Malicious Link Detection in QR Codes: A Comparison of Supervised and Unsupervised Machine Learning Algorithms

CSS 590 Final Project Report

Sav Wheeler and Andreas Hindman

***Abstract*—QR codes, an increasingly popular method of link-sharing for mobile environments, pose significant security threats such as phishing, malware distribution, and SQL injection attacks. Existing methods of detecting such threats like URL blacklists have proven to be inadequate due to their inability to detect emergent threats promptly. This study proposes a novel machine-learning-based solution: integrating a URL detection algorithm with an Android camera front-end, to identify malicious links in real-time. We compare supervised and unsupervised machine learning algorithms to determine the most effective approach for this application. Our findings suggest that machine learning can significantly enhance the detection and prevention of QR code-based threats, offering a robust defense against a rapidly evolving digital threat landscape.**

I. Introduction

The proliferation of QR codes, short for "quick-response codes", as a tool for rapid and efficient information sharing in mobile environments [1][2] has been accompanied by a rise in their exploitation for malicious purposes. QR codes may contain URLs leading to phishing sites, malware installers, or other harmful content [3]. Despite the risks inherent to this technology, users are still less aware of the risks associated with QR codes than of generic URLs [4].

Traditional solutions to the general issue of malicious URL detection have focused on blacklisting known threats [5]. However, blacklists are unable to detect emergent threats. To this end, machine learning has been proposed as a solution, and several classification models have been explored in their effectiveness of detecting threats. Research to this end focuses predominantly on phishing links, and particularly in training supervised models only [6][7][8][9][10]. While some exploration has been done into employing unsupervised models to this end, there are few, if any, robust comparisons of the performance of supervised and unsupervised models in detecting malicious URLs on the same dataset[s].

This project is centered on analyzing and developing machine-learning algorithms to detect malicious URLs within QR codes, which can integrated with a mobile application. We propose a framework for the design of an application integrating an algorithm in order to classify the URL content of QR codes as benign or malicious. This innovation aims to alert users of potential threats immediately upon scanning a QR code, to prevent users from putting their security at risk by following a potentially malicious QR code link. The motivation behind this project stems from the increasing reliance on QR codes and the corresponding lack of widespread awareness about their potential security risks. By leveraging machine learning, our solution adapts to new threats more effectively than traditional methods like URL blacklists, which struggle to keep pace with the rapid emergence of new malicious URLs.

A. Individual Contributions

Most responsibilities were equally shared among our group for this project. Specific individual contributions are as follows:

- **Sav:**
  - Developed code for lexical extraction, some supervised models.
  - Developed application proposal and high-fidelity mockup.
  - Logged training times of all models.
- **Andreas:**
  - Developed code for additional supervised and unsupervised learning models.
  - Compared supervised and unsupervised model performances.
  - Identified and proposed fix for data leakage.

II. Related Works

Many studies in this domain have largely focused on different approaches to known exploit detection using machine learning algorithms. For instance, Kim et al.'s [11] WebMon tool uses a call-tree based approach. However, these studies primarily deal with known threats, leaving a gap in addressing newly emerging malicious URLs, particularly in the context of QR codes and detection in emergent environments.

A. Feature Selection and Extraction Methods

Sahoo et al. [12] outline five general categories of representative URL features in their survey of works, namely blacklist, lexical, host, content, and others. Blacklist features refer to whether the URL is found on a given blacklist or not; lexical features refer to the alphanumeric content of the URL (such as domain and path tokens); host features refer to the features of the URL's host (such as IP address, connection speed); content features refer to the content of the website that the URL directs to (such as number of hyperlinks or iframes); other features include context-based and popularity-based features of the website. Of these categories, lexical features are the most accessible and fastest to process. Sadique et al. [8] categorize features similarly, maintaining lexical and host features, but divide other features into domain-based features retrievable by the WHOIS database, and GeoIP features, locational features which can be extracted from the host's IP.

Lexical feature extraction itself has many potential approaches. One common method of lexical feature extraction is the bag-of-words method. In this method, all the words in every URL in the training dataset is compiled into a dictionary, and each URL is assigned a binary sequence based on whether each dictionary word is present. This method has some drawbacks; the order of the words in the original URL is lost, and processing large datasets with this method is cost-inefficient [12].

Other methods consider tokenization strategies of the URL; while Sayamber and Dixit [13] consider information on the domain and path token distribution of the URL itself, Lakshmanarao, Babu, and Krishna [14] compared three methods of lexical tokenization- count vectorization, TF-IDF vectorization, and hashing vectorization- for processing a URL dataset to train on four supervised models, and achieved the best overall accuracy using hashing vectorization in tandem with a Random Forest (RF) model.

Verma and Das [7], meanwhile, employ Character N-grams in an effort to speed up the process of lexical feature extraction for large datasets. Character N-grams are overlapping sequences of N consecutive characters for values of N from 1 to 10. This approach has the benefit of capturing the order of characters and words in sequence, accounting for misspellings and punctuation in the URLs.

The main benefit of using purely lexical features for classification, aside from their processing speed and accessibility [12], is that it avoids the security risks inherent to the retrieval of content features during live classification such as drive-by download and cryptojacking attacks [15]. This comes with a trade-off of model accuracy, however need source.

B. *Supervised Learning Methods*

Some of the existing research on supervised learning approaches to malicious URL detection and their reviewed models are as follows:

- Nowroozi et al. [6] compared RF, Gradient Boost, AdaBoost, and XGBoost for URL classification, achieving a maximum accuracy of 99.63% with XGBoost.
- Verma and Das [7] compare J48, PART, and RF trained on URL unigrams and bigrams for phishing site detection. They achieve a maximum accuracy of 99.83% from RF on bigrams.
- Sadique et al. [8] compare K-Nearest Neighbors (KNN), ADB, GDB, Decision Tree (DT), RF, GNB, LD, QD, Support Vector Classification(SVC), and NuSVC for phishing URL detection. They achieve a maximum accuracy of 90.51% from RF. They also compare online learning classifiers PE, SGD, PA, MDT, Modrian Forest (MDF), and MLP, and among those, achieve a maximum accuracy of 87.47% from MDF.
- Ahammad et al. [9] compare RF, DT, Light GBM, Logistic Regression (LR), and Support Vector Machine (SVM) for phishing URL detection, with a maximum accuracy of 86.0% on Light GBM.
- Alkhudair et al. [10] compared RF with J48 Decision Tree, BayesNet, and KNN models on a dataset of lexical, host, and network features. They achieved a maximum accuracy of 96% on RF.

C. *Unsupervised Learning Methods*

Little research has been conducted into the application of unsupervised learning approaches to malicious URL detection, likely because unsupervised learning algorithms like K-Means Clustering classify targets based on homogeneity of features, and as such would perform better at anomaly detection than binary classification. Yan et al. [16] proposed an unsupervised model to learn URL embedding, to the ends of detecting malicious websites. They achieved a model precision as high as 91% using 100-dimensional vectors, but this came at the cost of requiring large amounts of storage for the model. Afzal et al. [17] compared the performance of Long Short-Term Memory (LSTM) and K-Means Clustering models trained on lexical and semantic features with a hybrid deep-learning approach, achieving accuracies of 98.3% and 99.7% respectively.

D. *Deep Learning Methods*

Some newer research has investigated the applications of a deep learning approach to this classification problem. Sahoo et al. [18] evaluated the effectiveness of deep learning algorithms alongside traditional machine learning algorithms at classifying malicious URLs using lexical features, and found that the deep learning algorithms MLP and Conv1D performed the best overall with 99.73% and 99.72% accuracy respectively. These results are promising for future work employing deep learning approaches to malicious URL detection. Le et al. [19] developed a Convolutional Neural Network (CNN) to detect malicious URLs based on lexical features, achieving a maximum AUC of 99.29 with a training set of 5 million URLs analyzing both character and word features- though this method has tradeoffs in feasibility due to the memory required to support the parameters for large datasets with many unique words.

III. Method

Our approach employs machine learning algorithms to analyze lexical features of URLs extracted from QR codes. We utilize lexical features exclusively as our features for classification; in the context of a mobile app, this allows the user to classify a QR code in an offline setting, and prevents potentially bad actors from receiving the client's network traffic. The following specific lexical features were utilized:

- Domain Features
  - Top-Level Domain (TLD)
  - Domain token count
  - Domain token average length
  - Domain token max length

    - Domain token length standard deviation
- Path Features
    - Path token count
    - Path token average length
    - Path token max length
    - Path token length standard deviation
- Other Features
    - Hostname
    - URL Length
    - Does the URL host a file (does it have any of the following extensions:
    - Does the URL use a common shortener (bit.ly, ow.ly, etc.)?

To extract these features, we utilized the re (regex) library of python, generally splitting on special characters, and dropping empty strings. The first slash in the URL (after a single instance of a double-slash, '//'), represents the cutoff for domain tokens. We dropped common protocols ("http" and "https") and the "www" token from the domain vectors before calculating metrics on domain tokens. We considered the hostname, rather than the full domain, to only be its leading token. For example, the URL "https://en.wikipedia.org/wiki/QR_code" results in the hostname "en", the TLD "org", the domain tokens "en", "wikipedia", and "org", and the path tokens "wiki", "QR", and "code", with no file extension.

Categorical features such as the hostname, and TLD were encoded using one-hot encoding. This was done to reduce the storage necessary for the feature vectors and to allow for training unsupervised models on solely numerical data.

Using a random sample of 60,000 benign and 40,000 malicious URLs from a dataset of over 650,000 URLs [20], we trained six supervised models and five unsupervised models with an 80:20 training-testing split. All the models used were provided by the scikit-learn (sklearn) toolkit for Python. We trained the following models (acronyms provided for ease of repetition):

- Supervised Models:
    - Random Forest (RF)
    - K-Nearest Neighbors (KNN)
    - Logistic Regression (LR)
    - Stochastic Gradient Descent (SGD)
    - Support Vector Classification (SVC)
    - Gaussian Naive Bayes (NB)
- Unsupervised Models:
    - K-Means Clustering (KC)
    - Mini-Batch K-Means Clustering (MKC)
    - Spectral Clustering (SC)
    - Gaussian Mixture Model (GMM)
    - Density-based spatial clustering of applications with noise (DBSCAN)

The following software/OS/hardware environment was used for training, which influenced the training time of the models: Visual Studio Code in Windows 11 (64-bit) on a PC with 16 GB RAM and 29.4 GB total virtual memory.

For certain models with performance that might vary based on intrinsic factors, such as the number of clusters in the K-Means Clustering model, we performed preliminary accuracy checking on different values for their key factors to find the best value to train on before performing testing. The models with model-specific factors, and the highest-accuracy-producing values of those factors, are as follows:

- **Random Forest:** 100 trees at 20 depth.
- **K-Nearest Neighbors:** k = 1.
- **Stochastic Gradient Descent:** log_loss loss function with alpha of 0.0001.
- **Support Vector Classification:** rbf kernel with c = 10.
- **K-Means Clustering:** k = 160
- **Mini-Batch K-Means Clustering:** k = 2
- **Spectral Clustering:** k = 2
- **Gaussian Mixture Model:** n = 2 components
- **DBSCAN:** 5 minimum samples

A. *Study Limitations*

We originally intended to use a combination of lexical and content features for this study, but had to restrict our scope to lexical features, as a number of the sample URLs' webpages have already been taken down. For the purpose of mobile development, this is a benefit to the project, as it does not put user security at risk of loading malicious sources when trying to retrieve content features, but it limits the robustness of our trained models. Some formats of URLs such as shortened content (bit.ly links, for example) may be indistinguishable between benign and malicious in terms of purely lexical features.

In evaluating our models, we necessarily prioritize false negative rates over false positive rates. Not only are false negatives more prevalent within the results of our training overall, but they pose a much greater security risk to the user, as the user may mistakenly visit a malicious site thinking it is benign.

While we proposed a mobile implementation of a QR code classifier and developed a high-fidelity mockup, we were unable to complete development of such an app ourselves due to issues arising regarding the programming environment. We hope that the proposed implementation provides a clear framework for development for software engineers working on similar products in the future.

IV. Results

Our evaluation of various machine learning models for malicious link detection in QR codes was done with an 80-20 training testing split. We assessed models on a unified set of metrics typical to supervised models – accuracy, precision, recall, F1 score, and ROC-AUC – as well as some additional model-specific metrics for certain unsupervised models. The training times were also recorded to evaluate the feasibility of each model in terms of computational resources and efficiency. Table I shows the results of our evaluation. Figure IV shows the confusion matrices of classifications produced by our models; matrices with lighter colors on the entire left side of the matrix, rather than the top left and bottom right, show high rates of false negatives.

TABLE I
PERFORMANCE METRICS OF TRAINED MODELS

Model	Accuracy	Precision	Recall	F1	ROC-AUC	Training Time (secs.)
RF	**90.41**	**92.25**	**82.99**	**87.37**	**48.34**	**14.0**
KNN	82.79	78.80	77.90	78.34	50.0	11.0
LR	69.16	72.66	36.62	48.70	40.79	0.2
SGD	69.35	73.29	36.69	48.90	40.99	0.4
SVC	**90.20**	**91.58**	**83.14**	**87.15**	**62.01**	**416**
NB	63.12	66.26	15.75	25.45	42.64	0.1
KC	68.97	69.49	39.87	50.67	64.11	0.8
MKC	54.14	40.30	30.64	34.81	49.79	0.2
SC	59.92	18.92	0.09	0.17	49.92	424.1
GMM	59.05	43.95	8.96	14.88	50.0	1.2
DBSCAN	60.10	100.0	0.16	0.32	49.92	323.2

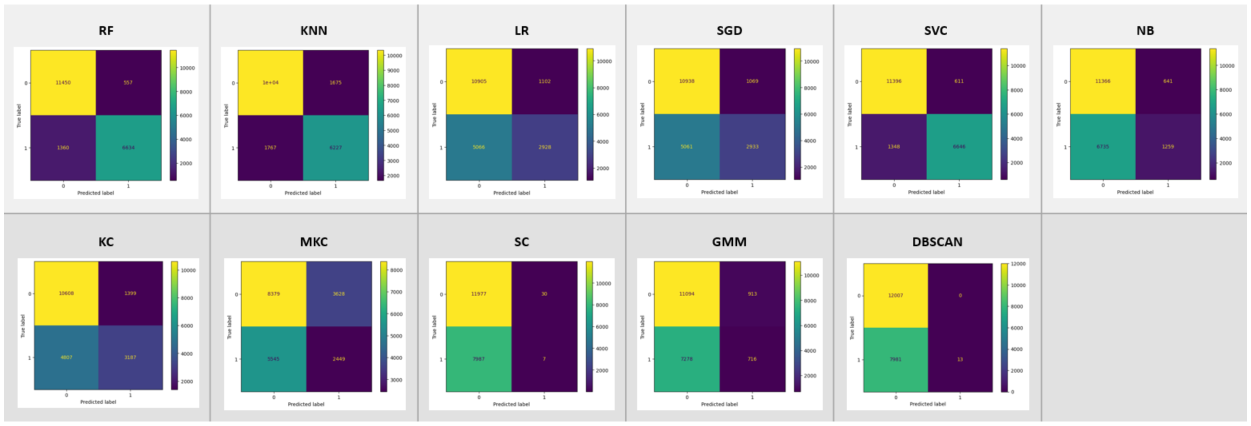

Fig. 1. Confusion Matrices for all Models

TABLE II
CLUSTERING METRICS OF UNSUPERVISED MODELS

Model	Homogeneity	Completeness	V-Measure
KC	14.13	1.97	3.45
MKC	0.0	0.0	0.0
SC	0.03	1.40	0.06
GMM	0.04	0.10	0.06
DBSCAN	0.51	0.50	0.50

A. Key Findings

1) Superior Performance of Supervised Models: Our study revealed a clear distinction in performance between supervised and unsupervised models. Our supervised models performed better overall than our unsupervised models. With enough labeled data available, most supervised learning models were reliable for the purpose of detecting malicious links in QR codes with high-confidence.

Supervised models, such as Random Forest, K-Nearest Neighbors, Logistic Regression, Stochastic Gradient Descent, and Support Vector Classification, consistently outperformed the unsupervised models that were trained and tested on the same data, using the same key metrics. The RF model achieved an accuracy of 90.41%, with precision and recall rates of 92.25% and 82.99%.

Similarly, SVC showed strong results with an accuracy of 90.20%, nearly matching the RF model, as well as a high ROC-AUC score of 62.01%.

Even the worse-performing supervised models – KNN, LR, and NB – demonstrated accuracy scores of 82.79%, 69.16%, 63.12% respectively, reinforcing the relatively strong efficacy of supervised learning for this application, compared to unsupervised counterparts.

The superior performance of the supervised models can be attributed to their ability to learn and generalize from labeled datasets. It enables the estimators to effectively identify patterns and anomalies in the labeled URL data that are indicative of malicious content. The fact that these models can be trained relatively rapidly enhances their practicality for security applications.

2) Limitations of Unsupervised Models: Many unsupervised models showed a high false negative rate. Particularly SC and DBSCAN showed extremely high false negative rates as indicated by their low recall values. This renders them unusable in a security context, as it implies a risk of failing to detect actual threats, giving users a false sense of security.

The unsupervised models showed lower accuracy and precision compared to supervised models. For instance, the KC model had an accuracy of 68.97% and precision of 69.49%, significantly lower than the RF model's 90.41% accuracy and 92.25% precision. The F1 score, which balances precision and recall, was notably lower in unsupervised models. For instance, Mini-Batch KMeans had an F1 score of 34.81%, compared to the 87.37% of the RF model.

The Receiver Operating Characteristic - Area Under Curve (ROC-AUC) scores for unsupervised models like SC and DBSCAN were close to 50%, which is only marginally better than random assignment. Failing to identify a malicious link can lead to security breaches.

In terms of clustering metrics, all unsupervised models performed poorly. Table II shows the homogeneity, completeness, and v-measure scores of the unsupervised models. With K-Means Clustering having the highest homogeneity score at 14.13% while all others are near zero, this may indicate that clustering algorithms are unfit for the dataset itself, with too much overlap in quantitative factors between benign and malicious URLs for meaningful clustering to occur.

Longer training times were also exhibited by unsupervised models. SC and DBSCAN took 424.1 seconds and 323.2 seconds, respectively, to be fitted on the same training set. The only supervised model which took over a minute to train was SVC, at 416 seconds.

Despite the limitations, the results indicate potential areas for enhancing unsupervised learning algorithms. Adjustments in feature selection and model parameters could potentially improve their performance in future studies.

3) Best Performing Model: Random Forest (RF): Of all models, Random Forest performed the best overall, reaching an accuracy of 90.41% with 100 decision trees at 20 depth each. This model also demonstrated a good balance across other metrics, with precision at 92.25% and recall at 82.99%. Furthermore, its short training time (14 seconds) indicates that the model may be a relatively practical choice for real-world applications.

The high accuracy of the RF model demonstrates a relatively strong ability to handle the complex features of URLs within QR codes. The robustness can likely be attributed to the ensemble learning approach, which combines the output of multiple decision trees. The ability of said decision trees to classify based on particular combinations of factors may also contribute to RF's good performance; for example, the specific combination of a long, obfuscating domain name with a file extension could potentially indicate a malicious download link. Additionally, the model's high precision and recall values are indicative of an effective balance between minimizing false positive and false negatives.

The high ROC-AUC score (48.34%) of the RF model, compared to other models, suggest better generalization capabilities. This means the model is less likely to over-fit to the training data and more likely to perform well on unseen data, which is imperative in real-world applications.

The scalability and efficiency of Random Forest models are especially relevant for diverse deployment environments, from mobile devices with limited computation capabilities to large-scale servers. Its ability to rapidly and accurately classify URLs without incurring delays or significant resource demands is promising for its suitability for real-time QR code scanning applications.

Lastly, there is potential for further enhancing the Random Forest model. By optimizing hyperparameters such as the number of trees and their depth, implementing advanced feature engineering and feature selection methods, and using data augmentation, the accuracy and operating efficiency can be improved. Continuous improvements are likely to further cement the RF model as the most promising tool for protecting users from QR code-based cyber threats.

4) Training Time Considerations: The results of our study highlights the importance of balancing accuracy with computational efficiency. While the Support Vector Classification (SVC) model performed well (90.20% accuracy), its training time was significantly longer (416 seconds). Furthermore, while Naive Bayes (NB) had the shortest training time (0.1 seconds), its lower accuracy (63.12%) and precision (66.26%) limit its effectiveness. This would limit its utility in scenarios where frequent training is required, such as live-trained or federated learning settings.

B. *Implications of the Study*

The outcomes of our research provide important insight into the applications of machine learning for improving QR code security, in a world where these codes – and attacks on them – are becoming more common. The outstanding performance of the Random Forest model in many of the evaluation metrics suggests that developers of cybersecurity software focusing on mobile environments should prioritize similar supervised learning algorithms. This is a crucial insight for developing robust security systems which aim to protect users from QR code based security attacks.

V. Mobile Implementation Proposal

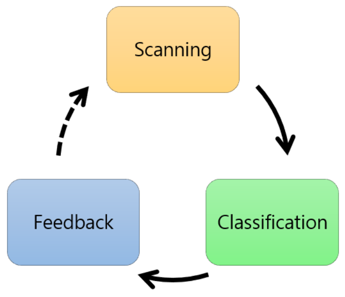

Fig. 2. Generalized Flow of Application Phases

Our proposed mobile implementation is based largely on the expected user experience of an app for scanning QR codes. Usage is divided into three phases: scanning, classification, and feedback. Figure V shows the flow of these phases, while Figure V shows our UML diagram for the design of this mobile app.

During the scanning phase, the app has not yet detected a QR code through the camera. When a QR code is detected, the app captures the data of the QR code, and enters the classification phase.

During the classification phase, the app extracts features from the data of the QR code and performs binary classification on the content based on its features. If the code's data does not represent a URL, it is automatically classified as benign, and enters the feedback phase (classifying non-URL threats is outside the scope of this application, though other developers may include it depending on the availability of additional classifiers). If the code's data represents a URL, it undergoes the process of lexical feature extraction described in section III of this paper. The model supported by the app-

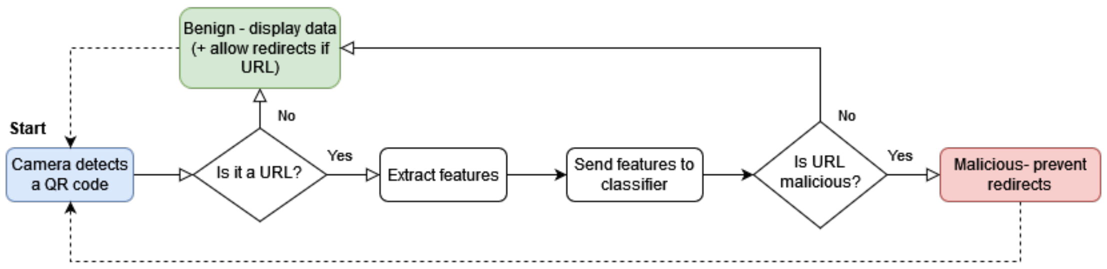
Benign - display data (+ allow redirects if URL)
Start
Camera detects a QR code
Is it a URL?
No
Yes
Extract features
Send features to classifier
Is URL malicious?
No
Yes
Malicious- prevent redirects

Fig. 3. UML Diagram for Mobile Implementation of QR Classifier

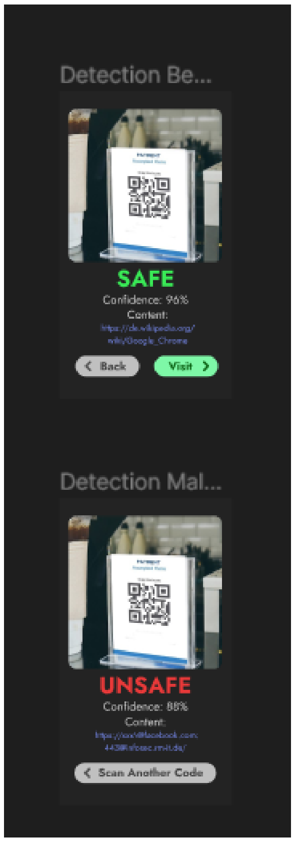

Fig. 4. High-Fidelity Mockup of Classification Result Displays

in the case of our comparisons, a Random Forest model, as it had the highest accuracy and precision overall- then classifies the feature set as either benign or malicious, and enters the feedback phase.

During the feedback phase, the app presents the user with the results of the classification. If the classification resulted in a benign URL, it allows the user to redirect to the URL in their browser; otherwise, it does not allow redirects. If the data in the QR code was not a URL, it also allows the user to preview this data if it is in a known image or text format. In either case, the app also allows the user to return to the scanning phase to scan other codes. Figure V shows our high-fidelity mockup of example displays for benign and malicious URLs in the feedback phase.

While originally intended to be developed in the .NET MAUI language for cross-compatibility between Windows, iOS, and Android environments, this proposed application architecture is broadly applicable to any mobile device which has a camera and supports a given QR code scanning service or library.

VI. Conclusion and Future Work

Our initial findings underscore the potential of machine learning in enhancing QR code security, and the viability of using a purely lexical approach to detecting malicious URLs through supervised learning. The proposal for a mobile implementation provided in this paper provides a clear architectural framework through which a mobile app implementing malicious QR code detection could be built and made publicly available- the logical next step is to complete such an implementation.

Future research could explore deeper integration with mobile operating systems for broader protection and investigate the use of more diverse data sets for training algorithms. Additionally, further comparison between different machine learning models could yield insights into optimizing accuracy and speed in real-world applications, particularly in the training phase, as continuous training is of importance to a long-term robust solution.

Some aspects of our feature selection, even limiting to lexical features, were limited. In an effort to achieve even higher accuracy, future models trained on lexical features could take the following into account:

- Verma and Das's [7] method of utilizing Character N-grams could help better capture some of the known obfuscation techniques used by phishing sites, such as mimicking legitimate domains.
- Ahammad et al. [9] suggest that the presence of IPs, '@' symbols, and a '-' prefix or suffix in a URL can be indicators of malicious links.
- In Alkhudair et al.'s [10] analysis of lexical, host, and network training features, they found that the following features were the six best predictors of malicious versus benign URLs:
  - Number of special characters
  - Country of server from WHOIS lookup
  - City/state of server from WHOIS lookup
  - Count of known ports different from TCP
  - Packets received by the server

– Bytes transferred to the server

With these in mind, investigating the feasibility of including WHOIS lookup features alongside lexical features without compromising the user's packets or offline mobile robustness may be worth investigating, if possible.

A hybrid approach to training detection models, using lexical data in tandem with a preexisting blacklist may also improve the accuracy to a degree. While blacklists may not be able to capture emergent threats on their own, they can improve the overall performance of a model by compensating for poor classification of known threats, and help classify emergent threats that use similar lexical schemes to known threats.

It may also be worth looking deeper into the current state of risk associated with QR codes, in terms of users' susceptibility to QR threats rather than their awareness of them. To this end, we recommend a field study utilizing web traffic analytics and mock-ups of QR attack vectors (such as fake advertisements, newsletters, etc.) to record the number of users visiting a site through misleading or fraudulent QR codes.

One final important consideration specific to the domain of malicious QR codes is non-URL threats, such as SQL injection and command injections [21]. While our research only covered the detection of malicious URLs and our proposal for an application framework disregarded non-URL data as benign automatically, it would likely be possible and feasible to detect other forms of malicious QR codes such as these attack vectors. Other studies could build on this research to include these alternate attack vectors in detection, either as part of the same classification model (utilizing a different tokenization technique for lexical feature encoding) or as a separate model selectively used when an app detects non-URL data from a QR code. Evaluating whether there is a significant difference in performance between a mixed singular model or separate models would help developers choose an approach with respect to computing resources.

References

[1] J. Coleman, "QR Codes: What Are They and Why Should You Care?" in *Kansas Library Association College and University Libraries Section Proceedings*, vol. 1, no. 1, pp. 16-23, 2011, doi: 10.4148/culs.v1i0.1355.

[2] S. Tiwari, "An Introduction to QR Code Technology," in *2016 International Conference on Information Technology (ICIT)*, pp. 39-44, 2016, doi: 10.1109/ICIT.2016.021.

[3] P. Kieseberg, S. Schrittwieser, M. Leithner, M. Mulazzani, E. Weippl, L. Munroe, and M. Sinha, "Malicious Pixels Using QR Codes as Attack Vector," *Trustworthy ubiquitous computing*, pp. 21-38, 2012, doi: https://doi.org/10.2991/978-94-91216-71-8_2.

[4] N. Kumar, S. Jain, M. Shukla, and S. Lodha, "Investigating Users' Perception, Security Awareness and Cyber-Hygiene Behaviour Concerning QR Code as an Attack Vector," in *Int. Conf. Human-Computer Interact.*, pp. 506-513, Jun. 2022.

[5] S. Sinha, M. Bailey, and F. Jahanian, "Shades of Grey: On the effectiveness of reputation-based 'blacklists'," in *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, 2008, doi: 10.1109/MALWARE14503.2008.

[6] E. Nowroozi, Abhishek, M. Mohammadi and M. Conti, "An Adversarial Attack Analysis on Malicious Advertisement URL Detection Framework," in *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1332-1344, June 2023, doi: 10.1109/TNSM.2022.3225217.

[7] R. Verma and A. Das, "What's in a URL: Fast Feature Extraction and Malicious URL Detection," in *IWSPA '17: Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, Scottsdale, AZ, USA, 2017, pp. 55-63, doi: 10.1145/3041008.3041016.

[8] F. Sadique, R. Kaul, S. Badsha and S. Sengupta, "An Automated Framework for Real-time Phishing URL Detection," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2020, pp. 0335-0341, doi: 10.1109/CCWC47524.2020.9031269.

[9] S. K. H. Ahammad, S. D. Kale, G. D. Upadhye, S. D. Pande, E. V. Babu, A. V. Dhumane, and D. K. J. Bahadur, "Phishing URL detection using machine learning methods", *Advances in Engineering Software*, vol. 173, Nov. 2022, doi: 10.1016/j.advengsoft.2022.103288.

[10] F. Alkhudair, M. Alassaf, R. U. Khan, and S. Alfarraj, "Detecting Malicious URL," in *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, Tabuk, Saudi Arabia, 2020, pp. 1-5, doi: 10.1109/ICCIT-144147971.2020.9213792

[11] S. Kim, J. Kim, S. Nam, and D. Kim, "WebMon: ML- and YARA-based malicious webpage detection," *Computer Networks*, vol. 137, pp. 119-131, Jun. 2018, doi: 10.1016/j.comnet.2018.03.006.

[12] D. Sahoo, C. Liu, and S. C. H. Hoi, 'Malicious URL Detection using Machine Learning: A Survey', arXiv [cs.LG]. 2019.

[13] A. B. Sayamber and A. M. Dixit, "Malicious URL Detection and Identification," *International Journal of Computer Applications*, vol. 99, no. 17, Aug. 2014.

[14] A. Lakshmanarao, M. R. Babu and M. M. Bala Krishna, "Malicious URL Detection using NLP, Machine Learning and FLASK", *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, Chennai, India, 2021, pp. 1-4, doi: 10.1109/ICSES52305.2021.9633889.

[15] H. Tupsamudre, A. K. Singh, and S. Lodha, "Everything Is in the Name – A URL Based Approach for Phishing Detection", in *CSCML 2019: Cyber Security Cryptography and Machine Learning*, S. Dolev, D. Hendler, S. Lodha, and M. Yung, Eds. Beer-Sheva, Israel, 2019, pp. 231–248, doi: 10.1007/978-3-030-20951-3_21.

[16] X. Yan, Y. Xu, B. Cui, S. Zhang, T. Guo, C. Li, "Learning URL Embedding for Malicious Website Detection," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, 2020, pp. 6673-6681, doi: 10.1109/TII.2020.2977886

[17] S. Afzal, M. Asim, A. R. Javed, M. O. Beg, and T. Baker, "URLdeepDetect: A Deep Learning Approach for Detecting Malicious URLs Using Semantic Vector Models," *Journal of Network and Systems Management*, vol. 29, no. 21, 2021, doi: 10.1007/s10922-021-09587-8.

[18] V. K. Sahoo, V. Singh, M. K. Gourisaria, and A. K. Acharya, "URL Classification on Extracted Feature Using Deep Learning", in *Computer Vision and Machine Intelligence*, M. Tistarelli, Ed. Singapore, 2023, pp. 415–428, doi: https://doi.org/10.1007/978-981-19-7867-8_33.

[19] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection", arXiv [cs.CR]. 2018.

[20] M. Siddhartha, "Malicious URLs dataset," [Online]. Available: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset/

[21] P. Kieseberg, M. Leithner, M. Mulazzani, L. Munroe, S. Schrittwieser, M. Sinha, and E. Weippl, "QR code security," in *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, New York, NY, USA, 2010, pp. 430–435, doi: 10.1145/1971519.1971593.