IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 June 2023, accepted 25 June 2023, date of publication 3 July 2023, date of current version 31 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3291811

RESEARCH ARTICLE

QsecR: Secure QR Code Scanner According to a Novel Malicious URL Detection Framework

AHMAD SAHBAN RAFSANJANI[1], NORSHALIZA BINTI KAMARUDDIN[2], HAZLIFAH MOHD RUSLI[2], (Member, IEEE), AND MOHAMMAD DABBAGH[3]

[1]Department of Computing and Information Systems, School of Engineering and Technology, Sunway University, Selangor 47500, Malaysia
[2]Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur 54100, Malaysia
[3]School of Information Technology and Engineering, MIT Sydney, Sydney, NSW 2000, Australia

Corresponding Author: Ahmad Sahban Rafsanjani (ahmadsahban@sunway.edu.my)

ABSTRACT Malicious Uniform Resource Locators (URLs) are the major issue posed by cybersecurity threats. Cyberattackers spread malicious URLs to carry out attacks such as phishing and malware, which lead unsuspecting visitors into scams, resulting in monetary loss and information theft. The adoption of Quick Response (QR) codes with malicious URLs is a growing concern and is an open security issue. The existing QR link detection scanner applications mostly utilize the blacklist method to detect malicious URLs, which is not the optimal method for detecting new websites. Recently, machine learning methods have gained popularity as a means of enhancing the detection of malicious URLs. However, these methods are entirely data-dependent, and a large and updated dataset is required for the training to create an effective detection method. This research proposes QsecR, a secure and privacy-friendly QR code scanner, according to a malicious URL detection framework. QsecR is an Android QR code scanner based on predefined static feature classification by employing 39 classes of blacklist, lexical, host-based, and content-based features. A dataset containing 4000 real-world random URLs was gathered from URLhaus and PhishTank. The QsecR is evaluated by several QR code scanners in terms of security and privacy. The experimental result shows that QsecR outperforms others and achieves a detection accuracy of 93.50% and a precision value of 93.80%, which is significantly higher than the current secure QR code scanners. Also, QsecR is one of the most privacy-friendly application with the least privilege permission.

INDEX TERMS Android security, malicious URL detection, privacy-friendly application, QR code scanner, QR code security.

I. INTRODUCTION

Android is the most popular operating system in the world, with more than 2.5 billion active users and a market share of about 75% among mobile operating systems [1]. The Google Play Store has significantly risen over the previous decade and reached $38.6 billion in 2020 and its revenue has grown by 167% during the past four years [2]. Also, it offered approximately 2.9 million apps in 2020, which were downloaded 108 billion times [1]. With the increasing popularity of this operating system on the global market, its threats are also expanding rapidly [3]. Android was the most vulnerable operating system in the past few years [4].

One particular threat vector in Android is QR codes. The Quick Response code (QR code) is the most popular two-dimensional barcode, and it was invented by the Japanese company Denso Wave in 1994. A QR code has the capability to encode a variety of data types such as numerals, alphabetical, Kanji, katakana, and hiragana characters, symbols, binary data, control codes, and others [5]. They may be used to display text to the user, to add a Vcard, to connect to a wireless network, as well as open a Uniform Resource Locator (URL) that links to a webpage [6], [7]. QR codes are extensively utilized in a variety of sectors such as payment, advertising, access control, product identification, and at this time for the

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamad Afendee Mohamed.

Covid-19 vaccine and tracking [8]. QR codes are widely used because of their high data capacity, readability speed, and reliability [9].

In general, security issues build up when technology is widely used and becomes popular. The QR code is no exception, and besides a broad range of its advantages, it attracts the attention of cyberattackers and has been misused as an attack vector [10]. The first cybercriminal attempts using QR codes were observed in September 2011 [11]. Attackers utilize QR code embedded with a malicious link to lead visitors to a malicious website to unconsciously download Jimm that is infected by TrojanSMS.AndroidOS.Jfake.f malware and sends SMS to premium rate numbers. According to [12], financial crimes employing QR codes are on the rise, with around US $13 million reported stolen via QR code scams in China.

There is no doubt that the majority of Internet users lack the basic skills to access the Internet safely. Specifically, they are unable to distinguish between malicious and benign (safe) websites [13]. The success and widespread use of QR codes in attacks is due to the fact that they are unreadable by human eyes and can be read only using specific scanning devices [14], [15]. QR code link redirects the user to a website that has already been modified by an attacker with the aim of gaining access to the victim's sensitive information.

Depending on whether the QR code scanner is a human or an automated program, several attack scenarios are possible. These attacks are phishing, malware propagation, cross-site scripting (XSS), SQL injection, command injection, and attacks on the scanner applications [16], [17], [18]. Phishing and malware propagation attacks that involve human intervention are the focus of this paper. Also, focus on the potential threats to the privacy of Android QR code scanner applications by requesting unusual permissions from users during installation.

A QRishing attack is a form of phishing attack in which the attacker encodes a phishing website in a QR code [19], [20]. Phishing is the most frequent attack using QR codes in which the victim scans the QR code image with their smartphone and is led to a fake website that appears to be legitimate in order to steal sensitive information such as login details and credit card numbers [21], [22].

Malicious websites are frequently used by attackers to deliver malware software, and the adoption of QR codes together with malware propagation is a growing concern [23]. In this method, the attacker encodes a malicious URL in a QR code and once scanned by a QR code scanner, it will direct the victim to a webpage from where they can be driven by a download attack. The attacker can infect the user's systems and cause serious harm through viruses, ransomware, spyware, botnets, Trojan horses, or worms [8].

Existing security techniques for detecting malicious QR links concentrate mostly on the security of the web browser. The efficiency and accuracy of the applications are entirely dependent on the capability of the browser and its malicious link detection method and plug-ins [24]. Nonetheless, the majority of these browsers rely on the blacklist approach, which is not the most effective means of identifying recently developed websites. Nevertheless, this approach is limited in its scope and fails to recognize newly generated or obfuscated malicious URLs.

Recently, some scanners have utilized machine learning techniques for malicious URL detection. Despite the fact that machine learning techniques have made significant improvements in detecting malicious URLs over the past decade and overcome blacklist method limitations, this approach is not without its limitations, the most prominent being its dependence on data, which poses a significant drawback.

To contribute to the research gap, we propose a malicious URL detection framework according to predefined static feature classification. Then we expanded the concept to the detection of malicious QR codes and developed QsecR, a secure and privacy-friendly QR code Android scanner. Specifically,

- We designed a heuristic malicious URL detection framework that is able to detect new malicious URLs in real time with a high level of accuracy.
- We provide a predefined static feature classification for detecting malicious URLs. The predefined values are assigned a range of values for the classes to do analyses and comparisons. The 39 classes of blacklist, lexical, host-based, and content-based features are extracted and classified.
- We have devised a feature evaluation method that assesses the value provided by individual features. It evaluates whether all features contribute to the final calculation, and if any feature does not provide a value, the method will make a decision to use the value of another feature instead, based on specific conditions.
- We developed QsecR, a secure and privacy-friendly QR code Android scanner against malicious QR links and examined the outcomes of the comparison. QsecR achieves a detection accuracy of 93.50% and overcome other secure Android QR code scanner applications.

The rest of the paper is structured as follows. Section II reviews the literature on the security and privacy of QR code scanners. Also, in details, we reviewed the malicious URL detection methods used by secure QR code scanner applications. Section III introduces the design and overall framework of QsecR. This section includes three phases: the redirection, feature extraction and classification, and malicious URL detection phases. Section IV describes the performance evaluation. This section starts with a description of the dataset, which includes 4000 real-world URLs. Then it presents several evaluation metrics to measure the performance of the scanners. Afterward, evaluate the current Android QR code scanner application in terms of security and privacy. In this part, some malware and phishing websites are selected to evaluate the security performance of QR code scanners in terms of detecting malicious URLs and analyzing

the permission requested from the scanner during installation. Finally, the performance evaluation of QsecR is given in the last part of this section. Findings indicate that QsecR achieves an average accuracy rate of 93.50% and a precision value of 93.80%. Section V figures out the conclusion of this paper and shows the future work for this research.

II. RELATED WORK

The security and privacy issues of QR code scanners have been relatively well studied over the past few years [6], [7], [8], [9], [15], [16], [17], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]. We will briefly review the literature on the security and privacy of current QR code scanners.

A. SECURITY OF QR CODE SCANNERS

QR code attacks are still a real concern for scanners [9], [24], [26]. QR-in-QR attacks, QR code payment attacks, QR code counterfeiting, and QR code information leakage have been identified as potential QR code security threats. Reference [31] describes a novel two-layer QR code attacks using an automated method that can encode two distinct messages in a QR code that can be decoded individually by switching scanning orientations. Reference [16] present QR code attack scenarios for Bitcoin payment by altering wallet address characters via a QR code generator. Reference [14] suggest a nested QR code, which combines two QR codes simultaneously in a square space while both are clearly readable depending on the scanner's orientation.

There are more than 300 applications by searching for the words ''QR code scanner'' or ''QR code reader'' in the Google Play Store, and most of them provide scanning services. A few of them provide a security method to prevent users from link threats, while others are without any security concerns despite their popularity. The average user has difficulty to distinguish between benign and malicious QR codes due to the fact that they are unreadable by human eyes and require specific scanning devices. QR code security scanner applications that offer security are categorized into several methods [9], [30], but in this research they are categorized as link security-based and cryptographic-based. [8].

1) CRYPTOGRAPHIC-BASED METHOD

Cryptographic-based methods are utilized in the QR code scanners to encrypt, sign, and control access to the content to provides confidentiality and privacy [9]. Furthermore, digital signatures can accomplish authentication, integrity, and non-repudiation [21]. There are limited number of applications that support generating and scanning cryptographic QR codes [9]. [32] BarSec is a comprehensive barcode security scanner, by adopting symmetric and asymmetric cryptographic mechanisms and offers barcode authentication, data integrity, access control, and confidentiality. AMP QR Code scanner present an anti-malware and phishing detection method, which provides encryption for QR codes by using the AES mechanism [27].

Despite the fact that some QR code security scanners provide cryptographic features, they are still vulnerable and have several fundamental drawbacks. Their main drawbacks are using weak algorithms, short key lengths, lack of following standard encryption structures or optimal encoding schemes, size overhead, and most importantly, requiring to use the same application (generator) to decode the QR code [17]. With these factors in mind, it's easy to comprehend why cryptographic-based methods haven't seen widespread adoption and utilization.

2) LINK SECURITY-BASED (URL-BASED) METHOD

Link security-based methods are an online protection technique that is provided by the QR code scanners that analyze the URLs encoded in QR codes and prevent users from being redirected to phishing and malware websites [24]. The secure QR code scanners generally utilize two protection methods for link security-based methods. These methods are blacklist and machine learning, which are explained in the following section [8], [9].

The blacklist method is the most preferred link security-based approach for QR code scanners. The URLs that have already been identified as potentially harmful (phishing, malware) are located in blacklist databases and have gathered over time [29]. It is assumed that a URL is harmful if it appears in the blacklist database and a warning is produced; otherwise, it is benign. This strategy is extremely fast and simple to implement due to the minimal query overhead, and it produces very low false positive errors [33], [34].

Reference [10] proposes SafeQR, a QR code scanner which is able to detect phishing and malware attacks by invoking the Application Programming Interface (API) of two famous blacklist databases, Google Safe Browsing [35] and PhishTank [36]. AMP QR Code Android scanner application [27] present an anti-malware and phishing detection method, by calling the VirusTotal [37] API.

The primary objective of malicious URL detection methods is to defend users against online attacks and related threats in real-time [38]. However, detection of malicious URLs has traditionally been done mostly via blacklists, bu this method is not exhaustive and cannot identify newly created or obfuscated malicious URLs [39]. Since attackers create hundreds of new websites daily (due to the short-lived of malicious URLs because they are suspended after recognition), the blacklist method is unable to identify fresh URLs that have not yet been added to the blacklist database [40], [41].

As big data becomes increasingly popular, machine learning techniques that are both generalizable and resistant to real attacks have evolved as the most widely used means of detecting malicious URLs [39]. Machine learning is a type of artificial intelligence and is the study of computer algorithms that improve detection models automatically via experience and analyzing data [42].

The main goal of machine learning is to develop a malicious URL detection model by utilizing sample data, referred to as a training dataset, and finding patterns in it [38]. These patterns can then be used to make predictions, categorize, and cluster objects without explicit programming [43]. This method classifies the features represented in a URL by extracting the APIs and other components of a website and then trains a prediction model on a dataset that includes both malicious and benign URLs.

The machine learning detection method is utilized in limited number of QR code scanners [8], [24], [28]. QR fence [24] presents, a threat-oriented QR malicious link detection framework, based on a novel machine learning model which integrates multiple classification algorithms, such as IBK, NB, RT, J48 and Logistic to train 31 lexical and content-based features. Reference [28] proposes QRphish, an automated QR code phishing detection approach based on a Bayes classifier machine learning model to train 20 lexical and host-based features. Reference [8] describes BarAI, secure real-time artificial intelligence system against malicious QR Code links. They used multiple machine learning classifications such as Naive Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR), K-nearest neighbours (KNN), and decision tree J48 (DT) classifier, including 17 lexical features.

Despite the fact that machine learning techniques have made significant improvements in detecting malicious URLs over the past decade and overcome blacklist method limitations, there are still numerous critical weaknesses that remain. The main drawbacks of machine learning are [34], [44], and [45]:

- **Label Dataset.** The model requires tedious and time-consuming labelling datasets for training supervised machine learning.
- **Massive Stores of Training Data.** A huge dataset is required for the training of machine learning algorithms in order to create an appropriate model for detecting malicious URLs.
- **Data Dependent (Quality of Dataset).** The detection model's reliability and level of accuracy are entirely dependent on the quality of the dataset. According to [46] the models that were built based on a training dataset with a high level of accuracy are ineffective for detecting URLs in another dataset.
- **Incur high retaining cost.** Continuous retraining with an updated dataset enables a method to identify and interact with new and real-world malicious URLs, which requires a significant amount of time and software resources [41].
- **Difficulty of Selecting Features.** Feature selection is incredibly difficult and requires expertise to select informative features and classes that enhance the detection performance of a method.

Recently, deep learning (a type of machine learning) has been applied to the challenge of detecting malicious URLs, and it has been effective in tackling some problems in machine learning [47]. It eliminates the feature selection procedure, which increases system performance and prevents the loss caused by the selection of incompatible features [48]. It doesn't require tedious feature extraction, which leads to a training model with minimal effort and results in an appropriate pattern for detecting malicious URLs [44], [49]. While there is progress, there are still significant issues that remain. Table 1 presents the limitation of current QR code scanner's detection methods.

TABLE 1. Weakness of current QR code scanner's detection methods.

Method	Weakness
Blacklist-based	• Unable to detect new generated URLs • Unable to detect obfuscated URLs
Machine learning-based	• Require labelling dataset for supervised machine learning • Massive Stores of Training Data • Data Dependent (Quality of Dataset) • Incur High Retaining Cost • Difficulty of Selecting Features

B. PRIVACY OF QR CODE SCANNERS

The safety of QR code scanner applications is a major concern for QR code security [20], [29]. There are several potential threats to the privacy of Android devices, but the most severe is an application's request for excessive permissions [50]. There is always the possibility that an attacker would discover a vulnerability in a QR code scanner application, which may lead to gaining access control over the entire smartphone and acquiring entry to the user's sensitive data [9]. The reason that is resulting to this vulnerability is that the application is seeking full permission to access the user's smartphone resources during the installation process [24].

The APK file is the list of all the information related to an Android application. Android utilizes a permission system to limit application access to system resources. The application should seek the required permissions via the AndroidManifest.xml file if it attempts to access hardware or software resources [51]. The number of Android built-in permissions is continuously increasing, from 166 permissions at API level 15 to 325 permissions at API level 28. More permissions mean more opportunities for exploitation [52].

Developers request a variety of permissions from the end user's smartphone, but they may be unaware of the risks associated with obtaining these permissions. Some developers just request the permissions necessary for their apps, however others believe that getting unnecessary permissions will ensure that their apps continue to run under all circumstances [50]. According to [8] and [9], privacy-friendly QR code scanner applications need standard architectural choices for developers to build applications with the least privilege permission. The necessary permissions that should be requested from applications are camera (scan the QR code) and internet (check the URL link). The least-privileged permissions for Save-Privacy QR code scanners are [9]:

- Camera: takes pictures and videos;
- Network: gives network access and views network connections;
- Wi-Fi: view Wi-Fi connections;

However, some of these scanners request unusual permissions which can lead to exploitation, such as changing or erasing the contents of the user's SD card, location, microphone, Bluetooth, telephone access for the purpose of directly calling phone numbers, SMS, and drawing over the other apps to modify system settings, etc.

These permissions may expose scanners to vulnerabilities. [53] discovered that a popular QR code scanner application downloaded over 10 million times from the Google Play Store has infected up to 10 million devices via a software update in December 2020. According to [54], six malicious QR code scanner applications with more than 500,000 downloads were discovered in the Google Play Store in 2018, which propagated a virus known as Andr/HiddnAD-AJ. These applications are able to evade Google's scanning by hiding malicious code and delaying the start of operation until six hours after installation, allowing them to avoid detection. Based on the findings of [55], Zebra Crossing or ZXing [56] with over 126 million installations in 2016 contained three specific vulnerabilities, namely code injection, unauthorized actions, and information leakage. Table 2 illustrates a summary of the related works, including our proposed scanner.

TABLE 2. Summary of related work.

Scanner	Detection Method	Feature Extraction	Classifier
[24]	Blacklist and Machine learning	Host-based and content-based (22 features)	J48, Random Tree, Naïve Bayes, IBk and Logistic and their combinations
[32]	Blacklist	Google Safe Browsing, PhishTank	-
[28]	Blacklist and machine learning	Blacklist, lexical and host-based (20 features)	BN
[8]	Machine learning	Lexical feature (11 features)	NB, SVM, LR, K-NN, and DT
[57]	Blacklist	Blacklist feature (G Data security)	-
[58]	Blacklist	Blacklist feature (Lionic malicious web cloud database)	-
[59] **QsecR**	Novel method (predefine static feature classification)	Redirection, Blacklist, Lexical, Host-based, Content-based (39 features)	-

III. QsecR FRAMEWORK

The proposed framework of QsecR is shown in Figure. 1. The malicious URL detection framework is separated into three phases. The first phase checks the URL for redirection. The specific purpose of this phase is that if a shortened or redirected URL is used, it redirects it to the original website in order to evaluate by features. The second phase is feature classification. This phase consists of four features, each of which consists of various classes, and its primary objective is to extract relevant information from the URL in order to detect malicious URLs. The third phase is to evaluate the result and detect malicious URLs. It evaluates the value given by features. If all the features provide a value, it proceeds to the final computation; however, if any of them fails to deliver a value, it employs other features' values and utilizes all these values for detection.

A. URL REDIRECTION

Cybercriminals attempt to bypass malicious URL detection techniques using obfuscation methods. Short URLs have established as the most effective obfuscation method to trick users by displaying malicious URLs as legitimate, and are widely used in phishing and malware websites [47], [60]. The redirection phase describes the procedure to take the input URL from the QR code and then run the algorithm for detecting obfuscated URLs. The specific purpose of this phase is to redirect URLs to the original website in order to evaluate by features if a short URL is used. This method helps to improve the detection accuracy of the framework by sending the original URLs to the feature extraction and classification part and evaluating them by features. If a URL does not redirect and send to these features, some of them will wrongly evaluate the website and provide an incorrect value, resulting in a high false positive rate.

The procedure for this phase begins with reading the QR code image. If the QR code contains text format, the application shows the user that the QR is benign and displays content to the user. However, if it includes a link, it checks if the URL includes a list of data formats and file extensions (URLs include IP addresses, executive files, and multimedia file formats). If yes, it displays the original URL and ends this phase (sending the URL to the next phase), and if no, it opens the website in a WebView. Android WebView is a Chrome-powered system component that allows Android apps to show online information. In this part, the WebView checks if the inserted URL is original or redirected. If original, it displays the original URL and ends this phase. If not, it opens the original website in the second WebView and redirects to the first WebView by running a method that is called ''overrides URL loading'' and repeating it until the original URL is displayed [61]. The first WebView counts the number of redirections, and the second one shows the original URL. The significance of this phase is that it returns the website to its original URL if it has been redirected even more than ten times. In the last step, the URL will be displayed to the user, and if a redirection occurs, it will be displayed as well. Here, the URL is ready to go on to the next phase and be processed through the features.

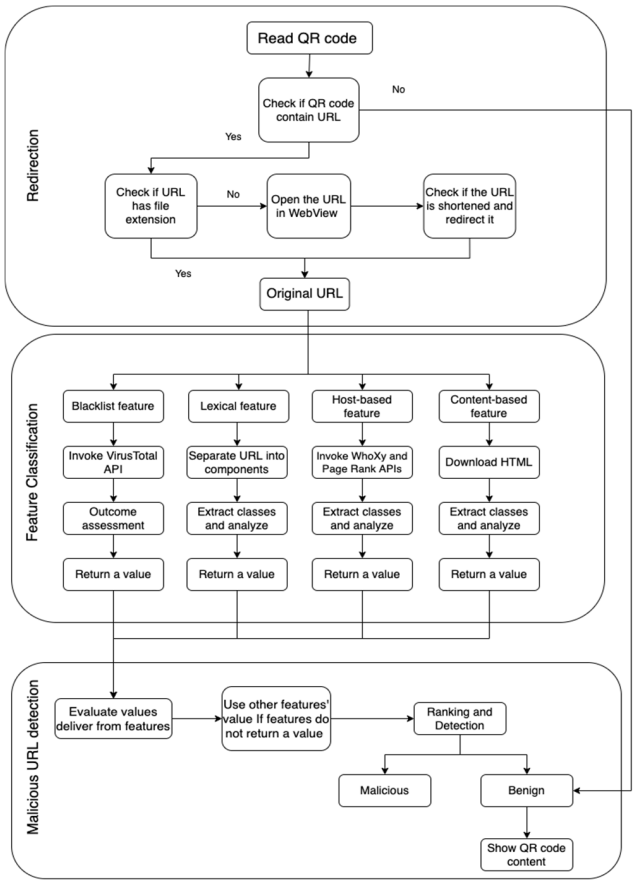

FIGURE 1. Proposed malicious QR Code detection framework.

B. FEATURE EXTRACTION AND CLASSIFICATION

There are specific features that must be selected and implemented in order to have a successful and effective detection system. Also, selecting features is extremely hard and needs expertise, and may lead to decreased detection performance due to ineffective feature selection [62]. The initial step is to classify and extract relevant information that adequately characterize the URL. These classes are extracted through parsing and analyzing various components of URLs, as well as by utilizing several APIs to provide valuable information.

In this research, the feature classes are gathered from previous malicious URL detection models, which achieved high level of accuracy [43], [63]. A malicious URL detection method based on supervised batch and online machine learning classifiers [43]. This research focuses on multi feature classification by employing 117 static and dynamic lexical, host-based, redirection, and content-based features. In the other research [63], authors proposed a machine learning approach for malicious URL detection by combining linear and non-linear space transformation approaches. It utilized 62 classes for feature classification, extracted from blacklist, lexical, host-based, and content-based features.

Although it is possible to utilize numerous classes, and this may improve the detection accuracy slightly, it is overwhelming and increases the response time. However, in this research, the classification is enhanced by having new classes and by utilizing only critical classes that assist in detecting malicious URLs to increase accuracy.

The critical classes refer to types of classes that assist in extracting essential statistical information from a URL, which leads to detecting whether this website is malicious or benign. These classes give relevant information regarding the website, which results in increased detection accuracy. The critical classes are identified according to the results of detecting malicious URLs in several datasets. Also, in this research, some new classes are added in the certificate and host-based features parts.

This phase consists of four features, each of which consists of various classes, and its primary objective is to extract relevant information from the URL in order to detect malicious URLs. These features are the blacklist, lexical, host-based, and content-based. In QsecR framework, we employed 39 classes, which are shown and described in detail in table 3.

The blacklist feature searches the URL in several databases and checks if it has already been discovered to be harmful (Phishing, Malware) or not. The VirusTotal [64] API is used to search over 90 different blacklists and web security service websites to determine how many of them contain the provided URL.

The lexical feature is the textual properties of a URL and extracts various details from the URL strings. The URL is broken into multiple components to enhance feature classification, such as the entire URL, hostname, path, and top-level domain (TLD), and each of them is inspected individually for analysis.

The host-based feature extracts a variety of information about the host and web rank. It looks at a website's traffic statistics and popularity, as well as the host details and the website's owner's personal information. This feature invoked two APIs that are called WhoXy [65] and OpenPageRank [66]. WhoXy is a type of WHOIS API, which is a hosted web service and returns well-parsed WHOIS information in multiple formats. The OpenPageRank aims to share the proposed host ranks and visit metrics.

The content-based feature is statistical information, which the website should completely download from the server and extract information from the raw content. It provides a lot of valuable information and is categorized into HTML, JavaScript, and certificate parts. Each part has various classes and searches through the programming functions.

The URL that was delivered in the previous phase will be used for each feature. Depending on the functionality, it may collect information by invoking APIs, parsing the URL, or downloading HTML. Then extract the information from them according to predefined values that shown in table 3. The classes are defined and categorized via numeric and binary values. The numerical values represent the sorts of classes required to count the objects, and binary classes are defined by the presence of unique objects.

TABLE 3. List of selected features along with their characteristics and reasons for their selection.

	No	Classes	Reasons	Type
Lexical	1	%	Used to identify URL HTML encoding, existence of IDs (i.e., Session IDs, Affiliate IDs, Tracking IDs, Referrer IDs) and time stamps.	Numeric
	2	//	It is used to identify segment part.	Numeric
	3	/	It is used to compute path's length in the URL.	Numeric
	4	.	It is used to determines the relative references within path hierarchy, domain name, SLD, and TLD.	Numeric
	5	=	It is used to identify assignment of values in URL.	Numeric
	6		It is used to count number of word joiners.	Numeric
	7	-	It is used to count number of word separators.	Numeric
	8	@	It used to extract and analyzed whether any user information exists in the URL or not.	Numeric
	9	?	It is used to identify the query section of the URL, if any.	Numeric
	10	:	It has been used to determine whether a port is in use.	Numeric
	11	~	It has been used to determine whether the URL has any reference or not.	Numeric
	12	URL Length	It has been used to identify the obfuscation technique mostly employed in phishing attacks.	Numeric
	13	Hostname Length	It used to identify the obfuscation technique mostly employed in phishing attacks.	Numeric
	14	Path Length	It used to identify the obfuscation technique mostly employed in phishing attacks.	Numeric
	15	Count Redirect	It used to identify the obfuscation technique, it mostly employed to pass the detection models. This information of this class gathered from preprocessing phase.	Numeric
	16	Count Digits	It used to count the number of digits has been used in the URL.	Numeric
	17	Exist IP	Some malware hosting websites don't have domain names and are instead identified by their IP addresses. Mostly used for drive by download attack.	Binary
	18	Exist Port	Using certain ports creates a connection between the web browser and the web servers, exposing sensitive user data to cybercriminals and potentially resulting in severe data misuse.	Binary
	19	Absolute URL	An absolute URL contains more information from protocol to TLD. However, a relative URL does not use the full web address and only contains the location following the domain name and is mostly used for obfuscation purposes.	Binary
	20	Suspicious TLD	The lengthy list of suspicious top-level domains with 2 levels of high risk and mid-risk is mostly used for phishing and malware attacks.	Binary
	21	Suspicious Words	The lengthy list of suspicious words has 2 levels of high risk and mid-risk, which are mostly used for phishing and malware attacks.	Binary
	22	File Extension	It has been mostly used by attackers to launch a drive-by-download attack on their victims by downloading executive files. It is a list of file extensions that includes all the executive files running in different operating systems.	Binary
Host	23	Create Date	According to the fact that the malicious website only exists for a short period of time.	Numeric
	24	Expiry Date	The domains that are known to be trustworthy are frequently paid in advance for several years. Usually malicious websites are expired and kindly not lived after quick time.	Numeric
	25	Update Date	Base on the fact that the trusted websites update their information frequently.	Numeric
	26	Owner Details	It checks if the website's owner provided the details. Most malicious URLs don't provide any details and mostly use rental domains.	Numeric
	27	Global Rank	Since malicious URLs are only active for a short period of time, they have a low rank and check if the website is ranked at the top of 100,000 websites. However, around 25% of malicious URLs are hosted on trustworthy domains, as they generate less suspicion and are more difficult to detect.	Numeric
	28	Total Visits	Since phishing and malware websites live for only a short period of time, the database measures the average number of monthly users visiting a website.	Numeric
	29	Country	It is important to look at the list of countries where the majority of malicious domains are located.	Binary
	30	City	It is important to look at the list of cities where the majority of malicious domains are located.	Binary
HTML	31	Iframe	Iframes are commonly used to embed dynamic content from another website onto a page on your site. iframes inserted in Web pages may have been used by attackers to successfully redirect visitors. As a result, attackers frequently utilize hidden iframes to trick users into visiting a malicious site.	Binary
	32	Mailto	Using Mailto in HTML may result in a number of security issues.	Binary
	33	Webpage size-based	Malicious websites are often comprised of a single lengthy line of HTML code. According to [72], 25% of malicious URLs with distinct domain names included the identical total number of lines of code.	Numeric
JavaScript	34	Popup windows	JavaScript Window Open() pop-ups are often used for advertisements and injecting exploits and used mostly by spammers.	Binary
	35	JavaScript functions	The native JavaScript functions that are often Web-based malware distribution. escape(), eval(), unescape(), exec(), and search() functions.	Binary
	36	DOM functions	The DOM structure of a webpage may be manipulated by an attacker using JavaScript. Attackers use DOM elements to get unauthorised access to user data. This study checked the source code for DOM functions like appendChild and createElement to see if they were present in a webpage.	Binary
	37	JavaScript obfuscation, suspicious functions	Attackers employ obfuscation techniques that make code analysis more complex to avoid detection. There are a number of suspicious functions like ActiveXObject, CreateObject, CreateTextFile, FileSystemObject, and FileExists that may be used to access files and directories as well as to create a backdoor to observe computer activity.	Binary
Certificate	38	Certificate	These class are used to check and confirm the certificate's existence on the server side in order to provide a secure connection.	Binary

The feature's values could be predefined (fixed) or dynamic according to the value specified for a class. In the malicious URL detection methods machine learning and deep learning methods are define the dynamic value for the classes after training the dataset and allocate a range for it. The predefined methods are determined and set the values and

the rules of the classes before the feature extraction [67]. This method is assigned to overcome the data-dependent limitations of learning methods.

The predefined method assists in creating a range of classes and overcoming the majority of the limitations experienced by current malicious URL detection methods. The predefined values are generated and configured based on the static features and characteristics of URLs that are collected according to various conditions in multiple datasets [68]. It assigns the value for the classes with the greatest performance to detect malicious websites, and its performance is independent of the quality of the dataset. The predefined values of the classes in table 3 are stored in the GitHub account [69].

Afterward, it does certain comparisons based on the predefined classes and returns a result. Each class has a value between 1 and 5, which means 5 indicates that the URL is more likely to be malicious and 1 indicates that it is benign. Due to the comparison, each class should return a value. Finally, the overall outcome will be determined by averaging the class values, and a feature should return a final value of 1 to 5. These values will be sent to the next phase for detection.

Although obtaining so many additional classes is possible and may provide a new perspective on the research, it is overwhelming, increases the response time, and may provide some security challenges. Furthermore, due to the fact that this framework is data-independent, new classes can be added easily without having to retrain the entire framework.

C. MALICIOUS URL DETECTION

The detection framework was developed using the quantity of accessible data and the total amount of data and is constructed as follows. The proposed malicious URL detection framework (DF) is based on a predefined static feature classification method [68], [70] and is presented in equation 1.

$$DF = \sum_{i=1}^{i=n} (F_i * 20) \tag{1}$$

F_i represents the feature value, which i={1, 2,..., n} shows the number of features and the total value is multiplied by 20 to determine the wealth of each feature, which is out of 100. DM is compared to the threshold value, which is 200; if it exceeds the threshold, the URL is malicious; otherwise, it is benign. The rule below is the feature evaluation method which is implemented in this research.

Rule:

if any $F_i = -1$ *and other* $F_i \geq 3.5$ *or blacklist* ≥ 3 *then* F_i *value assign to greatest feature value*

This method evaluates the value delivered from features. It determines if all of the features deliver value for the final calculation, and if any of them fails to deliver a value, the method will decide to use the other feature's value instead for other features according to various conditions. The F_i represents the feature value and is obtained by using Equation 2.

$$F_i = \sum_{i=1}^{i=n} (C_i) \tag{2}$$

C_i indicates the value of the class, where i = {1, 2,..., n} displays the number of classes of a feature, and each class can have a value of 1, 3, or 5 depending on the numerous comparisons and conditions that are present in [69]. The predefined values are assigned a range of values for the classes to do analyses and comparisons and return a result.

Drawbacks of the existing secure QR code scanners were discovered through observation and experimentation. Feature evaluation method was applied in this phase to enhance the detection accuracy of malicious URLs, which solved the majority of detection problems. This method evaluates the value delivered from features in the feature classification phase. It checks to see if all the features provide the value for final calculation (except lexical feature that always return a value), and if any of them fails to deliver a value, it needs to provide future actions. This scenario might play out if an application programming interface (API) fails to respond or if a server unexpectedly goes down. This is the novelty of this research, which includes an overwhelming calculation, and the framework will decide to use the other feature's value instead for detection. Even if two features do not respond, this framework is able to detect malicious URLs with a high level of accuracy. Figure 2 shows the pseudocode of the malicious detection framework.

IV. PERFORMANCE EVALUATION AND EXPERIMENTAL SETUP

In this section, we evaluate the performance of the QsecR framework. First, we introduce the experimental dataset used in this research. The dataset contains 4000 real-world random URLs. Second, we present evaluation metrics, which include five metrics and equations. Third, we assess a comprehensive systematic review of QR code scanners and compare several applications from security and privacy perspectives. Fourth, we present the design and development of the QsecR application. Fifth, we evaluate the performance of QsecR and benchmark it with other secure QR code scanners using the proposed dataset. Sixth, we discuss the factors contributing that lead to QsecR's outperformance of other scanners in terms of accuracy.

A. DATASET

The proposed malicious URL detection framework utilizes a training dataset that includes 5,500 samples. The 1500 malicious URLs were collected from a malicious URL dataset in Kaggle that was collected from 2020 to 2022, and 4000 benign samples were collected from the top 4000 site links of Alexa in 2022.

The experimental dataset contains 4000 real-world URLs that were gathered recently and contains 2000 benign and 2000 malicious URLs (1000 phishing and 1000 malware).

Malicious URL Detection Framework

Input: The value of each feature read from feature classification phase

Output: The outcome value (X) compared to the threshold value of 200 to determine if a URL is malicious or benign

Variable:

B_v Blacklist feature value
L_v Lexical feature value
H_v Host-based feature value
C_v Content-based feature value
X The total value

Begin

1: If B_v=-1 // If blacklist feature return -1
2: If L_v or H_v or C_v >= 3.5 // If other features value are equal or greater than 3.5
3: L_v or H_v or C_v ←B_v // The blacklist feature value will use for other feature with highest value
4: End if
5: End if
6: Else B_v=0 // If B_v=-1 and other features value are less than 3.5, the value of B_v is 0
7: X=X+ B_v*20 // Sum the value of features by multiplying to 20 (the worth of each level is out of 100)
8: X=X+L_v*20 // Lexical feature always return a value
9: If H_v=-1 // If host-based feature return -1
10: If L_v or C_v>= 3.5 or B_v>=3 // If other features values are equal or greater than 3.5
11: L_v or B_v or C_v ← H_v // The host-based feature value use for other feature with highest value
12: End if
13: End if
14: Else H_v=0 // If other features values are less than 3.5, the value of H_v is 0
15: X=X+ H_v*20 // Sum the value of features by multiplying to 20 (the worth of each level is out of 100)
16: If C_v=-1 // If content-based feature return -1
17: If L_v or H_v>= 3.5 or B_v>=3 // If other features value are equal or greater than 3.5
18: L_v or H_v or B_v ← C_v // The content-based feature value use for other feature with highest value
19: End if
20:End if
21: Else C_v=0 // If other features values are less than 3.5, the value of C_v is 0
22: X=X+C_v*20 // Sum the value of features by multiplying to 20 (the worth of each level is out of 100)
23: If X>=200 // If the total value of the all features is equal or smaller than 200
24: Return URL is malicious
25:Else // If the total value of the all features is greater than 200
26: Return URL is benign

End

FIGURE 2. Pseudocode of the malicious detection framework.

The malicious URLs were collected from two of the most well-known malware and phishing databases, URLhaus and PhishTank [72], [73]. The CSV format is used to store the entire dataset [59]. Each URL in the dataset is labelled as malicious or benign.

All the websites have gathered for the dataset are online, and the servers are responding, which can help to evaluate our research in real time. The URLs are checked by different tools for verification and labelled as benign or malicious. For evaluating the framework's performance, a variety of URL characteristics were picked for this dataset, such as shortened URLs, URLs with IP addresses, obfuscated URLs, extremely lengthy and short URLs, and URLs that were redirected more than twice (Table 4).

TABLE 4. Types of challenging URLs used in dataset.

Types of Challenging URLs Used in Dataset	Description
Shortened URLs	The URLs that used by shortening services (Bitly) to reduce the number of characters.
URL redirection	URLs are redirected to the other domain. This dataset made use of URLs that were redirected 5 times.
URL with IP address	URLs that employ IP addresses rather than domain names are more likely to be malicious and exploited in drive-by download attacks.
Lengthy URLs	Types URLs that used long domain name and are hard to detect, which mostly used by phishing attacks.
Obfuscated URLs	types of URLs that were obfuscated using various techniques.

B. EVALUATION METRICS

We compared the results using the confusion matrix, a table designed to visualize the performance of QR link detection (Table 5). It includes the following prediction quality measures:

- **True Positive (TP)** indicates the number of URLs that were correctly detected as malicious.
- **True Negative (TN)** indicates the number of benign URLs correctly detected as benign.
- **False Positive (FP)** indicates the number of benign URLs that were incorrectly detected as malicious.
- **False Negative (FN)** indicates the number of malicious URLs that were incorrectly detected as benign.

TABLE 5. Confusion matrix.

Actual Class		True Class: Malicious	True Class: Benign
Tested Class	Malicious	True Positives	False Positives
	Benign	False Negatives	True Negatives

Besides, to comprehensively represent QsecR performance, we evaluate it using some metrics which are: Accuracy (Acc), False Positive Rate (FPR), Precision (Pre), Recall (Rec), and F-1 score (F1). The evaluation metrics are shows in Figure 3.

C. COMPARISON OF QR CODE SCANNERS

In this section, we present a comprehensive systematic review of QR code scanner applications on Android and evaluate

Metrics	Definition	Equation
Accuracy	Acc reflects the correct classification proportion.	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
Precision	Pre represents the percentage of malicious URLs which we detect correctly in all the predicted malicious URLs.	$\text{Precision} = \frac{TP}{TP + FP}$
False Positive Rate	FPR states the percentage of predicted malicious URLs which are truly legitimate in all the legitimate URLs	$FPR = \frac{FP}{FP + TN}$
Recall	Rec denotes the percentage of malicious URLs we detect successfully in all the malicious URLs.	$\text{Recall} = \frac{TP}{TP + FN}$
F-1 score	F1 is the harmonic average of the recall and the precision rate	$F1 = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}$

FIGURE 3. Evaluation metrics.

several scanner applications from the security and privacy perspective. These apps were selected randomly based on previous researches, popularity, and security features. Unfortunately, related articles that developed secure QR code scanner do not provide public access for evaluation of the applications [24], [28].

Table 6 shows the details of the 15 secure QR code scanner applications as well as QsecR that were compared [74]. The items that are considered are version, number of downloads, users' rate, and security features. The information in this table demonstrates the lack of security features in various scanners that have received millions of downloads. These scanners do not provide adequate protection to safeguard users from potential threats [75], [76].

The next step is to evaluate the secure QR code scanners which provide security features. Table 7 illustrates the components that were evaluated for these applications. These components are check URL, display URL, get full URL (redirect), directly open URL, URL checking method, and detection framework. As seen in this table, the majority of secure QR code scanners do not meet baseline security requirements. [9], [25] present the criteria to develop a secure and usable QR code scanner. [58], [77], [78], [79], [80], [81], [82] are unable to redirect a URL and obtain the full URL. Even though some of them [77], [79] immediately access the website without user permission. Furthermore, the major detection method utilized in these applications is the blacklist method which has the lowest accuracy in detecting malicious URLs, and hardly can discover scanners that employ alternative detection methods [32].

The malicious URLs have been generated to evaluate the security strength of QR code scanners. The Zoo [87] and Zphisher [88] toolkits have been used to generate malicious URLs for phishing and malware propagation toolkits on GitHub [69]. The Zoo is a very popular malware repository for analysis and enables researchers who are interested to evaluate around 350 live malware projects. It contains 11 android destructive tools in the different categories of viruses, botnets, Trojans, and ransomware. Since Dendroid and Android Spy iBanking have the Android application file format (APK), they are selected from these tools and are suitable to test the QR code scanners. The Dendroid is a trojan Android remote administration toolkit that provides a variety of spying options such as taking photos, downloading pictures, recording audio, recording video, recording calls, sending texts, and more, as well as getting full permission from the user during installation [89]. The Android Spy iBanking is a botnet and is interesting for spying users' specific capabilities, including SMS messages, redirecting incoming voice calls, and capturing audio and video. It uses various banking Trojans in an effort to bypass a mobile two-factor authentication method.

Also, Zphisher provides multiple updated phishing websites and allows users to perform phishing attacks on several sites and social media, such as Facebook, Twitter, PayPal, Instagram, Netflix, and many more. In this study, PayPal website was selected to evaluate the efficiently of the QR code scanners to detect phishing attacks.

Finally, the most well-known URL shortening websites, Bitly [90] and Rebrandly [91], were utilized to generate short URLs to evaluate the usability of existing scanners. The main reason for using short URLs is that the most secure QR code scanners (malicious URL detection frameworks) cannot detect URL redirection, and those that can are not able to detect it two times.

Table 8 demonstrates the security performance of QR code scanners as well as QsecR. There are nine malware and phishing QR codes utilizing obfuscation techniques to evaluate the effectiveness of these scanners. Figure 4 shows the result of this evaluation and the number of detected phishing and malware QR codes by apps. The outcome reveals that [32] are capable of detecting some harmful websites, and [57] are suspicious about phishing attacks, while others lack the ability to detect malicious URLs. Also, our proposed scanner detected all the phishing and malware samples.

Existing secure QR code scanners are susceptible to flaws for a number of reasons. First, current scanners are incapable of redirecting shortened URLs and send the fake URL for detection. Second, the primary detection method of existing scanners is blacklist-based, which is incapable of identifying newly created or obfuscated malicious URLs. Third, the scanners that employ machine learning for detecting malicious URLs utilize weak feature classification and do not implement any method if the features do not deliver value for the final calculation.

The other weakness of some scanners is that they feature a suspicious-URL option that might make it difficult to distinguish between malicious and safe links. However, QsecR solves this weakness.

Furthermore, we have reviewed all the applications based on the permissions they require during installation. The safety of QR code scanner applications is a major concern for QR code security [20], [29]. There are several potential threats to the privacy of Android devices, but the most severe is an

TABLE 6. Qualitative comparison of QR code scanners.

App Developer	Version	Size (MB)	Downloads	Rate	Security Features
[75]	2.2.12	5.2	100M+	4.7	N/A
[78]	2.7.1-L	2.72	100M+	4.6	URL Checking
[76]	2.2.8.GP	6.26	10M+	4.8	N/A
[77]	1.1.0	8.7	500K+	4.7	URL Checking
[57]	1.0.3.14d044e2	3.2	10K+	2.9	URL Checking
[79]	1.8.4.260	18.2	5M+	4.5	URL Checking
[80]	7.0.6	10.1	50M+	4.3	URL Checking & Encryption
[81]	1.1.0	1.9	100K	4.3	URL Checking
[82]	9.6.3434	21	1M+	4.3	URL Checking
[83]	1.3	5.85	100K+	4.3	N/A
[84]	1.0.18	10	100K+	4.5	N/A
[85]	1.0.3.18	2.16	100+	4.3	N/A
[86]	1.0.2	21.1	10+	-	N/A
[32]	1.3	2.7	-	-	URL Checking & Encryption
[58]	1.0.0	10.63	10+	-	URL Checking
[59] **QsecR**	1.0.0	3.2	-	-	URL Checking

TABLE 7. Evaluate security features of QR code scanners.

APP Developer	Check URL	Display URL	Get Full URL	Direct Open	URL Checking Methods	Detecting Methods
[78]	Yes	Yes	No	No	Google Safe Browsing	Blacklist
[77]	Yes	Yes	No	If safe Yes unless No	Trend Micro security	Blacklist
[57]	Yes	Yes	Yes	No	G Data security	Blacklist
[79]	Yes	No	No	Yes	Kaspersky Virus desk	Blacklist
[80]	Yes	Yes	No	No	Google Safe Browsing	Blacklist
[81]	Yes	No	No	No	Cheetah Mobile browser	Blacklist
[82]	Yes	Yes	No	No	Intelligence Sophos Lab security	Blacklist
[32]	Yes	Yes	Yes	No	Google Safe Browsing, PhishTank and lexical feature	Blacklist and machine learning
[58]	Yes	Yes	No	No	Lionic malicious web cloud database.	Blacklist
[59] **QsecR**	Yes	Yes	Yes	No	Redirection, Blacklist, Lexical, Host-based, Content-based	Novel technique

application's request for excessive permissions [50], which may lead to gaining access control over the entire smartphone and acquiring entry to the user's sensitive data.

The necessary permissions that should be requested from applications are camera and internet (Wi-Fi, network) [9]. Table 9 shows the requested access from the applications.

- *__Camera (Cam):__ Take pictures and videos.
- __Storage (Stg):__1- Read the contents of your USB storage.2- modify or delete the contents of your USB storage.
- __Location (Loc):__1-Approximate location (network-based), 2-Precise location (GPS and network-based).
- __Contacts (Cont):__1- Read your contacts. 2- Modify your contacts.
- *__Wi-Fi Connection Information (wi-fi):__1- View Wi-Fi connects. 2- Connect and disconnect from Wi-Fi. 3- Allow Wi-Fi multicast Reception.
- __Photos/ Media/ Files (Files):__1- Read the contents of your USB storage. 2- Modify or delete the contents of your USB storage.
- __Phone (Ph):__1- Read phone status and identity. 2- Directly call phone numbers.
- __Device ID & Call Information (DevID):__ Read phone status and identity.
- __Device & APP History (DevHis):__ Read your web bookmark and history.
- __Calendar (Cal):__ Read calendar events plus confidential information.
- *__Network (Net):__1- Full network access. 2- View network connections. 3- Receive data from the internet. 4- Change network connectivity.
- __Others (Oths):__1- Control Flashlight. 2-Control vibration. 3- Prevent device from sleeping 4- Disable your screen lock. 5- Run at startup. 6- Draw over other apps 7- Install shortcuts 8- Google play license 9- microphone.

TABLE 8. Evaluate the security performance of QR code scanners utilizing nine malware and phishing QR codes.

APP Developer	Dendroid QR code	Dendroid Bitly QR code	Dendroid Rebrandly QR code	iBanking QR code	iBanking Bitly QR code	iBanking Rebrandly QR code	PayPal original tunnel QR code	PayPal Bitly QR code	PayPal QR code generator
[78]	No	No	No	No	No	No	No	No	No
[77]	*Un	No	No	Un	No	No	No	No	No
[57]	No	No	No	No	No	No	†Sus	Sus	Sus
[79]	No	No	No	No	No	No	No	No	No
[80]	No	No	No	No	No	No	No	No	No
[81]	No	No	No	No	No	No	No	No	No
[82]	No	No	No	No	No	No	No	No	No
[30]	Yes	Yes	No	Yes	Yes	No	Yes	Yes	No
[58]	No	No	No	No	No	No	No	No	No
[59] **QsecR**	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

TABLE 9. Requested permission from applications during installation.

APP Developer	Cam	Stg	Loc	Cont	wi-fi	Files	Ph	DevID	DevHis	Cal	Net	Oths
[75]	□	□	□		□	□					□	□
[78]	□	□	□	□	□						□	□
[76]	□	□			□						□	□
[77]	□	□		□	□						□	□
[57]	□	□	□		□						□	□
[79]	□	□			□					□	□	□
[80]	□	□	□	□	□						□	□
[81]	□	□		□	□		□				□	□
[82]	□	□	□	□	□	□	□	□			□	□
[83]	□	□	□		□		□				□	□
[84]	□	□			□		□	□	□		□	□
[85]	□	□			□						□	□
[86]	□	□	□	□	□		□			□	□	□
[30]	□	□			□						□	
[58]	□	□		□	□						□	□
[59]**QsecR**	□				□						□	

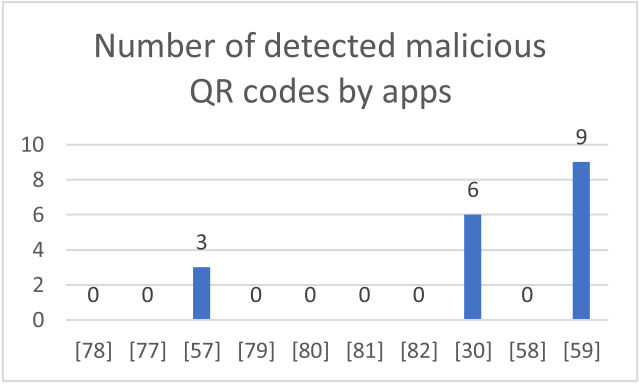

FIGURE 4. The number of detected phishing and malware QR codes by secure scanners.

According to [8] and [9], save privacy QR code scanner applications with the least privilege permission must only request access to the camera, Wi-Fi, and network. However, some of these applications request unusual permission. Most applications request access to storage to read, modify, or delete the contents. Reference [77] request for microphone access; [80] asking for Bluetooth permission; [57], [75], [78], [80], [82], [83], [86] requesting access to the location. Reference [81] asking for uncommon permissions to read contacts, call, SMS, and draw over the other apps' modified system settings. Reference [82] requests extremely abnormal access to draw other apps, control near-field communication, retrieve and run other apps, and read phone status and identity. Reference [83] request access to the telephone for the purpose of directly calling phone numbers. Reference [79] and [86] requesting permission to access the calendar.

The proposed QR code scanner (QsecR) [59] is one of the most privacy-friendly application with the least privilege permission, which only requests access to the camera, Wi-Fi, and network. After that is [32] with request to access the camera, Wi-Fi, storage, and network.

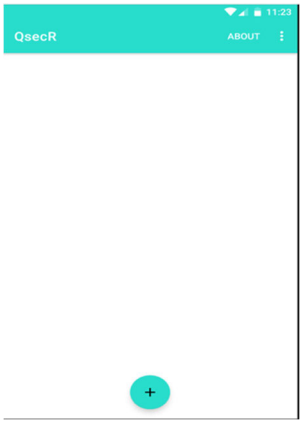

FIGURE 5.1

FIGURE 5.2

FIGURE 5.3

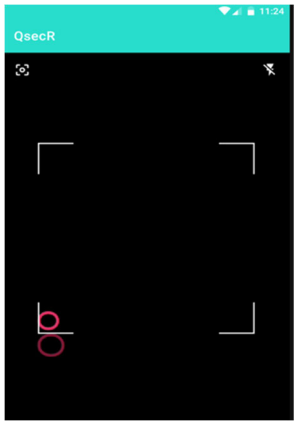

FIGURE 5.4

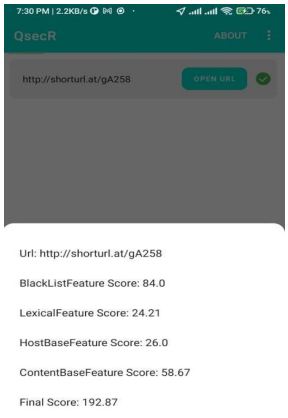

Figure 5.5

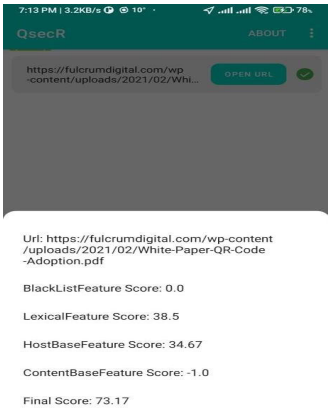

Figure 5.6

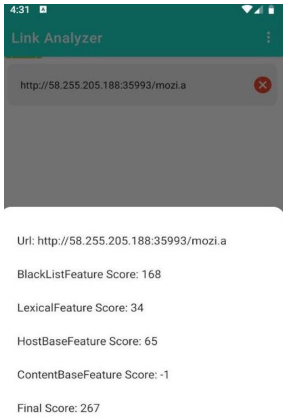

Figure 5.7

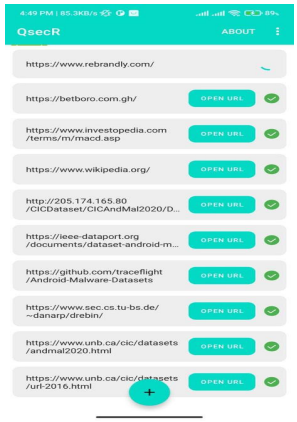

Figure 5.8

FIGURE 5. The interface, features, and process of detecting malicious URLs of QsecR.

D. THE QsecR APPLICATION

According to the recommendations presented in the [9], [26], and [27], we have developed the QsecR application (available at the GitHub account [59]). The application is developed based on the MVVM architecture in Kotlin programming language on Android. The QR code scanner was developed according to the 2.1.2 library based on ZXing [93]. Figure 5 shows the interface of QsecR, its features, and the process of detecting malicious URLs.

Figure 5.1 presents the main interface of the QsecR application. This application contains an insert button that is able to manually insert a URL or scan a QR code, as shown in Figure 5.2. The Figure 5.3 demonstrate the process of manually inserting the website for detecting malicious URLs. Figure 5.4 illustrates the process of scanning a QR code using the scanner. If the application detects that the QR code contains text, it shows the message directly. The Figure 5.5 shows the result of scanned QR code. The application detected that the website is benign. The application shows the detection details by clicking on the website name. This part shows each feature's value and the final score. Since the final score is less than the threshold value of 200, it is benign. Furthermore, since it is a safe website, the application allows the user to open the link by clicking on the open URL button. Figure 5.6 shows the procedure for detecting a URL when a feature does not respond. In this part, the content-based feature did not respond because the server was down and could not download the website and extract information. Since the other features did not detect that the URL was malicious (according to Figure 2), it did not employ the other features' value for the content-based feature. In the same situation, Figure 5.7 shows the procedure for detecting malicious URLs when the content-based feature does not respond. Since the blacklist features detect that the URL is malicious (according to Figure 2), the value of content-based is employed for the blacklist feature. Also, the application does not allow the user to open the link. Figure 5.8 shows the result of scanning several QR code images that remained in the history.

E. PERFORMANCE EVALUATION

The performance of secure QR code scanners was evaluated in the previous section. The outcome reveals that only Barsec

[32] is capable of detecting some harmful websites, while others lack the ability to detect malicious URLs in real time. To show the feasibility of the QsecR scanner, we evaluate the performance of QsecR and Barsec utilizing the proposed dataset, which contains 4000 real-world URLs. The confusion matrix of QsecR is shown in Table 10. The confusion matrix is used to measure the detection performance of a scanner. The QsecR correctly detected 1876 benign URLs out of 2000 and with 124 undetected websites. Also, it correctly detected 1780 malicious URLs out of 2000 phishing and malware websites and only could not identify 130 websites, which shows great detection performance.

The confusion matrix of BarSec is illustrated in Table 11. It accurately identified 1546 out of 2000 benign URLs and 454 false positives. In addition, it accurately identified 1599 malicious URLs out of a total of 2000 websites, with a false-negative rate of 401.

TABLE 10. Confusion matrix of QsecR.

	Detect Benign URLs	Detect Malicious URLs
Benign URLs	1876	124
Malicious URLs	130	1780

TABLE 11. Confusion matrix of BarSec.

	Detect Benign URLs	Detect Malicious URLs
Benign URLs	1546	454
Malicious URLs	401	1599

The details of the performance evaluation of the QsecR and Barsec are shown in Table 12 and Figure 6 illustrate. As shown in this table, the proposed QR link security detection framework outperforms the other scanners. The accuracy and precision of QsecR are 93.50% and 93.80%, respectively, compared to Barsec, whereas they are 78.63% and 77.30%. The findings demonstrate that QsecR improved the malicious URL detection accuracy of Android QR code scanners.

F. DISCUSSION

In this research, we propose QsecR, a secure, privacy-friendly, and usable QR code scanner, according to a data-independent malicious URL detection framework. It outperforms other scanners by an accuracy of 93.50 %, and the factors contributing to that will be discussed here.

The redirection section plays a key role in detecting obfuscated URLs (short URLs) and sending the original URLs to the feature extraction and classification part, which evaluates the original URLs by features. If a URL does not redirect and send to these features, some of them will wrongly evaluate the website and provide an incorrect value, resulting in a high false-positive rate.

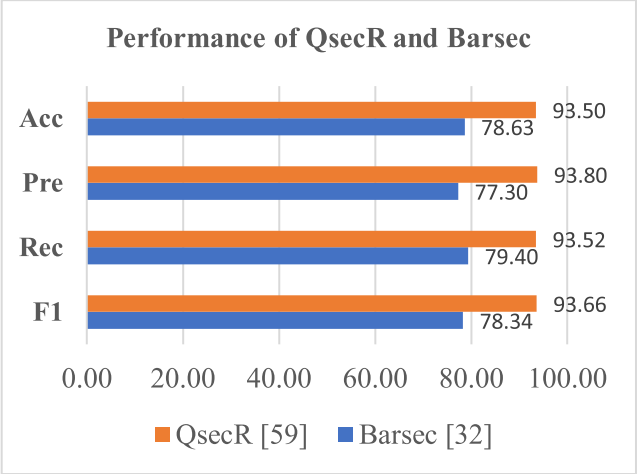

FIGURE 6. The performance of the QsecR and Barsec.

The other essential component of the malicious URL detection framework is effective feature classification. The feature classification is based on a predefined static classification, which assigns a value to the classes with the greatest performance in detecting malicious websites. The predefined method overcomes the majority of the limitations experienced by current malicious URL detection methods. Also, we employ 39 classes of blacklist, lexical, host-based, and content-based features. The classes are selected according to their importance and effectiveness in detecting malicious URLs. Although it is possible to implement over 150 different classes and this may improve the detection accuracy slightly, it is overwhelming, increases the response time, requires complicated calculations, and may present some security challenges.

The crucial aspect of this detection framework is the evaluation of the value that is delivered by features and situations where the features do not deliver a value, which is hard to find in the other frameworks. This scenario may play out if an API call is unsuccessful or if a server goes down. It will employ alternative feature values for detection based on a variety of circumstances.

The other important principle that sets QsecR (malicious URL detection framework) apart from other scanners is that it classifies websites exclusively as harmful or benign (not a third option such as suspicious).

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed QsecR, a secure and privacy-friendly QR code scanner. It was according to a data-independent malicious URL detection framework based on predefined static feature classification. We employed 39 classes of blacklist, lexical, host-based, and content-based features. Furthermore, we implemented a feature evaluation

TABLE 12. Detail performance evaluation of the QsecR and Barsec.

QR code scanner	Acc (%)	Pre (%)	FPR	Rec	F1
BarSec[32]	78.63	77.30	0.2111	79.40	78.34
QsecR [59]	**93.50**	**93.80**	**0.0651**	**93.52**	**93.66**

method that evaluated the value that was delivered by features and utilized the value of other features in the absence of value from any of the other features. We evaluated the performance of the proposed scanner using 4000 real-world datasets and compared the result with other secure QR code scanners in terms of security and privacy. The result showed QsecR outperforms others with accuracy and precision of 93.50% and 93.80%, respectively. Furthermore, it is one of the most privacy-friendly application with the least privilege permission.

Although the proposed framework performs well, further improvement is still required along with further studies to improve the entire system. Our objective is to improve the accuracy of malicious URL detection by assigning a priority coefficient to the classes and features according to their level of importance. Besides, enhance the feature classification by expanding the number of classes to get a more accurate result in the detection of malicious URLs. Also, another potential research agenda is evaluating secure QR code scanners in terms of response time. Last but not least, future research should focus on conducting comprehensive measurements of device resource usage performance. Specifically, an investigation into memory utilization, network usage, CPU usage, etc. Measuring device resources during experiments allows researchers to assess the performance and efficiency of applications and frameworks and how they operate under various conditions. This data helps identify bottlenecks, optimize resource allocation, and enhance overall system performance.

REFERENCES

[1] D. Curry. (2022). *Android Statistics*. Accessed: Sep. 14, 2022. [Online]. Available: https://www.businessofapps.com/data/android-statistics/#:~:text=Android%20has%2038%20percent%20market,had%2045%20percent%20market%20share

[2] Kapil. (2021). *Top Most Google Play Store App Statistics*. Accessed: Sep. 14, 2022. [Online]. Available: https://www.teamtweaks.com/blog/play-store-app-statistics-2021/

[3] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of Android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116363–116379, 2020, doi: 10.1109/ACCESS.2020.3002842.

[4] TheBestVPN.com. (2020). *Vulnerability Alerts*. [Online]. Available: https://thebestvpn.com/vulnerability-alerts/

[5] Denso Wave. (2022). *Quick Response Code (QR Code)*. [Online]. Available: https://www.denso-wave.com/en/

[6] R. Focardi, F. L. Luccio, and H. A. M. Wahsheh, "Usable security for QR code," *J. Inf. Secur. Appl.*, vol. 48, Oct. 2019, Art. no. 102369, doi: 10.1016/j.jisa.2019.102369.

[7] A. S. Rafsanjani, "Comparison cover image of digital watermarking based on discrete cosine transform by using quick response code," in *Proc. 1st Int. Conf. Emerg. Trends Eng., Technol. Social Sci. (ICETS)*, 2018, pp. 1–9.

[8] M. S. Al-Zahrani, H. A. M. Wahsheh, and F. W. Alsaade, "Secure real-time artificial intelligence system against malicious QR code links," *Secur. Commun. Netw.*, vol. 2021, Dec. 2021, Art. no. 5540670, doi: 10.1155/2021/5540670.

[9] H. A. M. Wahsheh and F. L. Luccio, "Security and privacy of QR code applications: A comprehensive study, general guidelines and solutions," *Information*, vol. 11, no. 4, p. 217, Apr. 2020, doi: 10.3390/INFO11040217.

[10] H. Yao and D. Shin, "Towards preventing QR code based attacks on Android phone using security warnings," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, May 2013, pp. 341–346.

[11] Kaspersky Lab. (2011). *Malicious QR Codes: Attack Methods & Techniques Infographic*. [Online]. Available: https://usa.kaspersky.com/about/press-releases/2011_malicious-qr-codes-attack-methods–techniques-infographic

[12] L. Tao. (2017). *QR Code Scams Rise in China, Putting E-Payment Security in Spotlight*. [Online]. Available: https://www.scmp.com/business/china-business/article/2080841/rise-qr-code-scams-china-puts-online-payment-security?module=perpetual_scroll&pgtype=article&campaign=2080841

[13] D. K. Mondal, B. C. Singh, H. Hu, S. Biswas, Z. Alom, and M. A. Azim, "SeizeMaliciousURL: A novel learning approach to detect malicious URLs," *J. Inf. Secur. Appl.*, vol. 62, Nov. 2021, Art. no. 102967, doi: 10.1016/j.jisa.2021.102967.

[14] G.-J. Chou and R.-Z. Wang, "The nested QR code," *IEEE Signal Process. Lett.*, vol. 27, pp. 1230–1234, 2020, doi: 10.1109/LSP.2020.3006375.

[15] A. Zhou, G. Su, S. Zhu, and H. Ma, "Invisible QR code hijacking using smart LED," in *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 3, pp. 1–23, Sep. 2019, doi: 10.1145/3351284.

[16] A. Averin and N. Zyulyarkina, "Malicious Qr-code threats and vulnerability of blockchain," in *Proc. Global Smart Industry Conf. (GloSIC)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Nov. 2020, pp. 82–86, doi: 10.1109/GloSIC50886.2020.9267840.

[17] R. Focardi, F. L. Luccio, and H. A. Wahsheh, "Security threats and solutions for two-dimensional barcodes: a comparative study," in *Computer and Network Security Essentials*. 2018, pp. 207–219.

[18] R. Chiramdasu, G. Srivastava, S. Bhattacharya, P. K. Reddy, and T. R. Gadekallu, "Malicious URL detection using logistic regression," in *Proc. IEEE Int. Conf. Omni-Layer Intell. Syst. (COINS)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Aug. 2021, pp. 1–17, doi: 10.1109/COINS51742.2021.9524269.

[19] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. F. Cranor, and N. Christin, "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*. Berlin, Germany: Springer, 2013, pp. 52–69.

[20] A. Kusyanti and A. Arifin. (2017). *QRishing: A User Perspective*. [Online]. Available: https://www.ijacsa.thesai.org

[21] K. S. C. Yong, K. L. Chiew, and C. L. Tan, "A survey of the QR code phishing: The current attacks and countermeasures," in *Proc. 7th Int. Conf. Smart Comput. Commun. (ICSCC)*, Jun. 2019, pp. 1–5.

[22] V. Mavroeidis and M. Nicho, "Quick response code secure: A cryptographically secure anti-phishing tool for QR code attacks," in *Computer Network Security* (Lecture Notes in Computer Science), vol. 10446. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-65127-9.

[23] V. Sharma, "A study of malicious QR codes," *Int. J. Comput. Intell. Inf. Secur.*, vol. 3, no. 5, pp. 1–15, 2012.

[24] J. Song, K. Gao, X. Shen, X. Qi, R. Liu, and K.-K.-R. Choo, "QRFence: A flexible and scalable QR link security detection framework for Android devices," *Future Gener. Comput. Syst.*, vol. 88, pp. 663–674, Nov. 2018, doi: 10.1016/j.future.2018.05.082.

[25] R. Dudheria, "Evaluating features and effectiveness of secure QR code scanners," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery, CyberC*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Jul. 2017, pp. 40–49, doi: 10.1109/CyberC.2017.23.

[26] K. Krombholz, P. Frühwirt, T. Rieder, I. Kapsalis, J. Ullrich, and E. Weippl, "QR code security—How secure and usable apps can protect users against malicious QR codes," in *Proc. 10th Int. Conf. Availability, Rel. Secur.*, Aug. 2015, pp. 230–237, doi: 10.1109/ARES.2015.84.
[27] N. Hegde, R. Bharti, and R. Sur. (2018). *Anti-Malware Phishing QR Scanner*. [Online]. Available: https://www.ijisrt.com
[28] A. Y. Alnajjar, M. Anbar, S. Manickam, O. Elejla, and H. El-Taj, "QRphish: An automated QR code phishing detection approach," *J. Eng. Appl. Sci.*, vol. 11, no. 3, pp. 553–560, 2016.
[29] K. A. Latif, "Anti-qrishing real-time technique on the QR code using the address bar-based and domain-based approach on smartphone," *Int. J. Cyber-Security Digit. Forensics*, vol. 8, no. 2, pp. 134–143, 2019. [Online]. Available: http://125.98.3.123/fake.html
[30] H. Wahsheh and F. Luccio, "Evaluating security, privacy and usability features of QR code readers," in *Proc. 5th Int. Conf. Inf. Syst. Secur. Privacy*, 2019, pp. 266–273, doi: 10.5220/0007346202660273.
[31] T. Yuan, Y. Wang, K. Xu, R. R. Martin, and S.-M. Hu, "Two-layer QR codes," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4413–4428, Sep. 2019, doi: 10.1109/TIP.2019.2908490.
[32] H. Wahsheh. (2018). *BarSec Droid*. Accessed: Jul. 28, 2022. [Online]. Available: https://m.apkpure.com/barsec-droid/barcode_security.heider.bsr
[33] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," 2018, *arXiv:1802.03162*.
[34] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," 2017, *arXiv:1701.07179*.
[35] *Google Safe Browsing*. Accessed: Jul. 28, 2022. [Online]. Available: https://safebrowsing.google.com/
[36] (2022). *PhishTank*. [Online]. Available: https://www.phishtank.com/
[37] Hispasec Sistemas. (2022). *VirusTotal*. [Online]. Available: https://www.virustotal.com/gui/home/url
[38] C. Rupa, G. Srivastava, S. Bhattacharya, P. Reddy, and T. R. Gadekallu, "A machine learning driven threat intelligence system for malicious URL detection," in *Proc. 16th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 1–15, doi: 10.1145/3465481.3470029.
[39] N. A. Alfouzan and C. Narmatha, "A systematic approach for malware URL recognition," in *Proc. 2nd Int. Conf. Comput. Inf. Technol. (ICCIT)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Jan. 2022, pp. 325–329, doi: 10.1109/ICCIT52419.2022.9711614.
[40] X. Xiao, D. Zhang, G. Hu, Y. Jiang, and S. Xia, "CNN–MHSA: A convolutional neural network and multi-head self-attention combined approach for detecting phishing websites," *Neural Netw.*, vol. 125, pp. 303–312, May 2020, doi: 10.1016/j.neunet.2020.02.013.
[41] G. Sonowal and K. S. Kuppusamy, "PhiDMA—A phishing detection model with multi-filter approach," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 1, pp. 99–112, Jan. 2020, doi: 10.1016/j.jksuci.2017.07.005.
[42] J. Yuan, Y. Liu, and L. Yu, "A novel approach for malicious URL detection based on the joint model," *Secur. Commun. Netw.*, vol. 2021, pp. 1–12, Dec. 2021, doi: 10.1155/2021/4917016.
[43] D. R. Patil and J. B. Patil, "Feature-based malicious URL and attack type detection using multi-class classification," *Int. J. Inf. Secur.*, vol. 10, no. 2, pp. 141–162, 2018. [Online]. Available: http://www.isecure-journal.org
[44] Y. Liang, Q. Wang, K. Xiong, X. Zheng, Z. Yu, and D. Zeng, "Robust detection of malicious URLs with self-paced wide & deep learning," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 717–730, Mar. 2022, doi: 10.1109/TDSC.2021.3121388.
[45] M. Trevisan and I. Drago, "Robust URL classification with generative adversarial networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 143–146, 2019. [Online]. Available: https://github.com/marty90/URL-generator
[46] Shantanu, B. Janet, and R. J. A. Kumar, "Malicious URL detection: A comparative study," in *Proc. Int. Conf. Artif. Intell. Smart Syst. (ICAIS)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Mar. 2021, pp. 1147–1151, doi: 10.1109/ICAIS50930.2021.9396014.
[47] 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," *J. Netw. Syst. Manage.*, vol. 29, no. 3, p. 21, Jul. 2021, doi: 10.1007/s10922-021-09587-8.
[48] C. Rupa, M. Harshitha, G. Srivastava, T. R. Gadekallu, and P. K. R. Maddikunta, "Securing multimedia using a deep learning based chaotic logistic map," *IEEE J. Biomed. Health Informat.*, vol. 27, no. 3, pp. 1154–1162, Mar. 2023, doi: 10.1109/jbhi.2022.3178629.
[49] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," *Secur. Commun. Netw.*, vol. 2019, pp. 1–15, Oct. 2019, doi: 10.1155/2019/2595794.
[50] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant permission identification for machine-learning-based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.
[51] M. Kim, D. Kim, C. Hwang, S. Cho, S. Han, and M. Park, "Machine-learning-based Android malware family classification using built-in and custom permissions," *Appl. Sci.*, vol. 11, no. 21, p. 10244, Nov. 2021, doi: 10.3390/app112110244.
[52] A. Mathur, L. M. Podila, K. Kulkarni, Q. Niyaz, and A. Y. Javaid, "NATICUSdroid: A malware detection framework for Android using native and custom permissions," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102696, doi: 10.1016/J.JISA.2020.102696.
[53] N. Collier. (2021). *Android Barcode Scanner App on Google Play Infects 10 Million Users With One Update*. [Online]. Available: https://blog.malwarebytes.com/android/2021/02/barcode-scanner-app-on-google-play-infects-10-million-users-with-one-update/
[54] P. Ducklin. (2018). *Crooks Infiltrate Google Play With malware in QR Reading Utilities*. [Online]. Available: https://nakedsecurity.sophos.com/2018/03/23/crooks-infiltrate-google-play-with-malware-lurking-in-qr-reading-utilities/
[55] K. Peng, H. Sanabria, D. Wu, and C. Zhu, "Security overview of QR codes," 2014.
[56] ZXing. (2020). *Zebra Crossing*. [Online]. Available: https://github.com/zxing/zxing
[57] G Data CyberDefense AG. (2021). *G DATA QR Code Scanner*. [Online]. Available: https://www.gdatasoftware.com/
[58] Lionic. (2022). *Lionic Secure QR Code Scanner*. [Online]. Available: https://play.google.com/store/apps/details?id=com.lionic.scanner.qrcode
[59] A. S. Rafsanjani. (2022). *QsecR: Secure QR code Scanner*. [Online]. Available: https://github.com/ahmadsa63/QsecR
[60] Y. Mourtaji, M. Bouhorma, D. Alghazzawi, G. Aldabbagh, and A. Alghamdi, "Hybrid rule-based solution for phishing URL detection using convolutional neural network," *Wireless Commun. Mobile Comput.*, vol. 2021, Sep. 2021, Art. no. 8241104, doi: 10.1155/2021/8241104.
[61] P. Mutchler, A. Doupé, J. Mitchell, C. Kruegel, and G. Vigna, "A large-scale study of mobile web app security," 2015.
[62] M. Alshehri, A. Abugabah, A. Algarni, and S. Almotairi, "Character-level word encoding deep learning model for combating cyber threats in phishing URL detection," *Comput. Electr. Eng.*, vol. 100, May 2022, Art. no. 107868, doi: 10.1016/j.compeleceng.2022.107868.
[63] T. Li, G. Kou, and Y. Peng, "Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods," *Inf. Syst.*, vol. 91, Jul. 2020, Art. no. 101494, doi: 10.1016/j.is.2020.101494.
[64] VirusTotal. (2022). *Analyze Suspicious Files, Domains, IPs and URLs to Detect Malware and Other Breaches, Automatically Share Them With the Security Community*. [Online]. Available: https://www.virustotal.com/gui/home/url
[65] (2022). *WhoXY*. [Online]. Available: https://www.whoxy.com/
[66] DomCop. (2022). *OpenPageRank*. [Online]. Available: https://www.domcop.com/openpagerank/
[67] V. K. Nadar, B. Patel, V. Devmane, and U. Bhave, "Detection of phishing websites using machine learning approach," in *Proc. 2nd Global Conf. Advancement Technol. (GCAT)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Oct. 2021, pp. 1–8, doi: 10.1109/GCAT52182.2021.9587682.
[68] C. do Xuan, H. Dinh Nguyen, and T. Victor Nikolaevich. (2020). *Malicious URL Detection based on Machine Learning*. [Online]. Available: https://www.ijacsa.thesai.org
[69] GitHub-Inc. (2022). *GitHub*. [Online]. Available: https://github.com/
[70] C.-M. Wu, M. Li, L. Ye, X.-C. Zou, and B.-H. Qiang. (2018). *Malicious Website Detection Based on URLs Static Features*. [Online]. Available: http://www.phishtank.com/
[71] S. Kumi, C. Lim, and S. G. Lee, "Malicious URL detection based on associative classification," *Entropy*, vol. 23, no. 2, pp. 1–12, Feb. 2021, doi: 10.3390/e23020182.
[72] (2022). *Urlhaus*. [Online]. Available: https://urlhaus.abuse.ch/

[73] D. Ulevitch. (2022). *PhishTank*. [Online]. Available: https://www.phishtank.com/
[74] A. S. Rafsanjani, D. N. Kamaruddin, A. Sjariff, N. Firdaus, N. Maarop, and H. M. Rusli, "A evaluating security and privacy features of quick response code scanners: A comparative study," *Open Int. J. Informat.*, vol. 10, no. 2, pp. 197–207, 2022, Accessed: Feb. 10, 2023. [Online]. Available: https://oiji.utm.my/index.php/oiji/article/view/201
[75] Gamma-Play. (2021). *QR & Barcode Scanner*. Accessed: Jul. 30, 2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.gamma.scan
[76] InShot-Inc. (2021). *Free QR Scanner—Barcode Scanner, QR Code Reader*. [Online]. Available: https://play.google.com/store/apps/details?id=qrscanner.barcodescanner.barcodereader.qrcodereader
[77] Trend-Micro. (2021). *QR Scanner—Free, Safe QR Code Reader, Zero Ads*. [Online]. Available: https://play.google.com/store/apps/details?id=com.trendmicro.qrscan
[78] TeaCapps. (2021). *QR & Barcode Reader*. Accessed: Jul. 30, 2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.teacapps.barcodescanner
[79] Kaspersky. (2021). *QR Code Reader and Scanner: App for Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.kaspersky.qrscanner&referrer=af_tranid%3DFvrg4d22ypPkZQbxXQKAVQ%26pid%3Dacq%26c%3Dacq-freekasp-COM-QRS-Android%26af_web_id%3Dae14f451-2652-4bf8-b3e3-394a0713e12c-c
[80] DroidLa. (2021). *QR Droid*. [Online]. Available: https://play.google.com/store/apps/details?id=la.droid.qr
[81] Cheetah-Mobile. (2020). *Cheetah Mobile QR code & Bar Code Scanner*. [Online]. Available: https://cm-qrcode.en.aptoide.com/app
[82] *Sophos Intercept X for Mobile*, Sophos-Ltd., U.K., 2021.
[83] Falcon-Security-Lab. (2021). *QR Code Scanner & Barcode Reader*. [Online]. Available: https://play.google.com/store/apps/details?id=com.falcon.barcodescanner
[84] Easy-TechMobile. (2021). *Safe Scanner-Best QR Code Reader, Barcode Scanner*. [Online]. Available: https://play.google.com/store/apps/details?id=app.safe.barcode.qrcode.scanner
[85] A. Unger. (2021). *SafeQR*. [Online]. Available: https://play.google.com/store/apps/details?id=biz.ungerware.safeqr&hl=en&gl=U.S
[86] S. Bifalco. (2021). *SafeQR*. [Online]. Available: https://play.google.com/store/apps/details?id=io.redcel.safeqr&hl=en&gl=U.S
[87] Ytisf. (2014). *TheZoo—A Live Malware Repository*. [Online]. Available: https://github.com/ytisf/theZoo
[88] Htr-Tech. (2021). *Zphisher*. [Online]. Available: https://github.com/htr-tech/zphisher
[89] Lucian Constantin. (2014). *New Crimeware Tool Dendroid Makes it Easier to Create Android Malware, Researchers Warn*. [Online]. Available: https://www.pcworld.com/article/2105500/new-crimeware-tool-dendroid-makes-it-easier-to-create-android-malware-researchers-warn.html
[90] (2022).*Bitly*. [Online]. Available: https://bitly.com/
[91] (2022). *Rebrandly*. [Online]. Available: https://www.rebrandly.com/
[92] Heider Wahsheh. (2021). *BarAI*. [Online]. Available: https://sites.google.com/site/heiderawahsheh/apps
[93] (2022). Accessed: Nov. 2, 2022. *GitHub—Zxing/Zxing: ZXing ('Zebra Crossing') Barcode Scanning Library for Java, Android*. [Online]. Available: https://github.com/zxing/zxing/

AHMAD SAHBAN RAFSANJANI received the Ph.D. degree in computer science, major in network security from Universiti Teknologi Malaysia, Malaysia, in 2022. He is currently a Lecturer with the Department of Computing and Information Systems, School of Engineering and Technology, Sunway University, Malaysia. He has published research papers in journals and conference proceedings. His research interests include network security, data hiding, cryptography, malware analysis, and information security. He is also a member of the IEEE Computer Society.

NORSHALIZA BINTI KAMARUDDIN received the B.S. degree in information technology from Universiti Utara Malaysia and the M.S. degree in computer science and the Ph.D. degree in image processing from the University of Malaya, in 2003 and 2016, respectively. From 2001 to 2018, she was a Lecturer in some private university in Malaysia, before she joined Universiti Technologi Malaysia as a Senior Lecturer, in 2019. Currently, she is also actively doing research on mental health issues based on text analysis and real time series data with machine learning techniques. Her research interests include image processing, machine learning, and artificial intelligence.

HAZLIFAH MOHD RUSLI (Member, IEEE) received the B.S. degree in computer science from the University of Sheffield, U.K., in 1997, and the M.S. and Ph.D. degrees in computer science from Universiti Teknologi Malaysia, in 2003 and 2017, respectively. Her Ph.D. thesis was on web services choreography testing using semantic service description. She is currently a Senior Lecturer with the Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia. Prior to that, she was with Universiti Teknologi Mara, Malaysia, from 2004 to 2020. She has four years of industry experience as a system analyst in the financial sector. Her current research interests include microservices architecture and domain-driven design.

MOHAMMAD DABBAGH received the Ph.D. degree in computer science, specialization in software engineering from the University of Malaya, Malaysia, in 2015. He has acquired enormous working experiences as a lecturer and a researcher in the field of computer science. He is currently a Senior Lecturer with the School of IT and Engineering, MIT Sydney. He has published several research papers in prestigious international journals and conference proceedings. His research interests include but not limited to blockchain, requirements engineering, empirical software engineering, big data analytics, and the Internet of Things. He has been recognized as a Certified Professional in Requirements Engineering by the International Requirements Engineering Board. He is also a Senior Member of the IEEE Computer Society.