

Detecting Quishing Attacks with Machine Learning Techniques Through QR Code Analysis*

Fouad Trad and Ali Chehab

Electrical and Computer Engineering, American University of Beirut, Beirut,
Lebanon
fat10@mail.aub.edu, chehab@aub.edu.lb

Abstract. The rise of QR code-based phishing (“Quishing”) poses a growing cybersecurity threat, as attackers increasingly exploit QR codes to bypass traditional phishing defenses. Existing detection methods predominantly focus on URL analysis, which requires the extraction of the QR code payload, and may inadvertently expose users to malicious content. Moreover, QR codes can encode various types of data beyond URLs, such as Wi-Fi credentials and payment information, making URL-based detection insufficient for broader security concerns. To address these gaps, we propose the first framework for quishing detection that directly analyzes QR code structure and pixel patterns without extracting the embedded content. We generated a dataset of phishing and benign QR codes and we used it to train and evaluate multiple machine learning models, including Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, LightGBM, and XGBoost. Our best-performing model (XGBoost) achieves an AUC of 0.9106, demonstrating the feasibility of QR-centric detection. Through feature importance analysis, we identify key visual indicators of malicious intent and refine our feature set by removing non-informative pixels, improving performance to an AUC of 0.9133 with a reduced feature space. Our findings reveal that the structural features of QR code correlate strongly with phishing risk. This work establishes a foundation for quishing mitigation and highlights the potential of direct QR analysis as a critical layer in modern phishing defenses.

Keywords: Quishing Detection · QR code analysis · Machine Learning · Feature Selection

1 Introduction

QR codes have become an integral part of modern digital interactions, facilitating seamless access to websites, payments, authentication systems, and other online services. However, their widespread adoption has also given rise to security threats, particularly in the form of QR code-based phishing attacks, commonly

* Supported by the Maroun Semaan Faculty of Engineering (MSFEA) at the American University of Beirut (AUB)

referred to as Quishing [9]. In quishing attacks, cyber criminals exploit QR codes to deceive users into scanning them, often leading to credential theft, malware downloads, or financial fraud [14]. Unlike traditional phishing attacks that rely on deceptive emails or messages with visible URLs, quishing leverages the opaque nature of QR codes, making it difficult for users to assess their legitimacy before scanning [16].

While URLs are the most common payload in quishing attacks, QR codes can encode a variety of data types beyond web links, broadening the attack surface. They can be used to store Wi-Fi credentials, trigger app deep links, initiate cryptocurrency transactions, add contact details, share geolocation data, send SMS messages, schedule calendar events, or even display plaintext phishing messages. This versatility allows attackers to craft social engineering tactics that do not rely solely on malicious URLs, further complicating detection efforts.

Existing phishing detection methods focus primarily on analyzing URLs and website content, often requiring the resolution of the QR code payload. This approach presents a critical security risk, as accessing the embedded content could expose users or automated security systems to malicious actions before a threat is identified. In addition, obfuscation techniques such as URL shorteners, redirections, and encoded payloads further complicate the detection [5]. Moreover, these techniques are limited to QR codes that embed URLs and do not generalize to QR codes containing other types of data. These challenges highlight the need for an alternative approach that can assess QR code security without resolving its payload.

In this study, we propose the first machine learning framework for quishing detection that directly analyzes QR code structure and pixel patterns without relying on URL extraction. Instead of treating QR codes solely as carriers of encoded text, our method leverages their visual and structural properties to distinguish between benign and malicious codes. As such, we created a dataset of QR codes labeled as phishing or benign, and evaluated multiple machine learning models, including Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, LightGBM, and XGBoost. Our results demonstrate that QR-centric detection is not only feasible but also highly effective, with our best-performing model (XGBoost) achieving an AUC of 0.9106.

Furthermore, we conducted a feature importance analysis to identify key visual indicators of malicious intent. By refining our feature set and excluding non-informative pixels, we further improved detection performance to an AUC of 0.9133.

This work establishes a foundation for pre-scan quishing mitigation, demonstrating the viability of direct QR code analysis as a proactive cybersecurity measure. By eliminating the need to extract or resolve QR payloads, whether they contain URLs or other data types, our approach strengthens phishing defense strategies and improves security in a world where QR codes are widely used.

In summary, the main contributions of this paper are as follows:

- Proposing the first machine learning framework for detecting QR code-based phishing attacks (*quishing*) by directly analyzing QR code structure and pixel patterns without extracting or resolving their embedded content.
- Creating a dataset of QR codes (available on GitHub) labeled as phishing or benign, enabling the evaluation of QR-centric detection models.
- Evaluating multiple machine learning models, including Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, LightGBM, and XGBoost, to assess their performance in quishing detection.
- Conducting feature importance analysis on raw QR code pixels to identify critical visual indicators of malicious intent and reducing the feature set to improve performance.

The structure of this paper is as follows: Section 2 provides a comprehensive review of the existing literature on phishing and quishing detection. Section 3 outlines the experimental setup used to create the Quishing dataset, detailing the machine learning models developed, the evaluation metrics employed, and the rationale behind the chosen methodologies. Section 4 presents the experiments conducted, followed by a discussion of the results. Finally, Section 5 concludes the paper with a summary of our findings, contributions to the field, and potential avenues for future research.

2 Related Work

Phishing detection has been a critical area of research in cybersecurity, focusing predominantly on identifying malicious URLs through various analytical and machine learning techniques. Traditional approaches to phishing detection have explored features such as URL lexical characteristics, website content analysis, and the use of third-party services like blacklists and WHOIS information to classify URLs as phishing or legitimate. Multiple studies have used machine learning algorithms to analyze URL features, demonstrating significant success in identifying phishing attempts [7,1,4].

Further advances have included the implementation of deep learning models, to analyze the textual features of URLs for improved phishing detection accuracy [2,15,10]. These techniques have proven effective in identifying malicious URLs by examining their lexical and host-based features and page content to discern patterns indicative of phishing. Very recently, large language models (LLMs) have been explored for this task and are achieving state-of-the-art performance [12,13,11].

Despite these advances in phishing detection, the domain of QR code-based phishing, or "quishing," remains largely unexplored. QR codes present unique challenges, as they can encode diverse types of information, allowing attackers to bypass traditional detection mechanisms by embedding malicious content in various forms. Sharevski et al. were among the first to investigate quishing, which gained traction during the COVID-19 pandemic [9]. Using a deceptive sign-up for a COVID-19 digital passport, they found that a majority of the 173 participants (67%) were willing to use their social media credentials for convenience, despite

the risks. The study highlighted the threat of quishing, but focused primarily on raising awareness and proposing educational guidelines rather than developing direct countermeasures to mitigate the risk. A study by Amoah and J.B. [3] represents one of the few attempts to address quishing. In their approach, QR codes are first decoded and converted into URLs, which are then subjected to conventional phishing detection methods. Similarly, Rafsanjani et al. introduced QsecR, an Android application that extracts the URL from the QRCode and analyzes it by extracting some static URL features [8]. These approaches, while innovative, introduce inherent risks by requiring URL resolution, potentially exposing users to malicious content, and do not exploit the full potential of direct QR code analysis for phishing detection since they are only limited to URLs.

Our work addresses this gap by introducing a novel machine learning framework that directly analyzes the visual and structural patterns within QR codes, bypassing the intermediary step of payload extraction. By focusing on QR code analysis, we can identify indicators of phishing risk regardless of the type of data encoded. This method provides a proactive pre-scan defense mechanism that automatically assesses the safety of QR codes before they are scanned, thereby enhancing digital security in an era where QR codes serve a multitude of functions.

3 Experimental Setup

3.1 Dataset

To our knowledge, no publicly available Quishing dataset currently exists. Therefore, a key contribution of this study is the creation of a Quishing dataset, which will be made publicly available to researchers. While QR codes can encode a wide range of payloads, such as Wi-Fi credentials or app links, our dataset focuses specifically on URL-derived QR codes. This focus is motivated by the availability of verified labels (phishing/benign) for URLs in existing phishing datasets, which allows for reliable benchmarking. In contrast, no equivalent datasets currently exist for non-URL QR payloads (e.g., malicious Wi-Fi configurations), as manual verification would require physical device interaction or proprietary threat intelligence. This dataset represents a practical starting point for QR code-based phishing detection, providing a foundation for future work that could extend this approach to other types of payloads. The proposed method is expected to generalize well to other payload types, as it is based on the direct analysis of QR code structure rather than the specific content it encodes.

The dataset we created is derived from the PhishStorm dataset [6] by selecting 10,000 samples, equally categorized into 5,000 legitimate and 5,000 phishing URLs. We then used the 'qrcode' Python library to generate QR codes corresponding to the URLs.

When generating QR codes, we can specify several parameters including:

- **Version:** An integer from 1 to 40, specifying the number of elements in the QR Code.

- **Error Correction:** A choice among 'high', 'medium', and 'low', indicating the degree of error we can tolerate in the QR Code while still navigating to the correct link without errors.
- **Box Size:** The size of an element in pixels (each cell in a QR Code represents a certain number of pixels).
- **Border:** The size of the borders.

After investigating the characteristics of our URL data and noting that not every URL can be encoded using any QR code version, we selected version 13 as our standard, which was able to encode most of the URLs (9,987 samples). This choice yields QR codes of a consistent size (69×69 pixels), which is advantageous for machine learning applications, as it allows for uniform pixel-based feature extraction and prediction. Moreover, using a single version simplifies the preprocessing pipeline and enhances the comparability of features across samples. The error correction level was set to 'low', which can tolerate up to 15% loss while still retrieving the URL correctly. Additionally, the box size was set to 1 (minimizing computational demands) and the border to 0 (eliminating extraneous pixels). Although future research may explore varying these parameters to create a more diversified dataset, this uniform approach serves as an effective starting point in our investigation. The created dataset can be accessed on GitHub through the following link: <https://github.com/fouadtrad/Detecting-Quishing-Attacks-with-Machine-Learning-Techniques-Through-QR-Code-Analysis>.

Figure 1 shows 10 samples of the generated QR Codes. The dataset was divided between 80% training and 20% testing. We later performed 10-fold cross-validation on the training set to tune the hyperparameters of the used models.

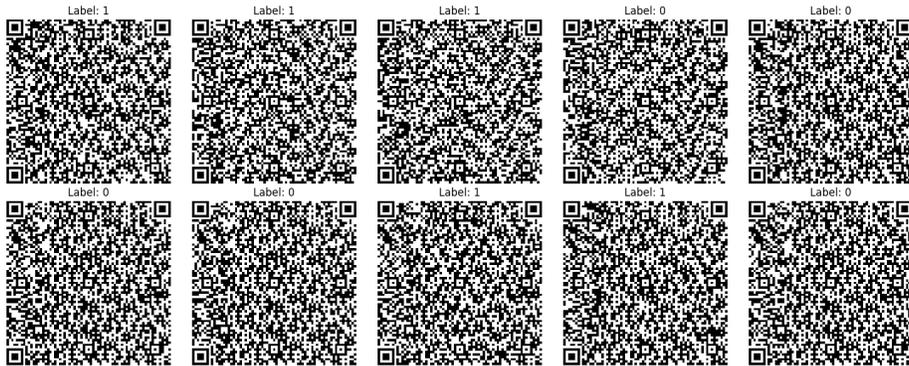


Fig. 1. Ten samples of the generated QR Codes

3.2 Models

We employ multiple machine learning models, including Logistic Regression, Decision Tree, Naïve Bayes, Random Forest, LightGBM, and XGBoost. Because

QR codes adhere to a standardized fixed structure, each QR code image can be reliably flattened into a uniform array, with every pixel treated as an individual feature. This consistency enables effective analysis and comparison across models in a QR code setting. Hyperparameter tuning was performed using a randomized search with 10-fold cross-validation on the training set. The optimal hyperparameters for each model are summarized in Table 1 to facilitate reproducibility.

Table 1. Hyperparameters of the chosen models

Model	Hyperparameters
Logistic Regression	'C': 0.1, 'solver': 'liblinear'
Decision Tree	'max_depth': 3, 'min_samples_leaf': 1
Random Forest	'max_depth': 20, 'n_estimators': 100
LightGBM	'learning_rate': 0.1, 'n_estimators': 200
XGBoost	'learning_rate': 0.2, 'n_estimators': 150

3.3 Metrics

The evaluation of our models relied on several classical classification metrics:

- **Accuracy:** This metric measures the overall proportion of correctly classified instances, providing a straightforward assessment of model performance.
- **Precision:** Precision quantifies the proportion of true positive predictions among all positive predictions, indicating the model’s ability to minimize false positives.
- **Recall:** Also known as sensitivity, recall measures the proportion of actual positives that are correctly identified by the model, reflecting its capability to detect positive cases.
- **F1-score:** As the harmonic mean of precision and recall, the F1-score offers a balanced evaluation of the model’s performance, particularly in scenarios where class distribution is uneven.
- **Area Under the ROC Curve (AUC):** The AUC metric evaluates the model’s overall ability to discriminate between classes across different threshold settings, providing a comprehensive measure of classification performance. It is worth noting that the previous metrics are reported based on the default classification threshold of 0.5.

4 Experiments and Results

4.1 Experiment 1: Training and testing the models

After identifying the optimal hyperparameters using 10-fold cross-validation, we trained each model on the full training set and evaluated their performance on the testing set. Table 2 presents the computed metrics for each model.

As shown in the results, most models perform well on this task, with the exception of Naïve Bayes, which achieved an AUC of 0.6531, significantly lower than the other models. All other models achieved an AUC above 0.81, demonstrating strong discriminatory ability. Among the tested models, XGBoost, LightGBM, and Random Forest classifiers exhibit the best performance, with AUC scores ranging from 0.8908 to 0.9106.

While AUC provides a threshold-independent measure of model performance, the accuracy and F1-score at the default 0.5 threshold offer additional insights. The LightGBM and XGBoost models achieve the highest accuracy, at 0.8293 and 0.8258, respectively, indicating strong overall classification performance. These models also maintain the highest F1-scores, at 0.8214 and 0.8184, balancing precision and recall effectively. However, it is important to note that classification metrics such as accuracy and F1-score depend on the chosen threshold. Given that AUC is relatively high for these models, adjusting the decision threshold could further optimize the trade-off between precision and recall, particularly in real-world scenarios where minimizing false positives or false negatives is a priority.

These results suggest that QR code-based phishing detection using direct QR analysis is feasible, with tree-based ensemble models such as LightGBM and XGBoost demonstrating the most promising performance. Further exploration of threshold tuning could refine the detection process based on specific application requirements.

Table 2. Performance metrics on the testing set

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.7983	0.8129	0.7621	0.7867	0.8737
Decision Tree Classifier	0.7578	0.7358	0.7856	0.7599	0.8138
Random Forest Classifier	0.7993	0.8570	0.7067	0.7746	0.8908
Gaussian NB	0.6376	0.8680	0.3036	0.4498	0.6531
XGBoost Classifier	0.8258	0.8332	0.8041	0.8184	0.9083
LGBM Classifier	0.8293	0.8394	0.8041	0.8214	0.9106

4.2 Experiment 2: Deriving Feature Importance

Based on the results of Experiment 1, we select the top three performing models (Random Forest, LightGBM, and XGBoost) and analyze their feature importance. To visualize the impact of individual pixels on the classification decision, we plot the feature importance values on a 69×69 grid, corresponding to the QR code size, as shown in Figure 2.

The results reveal that a significant portion of the QR code remains unused in the prediction process. Large black regions in the feature importance maps indicate that the majority of pixels contribute little to no information for distinguishing phishing from benign QR codes. This suggests that quishing detection

is primarily influenced by specific regions of the QR code rather than its full structure.

To further illustrate this observation, we provide two additional visualizations. Figure 3 highlights the pixels considered important by the XGBoost model, shown in gray, while Figure 4 highlights the excluded pixels, shown in yellow. These figures reinforce the finding that only a subset of pixels is relevant to the classification decision.

Additionally, Figure 5 presents the distribution of feature importance values for each of the three models. The distribution confirms that the majority of pixels have an importance value of around zero across all models, while only a small fraction contributes meaningfully to the decision-making process. This finding suggests that refining the feature space by focusing on the most relevant pixels could further enhance the model's efficiency and performance.

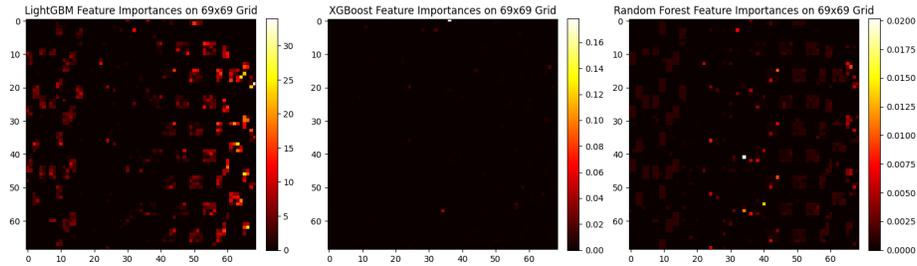


Fig. 2. Feature Importance of the top 3 models

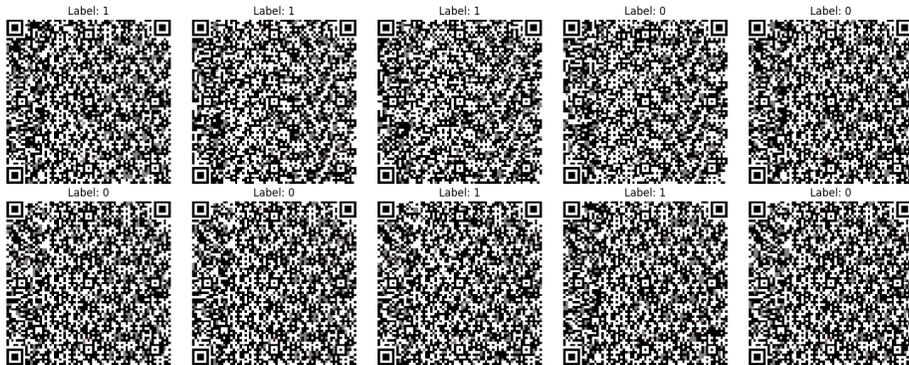


Fig. 3. Features taken into account when using XGBoost

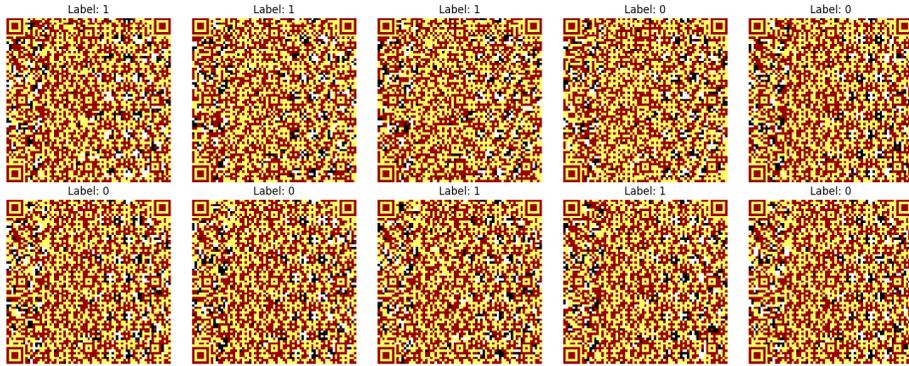


Fig. 4. Features not taken into account when using XGBoost

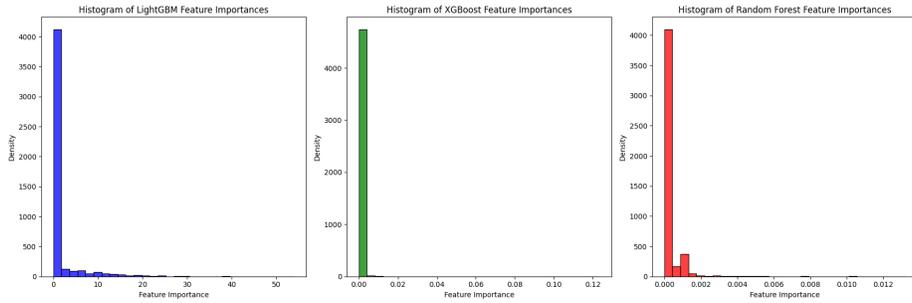


Fig. 5. Feature Importance distribution

4.3 Experiment 3: Feature Selection

In this experiment, we performed feature selection based on the most important features identified by each of the three best-performing models: Random Forest, LightGBM, and XGBoost. We then retrained all models using only these selected features and compared their performance to their original versions without feature selection. The results, presented in Table 3, show that applying feature selection consistently improves or maintains model performance. This suggests that many pixels in the original QR code images are non-informative and do not contribute to phishing detection.

Among the tested models, Decision Tree is the only one that exhibits no significant change in performance, which is expected given its inherent ability to focus on the most relevant features. The highest performance is achieved with LightGBM, reaching an AUC of 0.9133, further demonstrating the benefits of using a reduced but more informative feature set.

A particularly notable improvement is observed in Naïve Bayes, where AUC increases from 0.6531 (without feature selection) to as high as 0.8010 when using LightGBM-based feature selection. This suggests that Naïve Bayes, which

relies on strong independence assumptions, benefits significantly from a reduced feature set that removes irrelevant or noisy pixels. By limiting the input to only the most important features, the model is less affected by non-informative data, leading to better performance.

Additionally, the impact of different feature selection methods varies slightly across models. For instance, LightGBM achieves its highest AUC (0.9133) when feature selection is based on Random Forest, indicating that the features deemed important by Random Forest align well with LightGBM’s gradient boosting approach. This may be due to Random Forest’s ability to capture a diverse set of relevant features, which LightGBM can then refine further through its boosting mechanism. Similarly, XGBoost shows only slight variations across feature selection methods while consistently maintaining strong performance, reinforcing its robustness to different feature subsets. The relatively small differences in performance across feature selection strategies suggest that all three models are capturing meaningful predictors of phishing risk, further validating the effectiveness of QR code-based feature selection for this task.

Table 3. AUC Comparison for Each Model With and Without Feature Selection (FS)

Model	AUC (FS Based on XGBoost)	AUC (FS Based on LightGBM)	AUC (FS Based on Random Forest)	AUC (No FS)
Logistic Regression	0.8725	0.8720	0.8738	0.8737
Decision Tree	0.8138	0.8138	0.8138	0.8138
Random Forest	0.8984	0.8966	0.8923	0.8908
Naïve Bayes	0.7934	0.8010	0.7540	0.6531
XGBoost	0.9083	0.9104	0.9083	0.9083
LightGBM	0.9121	0.9106	0.9133	0.9106

5 Conclusion and Future Work

In conclusion, This study introduced a novel approach to QR code-based phishing (quishing) detection by directly analyzing QR code structure and pixel patterns without relying on URL extraction. Through machine learning models, trained on a newly created dataset, we demonstrate that QR-centric detection is both feasible and effective. Feature selection further improves or maintains model performance by identifying the most informative regions of the QR code while filtering out non-essential pixels.

While this work establishes an important foundation, there are several avenues for future research. One key direction is to extend the analysis beyond URL-encoded QR codes. Although URLs are the most common vector for quishing attacks, QR codes can also encode other forms of malicious payloads. Developing datasets that encompass these variations will enable a broader assessment

of QR code phishing risks and allow models to generalize to a wider range of attack strategies.

Another promising direction is the incorporation of deep learning techniques, particularly convolutional neural networks (CNNs) and vision transformers (ViTs), which could further enhance performance by capturing complex spatial patterns within QR codes. While shallow models performed well in this study, deep learning methods may better exploit subtle pixel-based variations indicative of phishing attempts.

Additionally, while this study operates under controlled conditions, practical implementation requires evaluating models on QR codes captured in varying environments, including different lighting conditions, distortions, and physical printouts. The robustness of QR-based phishing detection systems against adversarial attacks, such as perturbations that deceive machine learning models, is another critical research area.

Finally, integrating QR code phishing detection into real-world applications is an essential next step. This could involve developing mobile applications, browser extensions, or security software that can assess QR codes before they are scanned, providing users with real-time risk assessments.

By addressing these challenges, future research can further strengthen defenses against QR code phishing, ensuring that detection methods remain effective against evolving attack techniques. This study provides the groundwork for such advancements, demonstrating that direct QR analysis is a viable and valuable addition to modern phishing mitigation strategies.

References

1. Ahammad, S.H., Kale, S.D., Upadhye, G.D., Pande, S.D., Babu, E.V., Dhurane, A.V., Bahadur, M.D.K.J.: Phishing url detection using machine learning methods. *Advances in Engineering Software* **173**, 103288 (2022)
2. Aljabri, M., Mirza, S.: Phishing attacks detection using machine learning and deep learning models. In: *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*. pp. 175–180. IEEE (2022)
3. Amoah, G.A., Hayfron-Acquah, J.: Qr code security: mitigating the issue of quishing (qr code phishing). *International journal of computer applications* **184**(33), 34–39 (2022)
4. Hannousse, A., Yahiouche, S.: Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Engineering Applications of Artificial Intelligence* **104**, 104347 (2021)
5. Le Page, S., Jourdan, G.V., Bochmann, G.V., Flood, J., Onut, I.V.: Using url shorteners to compare phishing and malware attacks. In: *2018 APWG Symposium on Electronic Crime Research (eCrime)*. pp. 1–13. IEEE (2018)
6. Marchal, S., François, J., State, R., Engel, T.: Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management* **11**(4), 458–471 (2014)
7. Opara, C., Chen, Y., Wei, B.: Look before you leap: Detecting phishing web pages by exploiting raw url and html characteristics. *Expert Systems with Applications* **236**, 121183 (2024)

8. Rafsanjani, A.S., Kamaruddin, N.B., Rusli, H.M., Dabbagh, M.: Qsecr: Secure qr code scanner according to a novel malicious url detection framework. *IEEE Access* (2023)
9. Sharevski, F., Devine, A., Pieroni, E., Jachim, P.: Phishing with malicious qr codes. In: *Proceedings of the 2022 European Symposium on Usable Security*. pp. 160–171 (2022)
10. Tajaddodianfar, F., Stokes, J.W., Gururajan, A.: Texception: a character/word-level deep learning model for phishing url detection. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 2857–2861. IEEE (2020)
11. Trad, F., Chehab, A.: Large multimodal agents for accurate phishing detection with enhanced token optimization and cost reduction. In: *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*. pp. 229–237. IEEE (2024)
12. Trad, F., Chehab, A.: Prompt engineering or fine-tuning? a case study on phishing detection with large language models. *Machine Learning and Knowledge Extraction* **6**(1), 367–384 (2024)
13. Trad, F., Chehab, A.: To Ensemble or Not: Assessing Majority Voting Strategies for Phishing Detection with Large Language Models. In: *Intelligent Systems and Pattern Recognition*. pp. 158–173. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-82150-9_13
14. Vidas, T., Owusu, E., Wang, S., Zeng, C., Cranor, L.F., Christin, N.: Qrishing: The susceptibility of smartphone users to qr code phishing attacks. In: *Financial Cryptography and Data Security: FC 2013 Workshops, USEC and WAHC 2013, Okinawa, Japan, April 1, 2013, Revised Selected Papers 17*. pp. 52–69. Springer (2013)
15. Wang, Y., Zhu, W., Xu, H., Qin, Z., Ren, K., Ma, W.: A large-scale pretrained deep model for phishing url detection. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 1–5. IEEE (2023)
16. Yong, K.S., Chiew, K.L., Tan, C.L.: A survey of the qr code phishing: the current attacks and countermeasures. In: *2019 7th International Conference on Smart Computing & Communications (ICSCC)*. pp. 1–5. IEEE (2019)