

Exemplifying Emerging Phishing: QR-based Browser-in-The-Browser (BiTB) Attack

Muhammad Wahid Akram*, Keshav Sood*, Muneeb Ul Hassan*, and Basant Subba⁺

^{*}*School of IT, Deakin University, Geelong, Australia*

⁺*Indian Institute of Technology Ropar (IIT Ropar), Punjab, India*

Email: {s224289198, keshav.sood}@deakin.edu.au, muneebmh1@gmail.com, basant.subba@iitrpr.ac.in

Abstract—Lately, cybercriminals constantly formulate productive approaches to exploit individuals. This article exemplifies an innovative attack, namely QR-based Browser-in-The-Browser (BiTB), using proficiencies of Large Language Model (LLM) i.e. Google Gemini. The presented attack is a fusion of two emerging attacks: BiTB and Quishing (QR code phishing). Our study underscores attack’s simplistic implementation utilizing malicious prompts provided to Gemini-LLM. Moreover, we presented a case study to highlight a lucrative attack method, we also performed an experiment to comprehend the attack execution on victims’ device. The findings of this work obligate the researchers’ contributions in confronting this type of phishing attempts through LLMs.

Index Terms—Phishing, Quick response (QR) code, browser-in-the-browser, quishing attacks, large language models (LLMs)

I. INTRODUCTION AND BACKGROUND

Over almost last three decades, since the inception of digitization, humans are continuously being the target of phishing scams. In cybersecurity, humans are especially considered the most vulnerable link in the success of phishing attacks [1]. The authors in [2] comprehensively classified various classes of mobile phishing. This includes smishing (SMS phishing), vishing (voice phishing), QRishing or Quishing (Quick Response (QR) code phishing), etc. Similarly, in [3] the authors proposed an innovative form of phishing attack named Browser-in-The-Browser (BiTB) attack. In fact, attackers can also launch these attacks in combination, such as sending a phishing email containing a QR code attachment that leads to a malicious website. Furthermore, this article showcased how Large Language Models (LLMs) i.e. Google Gemini can help in generating malicious content for the successful execution of QR-based BiTB phishing attacks.

Currently, many individuals and businesses use QR codes for multiple benefits, for example, embedding website URLs in it, sending via emails, and more. Similarly, LLMs have emerged as a significant tool across several applications including AI-driven chatbots, research, innovation, and content generation to data analysis. However, their availability and productiveness make them vulnerable to exploit for creation of malicious content for phishing attacks [4]. In the existing literature various works reported phishing attacks via QR codes (please note that phishing via QR codes is known as Quishing). In this regards, Bekavac et al. [5] underlines QR code tampering approach in which, attackers either physically replace the whole QR code with a fake one or interfere

with the pixels (white and black) of QR code. Moreover, [6] illuminates the effectiveness of barcode-in-barcode attack.

Regarding phishing with LLMs, [4] and [7] comprehensively studied LLMs and Vision Language Models (VLMs) models respectively. They highlighted the competencies of LLMs and VLMs in generating malicious content for specific phishing attacks including email phishing, Quishing, and BiTB. Generally, LLMs did not respond to given malicious prompts. However, if the same malicious prompt(s) submitted to the LLM in a sophisticated way as highlighted by [4] then attackers can be successful in producing phishing content. Similarly, in context of BiTB, [8] and [9] attempts to present the practical implementation of this attack. Whereas the proposed methodologies from both studies are far more complex to implement in realistic settings.

Motivated from QR-phishing and involvement of LLMs in social engineering schemes, in our work, we are highlighting an evolving attack, i.e., BiTB (in QR context) named as QR-based BiTB phishing attack. The authors in [3] have described BiTB attack (in non QR-code scenarios), where a popup modal shows a real browser which contains a phony URL and phishing web page. This paper has taken BiTB phishing attack one step ahead from an attacker’s perspective. We have shown the BiTB attack with malicious dynamic QR code and Gemini-LLM which renders it into a compelling attack vector. Therefore, the combination of dynamic QR code, and Gemini-LLM could empower BiTB attack to certainly bypass the standard phishing detection mechanisms.

Moreover, the aim of our study is not only to illuminate technique of this attack but also to emphasize its consequences on the attackers’ capabilities. Through evaluating its anticipated impact, we are looking to contribute in continuing discussion on productive counter frameworks. Below are the contributions of our study:

- We discussed a novel phishing attack namely QR-based BiTB which is a novel attack in QR-Codes platform and has never been studied before in the existing literature (in QR context). A step-by-step detailed discussion is given to better comprehend how to launch this attack. This discussion is extremely helpful from the reproducibility perspective (of our work).
- We utilized the Gemini-LLM in integrating QR-based BiTB phishing attack which generating the malicious content through providing malicious prompts. The malicious content further exploits while the execution of

QR-based BiTB attack in real-time.

- We presented a case study to demonstrate the execution phase of this attack which helps readers to understand a lucrative approach an attacker can use to successfully launch this attack. This is presented with an overall aim to further investigate novel threat models/approaches in future and to contribute in continuing discussion on productive counter frameworks applicable to mitigate this attack in wider attack scenarios.
- A Proof-of-Concept (PoC) is given which shows a successful launch of this attack using a practical and simple approach. This is to show how publicly available tools and resources, attacker could utilize to commence this attack which is practical and easy to launch.

II. OVERVIEW OF THE QR-BASED BROWSER-IN-THE-BROWSER (BiTB) ATTACK

In this section, firstly we described the capabilities of a dynamic QR code and then presented a case study to understand an attackers' approach to attract a potential victim. Rest of the section includes a high level attack framework, implementation setup, and execution part of proposed attack as depicted in Fig. 2.

A. Capabilities of Dynamic QR codes

QR codes can be of two types: static and dynamic. Static QR codes can be generated without any cost. The main difference is, when a static QR code created with some content (e.g. URL or text) it cannot be modified later. Whereas, dynamic QR codes has a feature to update the embedded content after its generation. Individuals and organizations can acquire further details whenever an individual scan a dynamic QR code such as the location, operating system (OS) of scanning device, browser type (where link opens), and current IP address of device, etc. To exemplify, we performed an experiment with an online QR code generator (which supports dynamic QR codes). We registered a trial account on QRFY¹ and generated a dynamic QR code. The results of this experiment is presented in Fig. 1 (i). It clearly indicates that user's personal information is being sent on this site through QR code.

B. Our Case Study

In our experiment, we consider an attacker, who offers a potential victim the lifetime 'YouTube Ads Free Access'. To redeem offer, a potential victim requires to scan the malicious dynamic QR code generated by the attacker. Upon scanning, victim redirects to the phishing site, where attacker mimicked the official web page of 'YouTube premium' access i.e. <https://www.youtube.com/premium> with additional BiTB attack settings which are real-time begin generating by providing specific malicious prompts to Gemini-LLM using its API. The moment malicious site opens on victims' browser, the popup modal 1 generates using Gemini-LLM and shows up as shown in Fig. 6 (a). This modal requests victim to input his/her 'First Name'. On submitting 'First Name', the popup

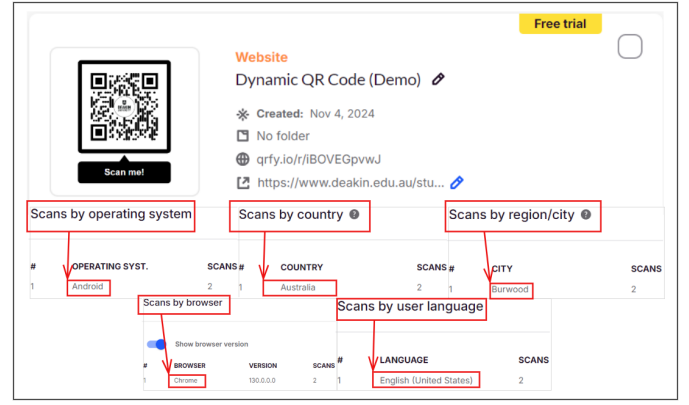


Fig. 1: QRFY results upon scanning dynamic QR code.

modal 2 generates using Gemini-LLM and appears with the Google signin form Fig. 6 (b). When victim provides his/her signin credentials, then malicious script redirects victim to the official app/site (app in case of mobile device and site in case of desktop) of 'YouTube'. Eventually, victims' credentials gets stored into the attackers' database. The detailed description of Fig. 6 is discussed in the following sections.

C. The High-Level Framework of the Proposed Attack

The step wise description of the high level framework in Fig. 2 is discussed below:

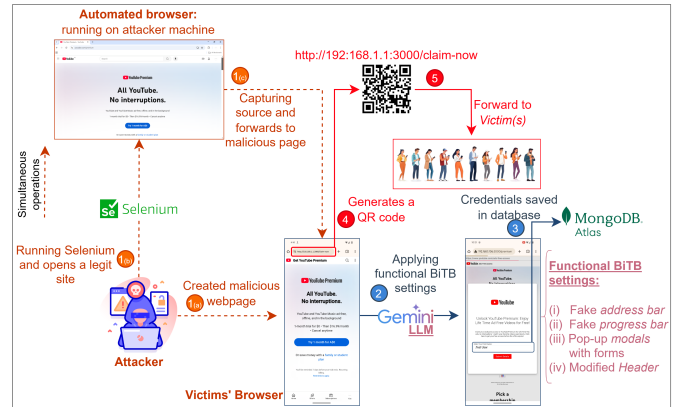


Fig. 2: The point-by-point flow of QR-based BiTB attack.

Step 1: The operations 1(a), 1(b), and 1(c) executes simultaneously. In 1(a), an attacker creates a malicious web page that shows up in victims' browser (Chrome in our experiment) running on local server link (i.e. <http://192.168.1.1:3000/claim-now>). Afterwards, 1(b), opens the automated browser with a legit site on attackers' machine. During 1(c), the source of legit site is captured and displayed on the malicious web page (see Algorithm 1) using Selenium.

Step 2: In operation 2, attacker modifies the extracted source by applying the BiTB settings on the malicious web page using API of Gemini-LLM. This includes, appending *fake address bar* on top repre-

¹<https://qrfy.com/>

senting legitimate URL, a *fake progress bar*, a *fake popup modals*, modified *header*, which are generated through Gemini-LLM API. The intention of modals is to prevent victims' interaction with the web page (like scrolling) and also to collect his/her sensitive credentials.

Step 3: If victim falls into the trap of BiTB settings of attacker and submits his/her credentials. Then in operation 3, these credentials saved to MongoDB database.

Step 4: In operation 4, the attacker embeds the server link into the QR code. So that, victim cannot recognize the malicious link before accessing it.

Step 5: Now, attacker is ready to disperse QR-based BiTB attack and can utilize any resource to forward malicious QR code to the victims such as by sending an email with fake offer like our case study or posting on social platforms, etc.

III. THE ATTACK IMPLEMENTATION SETUP: ATTACKERS' SIDE

This section discussed the integral components involved in our experiment to launch the QR-based Browser-in-The-Browser (BiTB) attack which is divided into three modules. Two of those modules are illustrated in this section. In addition, Algorithm 1 is also presented to better comprehend the step-by-step process of QR-based BiTB attack from attackers' end. Whereas, the third module is explained in the next section.

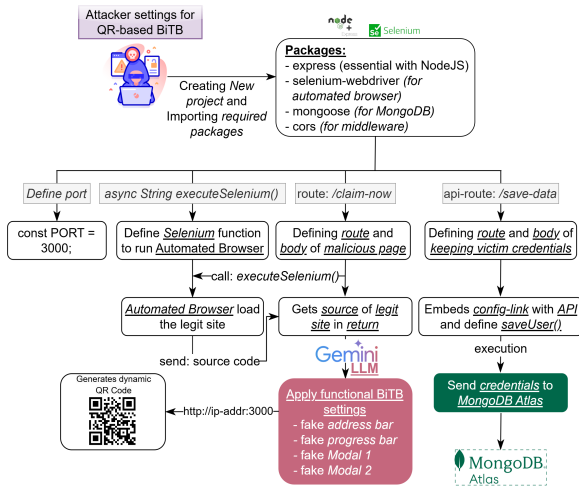


Fig. 3: Detailed architecture of attack implementation setup.

A. Module 1:- Writing and launching the BiTB Attack using Selenium:

In this module, the entire setup of Selenium with NodeJS including the Gemini-LLM's API is described and pictured in Fig. 3.

1) *New Project in NodeJS*: Once NodeJS is installed, the attacker creates a *New Project* with a server file *bitb_testing.js* in a specified directory. The entire malicious code is written

in NodeJS including the script of Selenium, database and Gemini-LLM's APIs to design settings of BiTB.

2) *Defining PORT*: To run project on local machine, a specific port is defined for example, *port = 3000* as shown in Fig. 3 and Algorithm 1.

3) *Selenium WebDriver*: Next, from Algorithm 1, a function *executeSelenium()* for Selenium script is defined which generates an automated browser (i.e. copy of Chrome) and opens a legitimate site as shown in simultaneous operations 1(a), 1(b), 1(c) in Fig. 2.

4) *Route for Malicious Web Page*: Now, attacker creates a route *'/claim-now'* for the malicious web page which appears on victims' browser and runs the malicious script. Firstly, the *executeSelenium()* executes from its body and returns the source of legitimate site. This process is outlined in operation 2 of Fig. 2 and Algorithm 1 where Gemini-LLM is also employed to generate malicious code for BiTB settings. Later, the attacker embeds this route with malicious link in the QR code.

5) *Route for Database API*: The attacker also establishes an API for database to save credentials of victim. For this, attacker defines a post request as showing in Algorithm 1 and connect the API with online database (*MongoDB Atlas*) using a *config-link*.

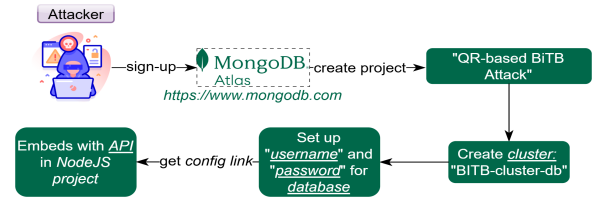


Fig. 4: MongoDB Atlas Setup.

B. Module 2:- Storing victim's sensitive credentials using MongoDB Atlas online database:

Fig. 4 illustrates the process of the attacker creating online database with MongoDB Atlas and connects this database with NodeJS project with a configuration link and API. The fundamental parts of this process are described below:

1) *MongoDB Atlas*: First, attacker signed up on official site of MongoDB Atlas (<https://www.mongodb.com/>) and creates a new project with name "*QR-based BiTB Attack*".

2) *Cluster and Database*: To generate database inside the project, attacker creates a cluster with name "*BiTB-cluster-db*" and set up the *username* and *password* credentials to access database.

3) *Config Link*: Following this, the created cluster provides a configuration link and attacker modifies this link with credentials to use in NodeJS project for database connectivity.

IV. QR-BASED BROWSER-IN-THE-BROWSER (BiTB) ATTACK EXECUTION: VICTIMS' SIDE

This section presents the execution of QR-based BiTB attack from victims' end. Moreover, the working of first (selenium module) and second (mongodb atlas module) module

Algorithm 1 Attackers' settings for QR-based BiTB Attack

```

1: import libraries and set-up PORT = 3000
2: function executeSelenium()
3:   pageSource ← callAutomatedBrowser('https://youtube.com')
4:   return pageSource
5: app.get('/claim-now')
6:   modifiedContent ← executeSelenium()
7:   modifiedContent ← pageContent + GeminiLLM_API('malicious prompts')
8:   send(modifiedContent)
9: app.post('/api/save-user')
10:   save_result ← saveUser(first_name, email, password)
11:   return 'User saved successfully'
12: {Gemini-LLM generate and operates below function}
13: function saveUser(first_name, email, password)
14:   user ← new User({first_name, email, password})
15:   return 'Success'
16: {Gemini-LLM generate and operates below functions}
17: function updateFakeAddressBar()
18:   Input new legitimate address into the address bar
19: function runFakeProgressBar()
20:   Progress bar executes
21: function modifyHeader()
22:   Changes applied to header

```

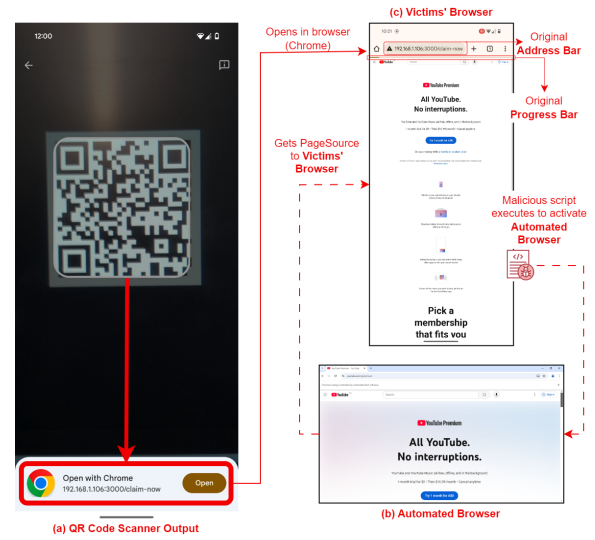


Fig. 5: (a) QR Code Scanner output, (b) Automated Browser, and (c) Victims' Browser.

can be seen in this (third) module. Additionally, Algorithm 2 is scripted to better comprehend the method.

For execution of BiTB attack, we deploy a Windows laptop where module 1 and module 2 are already configured. The NodeJS project is executed on local server url: 'http://localhost:3000'. By default, this URL is inaccessible on mobile. To enable this, attacker uses current IP address of machine which is connected with available Wi-Fi network. Afterwards, the mobile accessible URL embeds in malicious QR code was 'http://ip-addr:3000'. In Fig. 5 (a) and Algorithm 2, victim scans the malicious QR code using built-in QR scanner of mobile device. As a result, he/she gets embedded link to open in default browser. The moment link open in browser, the malicious script executes and initiates the *Automated Browser* on attackers' machine with a legitimate link of <https://www.youtube.com/premium> as scripted in Algorithm 2. The malicious script waits until page loads in *Automated Browser* and instantly it returns the source code to the victims' browser as shown in Fig. 5 (b) and (c), where, the original address and progress bars are also visible.

While loading of the malicious web page, the requests of malicious prompts also simultaneously sent to the Gemini-LLM by triggering its API to implement and present the BiTB attack settings on the malicious web page as highlighted in Fig. 6 (a) and Fig. 6 (b). This includes, alluring heading and paragraph texts, fake address bar displaying legit URL, fake progress bar, modified header, a fake popup modal 1, and modal 2 to acquire victim details. These settings from attacker can easily trick an individual. In Fig. 6 (a) and Algorithm 2, popup modal 1 requires victim to input his/her 'First Name' to proceed further with claiming the offer. On submission, the popup modal 1 hides and popup modal 2 shows up on the malicious page as represented in Fig. 6 (b). Also, header and progress bar of the page is further modified with 'First Name' of the victim using malicious script. Whereas, the URL in fake address bar is also altered to new legit URL. In addition, the

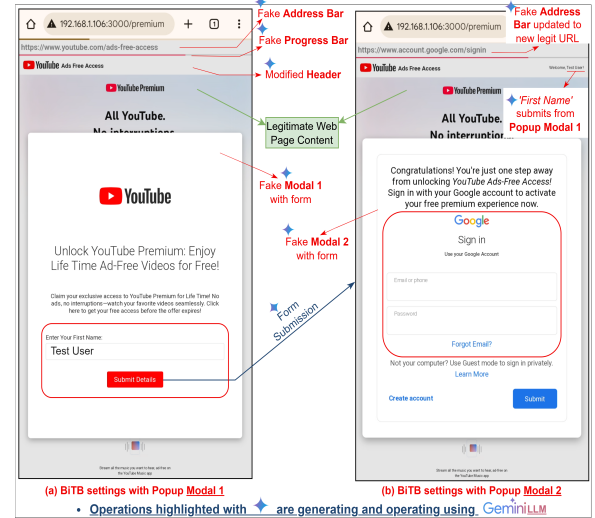


Fig. 6: (a) BiTB settings with Popup Modal 1 and (b) BiTB settings with Popup Modal 2.

attacker keeps entered 'First Name' in variable *first_name* as highlighted in Algorithm 2.

Modal 2 is the final stage of the BiTB attack, where attacker laid the trap of stealing victims' sensitive credentials. To manifest the legitimacy, attacker mimics the Google sign-in form in modal 2 as depicted in Fig. 7 (a) and Algorithm 2, whenever victim inputs his/her credentials in fake Google form and submits it. Then after submission, the fake progress bar refreshes again. Moreover, from Fig. 7 (b) and Algorithm 2, victim redirected to the official app of 'YouTube' if he/she is using a smartphone. For desktop users, they are redirected to official site of 'YouTube'. Simultaneously, victims' credentials gets stored in MongoDB database. In addition, dynamic QR codes captures individuals personal

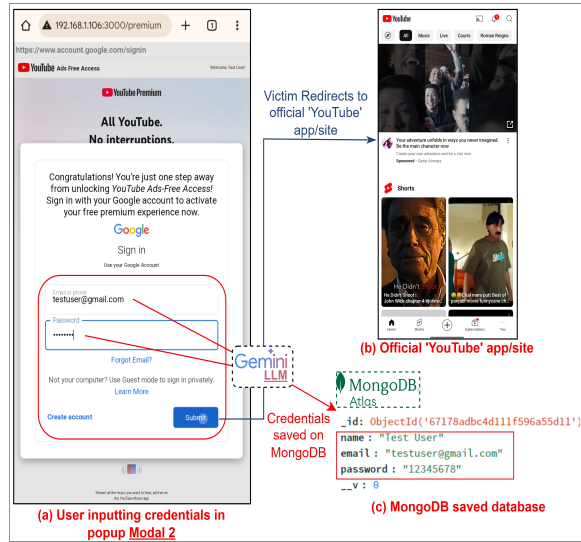


Fig. 7: (a) User inputting credentials in popup Modal 2, (b) User redirects to official 'YouTube' app/site, and (c) Screenshot from MongoDB database.

details upon scanning as shown in Fig. 2. This signifies that the attacker eventually holds the credentials of a victim using QR-based BiTB attack. Here, it is worth noticing that all of above steps are executed and generated in real-time using the malicious prompts which are requested to Gemini-LLM. This indicates the explicit illustration of how LLMs can be maliciously utilized in the execution of phishing attacks like the one proposed in this article i.e. QR-based BiTB.

Algorithm 2 Execution of QR-based BiTB attack from victims' end

Require: Algorithm 1 already defined, Malicious QR code embeds with a URL

```

1: Call app.get ('/claim-now') from Algorithm 1
2: Call function executeSelenium() from Algorithm 1
3: Modal 1 operates on victims' browser and require 'First Name' using Gemini-LLM
4: if Details submitted then
5:   Hide Modal 1 and keep input value in variable 'first_name'
6:   Call modifyHeader() from Algorithm 1
7:   Call updateFakeAddressBar() from Algorithm 1
8:   Call runFakeProgressBar() from Algorithm 1
9:   Reveal Modal 2
10: end if
11: Modal 2 operates on victims' browser and require 'Email' and 'Password' using Gemini-LLM
12: Victim inputs credentials and submits it
13: if Details submitted then
14:   Call runFakeProgressBar() from Algorithm 1
15:   Hide Modal 2 and keep input values in variable 'email, password'
16:   Call app.post ('/api/save-ser') from Algorithm 1
17:   Redirects victim to official app/site
18: end if

```

A. Key Insights

To sum up, the execution of QR-based BiTB attack from scanning malicious QR code to showcasing multiple BiTB settings, the attacker drives victim(s) during the whole process of phishing attempt. Below are the core findings of our work:

- In our experiments, the manipulation of fake configurations are much simple and effective in terms of vulnerabilities. We depicted how attacker utilizes easily accessible tools to execute the proposed phishing activity.
- QR-based BiTB using Gemini-LLM brought up as a novel social engineering attack technique that emphasizes essential demand to further strengthen defenses that can effectively withstand the evolving attack challenges.
- We demonstrated our case study using Chrome browser on Android mobile, however, because of attacks' generalizability initiated from scanning a QR code to a browser where subsequent steps executes, this attack is practically possible with any browser and smartphone device.

V. CONCLUSION AND FUTURE DIRECTIONS

Our work shows a peculiar form of a novel phishing attack i.e., QR-based BiTB attack using Gemini-LLM. The impact of this attack magnified due to the concatenation of two innovative attacks including Quishing and BiTB, and utilization of LLM. We demonstrated how simple compositions of open-source available resources can initiate this attack. Attackers continuously developing contemporary tactics to mislead users through presenting seemingly authentic information. The proposed QR-based BiTB is a prime example of it.

Lastly, this study has few shortcomings. Firstly, we utilized Selenium webdriver tool on local server, however, this tool is not supported by online servers due to its appearance in GUI form like an actual browser (i.e. Chrome). Secondly, operating this attack on local server requires attackers' machine and victims devices to connected on same network. This is because an attacker needs to use the IP address while connecting with network. We aim to address these gaps in future.

REFERENCES

- [1] L. Huang, S. Jia, E. Balcetis, and Q. Zhu, "Advert: an adaptive and data-driven attention enhancement mechanism for phishing prevention," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2585–2597, 2022.
- [2] R. Goenka, M. Chawla, and N. Tiwari, "A comprehensive survey of phishing: Mediums, intended targets, attack and defence techniques and a novel taxonomy," *International Journal of Information Security*, vol. 23, no. 2, pp. 819–848, 2024.
- [3] S. Asiri, Y. Xiao, S. Alzahrani, and T. Li, "Phishingtrds: A real-time detection system for phishing attacks using a deep learning model," *Computers & Security*, vol. 141, p. 103843, 2024.
- [4] S. S. Roy, P. Thota, K. V. Naragam, and S. Nilizadeh, "From chatbots to phishbots?: Phishing scam generation in commercial large language models," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 36–54.
- [5] L. J. L. Bekavac, S. Mayer, and J. Strecker, "Qr-code integrity by design," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–9.
- [6] M. S. Al-Zahrani, H. A. Wahsheh, and F. W. Alsaade, "Secure real-time artificial intelligence system against malicious qr code links," *Security and Communication Networks*, vol. 2021, no. 1, p. 5540670, 2021.
- [7] M. Taeb, J. Wang, M. H. Weatherspoon, S. Bernadin, and H. Chi, "Seeing the unseen: A forecast of cybersecurity threats posed by vision language models," in *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 2024, pp. 5664–5673.
- [8] F. Tommasi, C. Catalano, and I. Taurino, "Browser-in-the-middle (bitm) attack," *International Journal of Information Security*, vol. 21, no. 2, pp. 179–189, 2022.
- [9] J. Tzschoppe and H. Löhr, "Browser-in-the-middle-evaluation of a modern approach to phishing," in *Proceedings of the 16th European Workshop on System Security*, 2023, pp. 15–20.