not examined, and functional deployment testing was limited to six representative QR codes. Larger independent evaluations, feature-importance analysis, real-world testing, performance benchmarking, and controlled comparison with deep-learning approaches are therefore required. Overall, aQRace demonstrates the feasibility of incorporating malicious URL classification into an operational QR code scanning workflow. By placing security analysis between QR code acquisition and destination access, the approach provides a foundation for further development of QR code scanning systems that can warn users before interacting with potentially malicious destinations.

REFERENCES

1 Alsawi, W., Ibrahim, D. M. QR Code-Based Access Control Systems: Architectural Taxonomy, Security Landscape, and Future Research Directions. International Journal of Advanced Computer Science and Applications, 17(4), (2026).

2 Njuguna, D., Ndia, J. Quick Response Code Security Attacks and Countermeasures: A Systematic Literature Review. Journal of Cybersecurity (2579-0072), 7(1), 1 (2025). https://doi.org/10.32604/jcs.2025.059398

3 Trad, F., Chehab, A. Detecting quishing attacks with machine learning techniques through qr code analysis. IFIP International Conference on Artificial Intelligence Applications and Innovations, 194–206 (2026). https://doi.org/10.1007/978-3-032-30805-4_14

4 Vaithilingam, S., Shankar, S. a. M. Enhancing security in QR code technology using AI: Exploration and mitigation strategies. International Journal of Intelligence Science, 14(02), 49–57 (2024). https://doi.org/10.4236/ijis.2024.142003

5 Al-Zahrani, M. S., Wahsheh, H. A., Alsaade, F. W. Secure Real-Time Artificial Intelligence System against Malicious QR Code Links. Security and Communication Networks, 2021(1), 5540670 (2021). https://doi.org/10.1155/2021/5540670

6 Sarkhi, M., Mishra, S. Detection of QR code-based cyberattacks using a lightweight deep learning model. Engineering, Technology & Applied Science Research, 14(4), 15209–15216 (2024). https://www.etasr.com/index.php/ETASR/article/view/7777/3803

7 Herlina, B., Soeparno, H. Machine learning model to improve classification performance in the process of detecting phishing URLs in QR codes. Journal of Theoretical and Applied Information Technology, 101(18), 5755–5765 (2023). http://www.jatit.org/volumes/Vol101No18/27Vol101No18.pdf

8 Pawar, A., Fatnani, C., Sonavane, R., Waghmare, R., Saoji, S. Secure qr code scanner to detect malicious url using machine learning. 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), 1-8 (2022). https://doi.org/10.1109/ASIANCON55314.2022.9908759

9 Furqan, M., Katuk, N., Hartama, D. Multiclass Skin Lesion Classification Algorithm using Attention-Based Vision Transformer with Metadata Fusion. Journal of Applied Data Sciences, 7(1), 203–217 (2026). https://doi.org/10.47738/jads.v7i1.1017

10 Alsulami, A.-A., Al-Haija, Q.-A., Alturki, B., Yafoz, A., Alqahtani, A., Alsini, R., Binyamin, S.-S. Efficient Malicious QR Code Detection System Using an Advanced Deep Learning Approach. Computer Modeling in Engineering & Sciences, 145(1), 1117–1140 (2025). https://doi.org/10.32604/cmes.2025.070745

11 Tian, Y., Zhu, E. Incremental Phishing Defense Method Based on QR Code Decoding and Multimodal Deep Learning. Proceedings of the 2026 5th International Conference on Cryptography, Network Security and Communication Technology, 209–214 (2026). https://doi.org/10.1145/3802927.3802959

12 Fauzi, M. A., Yang, B., Hayati, Y. S., Setiawan, B. D., Sari, I. N., Bayona, J., Katuk, N., Dewi, D. A., Surasak, T. Hybrid GCN-LSTM for Privacy-Preserving Fall Detection in Human Pose-Based Elderly Monitoring Systems. Proceedings of the 14th International Conference on Advances in Information Technology, 1–8 (2026). https://doi.org/10.1145/3816713.3819508

[1]*Норлиза Катук,
PhD, ORCID ID: 0000-0001-8805-2574,
*e-mail: k.norliza@uum.edu.my
[1]Хикма Нурдин,
бакалавр, ORCID ID: 0009-0001-3766-5482,
e-mail: hikmahnurdin750@gmail.com
[2]М. Али Фаузи,
PhD, ORCID ID: 0000-0002-9696-2807,
e-mail: moch.ali.fauzi@ub.ac.id
[3]Мхд. Фурқан,
PhD, ORCID ID: 0000-0002-7255-4454,
e-mail: mfurqan@uinsu.ac.id
[1]Есептеу техникасы мектебі,
Солтүстік Малайзия университеті, Малайзия
[2]Компьютерлік ғылымдар факультеті,
Бравиджая университеті, Индонезия
[3]Ғылым және технология факультеті,
Солтүстік Суматра мемлекеттік ислам университеті, Индонезия

AQRACE: QR-КОДТАРДАҒЫ ЗИЯНДЫ URL-МЕКЕНЖАЙЛАРДЫ НАҚТЫ УАҚЫТ РЕЖИМІНДЕ ҚАУІПСІЗДІКТІ БАҒАЛАУ ҮШІН МАШИНАЛЫҚ ОҚЫТУ НЕГІЗІНДЕ АНЫҚТАУ

Аңдатпа

Жылдам жауап беру (Quick Response, QR) кодтарының кеңінен қолданылуы QR-кодтарға негізделген фишинг, зиянды қайта бағыттау және зиянды бағдарламаларды тарату қаупін арттырды, өйткені пайдаланушылар QR-кодты сканерлеу алдында оның бағыттайтын мекенжайын тікелей тексере алмайды. Осы зерттеуде пайдаланушы URL-мекенжайға өтпес бұрын QR-кодқа енгізілген URL-мекенжайларды талдайтын, машиналық оқытуға негізделген QR-код қауіпсіздігіне арналған aQRace қосымшасы әзірленді. Зерттеу URL-мекенжай белгілерін бөліп алуды, машиналық оқыту модельдерін салыстырмалы бағалауды және оларды жұмыс істейтін жүйеге біріктіруді қамтиды. Жіктеу үшін URL-мекенжайлардың құрылымдық, лингвистикалық, техникалық және кеңейтілген сипаттамалары алынды. Бақыланатын машиналық оқытудың төрт алгоритмі – XGBoost, Random Forest, Decision Tree және Logistic Regression – 7000 қауіпсіз және 7000 зиянды үлгіден тұратын, жалпы саны 14 000 URL-мекенжайды қамтитын теңгерімді деректер жиынтығында оқытылып, бағаланды. Модельдердің өнімділігі accuracy, precision, recall және F1-score көрсеткіштері бойынша салыстырылды. XGBoost ең жоғары 84% accuracy және 0,80 precision мәндерін көрсетті, ал Random Forest ең жоғары 0,95 recall мәніне қол жеткізді. Екі модель де 0,84 F1-score нәтижесін көрсетті. Жалпы өнімділік көрсеткіштерінің үйлесімділігіне байланысты XGBoost жүйеге енгізу үшін таңдалып, Flutter негізіндегі мобильді интерфейс пен Flask негізіндегі бұлттық серверлік бөліктен тұратын клиент-серверлік қосымшаға біріктірілді. Қосымша QR-код кескіндерін жүктеуді және камера арқылы сканерлеуді қолдайды, сондай-ақ пайдаланушы мақсатты мекенжайға өтпес бұрын URL-мекенжайды өңдеу, белгілерді бөліп алу, жіктеу және қауіп туралы ескерту функцияларын орындайды. Қауіпсіз және зиянды QR-кодтардың үлгілік жиынтығымен жүргізілген функционалдық тестілеу анықтау үдерісінің толық жұмыс істейтінін растады. Нәтижелер URL-мекенжай белгілеріне негізделген дәстүрлі машиналық оқыту әдістерінің пайдаланушы

мақсатты мекенжайға өтпес бұрын QR-код қауіпсіздігін бағалауға мүмкіндік беретінін және күрделілігі артып келе жатқан терең оқыту тәсілдеріне практикалық балама ұсына алатынын көрсетті.

Түйін сөздер: QR-код қауіпсіздігі, quishing, зиянды URL-мекенжайларды анықтау, машиналық оқыту, XGBoost, фишингті анықтау, киберқауіпсіздік.

[1]*Норлиза Катук,
PhD, ORCID ID: 0000-0001-8805-2574,
*e-mail: k.norliza@uum.edu.my
[1]Хикма Нурдин,
бакалавр, ORCID ID: 0009-0001-3766-5482,
e-mail: hikmahnurdin750@gmail.com
[2]М. Али Фаузи,
PhD, ORCID ID: 0000-0002-9696-2807,
e-mail: moch.ali.fauzi@ub.ac.id
[3]Мхд. Фуркан,
PhD, ORCID ID: 0000-0002-7255-4454,
e-mail: mfurqan@uinsu.ac.id
[1]Школа вычислительной техники,
Университет Утара Малайзия, Малайзия
[2]Факультет компьютерных наук,
Университет Бравиджая, Индонезия
[3]Факультет науки и технологий,
Государственный исламский университет Северной Суматры, Индонезия

AQRACE: ОБНАРУЖЕНИЕ ВРЕДОНОСНЫХ URL-АДРЕСОВ В QR-КОДАХ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ОЦЕНКИ БЕЗОПАСНОСТИ В РЕАЛЬНОМ ВРЕМЕНИ

Аннотация

Широкое распространение кодов быстрого реагирования (Quick Response, QR) увеличило риск QR-фишинга, вредоносной переадресации и распространения вредоносного программного обеспечения, поскольку пользователи не могут непосредственно проверить адрес назначения до сканирования QR-кода. В данном исследовании разработано приложение aQRace для обеспечения безопасности QR-кодов на основе машинного обучения, которое анализирует встроенные URL-адреса до перехода пользователя по соответствующей ссылке. Исследование объединяет извлечение признаков URL-адресов, сравнительную оценку моделей машинного обучения и их интеграцию в функционирующую систему. Для классификации были извлечены структурные, лингвистические, технические и расширенные характеристики URL-адресов. Четыре алгоритма машинного обучения с учителем – XGBoost, Random Forest, Decision Tree и Logistic Regression – были обучены и оценены на сбалансированном наборе данных, содержащем 14 000 URL-адресов, из которых 7000 являлись безопасными и 7000 – вредоносными. Производительность моделей сравнивалась по показателям accuracy, precision, recall и F1-score. XGBoost продемонстрировал наивысшую accuracy, равную 84%, и precision 0,80, тогда как Random Forest показал наивысший recall, равный 0,95. Обе модели достигли значения F1-score 0,84. На основе совокупности показателей производительности XGBoost был выбран для внедрения и интегрирован в клиент-серверное приложение, состоящее из мобильного интерфейса на Flutter и облачной серверной части на Flask. Приложение поддерживает загрузку изображений QR-кодов и сканирование с помощью камеры, а также выполняет обработку URL-адреса, извлечение признаков, классификацию и предупреждение пользователя до перехода по целевому адресу. Функциональное тестирование на репрезентативных безопасных и вредоносных QR-кодах подтвердило работоспособность полного процесса обнаружения угроз. Полученные результаты показывают, что традиционные методы машинного обучения на основе признаков URL-адресов могут использоваться для оценки безопасности QR-кодов до перехода пользователя по ссылке, обеспечивая практическую альтернативу все более сложным подходам на основе глубокого обучения.

Ключевые слова: безопасность QR-кодов, quishing, обнаружение вредоносных URL-адресов, машинное обучение, XGBoost, обнаружение фишинга, кибербезопасность.