National College of Ireland

Detecting Phishing URLs in QR codes using Heuristic Techniques

MSc Research Project
Cyber Security

Khan Ahmer Khalique
Student ID: 21201544

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	AHMER KHALIQUE KHAN
Student ID:	21201544
Programme:	MSCCYB1 **Year:** 1
Module:	RESEARCH IN COMPUTING
Supervisor:	MICHAEL PANTRIDGE
Submission Due Date:	18/09/2023
Project Title:	DETECTING PHISHING URLS IN QR CODES USING HEURISTIC TECHNIQUES
Word Count:	5537 (excluding references) **Page Count** 18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ahmer Khalique Khan

Date: 17/09/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Contents

Detecting Phishing URLs in QR codes using Heuristic Techniques

Ahmer Khalique Khan
21201544

Abstract

Quick Response (QR) codes are the most widely used barcodes today for various purposes including and not limited to payments, shopping, information sharing, etc. With massive application base, QRs become an easy target of threat actors. Since QR codes are easy to generate, a malicious user can easily generate one that looks genuine but stores the URL of a phishing website. There are many tools and techniques available today to tackle phishing which fall under heuristic, visual-similarity, list-based and machine-learning domains but focus majorly on web and email based phishing detection. This paper focusses on heuristic technique to detect phishing URLs in QR codes. The paper introduces a novel algorithm to generate a "phish-score" based on the lexical features of a URL. The application was tested extensively against its competitors using a curated list of mixed URLs stored in QRs and achieved an accuracy of 85.9% with 87.2% precision.

Keywords: QR code, phishing detection, heuristic technique, PhishScan

1 Introduction

1.1 QR codes

Quick Response (QR) codes are the most used barcodes today as surveyed by (L. Ceci, 2021), which shows 86.37% of respondents had used a QR today, in the past week, past month, and in the past year. QR is a type of two-dimensional barcode that can be scanned using a smartphone or QR code reader to launch a specific action or access information stored digitally in it, such as Uniform Resource Locators (URL), contact information, email addresses, and text messages. QR codes are often used in advertising, product packaging, and other types of marketing materials, as well as for mobile payments, contact sharing, advertisement and authentication, where users interact directly with the QR, making it easy for bad actors to reach the end user. Since QR codes are very easy to generate, a malicious user can easily generate one that looks genuine but stores the URL of a phishing website (Yong et al., 2019). The end users scanning such QRs will be redirected unrestrictedly to these malicious websites and tricked into giving away sensitive information such as login credentials, credit card numbers, or personal information.

1.2 Existing Techniques

The need to verify phishing URLs has indeed crossed many minds, and research has been conducted wherein different techniques have been used to detect a phishing URL using artificial intelligence, machine learning, visual similarity, and heuristic techniques. For instance, (Feroz & Mengel, 2015) suggest ranking URLs using URL clustering, URL classification, and URL categorization mechanisms in conjunction. (Aljofey et al., 2022) have proposed a hybrid feature set including URL character sequence features without expert's knowledge, various hyperlink information, plaintext and noisy HTML data-based features within the HTML source code. These features are then used to create feature vector required for training the proposed approach by XGBoost classifier. Since none of the existing approaches guarantee success against zero-day attacks, the paper intends to propose a new algorithm to calculate phish-score or a URL using its lexical features with the help of existing heuristic techniques. We attained 85.9% accuracy in detecting phishing URLs in a QR code through our lightweight Android app "PhishScan" which is free to use and ethically sound as compared to its competitors.

1.3 Proposed Solution

After the COVID-19 pandemic, the use of QR codes has seen an exponential increase as surveyed by (Raynor de Best, 2022), and QRs are being adapted for various applications like advertising, product packaging, and other types of marketing materials, as well as for mobile payments and authentication. Most of these uses are carried out using a smartphone or a tablet. We discovered that there are only two applications on the Google Play store that provide phishing detection against QRs (as on April 2023), one of which the *Kaspersky Security & VPN* app (Kaspersky Lab Switzerland, 2023) forcefully blocks the URL without allowing the user to decide whether or not they want to visit the site anyway, while *QR Scanner-Safe QR Code Reader* app (Trend Micro, 2022) does not provide the probability measure of the URL being "**phishy**." Also, they are collecting web browsing history without providing a way for users to request that their data be deleted. As a result, we decided to develop a lightweight Android app to detect QR codes containing phishing URLs, using a "**phish-score**" generated from existing heuristic techniques of ranking lexical features of a URL without visiting the website, to achieve safety and maximum accuracy along with storing no user data to avoid ethical issues. We introduced an algorithm to provide the user with a phish-score and allow the user to decide if they still want to go ahead and visit the URL.

As a novel contribution to the field of research, I plan to propose a new algorithm which will generate a phish-score for a URL scanned from a QR code. This algorithm is built using existing heuristic techniques, focussing on 9 lexical and 10 host-based features of a URL that include and are not limited to domain name analysis, URL length, content analysis, IP address analysis, SSL certificates, etc, as researched and documented in (Silva et al., 2020), (Zhu et al., 2020), (Zhu et al., 2019) and (Darling et al., 2015), through our lightweight Android application "**PhishScan**".

PhishScan requires user permissions to access Camera and Internet. The app reads URL from the QR being scanned and provides the QR to the ranking mechanism. The URL will then be tested against different lexical features and a phish-score will be generated. This phish-score will produce a judgement for the user in three categories namely “Genuine”, “Maybe Phishy” and “Phishy”. Using this result, the user will be given an option to decide if they want to go ahead and visit the link or abort the operation.

1.4 Research Question

How effective are heuristic techniques in detecting phishing URLs in QR codes.

This research paper is organized as follows: Section 2 provides additional background necessary to comprehend this work and describes related work in brief along with the research niche. Our methodology, architecture, flowchart, evaluation and a list of tools used for testing purposes are described in Section 3. Section 4 describes the implementation, design and algorithm explanation of PhishScan app followed by Section 5 which showcases the results obtained and finally the paper concludes with a conclusion after analysing the result.

2 Literature Review

In this section, we will first introduce the reader to the concepts of phishing, QR codes and heuristic techniques, followed by brief discussions on related works relevant to this research.

2.1 Background

2.1.1 Phishing

The word "phishing" is derived from the word "fishing" since the attacker in a phishing attack mimics a fisherman by employing a "bait" to entice the victim into disclosing sensitive information (Khonji et al., 2013). To steal sensitive information from a victim, an attacker may employ several methods, such as direct deception or indirect payload delivery. It is noted by Anti-Phishing Working Group that there has been a steady increase in the number of worldwide phishing attacks, with 1,270,88 recorded in the third quarter of 2022 alone. (APWG, 2022). There are several ways of carrying out a phishing attack as listed in exhaustive research by (Chiew et al., 2018), a few of the most prominent ones are mentioned below:

a. *Spear Phishing*: is a targeted attack on a specific person, group, or establishment. Phishers now prefer spear phishing to more traditional methods of phishing, such as bulk and random email phishing.
b. *Whaling:* is a targeted assault similar to spear phishing but directed at senior executives with significant control over an organization's resources.
c. *Clickjacking*: The term "clickjacking" refers to an attack in which the user interface (UI) of a website is tampered with in order to trick the user into performing an action without their knowledge.

2.1.2 QR code

QR codes, also known as matrix bar codes or two-dimensional codes, were developed specifically for use with mobile devices. QR stands for “Quick Response” indicating that the information contained within the code has to be read very quickly at high speed. The code is represented as black square blocks on a white background. An in-depth introduction of QR technology has been discussed in (Tiwari, 2017) from which we can conclude that QR codes are very easy to generate and manipulate since its practically impossible for the human eye to distinguish between them.

2.1.3 Heuristic technique

Heuristic technique uses features derived from the phishing website. The information is compiled from a wide range of sources using these techniques, including URLs, text content, DNS, digital certificates, and website traffic. A part of heuristic technique is the analysis of lexical features of a URL, such as the length of the URL, the presence of particular letters, the number of subdomains, etc., which can be used to categories a malicious URL (Saiful et al., 2016).

With this background, we will now discuss in the next section, some related works on phishing detection using different techniques and also some proposed techniques to secure QR.

2.2 Related Work

Largely it is noticed that a lot of research work has been carried out on phishing detection in web and email-based systems and a few works been carried out on securing QR codes. There is still a wide gap in detecting phishing URLs in QR codes as an individual subject. It is only recently that QR code phishing is gaining mainstream attention with reports like (HP Wolf Security, 2023) wherein QR code "scan scam" tactics are noticed by HP on a regular basis. Attempting to fill this gap, we have carried out our background research on related works as listed below.

2.2.1 QR Security

(Goel et al., 2017) have proposed a method to generate QR which store data in an encrypted format. This proposed technique uses AES algorithm for encrypting data. They have suggested to first enter a password to encrypt the data, then a 128-bit key using AES algorithm is generated from the password. Next, the data to be stored in the QR is encrypted with AES algorithm and saved in the QR. For reading the QR, the user needs to enter the password from which the AES key is generated, and data is decrypted using the key. Though this technique does solve a problem but, in the bargain, it defies the basic purpose of a QR which is providing quick and easy to use way of accessing information.

Also, in this approach the user needs to remember and insert passwords every time he/she needs to access data from a QR. It is highly inconvenient to remember multiple passwords for different QR and with the wide use of QRs today in the simplest of applications, it is practically impossible to be implemented on a large scale. A detailed survey on current measures to secure QR codes has been carried out in (Yong et al., 2019) which concludes that QR security is largely ignored, and more focus has always been on web and email phishing. Also, it was observed that most of these countermeasures focus on the digital signing or encryption of the data rather than flagging the URL itself.

2.2.2 Visual similarity-based techniques.

These techniques focus on refining and standardizing image-based visual similarity detection. In other words, Websites are identified as phishing pages when their content is compared to that of well-known, reliable websites and an obvious visual likeness is found. similarity-based approaches count on the adversary's strong motivation to create pages that seem like reputable ones, as opposed to relying on heuristics, which are subject to change. Using visual similarity technique, (Abdelnabi et al., 2020) have proposed a system named VisualPhishNet which learns a visual profile of websites by calculating the degree to which any two pages inside the same domain are similar, despite their varied contents. Whereas (Aljofey et al., 2022) suggested a method to crawl HTML and produce a unique feature vector for each page using vectorization. Based on this vector, a decision is made if the website is malicious or not. Visual similarity solutions require the website to be visited, which in our case is not the desired solution as frontend technologies are ever evolving and such solutions can get outdated very easily. Also, visual similarity is inappropriate since our problem statement is to scan a QR rather than visiting a URL and these techniques are time consuming as compared to heuristic techniques.

2.2.3 Machine learning based techniques.

In machine learning based phishing detection algorithms common attributes such as URL information, website structure, and JavaScript features are gathered to categorise phishing URLs and related websites. Then, based on those features, phishing data sets are generated. Next, machine learning classifiers are trained to recognize the phishing domain using these characteristics. (Zhu et al., 2020). (Atari & Al-Mousa, 2022) has undertaken exhaustive research to compare the efficiency of very well-known machine leaning algorithms like RandomForest, K-Nearest Neighbors, Linear Support Vector Machine (SVC), Logistic Regression, Extreme Gradient Boosting Classifier (XGBoost), SVC with Polynomial Kernel and Voting Classifier.

On similar lines, (Fan, 2022) have developed a machine learning method in which the URL is first checked in the PhishTank database, if not found then its lexical features are extracted and compared with the learnt database to check for phishing features previously encountered. Though the researchers claim a high level of accuracy of the system, the basis of this system is the PhishTank database which needs to be updated regularly and such systems fail to detect zero-day attacks since the list takes a few hours to be updated.

2.2.4 Combined techniques approach.

Many researchers have also explored combining the above techniques to eliminate the drawbacks and improve accuracy of detection. For instance, (Gu, 2021) combined list and machine learning approaches. They have proposed a fast and accurate detection algorithm by first checking URL similarity using Minhash signature to quickly verify it from the blacklist. Next, they applied the GIST vector to compute page similarity between the entire website and the whitelisted functioning websites, and the k-means clustering technique was used to assess if the websites were similar based on GIST vector.

2.3 Research Niche

This research is mainly focussed on proving how effective are heuristic features of a URL in detecting phishing and benign URLs. Specifically emphasizing on the lexical features of a URL. Following are the proposed deliverables of this report.

- A decision-making algorithm to calculate phish-score by analysing the lexical features of a URL.
- A lightweight Android application to scan QR codes and implement the algorithm.
- The application (PhishScan) is designed to be effective against zero-day URLs, shortened URLs and redirection URLs.
- The user is prompted with a phish-score, and an option to visit the URL or abort and rescan.

3 Methodology

The proposed application in this paper focusses on detecting phishing URLs in QR codes. The architecture of the system is shown in Figure 1, while the method that will be used is shown in Figure 2, based on (Carolin Jeeva & Rajsingh, 2016). The process begins with the application scanning the QR code to extract the URL embedded. After reading the URL from the QR, the URL is examined to be a valid website URL or any other information like contact, social media details, payment link, etc. If the URL is invalid, then an error is displayed, and control is taken back to the start. Else if its valid, then we apply heuristic technique to examine the URL and generate a phish-score through our algorithm. This phish-score is then prompted to the user with the URLs details who can then decide if they want to visit the link or abort. If yes, then the link is opened in the smartphone's default browser. If not, then the control is taken back to the start of the application.

3.1 Present Solutions

We reviewed the Google Play Store thoroughly to identify the best QR code scanner applications that also offer security against malicious URLs. We could identify only 2 such active applications on play store (as on Aug 2023), which claim to provide detection of malicious URLs in QR codes. Both these apps are listed in Table 1 along with their particulars. Though there are numerous other apps providing QR scanning functionality, none of them are providing detection of phishing URLs.

Table 1: Particulars of the two competitor apps on Play Store.

Feature	Kaspersky Security & VPN	QR Scanner-Safe QR Code Reader
Developer	Kaspersky Lab Switzerland	Trend Micro
Latest Version Available	11.97.4.9681	1.2.1
No. of installs	100M+	1M+
Rating	4.8	4.6
Stores scan history	Yes	Yes

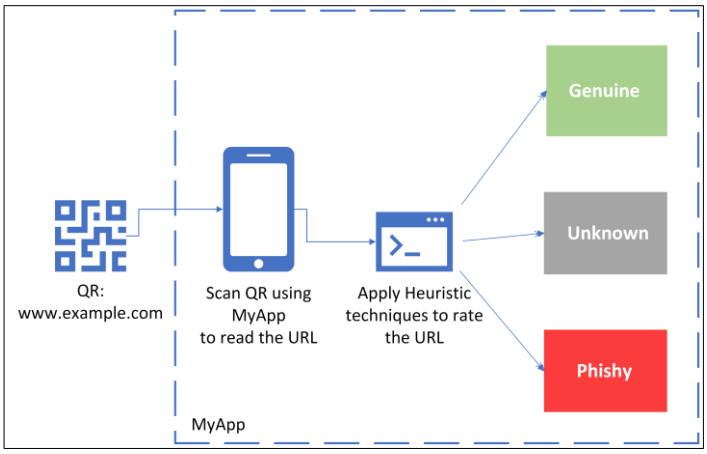

Figure 1: Architecture of the proposed system

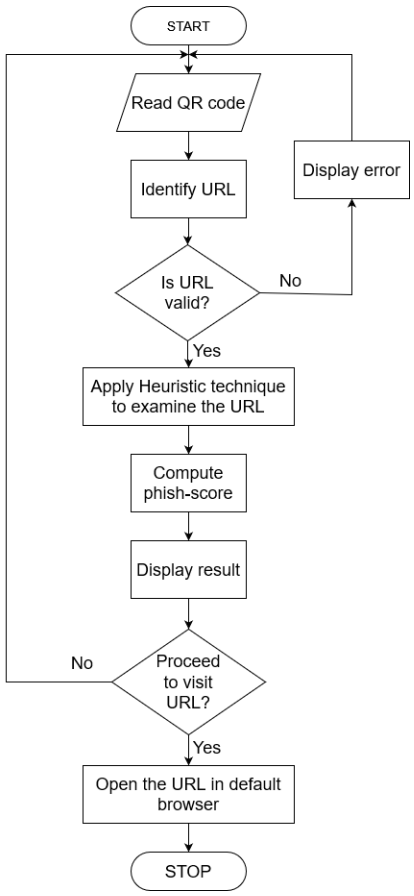

Figure 2: Flowchart of the proposed system

3.2 URL extraction

The URL is captured by reading the QR code using the app. PhishScan app uses code-scanner library[1] from GitHub to scan QR codes and extract the URL from it. Next, we check if the URL received is in actual form or shortened using third party URL shortening services like bitly, tinyurl, short.io, etc. threat actors usually use shorten URLs to disguise a user as the real information of the malicious URL is hidden in an encoded small URL. To tackle this, our application uses free APIs to unshorten URLs from unshorten.me[2]. These unshortened URLs are then supplied as an input to our novel algorithm to calculate the phish-score.

3.3 Phish-Score Algorithm

The proposed solution to our research question is the novel feature of this paper i.e., the phish ranking algorithm aka phish-score algorithm. The algorithm which we will use to rank the URL stored in a QR will house several layers of checks on the lexical features of the URL. Based on every check, an individual score will be allotted to the URL which will accumulate to the final

[1] https://github.com/yuriy-budiyev/code-scanner

[2] https://unshorten.me/api

score. This final score will be used to determine if the URL is benign or phishy. We call this score the “phish-score”.

The proposed algorithm ranks a URL in the range of 0 to 100 including decimals. Every heuristic feature holds its appropriate weight, and a final score is generated accumulating all the individual weights. This phish-score is then scaled in the range of 0 to 100 where a score closer to 100 is highly phishy. A detailed distribution of weight per feature is shown in Table 2. Depending on the phish-score, appropriate message is being given to the user to decide if they want to go ahead and visit the link or scan another QR.

Table 2: Weightage of each feature as used by the algorithm to generate Phish-Score.

No	Feature	Weight
Lexical Features		
1	IDN and homograph attacks	5
2	Suspicious characters	5
3	URL structure and parameters	5
4	URL length	5
5	Presence of ‘@’ symbol	5
6	URL Redirection	3
7	URL Shortening Service	2
8	Existence of protocol in domain part ‘http’ or ‘https’	2
9	Primary domain’s length	3
10	Number of dots	2
11	Average word length	2
12	Length of the longest word	3
Host-based Features		
13	Domain creation date	3
14	DNS Record	5
15	Domain similarity	15
16	Subdomain anomalies	10
17	Domain extensions	10
18	HTTPS and SSL certificates	10
19	Direct IP address usage in the URL	5

4 Implementation

We implemented the PhishScan which is purely based on lexical features of URLs on Android operating system. Traditional methods visit the site to detect visual features and rate the URL as phishing based on machine learning algorithm predictions. This method poses a risk of visiting the site which might trigger a download and infect the smartphone even without human interaction. To prevent this scenario, PhishScan rates the URL based on the URL’s lexical features purely without visiting the actual website. This section details the implementation of PhishScan to detect phishing URLs using lexical features.

4.1 Research Resources

To conduct this research, we have made use of several software and hardware tools as mentioned in Table 3.

Table 3: Tools to be used to complete the research.

Tool	Nature	Description
Android Studio	Software	official IDE for Android app development which has comprehensive set of tools for developers to create, test, and deploy Android apps.
Google Pixel 2	Hardware	Google smartphone with 5-inch 1080p OLED display, Snapdragon 835 processor, 4GB of RAM, and a 12.2-megapixel camera.
www.qrcode-monkey.com	Software	a free online QR code generator.
PhishTank	Software	a community-driven website that collects and shares information about phishing attacks. It is run by the non-profit OpenDNS. PhishTank collects data from users who submit suspected phishing websites and verifies them through a process of manual and automated checks, then adds these websites to the PhishTank database, which is freely available to the public.
AMSTO (testing)	Software	AMTSO stands for the Anti-Malware Testing Standards Organization. AMTSO provides a range of resources and tools on its website to help testers, developers, and users of anti-malware/anti-phishing software to better understand the complexities of anti-malware/anti-phishing testing. These resources include a library of test cases, guidance documents, and articles on best practices for testing anti-malware/anti-phishing products.

4.2 Design

Figure 3: Scan screen

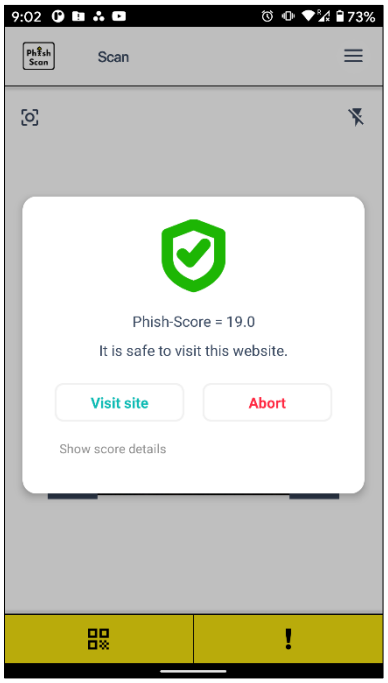

Figure 4: URL not phishy

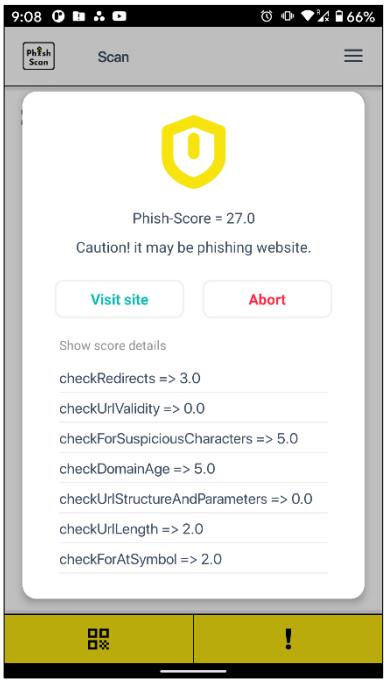

Figure 5: URL maybe phishy

Phish-Scan is developed for Android operating system and is compatible with Android SDK version 29 and above. The application has a simple interface, with the home screen housing a QR code scanner and a bottom bar with Scan and About navigation buttons (Figure 3). Once a QR is scanned, a popup is displayed with the Phish-Score, a warning text, a detail list of parameters which define the score and two buttons to visit the site and cancel to scan a new QR (Figure 4 and 5).

4.3 Phish-Scan Algorithm Breakdown

The proposed algorithm considers following heuristic features of a URL:

4.3.1 Lexical features

Following is a comprehensive list of lexical features being checked in the URL to determine its Phish-Score. All these methods are written in the ScanViewModel.kt class and the description of each can be found below.

a. IDN and homograph attacks:

Usage of Latin characters to fool a user is being checked in the method checkForIdnAndHomographAttacks(url: String), which accepts the scanned URL from the QR code and converts the URL into ASCII code. Following which, each character is compared with ASCII characters to check if it matches or not. If a character does not match the ASCII character set, then it means a Latin character is found and hence we increase the Phish-Score with the appropriate weight. The code for the same is shown in Figure 6.

```
//    todo: Checking Latin characters in the URL
new *
private fun checkForIdnAndHomographAttacks(url: String) {

    var isSuspicious = false

    // Convert the URL to its ASCII representation
    val asciiUrl = IDN.toASCII(url)

    // Check for non-Latin characters with similar Latin characters
    for (i in 0 ≤ until < url.length) {
        val char = url[i]
        val asciiChar = asciiUrl[i]

        if (char != asciiChar) {
            isSuspicious = true
        }
    }
```

Figure 6: Code snippet to check Latin characters present in the scanned URL.

b. Suspicious characters:

We have identified a list of suspicious characters which threat actors usually use in phishing URLs to deceive users. These characters are being checked in checkForSuspiciousCharacters(url: String) method and if the count of suspicious characters exceeds our threshold of 4 characters, then the Phish-Score is incremented according the

weight of this feature. Figure 7 shows the code snippet of counting suspicious characters in the URL.

```
286    private fun checkForSuspiciousCharacters(url: String) {
287        val suspiciousCharacters = listOf(
288            "@", "%", "!", "#", "$", "&", "*", "(", ")", "[", "]", "{", "}", "<", ">", "|", "\\",
289            "/", "?", ":", ";", "'", "\"", "`", "~", "+", "=", "^"
290        )
291
292        var count = 0
293
294        for (character in suspiciousCharacters) {
295            if (url.contains(character)) {
296                count++
297            }
298        }
299
300        if (count > 4) {
301            addPhishScore( points: 5.0)
302            errorList.add("checkForSuspiciousCharacters => 5.0")
303        }
304    }
```

Figure 7: Code snippet to check count suspicious characters present in the scanned URL

c. URL structure and parameters:

As it is well established in the previous sections of this paper that a URL consists of a protocol, host and a domain, we first check if the scanned URL has the host and protocol or not on line 363 in the checkUrlStructureAndParameters(url: String) method. If not then suspiciousScore is incremented. Following on, we check for the query half of the URL which is usually in the form of https://example.com/api/AuctionDetails?userId=10972&productId=12, where userId and productId are the query parameters which are always separated with a "=" sign and if more than one query parameters are present then they are separated with a "&" sign. We check these two in the URL at line 371 and if any misplacement is found, we increase the suspiciousScore accordingly.

```
356    private fun checkUrlStructureAndParameters(url: String) {
357        var suspicionScore = 0.0
358
359        try {
360            val url = URL(url)
361
362            // Check for anomalies in URL structure
363            if (url.protocol.isNullOrEmpty() || url.host.isNullOrEmpty()) {
364                suspicionScore++
365            }
366
367            // Check for anomalies in URL parameters
368            val queryParams = url.query?.split( ...delimiters: "&") ?: emptyList()
369            for (param in queryParams) {
370                val keyValue = param.split( ...delimiters: "=")
371                if (keyValue.size != 2 || keyValue[0].isNullOrEmpty() || keyValue[1].isNullOrEmpty()) {
372                    suspicionScore++
373                }
374            }
```

Figure 8: Code snippet to check query parameters of the scanned URL.

d. URL length:

An average URL length of a genuine URL can go up to 70 characters long as per industry standards. Following the norm, we check the length of the URL in checkUrlLength(url: String) method and add to the Phish-Score accordingly.

e. Presence of '@' symbol:

A URL usually does not require an "@" sign in it. It is most likely a phishing attempt wherein a threat actor will embed a second URL in the primary one to disguise the user. For instance consider the URL https://$%^&;****((@bit.ly/3vzLjtz#ZmluYW5jZUBuZ3BjYXAuY29t, the actual URL "bit.ly/3vzLjtz " is embedded in the primary URL and separated with an "@" symbol. We check the appearance of "@" symbol in checkForAtSymbol(url: String) method.

f. URL Redirection:

URL redirection is a very common technique used to deceive a user. A genuine URL is set to redirect to a malicious one and hence it is impossible for a user to identify a phishing URL. We have checked the connection of the domain by opening a connection on line 201 in checkRedirects(url: String) method and reading the response status. If the response code is anything between 300-400 which is a code for redirection, then our Phish-Score is incremented as this is suspicious and cannot be ignored.

g. URL Shortening Service:

Phishing URLs are usually shortened using URL shortening services like tinyurl and bit.ly because the actual domain is then hidden from the user. This becomes very difficult to judge if a URL is phishy or not in the first glance. We use URL expanding API from unshorten.me[3] website in expandURL(url: String) method, to expand the shortened URL and feed it to the algorithm for further scoring.

h. Existence of protocol ('http' or 'https') in domain part:

Every genuine URL will have the protocol as 'http' or 'https' at the beginning. If its absent then it is most likely phishy. We check this parameter in checkProtocolInDomain(url: String) and increase the Phish-Score accordingly.

i. Primary domain's length:

An average primary domain is 7 characters in length. While anything between 7 to 13 is questionable, above 14 is considered risky. This check is done in checkPrimaryDomainLength() method on line 446.

j. Number of dots:

As per industry standards and accepted norms, a genuine URL should ideally contain 2 dots or 3 in case of a subdomain added. Any number of dots above this probably phishy and

[3] https://unshorten.me/json

needs a closer look. For this reason, we have calculated the number of dots in the URL string and rated it accordingly in checkNumberOfDots(url: String) method.

k. Average word length:

Average word length in a URL is also worth noting since phishing URLs have found to have random long unreadable texts in the URL to confuse a user. An average acceptable length considered in our algorithm is 6 as can be seen in calculateAverageWordLength(url: String) method.

l. Length of the longest word.

4.3.2 Host-based features

a. Domain creation date:

The vast majority of phishing URLs are generated, registered, utilised, and subsequently removed within a span of a few months in order to evade detection. Our application uses ipty.de[4] free API service to determine the age of a domain in the method checkDomainAge(url: String). If the age is less than 1 year (365 days) then we mark it as suspicious and increase the Phish-Score.

b. DNS Record:

Phishing URLs have a history of forging domain names as they hold identity of the domain owner. We try to resolve the DNS record of the ULR using InetAddress class of Java in checkDNSRecord(url: String) method.

c. Domain similarity

An exhaustive list of common well known domains are pre-fed to the algorithm. These domains are most widely and most likely to be used to confuse a user. Some of them are paypal.com, payment, amazon, bank sites, etc. Phishing URLs use these domain names in conjunction with random characters like www.paypall.com which look similar to the original domain. Here in domainSimilarity(url: String) method we check for such things in the URL.

d. Subdomain anomalies:

Phishing URL also make extensive use of subdomains to forge URLs. For instance, www.paypal.com would be used as www.games.paypal.biz where no such subdomain exists for PayPal and games is unrelated to the domain PayPal which is a payment gateway. These subdomain anomalies are checked in the method checkSubdomainAnomalies(url: String).

e. Domain extensions:

An exhaustive list of tlds was curated from various sources like bleepingcomputer.com[5] and phishlabs.com[6] and fed to the algorithm to check for those widely used tlds in phishing

[4] https://ipty.de/domage/

[5] https://www.bleepingcomputer.com/news/security/these-are-the-top-level-domains-threat-actors-like-the-most/

[6] https://www.phishlabs.com/blog/top-10-tlds-abused/

URLs. The weightage given to this criteria has taken into account that genuine websites do use these tlds and they should not be flagged phishy.

```
186    private fun checkDomainSuspicion(url: String) {
187
188        val suspiciousTlds = listOf(
189            "xyz", "info", "biz", "top", "online", "club", "site", "ga", "cf",
190            "tk", "org", "ml", "pw", "top", "ga", "icu"
191        )
192
193        val tld = domain.substringAfterLast( delimiter: ".",  missingDelimiterValue: "")
194
195        if (tld in suspiciousTlds) {
196            addPhishScore( points: 10.0)
197            errorList.add("checkDomainSuspicion => 10.0")
198        }
199    }
```

Figure 9: Code snippet of checking domain extension of the scanned URL.

f. HTTPS and SSL certificates:

We check the status of X.509 certificates in checkUrlValidity(url: String) method. Many budding phishers try their had at phishing with minimum efforts and hence use phishing websites without registering a genuine certificate. It is always good practice to check for certificate validation when determining phishing URLs.

g. Direct IP address usage in the URL:

Checking if IP address is directly mentioned in the URL through checkForDirectIPAddressUsage(url: String) method.

4.4 Evaluation

Our novel Phish-Score algorithm determines the score accumulating weights from Table 2 and eventually making a decision if the URL is phishy or not based on the decided threshold formula. We have set the threshold for phishing website at a score of 40 and above (Figure 10) after intensive testing on several URLs form PhishTank database.

```
104
105            if (phishScore <= 12) {
106                result.value = NotPhishing
107            } else if (phishScore > 12 && phishScore <= 25) {
108                result.value = MaybePhishing
109            } else
110                result.value = Phishing
111
112            Log.e( tag: "11111",  msg: " PhishScore = $phishScore")
113
114        }
```

Figure 10: Code snippet for decision making threshold (line 105).

We tested the Phish-Scan algorithm by running a list of mixed phishing and genuine URLs to determine the efficiency and accuracy of our algorithm. An exhaustive list from PhishTank[7] was used to feed URLs to the algorithm and the results were compared with PhishScan's competitors.

5 Results

We tested our algorithm with exhaustive runs carried out on a selected mixed subset of URLs (1000 URLs) curated from PhishTank[8] which resulted in interesting results, a few results are shown in Table 4. All these URLs were stored in separate QR codes and scanned using the respective applications.

Table 4: Results of QR scans compared with competitors.

URL	PhishScan	Kasperky	Trend Micro
http://ex0dsrecovery.world/dialogue-box/	MP	**P**	**P**
www.monster.pay.pal.xyz/	**MP**	UV	UV
http://hdxvtqpliz.duckdns.org	MP	G	**P**
https://login-online.mbhbank.hu/welcome.htm#	**G**	G	G
https://scontent.fmaa10-1.fna.fbcdn.net/	**G**	G	G
https://gredsa.2waky.com/	MP	G	**P**
https://click.mail.lidl.de/?qs=bf63ce4204b6aa7e8d226355f7a327a177ec80bd1ddb28e2d62f023bb4e18fa2e919bcf3dc28f4ae4b9c024b8df7727b74abb49275b30ce5	UV	G	G
https://ipfs.io/ipfs/QmboDZd3aR5Sc9LiXYzVmNKe2n4pDpiihSsLqBwJKCpdBM?filename=bobosing.html	UV	G	**P**
https://pub-d66e22d63e6f4b69af66c870cfec509e.r2.dev/index.html	**P**	P	P
https://www.ncirl.ie/	**G**	G	G

P – Phishing
MP – Maybe Phishing
G – Genuine URL
UV – Unable to Verify

We notice that out of the 1000 URLs tested, PhishScan was able to flag correctly 234 as Phishing, 434 as Maybe Phishing and 191 as Genuine (Not Phishing). Out of the remaining 141 URLs scanned, PhishScan produced 98 false positives and 43 false negatives.

True Positives (TP) = 234 + 434 True Negatives (TN) = 191
False Positives (FP) = 98 False Negatives (FN) = 43

Accuracy = (TP + TN) / (TP + TN + FP + FN) = **0.859**

Precision = TP / (TP + FP) = **0.872**

[7] https://phishtank.org/

[8] https://phishtank.org/phish_search.php?valid=y&active=y&Search=Search

With accuracy of 85.9% and a precision of 87.2%, our algorithm stands strong amongst its competitors. Further, it was noted that for certain URLs with very less lexical feature anonymity, our algorithm was not very efficient. Though for zero-day URLs with distinct lexical features, our algorithm achieved highest level of accuracy in detecting phishing.

6 Conclusion and Future Work

An attempt was made in this paper to identify phishing URLs purely based on the most common heuristic features of a URL. We identified 19 such features which included lexical and host-based features. A novel Phish-Score algorithm was introduced which used individual weights of each of these features with respect to their importance in identifying phishing URLs. Though the algorithm was able to identify phishing URLs with an accuracy of 85.9%, it is concluded that heuristic techniques alone are not sufficient to identify phishing URLs.

Today's modern solutions heavily rely on machine learning algorithms, whitelisting and blacklisting techniques which produce more accurate results, but they also have their own limitations. For instance, in whitelisting technique the list needs to be highly updated with every new phishing URL being circulated and the application using such lists should also have it updated very regularly. This can be a daunting task with millions of new phishing URLs being generated every month (Palmer, 2017) and with AI generating websites on the fly (Vincent, 2023) any zero-day phishing URL or website can easily be missed.

Considering such limitations of existing techniques, we attempted to design a solution which could prevent zero-day attacks, which does not rely on lists being updated, which is fast and instant and a solution which does not pose a risk to the device by visiting the malicious site to check for visual features.

For future work, we intend to combine a few techniques along with heuristics to better the accuracy of our algorithm. Machine learning and whitelisting should be considered along with Phish-Scan algorithm to get an average score after combining the scores from all three methods. Also, to avoid potential harm to the device, we could use HTML DOM reading techniques to read the contents of the webpage and analysing them. Combining these techniques into one algorithm should provide a comprehensive output to detect phishing URLs and eventually can be implemented in our existing PhishScan app to improve the results drastically.

References

Abdelnabi, S., Krombholz, K., & Fritz, M. (2020). *VisualPhish-Net: Zero-Day Phishing Website Detection by Visual Similarity*. 18. https://doi.org/10.1145/3372297.3417233

Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., & Wang, Y. (2022). An effective detection approach for phishing websites using URL and HTML features. *Scientific Reports 2022 12:1*, *12*(1), 1–19. https://doi.org/10.1038/s41598-022-10841-5

APWG. (2022). *Phishing Activity Trends Report, 3rd Quarter 2022*. https://docs.apwg.org/reports/apwg_trends_report_q3_2022.pdf?_ga=2.1890163.351607279.1681390062-1713902678.1681390060&_gl=1*hk8tov*_ga*MTcxMzkwMjY3OC4xNjgxMzkwMDYw*_ga_55RF0RHXSR*MTY4MTM5MDA1OS4xLjEuMTY4MTM5MDM3OS4wLjAuMA..

Atari, M., & Al-Mousa, A. (2022). A Machine-Learning Based Approach for Detecting Phishing URLs. *2022 International Conference on Intelligent Data Science Technologies and Applications, IDSTA 2022*, 82–88. https://doi.org/10.1109/IDSTA55301.2022.9923050

Carolin Jeeva, S., & Rajsingh, E. B. (2016). *Intelligent phishing url detection using association rule mining*. https://doi.org/10.1186/s13673-016-0064-3

Chiew, K. L., Yong, K. S. C., & Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications*, *106*, 1–20. https://doi.org/10.1016/J.ESWA.2018.03.050

Darling, M., Heileman, G., Gressel, G., Ashok, A., & Poornachandran, P. (2015). A lexical approach for classifying malicious URLs. *Proceedings of the 2015 International Conference on High Performance Computing and Simulation, HPCS 2015*, 195–202. https://doi.org/10.1109/HPCSIM.2015.7237040

Fan, Z. (2022). *Detecting and Classifying Phishing Websites by Machine Learning*. 48–51. https://doi.org/10.1109/ICAML54311.2021.00018

Feroz, M. N., & Mengel, S. (2015). Phishing URL Detection Using URL Ranking. *Proceedings - 2015 IEEE International Congress on Big Data, BigData Congress 2015*, 635–638. https://doi.org/10.1109/BIGDATACONGRESS.2015.97

Goel, N., Sharma, A., & Goswami, S. (2017). A way to secure a QR code: SQR. *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, *2017-January*, 494–497. https://doi.org/10.1109/CCAA.2017.8229850

Gu, C. (2021). A Lightweight Phishing Website Detection Algorithm by Machine Learning. *Proceedings - 2021 International Conference on Signal Processing and Machine Learning, CONF-SPML 2021*, 245–249. https://doi.org/10.1109/CONF-SPML54095.2021.00054

HP Wolf Security. (2023, March 16). *HP Wolf Security Threat Insights Report Q4 2022 | HP Wolf Security*. https://threatresearch.ext.hp.com/hp-wolf-security-threat-insights-report-q4-2022/

Kaspersky Lab Switzerland. (2023, March 17). *Kaspersky Security & VPN - Apps on Google Play*. https://play.google.com/store/apps/details?id=com.kms.free

Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing detection: A literature survey. *IEEE Communications Surveys and Tutorials*, *15*(4), 2091–2121. https://doi.org/10.1109/SURV.2013.032213.00009

L. Ceci. (2021, July 7). *U.S. and UK QR code users last scan 2020 | Statista*. https://www.statista.com/statistics/199334/us-qr-code-scanners-last-time-scanned/?locale=en

Palmer, D. (2017, September 22). *1.4 million phishing websites are created every month: Here's who the scammers are pretending to be*. ZDNET. https://www.zdnet.com/article/1-4-million-phishing-websites-are-created-every-month-heres-who-the-scammers-are-pretending-to-be/

Raynor de Best. (2022, October 7). *QR code users in the U.S. 2019-2020, with forecasts to 2025*. https://www.statista.com/statistics/1337584/number-of-smartphone-qr-code-scanners-usa/

Saiful, M., Mamun, I., Rathore, M. A., Lashkari, A. H., Stakhanova, N., & Ghorbani, A. A. (2016). *Detecting Malicious URLs Using Lexical Analysis*. https://doi.org/10.1007/978-3-319-46298-1

Silva, C. M. R. da, Feitosa, E. L., & Garcia, V. C. (2020). Heuristic-based strategy for Phishing prediction: A survey of URL-based approach. *Computers & Security*, *88*, 101613. https://doi.org/10.1016/J.COSE.2019.101613

Tiwari, S. (2017). *An Introduction to QR Code Technology*. 39–44. https://doi.org/10.1109/ICIT.2016.021

Trend Micro. (2022). *QR Scanner-Safe QR Code Reader* (1.2.1). https://play.google.com/store/apps/details?id=com.trendmicro.qrscan

Vincent, J. (2023, May 2). *AI is being used to generate whole spam sites*. https://www.theverge.com/2023/5/2/23707788/ai-spam-content-farm-misinformation-reports-newsguard

Yong, K. S. C., Chiew, K. L., & Tan, C. L. (2019). A survey of the QR code phishing: The current attacks and countermeasures. *2019 7th International Conference on Smart Computing and Communications, ICSCC 2019*. https://doi.org/10.1109/ICSCC.2019.8843688

Zhu, E., Chen, Y., Ye, C., Li, X., & Liu, F. (2019). OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network. *IEEE Access*, *7*, 73271–73284. https://doi.org/10.1109/ACCESS.2019.2920655

Zhu, E., Ju, Y., Chen, Z., Liu, F., & Fang, X. (2020). DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features. *Applied Soft Computing*, *95*, 106505. https://doi.org/10.1016/J.ASOC.2020.106505