Demystifying the (In)Security of QR Code-based Login in Real-world Deployments

Xin Zhang[1], Xiaohan Zhang[1], Bo Zhao[1], Yuhong Nan[2], Zhichen Liu[1],
Jianzhou Chen[1], Huijun Zhou[1], and Min Yang[1]

[1]*Fudan University* [2]*Sun Yat-sen University*

Abstract

QR code-based Login (QRLogin) has emerged as a prevalent method for web account authentication, offering a more user-friendly alternative to traditional username and password entry. However, despite its growing popularity, the security of QRLogin has been overlooked. In particular, the lack of standardized QRLogin design and implementation guidelines, coupled with its wide deployment variability, raises significant concerns on the real-world deployments of QRLogin.

This paper presents the first systematic study on the security of QRLogin in real-world deployments. We begin our research with real-world studies to understand the deployment status of QRLogin and user perceptions of this novel authentication paradigm, which assists us in establishing a realistic threat model. We then proceed with a systematic security analysis by generalizing the typical workflow of QRLogin, examining how key variables adhere to common security principles, and ultimately exposing 6 potential flaws. We conduct security analysis on real-world QRLogin deployments with a semi-automatic detection pipeline, and reveal surprising results that 47 top websites (43% of tested) are vulnerable to at least one of the above flaws. These design and implementation flaws can lead to 5 types of attacks, including *Authorization Hijacking*, *Double Login*, *Brute-force Login*, *Universal Account Takeover*, and *Privacy Abuse*. We have responsibly reported all the identified issues and received 42 vulnerability IDs from official vulnerability repositories. We further provide an auditing tool and suggestions for developers and users, contributing a concerted step towards more secure implementations of QRLogin.

1 Introduction

With the proliferation of mobile devices and the widespread adoption of QR codes, a novel authentication mechanism, QR code-based Login, abbreviated as *QRLogin*, has gained increasing utilization. As shown in Figure 1, to log into a website, a user can opt to use a mobile app, typically operated by the same principal as the website or from the cooperating third parties[1], to scan a QR code displayed on the website. Subsequently, the user confirms the login via the mobile app and finally logs into the website.

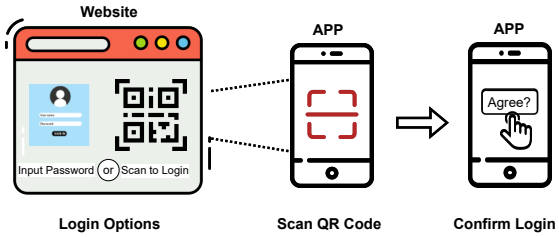

Figure 1: Demonstration of the QRLogin mechanism.

From the perspective of user convenience, the QRLogin mechanism eliminates the need for users to remember and type passwords, allowing them instead to use their authenticated QRLogin app, thus simplifying the authentication process. From a security standpoint, QRLogin requires users to have physical access to a personal mobile device and the QRLogin app, which may be secured by various methods such as fingerprint scanning and facial recognition. Therefore, it adds an additional layer of security, i.e., out-of-band security from the perspective of the website.

At first glance, QRLogin is considered more convenient and secure than traditional authentication methods such as password-based login on the website. However, as a new authentication paradigm, to the best of our knowledge, *there are no established standards or specifications on how to securely design and implement the QRLogin process*. Existing implementations are largely driven by individual interpretations and developments, leading to a diverse and complex array of QRLogin deployments with varying levels of security. Given that the QRLogin is an emerging and promising authentication scheme, the landscape and security of its real-world deploy-

[1]To ease illustration, the website deploying QRLogin will be called *QRLogin website*, while the scanning app is called *QRLogin app* in the rest of this paper.

ments remain open research questions, requiring a systematic, in-depth understanding and evaluation.

Our Work. In this paper, we present the first systematic study of QRLogin security in real-world deployments. We begin by collecting and analyzing 350 popular QRLogin websites to assess their prevalence and significance, as well as conducting a user study to investigate users' perceptions on QRLogin security.

To further explore the security issues in these websites, we summarize the common workflow for the QRLogin mechanism across its lifecycle, and pinpoint the key security-related variables involved in this workflow, i.e., *QrId*, *SessionID*, and *Tokens* (see §3.2 and §3.3). Following this, we examine the common security principles, including confidentiality, integrity, and consistency of and between these key variables. This principled analysis enables us to identify six potential implementation flaws, including *Unbound SessionId*, *Reusable QrId*, *Predictable QrId*, *Controllable QrId*, *Vulnerable Identity Verification*, and *Unintentional Privacy Leakage*.

Measurement and Findings. To further evaluate the presence and the impact of the aforementioned flaws, we employ a semi-automated pipeline to identify the insecurity of QRLogin in real-world deployments. Our findings reveal that 47 out of 109 (43%) tested websites exhibit at least one type of security flaws, which can lead to five types of attacks, including *Authorization Hijacking*, *Double Login*, *Brute-force Login*, *Universal Account Takeover*, and *Privacy Abuse* (see §4.3 for more details).

For example, a popular Russian social media platform employs QRLogin but its implementation exhibits two security flaws: *Unbound SessionId* and *Reusable QrId*. Consequently, if an attacker gains knowledge of the victim's *QrId* (the identifier of the QR code), they can log into the victim's account once the victim completes the QRLogin, leaving the victim unaware of the compromise, leading to the *Double Login Attack*. As another severe example, our study identifies two websites, including a leading network disk provider with more than 100 million users and a Chinese provincial government website[2], both exhibit the flaw *Vulnerable Identity Verification*. This vulnerability allows attackers to universally log into any account, provided they know just the phone number or account ID of the intended victim, leading to the *Universal Account Takeover Attack*. Given that both websites handle highly sensitive personal data (e.g., personal and business photos and files), the potential consequences of such breaches are particularly severe (more details in §4.3).

We responsibly report these vulnerabilities to the developers and regulators. At the time of paper submission, we have received a total of 42 acknowledgements from related parties, including the assignment of 17 CNVD and 25 NVDB IDs[3].

[2]Anonymized at the vendors' request.

[3]CNVD (China national vulnerability database, www.cnvd.org.cn) and NVDB (National vulnerability database, www.nvdb.org.cn) are two official vulnerability repositories in China.

To further enhance the security of QRLogin, we consider the stakeholders in QRLogin. In particular, for developers, we provide an automated auditing tool to evaluate and enhance their QRLogin implementations with specific mitigation advice. For users, we provide actionable suggestions for safer user practices.

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

- We perform the first systematic, in-depth study of security issues in the QRLogin scheme. Our research demystifies the workflow of QRLogin and then uncovers six types of unique flaws across the lifecycle of QRLogin.
- We perform a thorough analysis on 109 popular QRLogin websites in the wild. Our research identifies a total number of 47 websites that are vulnerable to various attacks in QRLogin.
- We enhance QRLogin security by providing developers with an automated auditing tool and offering mitigation suggestions for developers and users.

2 Understanding Real-world QRLogin

In this research, we start by understanding real-world QRLogin through two lenses: website deployments and user perceptions. We collect and confirm 350 websites using QRLogin from the Tranco top 100K list [24] (as of Nov. 24, 2023) and the results show the prevalence of QRLogin in top websites and the use in sensitive scenarios. Then, we perform a user study to understand user perceptions on QRLogin from the perspective of security.

2.1 QRLogin Deployments in the Wild

In this subsection, we first describe our approach to collecting QRLogin deployments in the real world and subsequently present the results.

Collecting Method. We employ a straightforward yet effective method to collect real-world QRLogin deployments. The core idea here is to find the presence of "QR Code-based Login" semantics on website login pages (e.g., the "scan to login" feature shown in Figure 1). Given the variability in QRLogin implementations, we use a best-effort strategy to automatically collect as many candidates as possible, and leave potential false positives for further manual confirmation.

Given a domain name, we employ the following four key steps to identify and confirm whether it is indeed with the QRLogin feature: 1) Identifying the login page by searching for login-related semantics within HTML elements and navigating to the relevant pages, including enhancements for diverse login page structures; 2) Detecting QR code semantics through keyword-based heuristics, excluding instances where QR codes are used for non-login purposes like resource downloading; 3) Finding the QRLogin app by analyzing prompts or links on the login page and conducting supplementary

searches in app markets; 4) Manually confirming the collected QRLogin websites by having two researchers independently verify the presence and functionality of QRLogin, cross-checking their findings to ensure data accuracy. We implement the approach based on *Playwright browser automation* tool [34], and process the various languages using the *googletrans* Python module [39]. We describe more details about this data collection process in Appendix A.

Table 1: The result of QRLogin deployments collection.

	Initial Candidates	Confirmed Websites
# Websites	611	350

Results. We use the above process to analyze the Tranco top 100K list [24] (as of Nov. 24, 2023). Overall, our approach finally identifies 350 websites using QRLogin. More specifically, as shown in Table 1, we first automatically collect 611 QRLogin website candidates through the first three steps. Then, our further manual verification confirms that 350 of them indeed implement QRLogin.

We further analyze the false positives and negatives in the automatic QRLogin website collection. False positives mainly stem from distracting content on the page, such as prompts to scan for presales consulting, descriptive text about QR codes unrelated to QRLogin, or websites offering related services like QR-code generators. These false positives can be filtered out through subsequent manual confirmation. False negatives, on the other hand, occur when websites require complex navigation paths to the QRLogin page, such as clicking on app icons without explicit textual guidance. While this limitation is common to dynamic testing crawlers, improving crawler coverage is considered as orthogonal to our work.

Prevalence and Categories of QRLogin Websites. As shown in the upper part of Figure 2, higher-ranked websites in Tranco are more likely to adopt QRLogin compared to lower-ranked ones. Notably, the QRLogin websites ranked in the top 30K account for over half (55%) of the total number of QRLogin websites. Besides, our analysis shows that these 350 QRLogin websites span 51 categories out of the 90 total categories featured in Sitereview [4], indicating a broad diversity of website types employing this authentication method. The lower part of Figure 2 shows the top 15 categories of QRLogin websites. As can be seen, these categories primarily consist of websites with sensitive security requirements, thereby highlighting the importance of secure QRLogin implementations.

2.2 User Perceptions on QRLogin Security

In the meantime, we conduct a user study that investigates various aspects of QRLogin usage, including their engagement, understanding of potential risks, and their behavior when using QRLogin. The findings from this user study provide insights for our security analysis of QRLogin implementations.

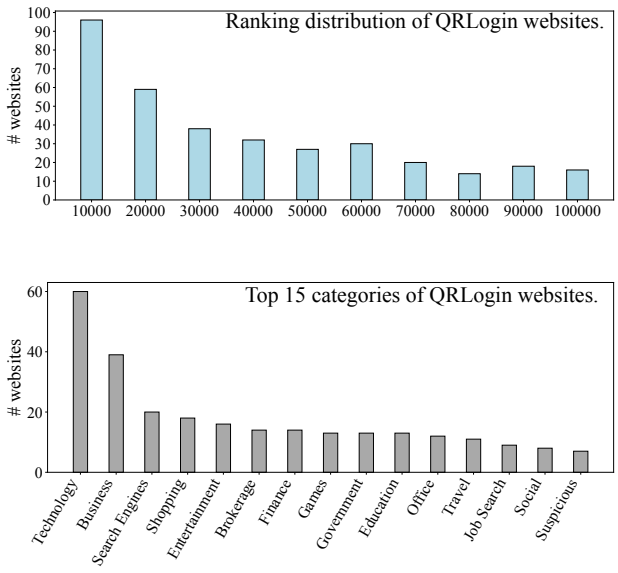

Figure 2: The ranking distribution and top 15 categories of 350 QRLogin websites. Note that in the ranking distribution figure, the x-axis value of 10,000 marks the number of websites ranked from 1 to 10,000.

Design of the User Study. In this study, we aim to answer the following research questions:

- RQ1: How do users engage with QRLogin?
- RQ2: What is the likelihood of QR code leakage in QRLogin?
- RQ3: How do users perceive the potential security risks in QRLogin?

To achieve our goal, we design a questionnaire consisting of 13 questions, aimed at gathering participants' responses. Initially, we present a real-world website example that uses QRLogin to ensure participants understand the QRLogin concept as discussed in this study. Next, participants are asked to confirm their prior experience with QRLogin by providing any websites or apps they have used that offer this functionality, aside from the example provided. If a participant provides an invalid response (i.e., the website does not support QRLogin), the participant will not be considered qualified. Following this, the rest participants will be asked to answer seven questions related to our research inquiries. Finally, we collect demographic information, including age, gender, education, and occupation, while ensuring the anonymity and privacy of all participants. The complete questionnaire, recruitment process, and demographic details are provided in Appendix B.

Results and Findings. The user study successfully obtained 180 valid reports from 180 individuals. The details on how

we design and conduct the user study, as well as the detailed result analysis are shown in Appendix B. Through this user study, we come up with the following key observations:

- *Users frequently engage in QRLogin in daily activities, and the categories of used websites are relatively sensitive.* In our survey, 39.44% of all participants use QRLogin every day, and the categories of used QRLogin websites include social, financial, governmental, etc.
- *Many users face the risk of QR code leakage when using QRLogin.* Specifically, 41.67% of all participants never or only occasionally took measures to prevent others from obtaining their QR codes. Moreover, 12.78% of them even believed it reasonable to provide their QR codes to others when they are requested, while 72.78% were uncertain.
- *QRLogin users have insufficient understanding and awareness of its associated risks.* Specifically, among the collected responses, only 15.00% of all participants correctly identified the risk of QR code leakage.

In summary, our findings underscore the critical role QRLogin plays in users' daily lives. More importantly, there exists a concerning gap in users' security awareness, probably leading to behaviors that expose them to potential threats like QR code leakage. These insights inspire our further security analysis, particularly in exploring the threat model where attackers can gain access to the victim's QR code.

3 Analyzing QRLogin Security

In this section, we analyze QRLogin security by first discussing the assumptions and threat model considered in this study. Then, we demystify the QRLogin workflow and pinpoint the security-critical variables in QRLogin to assess its compliance with security principles. Based on the analysis, we finally summarize six security flaws within the QRLogin workflow.

3.1 Threat Model and Analysis Procedure

Threat Model. To systematically analyze the security of QRLogin, we begin by defining the security assumptions and attacker capabilities based on our understanding on real-world QRLogin deployments.

- *Assumptions on channels.* We assume that all communication channels between the victim's browser, the victim's app, and the server are secure, and attackers cannot control any devices of the victim or the server.
- *Assumptions on users.* Our research concentrates on vulnerabilities caused by the QRLogin mechanism itself. In other words, we *do not* assume the QR code scanned by the user can be manipulated by the attacker (e.g., the QRLjacking attack [2] supplemented by phishing strategies).

On the other hand, our research considers the attacker who can obtain one of the following types of information:

- *The victim's QR code*. The QR code can be obtained through various methods, such as shoulder surfing [15], and social engineering to deceive the victim into sending the QR code to the attackers. The feasibility of this scenario is also confirmed by our previous user study (i.e., RQ2 and RQ3), as most normal users do not consider information about QR codes as sensitive data.
- *The victim's QR code id*. As one of the authentication factors in the QRLogin scheme, in most cases, the victim's QR id can be obtained from the QR code. Besides, even without the victim's QR code, it is also possible to brute-force the identifier of the QR code with weak randomness.
- *The victim's identifiers used for account login.* These identifiers may include the user's phone number and email address, which can be relatively easy to obtain given the widespread prevalence of information leaks. Besides, the user identifier can also be obtained via social engineering, which has been well studied by prior research [23].

Analysis Procedure. As the starting point, the entire process of demystifying the workflow of QRLogin primarily involves manual investigation. The analysis is based on a dataset of 153 QRLogin websites selected from those identified in §2.1. These websites are chosen from a total of 51 categories, with the top 3 websites from each category selected to ensure a diverse and representative sample. For each website, we first execute the complete QRLogin process and summarize the key entities involved in the workflow while capturing all network traffic exchanged between the browser, app, and server. Next, we observe and analyze the entire sequence of interactions between these entities based on the captured traffic. Finally, we examine the relevant fields in the requests and responses to identify their roles in the workflow. In cases where the role of certain fields is unclear, we perform controlled modifications to the traffic, such as altering request parameters and replaying the traffic to observe how the system responds. Through this iterative process, we can identify consistent patterns and reconstruct the typical sequence of interactions and the data involved in QRLogin. This approach allows us to gain a comprehensive understanding of how QRLogin functions across different websites.

Note that, given the heterogeneous implementations of QRLogin, such a process of manual investigation is mandatory and meaningful. As we will further show in §4, the process of understanding the QRLogin process leads to a semi-automated pipeline, allowing us to more efficiently discover 75 flaw instances in QRLogin deployments. In the meantime, the manual analysis further leads to an end-to-end auditing tool, named *QRLChecker* (see §5.1), for website developers to inspect their QRLogin implementations.

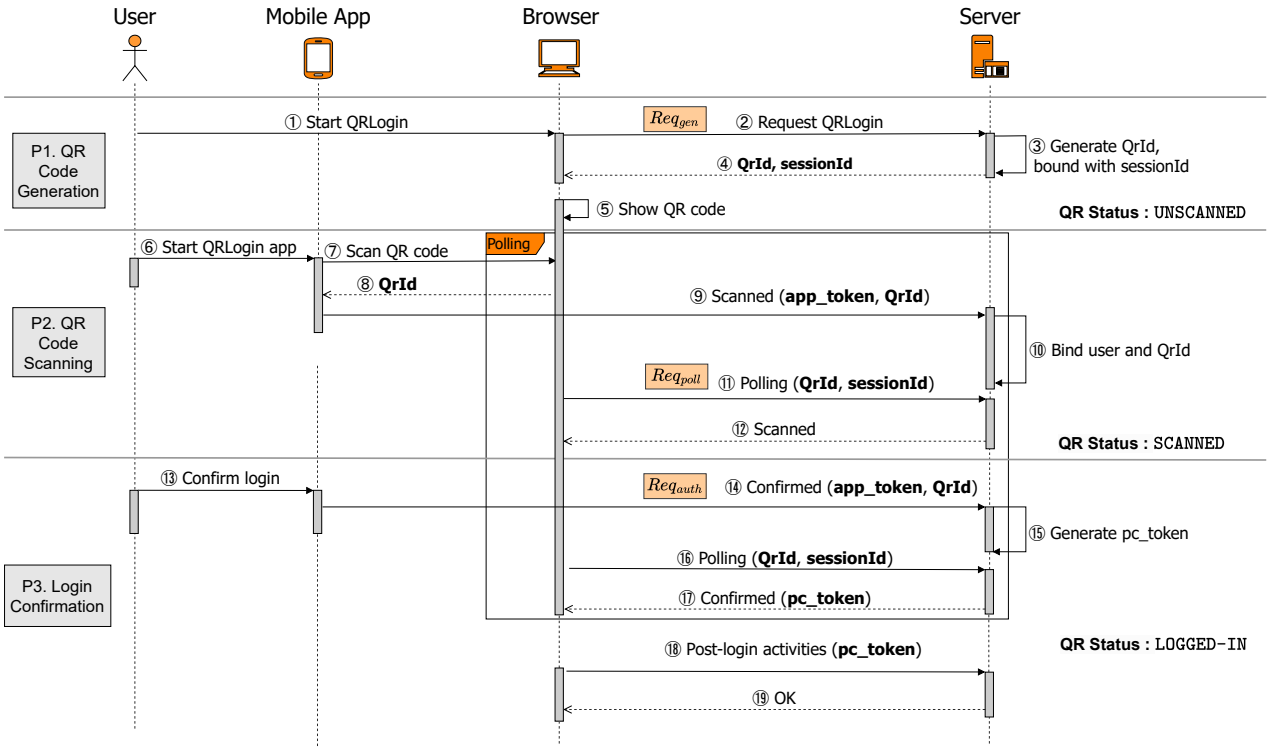

Figure 3: The abstracted workflow of QRLogin.

3.2 QRLogin Workflow

Figure 3 summarizes a generic workflow of the QRLogin scheme. It typically contains three main phases: *QR Code Generation*, *QR Code Scanning*, and *Login Confirmation*. We describe the details below.

P1. QR Code Generation. To begin with, when the user starts a QRLogin process on their browser, the browser will send a QR code generation request (Req_{gen}) to the server. Then, the server will generate and remember a *QrId* for this request (step ③), and send it back to the client browser, which will subsequently show a QR code encoding the *QrId* to the user. The server will maintain the status of the *QrId*, which at this phase is `UNSCANNED`. Furthermore, it is important for the server to bind the *QrId* with the request session, to distinguish this request from others.

P2. QR Code Scanning. The second phase is to link the user with the QR code to set up for later login authorization. In this phase, the user scans the QR code displayed on the browser using their authenticated mobile app, allowing the app to obtain the transmitted *QrId* by parsing the QR code. Subsequently, the app automatically sends a request to the server containing both the *QrId* and *app_token*, indicating that the user has scanned the corresponding QR code (step ⑨). The server, informed of the user's identity by the *app_token* sent from the app, binds the user account to the received *QrId* and updates the status of the corresponding QR code to `SCANNED`. Starting from the beginning of this phase, the browser will engage in continuous and periodic polling (Req_{poll}) of the server to retrieve the authorization status of the QRLogin.

P3. Login Confirmation. This phase is to ensure explicit user authorization via the mobile app. After scanning the QR code, the app prompts the user to opt for whether to authorize the website login request. Only upon the user's confirmation will the website successfully log in to the account. Specifically, after the user confirms authorization, the app sends a request to the server indicating the login confirmation (Req_{auth}, step ⑭). Then, the server generates a login token (*pc_token*) and updates the QR code status to `LOGGED-IN`, thereby authorizing the login. The polling request from the browser will obtain the *pc_token* and finish the website login process.

In real-world scenarios, some QRLogin implementations may exhibit minimal deviations from the generic workflow. In our analysis, we encountered two main types of variations: 1) Workflow variations, which refer to differences in the execution of specific phases, such as merging phases *P2* and *P3* or directly omit phase *P3*, bypassing explicit user confirmation; 2) Data variations, which involve differences in data generation and usage, such as generating *QrId* on the client side instead of server-side or sending the *app_token* only in step ⑭. While these deviations occur in some rare cases, they do not affect the overall workflow structure, as the core phases and interactions remain consistent. As a result, these variations do not impact the validity of our analysis.

Table 2: Summarized flaws with concerned key variables.

Phase	Flaw	Key Variable	Steps	Related Attack*
P1. QR Code Generation	F1. Unbound SessionId	sessionId	③⑪⑯	Authorization Hijacking, Double Login, Brute-force Login
	F2. Reusable QrId	QR code	⑰	Double Login
	F3. Predictable QrId		③	Brute-force Login
	F4. Controllable QrId		②③④	Authorization Hijacking
P2. QR Code Scanning	F5. Vulnerable Identity Verification	tokens	⑨⑩⑭	Universal Account Takeover
P3. Login Confirmation	F6. Unintentional Privacy Leakage	other data	⑰	Privacy Abuse

* More details of the attacks are described in §4.3.

3.3 Security-critical Variables in QRLogin

Based on the generic workflow of QRLogin, we proceed to conduct a systematic security analysis to identify potential flaws. To ensure a comprehensive analysis, we focus on security-critical variables involved in the QRLogin process, including the session identifier (*sessionId* for short), the QR code identifier (*QrId*), and the authentication tokens (*app_token* and *pc_token*). These variables are fundamental to the security of the authentication session.

Our analysis of these variables is guided by core security principles under our established threat model. Specifically, we examine each variable for *confidentiality* (can it be leaked or guessed?), *integrity* (can it be tampered with?), and *consistency* (is it maintained consistently across all phases of the workflow?). This systematic approach allows us to thoroughly evaluate the robustness of QRLogin mechanisms and uncover vulnerabilities. The detailed analysis goes as follows:

- *SessionId* is the identifier of the communication session between the victim's browser and the server. This variable cannot be leaked and tampered with under our threat model. However, if the server fails to consistently bind the *sessionId* with the corresponding *QrId* (step ③), it may give the attacker a chance to bind other unauthorized *sessionIds* with the authorized *QrId*. This breaks the consistency of the *sessionId* by allowing it to be replaced.
- *QrId* is the identifier of the QR code scanned by the mobile app. Under our threat model, it is possible for the *QrId* to be compromised and leaked to attackers, for example, by means of shoulder surfing, where attackers visually capture the QR code as it is being scanned by the victim. In addition, if the *QrId* is generated too simply, attackers might also be able to guess its value. Furthermore, if the server fails to properly manage the status of the *QrId*, for example, a *QrId* already authenticated can be re-scanned, which may allow the attackers to reuse the *QrId*.
- *Tokens*, including *app_token* and *pc_token*, are used to identify the user among the mobile app, the browser, and the server. Under our threat model, it is assumed that these tokens cannot be leaked or tampered with. However, a potential risk arises when these tokens are misused, leading to privilege confusion.

By examining the security-critical variables within the QRLogin workflow according to the core security principles, we systematically analyze the potential flaws that may exist in QRLogin implementations.

3.4 Common Flaws in QRLogin

Building on the analysis of security-critical variables, we examine real-world QRLogin deployments and have summarized six common implementation flaws. These flaws, along with the vulnerable key variables, the steps, and the phases of QRLogin in which they occur, are summarized in Table 2. We discuss the details of each flaw below.

Flaw-1: Unbound SessionId. In the QRLogin process, multiple users often simultaneously poll the server to obtain the latest status of their QR codes. The server determines which QR code status to return based on the *QrId* attached to the polling request. Once the QR code status is `LOGGED-IN`, subsequent polling requests for this QR code can lead to direct login into the corresponding account. Thus, the server needs to confirm if the polling request is from the legitimate user, i.e., the user who initiated the QRLogin.

Therefore, binding the *sessionId*, a natural identifier of the QRLogin session, to the *QrId* used in this session during its generation by the server is essential. During subsequent polling request processing, the server should reject polling requests that do not have the *sessionId* bound to the corresponding *QrId* to identify legitimate users. However, a security flaw arises if the server fails to correctly bind the *QrId* and *sessionId*, failing to distinguish legitimate users.

Flaw-2: Reusable QrId. As depicted in Figure 3, the QR code's status is continuously managed and updated by the server throughout the entire QRLogin process. It is essential for the server to invalidate the authorized status of the QR code immediately after the completion of a QRLogin session to prevent the reuse of the same QR code for authentication. However, a security flaw exists when the QR code's residual status is left unaddressed after the completion of the QRLogin, potentially allowing the QR code to be reused for unauthorized access.

Flaw-3: Predictable QrId. Due to the lack of uniform standards or guidance about the security implementation of QR-

Login, different developers may adopt different formats for *QrId*. Some developers may generate *QrIds* with weak randomness. This inadequate entropy makes the *QrIds* susceptible to prediction or brute-force attacks. Attackers exploiting this flaw can potentially know the value of *QrId* without needing to obtain the QR code directly, merely by predicting its value based on observable patterns or by systematically trying combinations in a brute-force manner.

Flaw-4: Controllable QrId. This flaw arises when the *QrId* is generated on the client side rather than securely by the server. In such scenarios, the responsibility of generating the *QrId* falls to the browser, which then transmits this identifier to the server for authentication and session management. This practice makes the *QrId* user-controllable, implying that the attackers can set their *QrId* to any value (as long as it meets basic format requirements) to deceive the server.

Flaw-5: Vulnerable Identity Verification. During QRLogin, the mobile app is required to send user credentials such as *app_token* to the server to authorize the website login request. However, the server may inadequately validate the *app_token*. For example, the app may send some other user identifiers such as a phone number along with the *app_token* in the request, but the server fails to verify the correspondence between the user identifiers and *app_token*, and directly authorizes the website login according to the provided user identifier. This flaw may allow attackers to tamper with the user identifier to bypass the authentication.

Flaw-6: Unintentional Privacy Leakage. In various specific implementations of QRLogin, sensitive user data may be embedded in the traffic between the browser and the server, which may lead to unintentional privacy data leakage. Specifically, some servers treat QRLogin as an alternative to passwords. When QRLogin is completed, the server sends the user's password back and automatically continues to authenticate using the password. In such scenarios, the leakage of sensitive data may cause additional security hazards.

The above flaws are identified based on our in-depth understanding of the QRLogin workflow and the security analysis considering how security principles are satisfied by the key variables, as well as our manual investigation of real-world QRLogin cases. In the following subsection, we will transform such human knowledge into a pipeline to analyze, and identify security issues of real-world QRLogin deployments.

4 Security Issues in the Wild

In this section, we use a semi-automated detection pipeline to investigate the real-world impact of the proposed security flaws based on our understanding of the QRLogin process. We then present how these flaws manifest in practice by detailing typical attack scenarios and providing real vulnerable cases. Our analysis reveals that a number of websites are vulnerable to these attacks. One such attack involves an attacker who, by obtaining the victim's QR code, can hijack and take over the victim's account. Another serious attack demonstrates that even with just the user's identifier, such as a phone number, an attacker can exploit the vulnerability in QRLogin implementations to gain unauthorized access to the user's account. These findings underscore the critical security risks present in current QRLogin deployments.

4.1 Detection Pipeline

To investigate the impact of these six flaws in the real world, we design a semi-automated pipeline to help detect the QRLogin implementations of websites. As shown in Figure 4, this pipeline includes four steps. First, it collects the traffic generated while the human tester completes the QRLogin process. Next, it tries to locate the key components of the QRLogin implementation, such as *QrId*, based on the collected traffic. Third, it conducts dynamic testing using these key components. Finally, the reported vulnerabilities are confirmed with the assistance of the human tester.

1) Traffic collection. We collect traffic between the browser and server, as well as between the QRLogin app and server, while a human tester performs QRLogin. This step is implemented using mitmproxy [12] to capture on the specific port while the human tester is completing QRLogin on the browser initiated by Selenium [3], confirming that the collected traffic is all generated by the QRLogin process.

2) Key component location. Based on the QRLogin traffic, we then need to identify the key components of QRLogin, including *QrId*, Req_{gen}, Req_{poll}, Req_{auth}. These components form the backbone of the QRLogin workflow, and identifying them is a prerequisite for detecting flaws.

However, identifying key components in QRLogin implementations is challenging due to significant variations across websites, such as differing *QrId* field names and values, and the lack of uniform patterns in URLs for polling requests. To solve this challenge, we leverage the fact that *QrId* serves as a bridge connecting the QRLogin website, app, and server, and appears in all three requests. This observation allows us to use *QrId* as a pivot for locating other components. We begin by identifying *QrId* through decoding the QR code used in QRLogin and comparing the decoded contents with the request fields in the collected traffic. Then, we use *QrId* to locate three types of requests by searching for the value of *QrId* in the collected traffic.

Specifically, we parse the contents of QR codes by developing a universal parser that handles various formats like JSON and further filter out unrelated values like timestamps. When comparing the common field in the QR code and requests captured during QRLogin to identify *QrId*, we sort these requests based on URL frequencies and start from the most frequent one to compare. This is inspired by the fact that Req_{poll} exhibits a notably high frequency and periodic nature due to websites requiring constant updates on the user's scanning

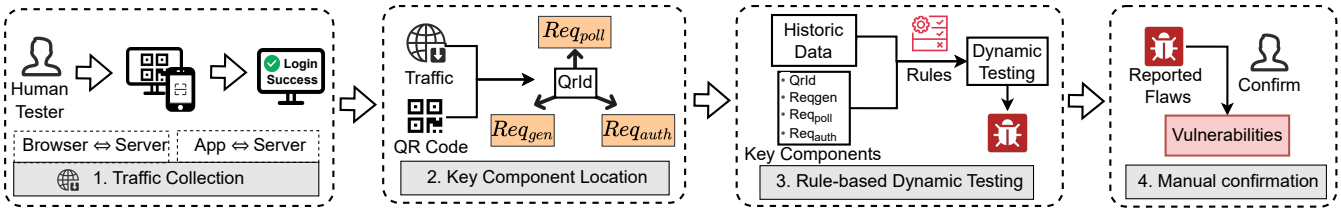

Figure 4: The semi-automated pipeline of analysis.

status. The tool will report the matched value and request as *QrId* and Req_{poll}. Then, searching the first occurrence of *QrId* in the network traffic can locate Req_{gen}, and the app's requests containing *QrId* are classified as Req_{auth}.

3) Rule-based dynamic testing. Based on the identified key components, the pipeline proceeds to detect flaws according to the predefined rules listed in Table 3. This process involves two main steps: setting up the premises and assessing the detection conditions in the rules. First, we need to obtain valid values of key variables, such as *QrId* and *sessionId*, which are essential for testing. However, due to the varying request formats and parameter requirements across websites, generating valid values that meet all site-specific requirements is challenging. To address this, we dynamically acquire valid values by learning from the collected historical traffic to simulate specific user requests. Specifically, we can acquire a new valid *QrId* value by replaying the QR code generation request (Req_{gen}) and parsing its response. Likewise, a valid *sessionId* is retrieved by identifying and replaying the request that initially sets the cookie containing the *sessionId*, thus obtaining a fresh one for testing.

Subsequently, to assess the rule conditions, we need to evaluate the responses with different statuses, such as `UNSCANNED` and `LOGGED-IN`. We adopt an approach to comparing them with the corresponding historical response with the specific status. Specifically, we first filter the random field values in the evaluated responses, including trace ID (used for tracking), timestamp, and packet signature, and then conduct a comparison. This approach helps distinguish responses of different statuses with high similarity on the same website and across different websites with varying response formats.

Specifically, the dynamic testing implementation involves parsing request and response fields, focusing on URLs, headers, and content where *QrId* might appear. Response contents are parsed based on the *Content-Type* header, with special handling for base64-encoded data. Additionally, since *QrId* may have different names across request types like Req_{poll} and Req_{gen}, we record these variations to accurately identify *QrId*. If client-generated QR codes are detected, we construct a new valid *QrId* by modifying a character in the original *QrId* and sending it to the server.

4) Manual confirmation. After the semi-automated tool reports flaws, we further perform manual confirmation to ensure the exploitability of the detected vulnerabilities.

4.2 Overall Results

We apply the above detection pipeline on the identified 350 QRLogin websites to uncover instances of the six flaws we proposed. To begin with, our analysis notices that part of the identified websites adopt identical QRLogin implementations. For example, those websites from the same company, or websites using the same third-party login apps. To avoid redundant analysis and results, we perform additional deduplication based on the QRLogin apps integrated by these websites. Finally, our research obtains a total number of 181 unique QRLogin implementations adopted by these 350 websites. Among them, 109 unique deployments are found to be testable. The overall results are shown in Figure 5.

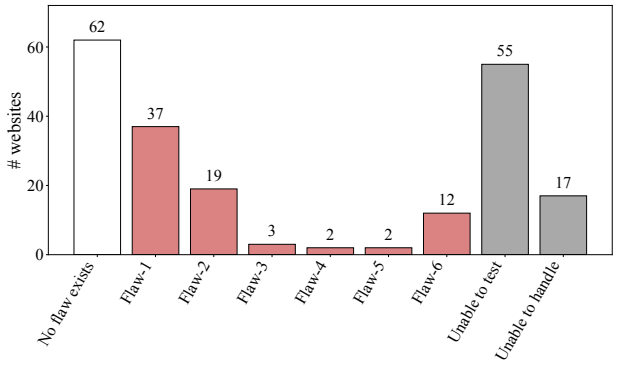

Figure 5: Distribution of the detected flaws in various websites.

Detected Flaws. Among the 109 testable websites, we find 47 (43%) websites with a total of 75 flaws in their QRLogin implementations. Note that one website may have multiple flaws. These websites with flaws span various sensitive categories, such as "Technology/Internet", "Business/Economy", and "Financial". Besides, nearly half of the flawed websites are ranked within the top thousands, indicating that QRLogin flaws are prevalent across a wide spectrum of websites with significant risks. As for specific flaws, the results show that

Table 3: Rules for detecting flaws in QRLogin. *sid*: sessionId; $Req(sid, qrid)$: request including *qrid* under session *sid*; $Authed(qrid)$: whether *qrid* has been authorized by a legitimate user; $Resp(sid, qrid)$: return the status of response for request including *qrid* under *sid*; $IsDigits(qrid)$: whether *qrid* consists solely of numeric digits; $GetFields(Req)$: return set of fields in the request.

Flaw	Premise	Rule Condition
F1. Unbound *sessionId*	$sid_1 \neq sid_2$, $qrid_1 \neq qrid_2$, $Req(sid_1, qrid_1)$, $Req(sid_2, qrid_2)$, $\neg Authed(qrid_1)$, $\neg Authed(qrid_2)$	$Resp(sid_2, qrid_1) =$ `UNSCANNED`
F2. Reusable *QR code*	$sid_1 \neq sid_2$, $qrid_1 \neq qrid_2$, $Req(sid_1, qrid_1)$, $Req(sid_2, qrid_2)$, $Authed(qrid_1)$, $\neg Authed(qrid_2)$	$Resp(sid_2, qrid_1) =$ `LOGGED-IN`
F3. Predictable *QrId*	$Req(sid, qrid)$	$Len(qrid) \leq 6 \wedge IsDigits(qrid)$
F4. Controllable *QrId*	$Req(sid, qrid)$	$qrid \in GetFields(Req_{gen})$
F5. Vulnerable Identity Verification	$A = GetFields(Req_{auth})$	$\exists a \in A, a \in PII$
F6. Unintentional Privacy Leakage	$\exists resp, A = GetFields(resp)$	$\exists a \in A, a \in PII$

Flaw-1 is relatively pervasive (34%). This suggests that websites often overlook the verification of the requester when implementing QRLogin, resulting in potential risks.

As for the 72 untestable websites, 55 of them have specific account creation requirements, such as needing a particular country's phone number or social ID, which makes testing infeasible. Additionally, 17 websites employ complex encryption techniques or specialized mechanisms, such as long connections or the use of cookies and other data for polling instead of the *QrId*, which deviates from the common QRLogin implementations we analyze.

We further analyze the false positives reported in the detection pipeline before manual confirmation, identifying 87 reported flaws, of which 12 are false positives across 7 websites. These false positives primarily stem from the additional website protections. Specifically, some websites may include and check special fields attached to certain custom requests, like device information. The detection pipeline reports flaws when polling request responses indicate a successful login, but manual verification reveals that the login is blocked later due to extra protection checks. For false negatives, we manually verify the websites that are detected without flaws and find 5 cases. Further analysis discovers that the reasons include the limitations and failed decoding of current QR code decoding tool. As these 5 cases are not detected by the tool, they are not included in the detection results listed above. Nevertheless, we have responsibly reported these vulnerabilities to their vendors.

4.3 Typical Attacks & Case Studies

In this subsection, we demonstrate 5 typical types of attacks caused by the identified flaws, including ① Authorization Hijacking, ② Double Login, ③ Brute-force Login, ④ Universal Account Takeover, and ⑤ Privacy Abuse.

The overview of all attack types and their contributing flaws is shown in Table 4. We also list the number of QRLogin websites vulnerable to each attack. Overall, many websites are confirmed vulnerable to these attacks, implying the widespread insecurity of QRLogin in real-world deployments. These attacks can lead to severe consequences such as account takeover, privacy violation, and even financial loss. We discuss the details of each type of attack below, along with some real-world case studies to better illustrate the attack process and consequence.

Table 4: Overview of attack types and contributing flaws.

No.	Attack	Concerned Flaws	# Websites
1	Authorization Hijacking	F1 | F4	37
2	Double Login	F1 & F2	17
3	Brute-force Login	F1 & F3	1
4	Universal Account Takeover	F5	2
5	Privacy Abuse	F6*	7

* Please note that the prerequisite for Attack 5 is successful login through any one of Attacks 1 to 3, followed by the addition of Flaw-6.

Attack-1: Authorization Hijacking. When a QRLogin website exhibits *Flaw-1*, it is vulnerable to authorization hijacking attacks. This attack is illustrated in Figure 6, which works as follows: 1) The victim starts the QRLogin process by sending Req_{gen} and receives a QR code containing a *QrId*; 2) The attacker steals the victim's *QrId* through methods like shoulder surfing or social engineering, and waits for the victim to authorize; 3) After the victim normally authorizes on the app (sending Req_{auth}), the attacker promptly races to send Req_{poll} with the victim's *QrId* to the server. Note that the attacker can poll at a higher frequency to hijack the victim's authorization within the attack time window marked as the red area in Figure 6. Since the server does not bind the *QrId* with the specific request session, the attack can successfully login to the victim's account before the victim does.

Additionally, *Flaw-4* also allows attackers to launch au-

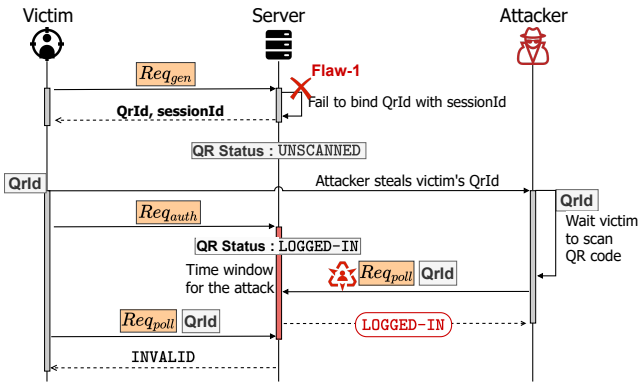

Figure 6: Authorization hijacking attack.

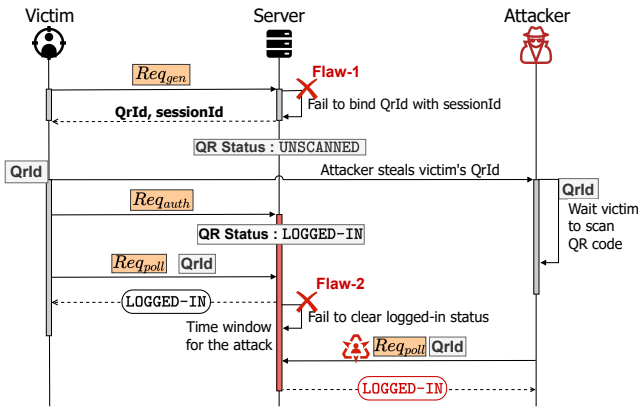

Figure 7: Double login attack.

thorization hijacking attacks. Specifically, since *QrId* is locally generated by the client, even if the server first binds the victim's *QrId* with their *sessionId*, attackers can still have a chance of re-binding. This is implemented by sending Req_{gen} with the victim's *QrId*, thereby re-binding it with the attacker's own *sessionId*. Then, attackers can achieve the authorization hijacking attack if their Req_{poll} reaches the server within the attack time window.

As for the practicality of *Attack-1*, attackers have ample opportunity to launch the attack given that users typically take several seconds to open the mobile app, scan, and confirm login. Moreover, attackers can impersonate the victim to poll at a higher frequency than the victim to race. Specifically, websites normally poll at second-level intervals, whereas an attacker can use millisecond-level intervals to effectively facilitate a successful attack.

Case-1: Taobao (taobao.com). Taobao is a leading shopping platform with nearly 500 million users[4], which exists *Flaw-1* and is vulnerable to authorization hijacking attacks. By exploiting this attack, the attacker can log into the victim's Taobao account on the website without authorization, thereby compromising sensitive information such as purchase history, browsing records, sensitive shipping address, phone number, as well as the recipient's information. We have reported this vulnerability to NVDB and received an NVDB ID as NVDB-CAPPVD-2024143978.

Attack-2: Double Login. When a QRLogin website exhibits both *Flaw-1* and *Flaw-2*, it is vulnerable to double login attacks. As shown in Figure 7, the attack workflow is as follows: 1) The victim starts the QRLogin process by sending Req_{gen} and receives a QR code containing a *QrId*; 2) The attacker steals the victim's *QrId* and waits for the victim to authorize; 3) After the victim normally logs into their account, the attacker sends Req_{poll} with the victim's *QrId* to the server. As the status of the *QrId* is reusable (Flaw-2) and it is not bound with a specific session (Flaw-1), the attacker can also log into the victim's account, leading to a double login of the victim's account.

In this type of attack, the attacker does not need to race against the victim. Therefore, this attack has a longer time window compared to *Attack-1*, as shown by the red area in Figure 7. Moreover, this attack does not disrupt the victim's normal login process, leaving the victim unaware of the attack.

*Case-2: ** (**.ru).* **[5] is a popular social media platform in Russia, which has a large number of users and ranks within the top 500 in the Tranco list. This website suffers from *Flaw-1* and *Flaw-2*, making it vulnerable to the double login attacks. Through exploiting the flaws, an attacker can gain unauthorized access to a victim's account unnoticed, thereby monitoring the victim's social activities and potentially impersonating them to perform malicious actions. We have reported this issue to the vendor but have yet to receive a response.

Attack-3: Brute-force Login. In the presence of both *Flaw-1* and *Flaw-3*, a website is vulnerable to brute-force login attacks. When the *QrId* is violently enumerable, attackers can execute brute-force attacks without the need to steal the *QrId* from the victim. By systematically traversing possible *QrId* values, attackers can collide with any victim's *QrId*, gaining access to their login authorization, as *Attack-1* did. In real-world scenarios, websites usually neglect to check and limit polling frequency, making such brute-force attacks feasible. Furthermore, if the website also suffers from *Flaw-2*, the chances of a successful attack are even higher. As long as the brute-forced *QrId* matches a valid QR code, attackers can access user accounts illegally.

*Case-3: ** (**.com).* **[5], a global virtual community service on the Internet with users from multiple countries such as the United States and Russia, ranks among the top 500 in the Tranco list. This website exhibits *Flaw-3*, making it vulnerable to brute-force attacks. Specifically, the *QrId* used in

[4] https://www.taobao.com/about/intro.php

[5] We anonymized the identity as we have not received any response from vendors yet.

its QRLogin consists solely of 6-digit pure numbers. Attackers can exploit this vulnerability by conducting brute-force enumeration, and upon collision with a victim's *QrId*, they can take control of the victim's account. We are actively contacting developers to facilitate the fix of the vulnerability.

Attack-4: Universal Account Takeover. When a website exhibits *Flaw-5*, its QRLogin process becomes vulnerable to universal account takeover attacks. Attackers only need to know the victim's account identifier, such as a phone number to log into the victim's account. The specific attack process is shown in Figure 8. This vulnerability originates from the server's failure to verify the *app_token* in Req_{auth} but only checking the account identifier. Consequently, attackers can take over any victim's account by simply substituting the account identifier in Req_{auth} with that of the victim.

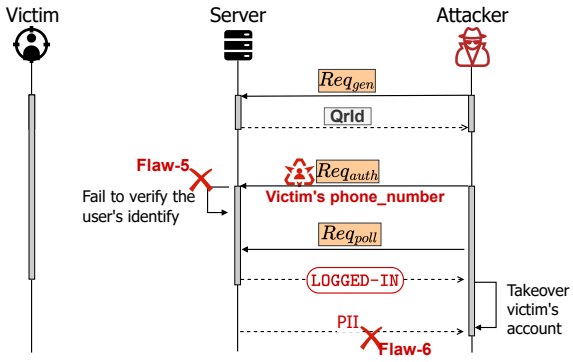

Figure 8: Universal account takeover attack.

*Case-4: **(**.cn) & **(**.cn).* A Chinese provincial government website and a popular cloud storage service provider with more than 100 million users are found to exhibit *Flaw-5*, making them vulnerable to account takeover attacks. Specifically, their servers directly authorize the login for the account linked to the phone number in Req_{auth} without validating the *app_token*. As a result, attackers can log into any victim's account using only the victim's phone number. This allows attackers to exploit the victim's government services on the government website, and illegally access private files stored on the cloud service website. We have reported these vulnerabilities and received a formal thanks letter from the government and an NVDB ID as NVDB-CAPPVD-2024672890, respectively.

Attack-5: Privacy Abuse. When a QRLogin website exhibits *Flaw-6*, it is potentially vulnerable to privacy abuse attacks, where attackers can acquire the victim's highly sensitive information such as passwords from the server's responses. Furthermore, these data can be abused by attackers to cause more harm. Note that to conduct such attacks, attackers have to first complete any one of *Attack-1*, *Attack-2*, or *Attack-3*.

Case-5: Chinese National Digital Library (nlc.cn). This is one of the largest library management platforms in China. However, this platform suffers from *Flaw-1*, *Flaw-2*, and *Flaw-6*, making it vulnerable to privacy abuse attacks. Once the attacker compromises the QRLogin of the website, they can obtain the victim's username and password in plaintext from the server's responses, thus gaining full control over the account. More seriously, considering that people often reuse passwords across different platforms [38], this vulnerability allows attackers to potentially compromise the victim's other accounts through credential stuffing [36]. We further explore why the website returns passwords and discover that the website uses QRLogin as a pre-step to password login, using the returned password to automatically complete the password login process. We have reported this vulnerability to NVDB and received an NVDB ID as NVDB-CAPPVD-2024678490.

4.4 Responsible Disclosure

We have responsibly disclosed the identified flaws and vulnerabilities to the respective vendors, providing comprehensive details, including descriptions of the vulnerabilities, proof of concept (POC) scripts, and other supporting documents to facilitate their understanding and mitigation efforts. At the time of this paper's submission, we have received acknowledgments in the form of 17 CNVD IDs and 25 NVDB IDs. Notably, there is no overlap between the vulnerabilities that obtained CNVD and NVDB IDs.

While some vendors have either not yet responded or have underestimated the risks, we are actively engaged in discussions with them to clarify and address the issues. We hope that our efforts in responsible disclosure will significantly contribute to enhancing the security of QRLogin systems in real-world applications.

5 Enhancing QRLogin Security

To enhance the security of QRLogin, we consider both the involved stakeholders, i.e., developers and users. For developers, we present *QRLChecker*, an automated auditing tool that evaluates QRLogin implementations and provides targeted measures to fix the existing vulnerabilities. For users, we provide tailored suggestions to improve their awareness and safety based on the insights from our user study (§2.2). This dual focus aims to strengthen QRLogin security from both technical and user perspectives.

5.1 *QRLChecker*: A QRLogin Auditing Toolkit

Overview. To enhance the security of real-world QRLogin implementations, we further propose a dedicated auditing tool for developers, called *QRLChecker*, as shown in Figure 9. This tool is designed to assist developers in assessing the security of their QRLogin implementations by leveraging prior knowledge provided by the developers themselves. With the pre-knowledge, supplied in the form of a configuration

file, *QRLChecker* can automatically detect and report existing security vulnerabilities without misidentification of key components in QRLogin implementations, while also offering tailored mitigation suggestions.

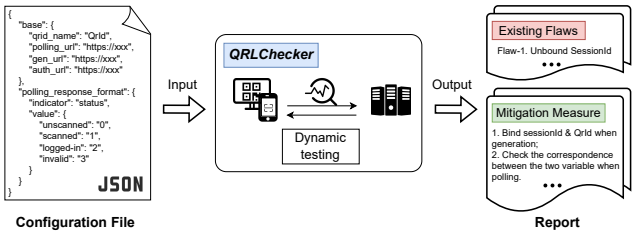

Figure 9: Overview of *QRLChecker*.

Developer Input. When using *QRLChecker* to audit a QRLogin deployment, developers are required first to provide a configuration file that serves as essential pre-knowledge for the tool. This configuration file is a JSON file containing key components of the QRLogin implementation. Specifically, the file should include: 1) the name of the field that holds the *QrId* in the requests and responses; 2) the URLs corresponding to the three types of requests (Req_{gen}, Req_{poll}, Req_{auth}); and 3) the responses format of the polling request, particularly the variable name indicating the status of the QRLogin process and corresponding values that represent different states, such as `UNSCANNED`, `LOGGED-IN`. By providing this detailed configuration, developers equip *QRLChecker* with the necessary context to accurately analyze the security of the QRLogin implementation.

***QRLChecker* Output.** Given the input configuration file, *QRLChecker* generates a comprehensive report that includes two primary sections. The first section identifies and details the security flaws present in the audited QRLogin implementation, outlining the specific vulnerabilities and their potential impacts on the authentication process. The second section offers targeted mitigation methods and recommendations for each identified flaw, providing developers with clear, actionable steps to address the vulnerabilities effectively and enhance the overall security of their QRLogin implementations.

Detailed Running Process. In an auditing process, *QRLChecker* starts by reading the developer-provided configuration file to gather the necessary information for the audit. It then launches an automated testing process by opening a browser to access the target website, where the developer is required to complete a QRLogin process like a normal user to allow the tool to capture the required traffic. Using the configuration details, *QRLChecker* identifies the relevant requests and fields in the captured traffic. It then conducts dynamic testing by modifying and replaying requests based on predefined rules (listed in Table 3). The results are compiled into a report that summarizes the identified vulnerabilities and offers practical measures to fix them, helping the developer improve the security of their QRLogin implementations.

This approach allows for more precise and relevant security assessments by incorporating site-specific knowledge into the detection process. As a result, *QRLChecker* enhances the accuracy of vulnerability detection and provides developers with actionable insights to strengthen the security of their QRLogin implementations, ultimately contributing to safer real-world QRLogin deployments.

In terms of the difference between our detection pipeline (§4.1) and *QRLChecker*, they serve distinct purposes with different workflows. The pipeline is to assist our human testers in identifying potential security flaws across a variety of QRLogin websites. It operates at scale, automatically detecting key components, which requires further human verification to confirm accuracy. In contrast, *QRLChecker* is designed for developers to audit and enhance the security of their QRLogin implementations. By requiring a configuration file with key component information as input, *QRLChecker* can provide precise auditing results and targeted mitigation advice.

5.2 Suggestions for Developers

In addition to the targeted mitigation measures provided in the *QRLChecker* audit reports, we offer best practices throughout the QRLogin lifecycle for developers to ensure secure QRLogin implementations. These suggestions serve as a comprehensive guide to bolster the security of QRLogin deployments, ensuring that common vulnerabilities are effectively avoided. Developers should take responsibility for avoiding security flaws through the following specific measures:

- *Generation of QR code.* Developers should delegate the generation of *QrId* to the server side using a secure algorithm and lengthy strings of mixed characters to avoid *Flaw-3* and *Flaw-4*. Moreover, the generated *QrId* should be bound with the *sessionId* of the user's session, provided in the request for QR code generation.
- *Utilization of QR code.* Developers should ensure that only polling requests from the legitimate user who initiates the QRLogin are considered valid by verifying the correspondence between the *sessionId* and *QrId* in the requests, to avoid *Flaw-1*. Besides, developers must promptly clear `LOGGED-IN` status of the QR code after the user completes the QRLogin to avoid *Flaw-2*.
- *Validation of tokens.* Developers should carefully issue valid *app_token* for authenticated QRLogin apps. When authorizing QRLogin, developers must verify the validity of *app_token* in Req_{auth} and only authorize the login for the account linked to that *app_token* to avoid *Flaw-5*.
- *Protection of sensitive data.* Developers should be cautious to prevent unintentional leakage of sensitive information in QRLogin responses to avoid *Flaw-6*. Meanwhile, necessary sensitive information should be transmitted encrypted.

Furthermore, QRLogin should be implemented as an independent authentication method to avoid posing additional threats to other authentication methods like passwords.

5.3 Suggestions for Users

Based on the findings in our user study (§2.2), we provide suggestions to enhance QRLogin security for users, helping users better protect their accounts.

- *Protecting QR codes actively.* Users should take proactive measures when using QRLogin, such as hiding the QR codes to protect them from being stolen or captured when scanning the QR code, to mitigate the attacks caused by the leakage of QR codes.
- *Raising security awareness.* Users should be fully aware of the potential risks in the QRLogin process and avoid intentionally sharing their QR codes with others.

6 Discussion

Comparison with Existing Protocols. Several existing protocols, such as RFC 8628 OAuth 2.0 Device Authorization Grant [13] and OpenID Connect Client-Initiated Backchannel Authentication Flow (CIBA) [16], address authentication in similar contexts. QRLogin shares common ground with them in that they all address authentication scenarios where the device initiating the authorization differs from the device used for authorization. However, their scopes and designs diverge significantly. Specifically, RFC 8628 is designed for input-constrained devices like printers and TVs, while QRLogin is primarily used for website authentication. CIBA, in turn, does not involve direct interactions between two devices as QRLogin does. Instead, the server communicates the authentication request directly to the user. As a result, the leakage channels, such as QR codes in QRLogin, do not exist in CIBA.

In our work, all identified websites rely on mobile apps for authorization and use QR code scanning for direct interactions between two devices, making RFC 8628 and CIBA protocols inapplicable. These differences further highlight QRLogin's unique security concerns. The flaws we identified, such as Flaw-2 and Flaw-3, are specific to QR code usage. Nevertheless, the principles underlying these flaws can be generalized to broader authentication systems, offering insights that extend beyond QRLogin.

Limitation. This paper adopts a dynamic testing approach to audit the security of the QRLogin process, thus it may incur the common shortcomings of dynamic testing. For example, for the 181 QRLogin websites with unique implementations, we find a large proportion of them (55) can not be tested due to certain account registration requirements. Also, when capturing and parsing the network traffic, we encounter the problem of potential parameter encoding and encryption, which brings challenges to all dynamic testing approaches [11]. We believe the advancement of dynamic techniques can further improve the performance of our proposed method.

Formal methods. When detecting implementation flaws, we apply a rule-based approach to examine the existence of certain network traffic patterns. We noticed that formal methods are widely used to evaluate the security and correctness of network or authentication protocols. However, the diversity and complexity of current QRLogin implementations pose significant challenges to the direct application of formal methods. Consequently, we advocate for future standardization efforts in this new scheme to enable more effective use of formal methods in evaluating QRLogin security.

Manual efforts. In our study, the heterogeneous nature of websites necessitates manual confirmation of the QRLogin websites collected by the tool to ensure accuracy. Additionally, the semi-automated detection pipeline requires manual steps for app preparation, QR code scanning, and vulnerability confirmation. Such manual efforts are unavoidable, as seen in similar research on app authentication [20].

Future Work. As an emerging authentication mechanism, there are multiple aspects worth exploring in the future. One particularly intriguing and crucial area for subsequent research is the usability or usable security of QRLogin. During our manual analysis, we found that some websites may overlook users' rights of awareness and cancelability during the QRLogin process. They may directly omit a critical phase (*P3*) in the workflow or do not provide an effective mechanism for users to manage the authenticated sessions. These usability issues may cause users not to be able to correctly perceive and protect the QRLogin processes. We think this could be an interesting research target in the future.

Third-party QRLogin. During our research, we observe that a subset of websites utilize third-party QRLogin mechanisms, employing popular apps such as *WeChat* to scan QR codes and authenticate users. This method is more like an *OAuth* procedure. Although current analysis has not revealed any specialized security flaws within these third-party implementations, the use of QRLogin via third-party apps requires further investigation to ensure its robustness and security.

7 Related Work

QR Code Security. Previous research on QR codes mainly focuses on security issues and defense mitigation measures during the QR code scanning process. Han et al. [19] proposed an attack named *Medusa*, which exploits vulnerabilities in QR code parsers to trigger the custom remotely accessible handlers (RAHs) and gain unauthorized access to the sensitive data within the mobile apps. To enhance the security of QR codes, various methods have been proposed, including automated detection of malicious QR codes [22], utilizing fingerprint to identify the display screen used for presenting the QR code [27], and employing novel QR code systems to pre-

vent scanning by unauthorized individuals [37]. Additionally, Mavroeidis et al. [33] introduced the Quick Response Code Secure (QRCS) approach, which employs digital signatures to verify the authenticity of QR codes, to mitigate the risks of man-in-the-middle and replay attacks.

In contrast, our research focuses on the security issues present in the QR code scanning-based login process. The most related work to ours is QRLJacking [2], wherein attackers subvert the authentication process by replacing the victim's QR code with one of their own. This can be achieved by tricking the victim into scanning a QR code controlled by attackers. Nonetheless, the success of such operations depends on the victim's interaction, which may raise suspicions if not carefully executed. This kind of attack is much more like a social phishing attack, while in this paper we study the security of the QRLogin process and the implementation flaws in the protocol design and implementations.

Authentication Security. The security of authentication mechanisms is a critical research topic in the security area, and many authentication methods have been studied [7, 17, 42, 43, 45]. Traditional password authentication is vulnerable to guessing attacks, brute-force attacks, and credential stuffing attacks [8, 32]. Facial recognition can be invaded by trojaned model provided by attackers [29]. A recent study [44] also systematically revisits the design and implementation details of face recognition in mobile apps, finding that their protocol and workflow suffer from various attacks. As a result, even with just a few photos of the victim, it is possible to bypass facial recognition completely.

SMS one-time passwords (OTP) is another widely used authentication scheme, which sends a one-time code to the users for login. However, SMS OTP may suffer from weak randomness issues [31]. Besides, attackers can intercept SMS OTP through malware [10, 26], SIM card swapping attacks [25], and wireless network hijacking [21], thus facilitating account hijacking [18]. By contrast, we investigate the QR code-based authentication scheme, which is a newly emerging authentication method increasingly adopted by popular websites. However, this new scheme receives less attention than others. As far as we know, we are the first to systematically evaluate the security of QRLogin deployed in practical settings.

Website Login Security. The security of website login mechanisms has gained significant attention among researchers. Some studies have investigated website cookies and identified various security vulnerabilities, such as the susceptibility of cookie integrity to compromise [41] and their susceptibility to hijacking attacks [14]. Additionally, research on website password managers has identified security issues and provided recommendations for enhancing existing password management solutions [28, 30, 35, 40]. Furthermore, through large-scale analysis of password-based website login implementations, Al Roomi et al. [9] have revealed insecure login policies existing in real-world deployments. In our work, we extend the study of web security to include QR code-based logins and conduct a systematic investigation. This work on the newly emerged QRLogin scheme broadens the scope and introduces new perspectives to the field of web login security research.

8 Conclusion

This paper conducts the first systematic analysis of QRLogin security in real-world deployments. We first understand the real-world QRLogin by measuring its deployments and exploring user perceptions on its security. Based on the findings, we set up a realistic threat model for QRLogin. We then summarize a typical QRLogin workflow, identify crucial variables, and assess compliance with security principles, during which process we uncover six common implementation flaws.

We evaluate 109 QRLogin websites using a semi-automated pipeline and find that 47 websites possess at least one security flaw type, posing serious risks like authorization hijacking and universal account takeover. We have responsibly disclosed these vulnerabilities, resulting in the issuance of 42 vulnerability IDs. Furthermore, we provide developers with an auditing tool and give mitigation suggestions for developers and users to enhance QRLogin security.

Acknowledgments

We would like to thank our shepherd and the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Natural Science Foundation of China (62102091, 62172104, 62172105, 62472096, 62102093, 62302101, 62402114, 62402116, 62202106), National Key Research and Development Program (2021YFB3101200). Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China.

Ethics Considerations

During this research, we strictly adhered to USENIX Security '25 Ethics Guidelines [5], ensuring that no potential harm was caused to any stakeholders involved. This section discusses the ethical considerations we addressed throughout our study, from performing the user study, testing QRLogin websites for security analysis, to the responsible disclosure of identified vulnerabilities.

User Study. Our study was approved by the Institutional Review Board (IRB). In our user study to investigate users' perceptions of QRLogin security, we collected the participants' anonymized background information to analyze and demonstrate the demographics of this user study. We ensured that participants were fully informed about the purpose of data collection, which is only for research, and we obtained

consent for data collection. The data gathered, which was limited to non-sensitive demographics as outlined in Questions 10-13 in §B.3, ensures anonymity, making it impossible to contact or deduce specific individuals from the collected information.

On the other hand, our questionnaire-based user study only asked about their habits, methods, and perceptions of using QRLogin. The survey did not request any confidential account information or credentials from participants.

Testing Procedures. During our testing, we prioritized ethical standards to ensure that no harm was caused to the users and the websites. On the one hand, all tests were exclusively conducted using our own devices and test accounts, avoiding any potential risk to other users' accounts.

To avoid imposing unnecessary burdens on external systems, our tests were carefully designed and executed to simulate the typical behavior of regular users. The number of requests made during testing was strictly controlled within the normal user request range (i.e., fewer than 10 requests) to ensure that no excessive traffic or computational load was placed on any servers, thereby preventing any disruption to the normal operation of the websites.

Moreover, as for the manual confirmation of the authorization hijacking attack, given it is a kind of race attack, launching a real attack would require the attacker to send requests to the server at a high frequency, which may cause stress on the server. Therefore, we adopted an approach of blocking the victim tester's polling requests to confirm the race condition vulnerability. This approach avoids the impact on the website servers that would result from simulating the attack with high-frequency polling requests.

Vulnerability Disclosure. All vulnerabilities identified in this study were promptly and responsibly reported to the relevant vendors following the specified ethical guidelines, along with proposed mitigation measures to minimize any potential risks. For vendors (websites) hosted in China, we reported the vulnerabilities through official platforms, including CNVD and CAPPVD, in compliance with the local regulation requirements (i.e., China). For vulnerabilities affecting other regions, we promptly contacted the affected vendors directly to report the security issues in their QRLogin implementations and provide practical measures to fix them, preventing potential exploitation. Note that while we initially attempted to report the vulnerabilities via the Common Vulnerabilities and Exposures (CVE) platform, the idea seems not to work. According to CVE's inclusion criteria, vulnerabilities on specific websites are not assigned CVE IDs (as per INC3 in Inclusion Decisions [1]). As of our submission, we did not disclose any information of a specific vendor, if we have not received their response.

Open Science

We comply with the USENIX Security open science policy by open-sourcing all research artifacts associated with our work. The artifacts include:

- **Detection Pipeline and Auditing Tool (*QRLChecker*).** The source code for these tools is open-sourced to vetted researchers and vendors upon request. This controlled release addresses ethical concerns, as unrestricted access may enable attackers to abuse these tools to identify and exploit potential vulnerabilities in the real world. By carefully vetting those who request access, we take responsibility for preventing such risks and ensuring that the tools are used only for legitimate purposes.
- **Dataset and Data Collection Scripts.** The dataset of websites we analyzed in this work and the scripts used to collect QRLogin websites are directly open-sourced.
- **User Study Data.** The raw data collected from our user study is publicly available.

The dataset, data collection scripts, and user study data are directly open-sourced at `https://doi.org/10.5281/zenodo.14676762`, while the detection pipeline and auditing tool (*QRLChecker*) are available through responsible release at `https://doi.org/10.5281/zenodo.14676842`, accessible to vetted researchers upon request.

References

[1] "Cve counting rules cve," https://cve.mitre.org/cve/list_rules_and_guidance/counting_rules.html, 2024.

[2] "Qrljacking, an attack introduced on owasp." https://owasp.org/www-community/attacks/Qrljacking, 2024.

[3] "Seleniumm, a suite of tools for browser automation." https://www.selenium.dev/, 2024.

[4] "Symantec sitereview," https://sitereview.bluecoat.com/#/, 2024.

[5] "Usenix security '25 ethics guidelines," https://www.usenix.org/conference/usenixsecurity25/ethics-guidelines, 2024.

[6] "Wenjuanxing user study," https://www.wjx.cn, 2024.

[7] A. Acien, A. Morales, R. Vera-Rodriguez, J. Fierrez, and R. Tolosana, "Multilock: Mobile active authentication based on multiple biometric and behavioral patterns," in *1st International Workshop on Multimodal Understanding and Learning for Embodied Applications*, 2019, pp. 53–59.

[8] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "Pasta: password-based threshold authentication," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2042–2059.

[9] S. Al Roomi and F. Li, "A {Large-Scale} measurement of website login policies," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 2061–2078.

[10] A. Center, "Additional requirements for the use of specific permissions," 2019.

[11] Q. Chen and A. Kapravelos, "Mystique: Uncovering information leakage from browser extensions," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1687–1700.

[12] A. Cortesi, "Mitmproxy: a man-in-the-middle proxy." https://mitmproxy.org/, 2024.

[13] W. Denniss, J. Bradley, M. B. Jones, and H. Tschofenig, "OAuth 2.0 Device Authorization Grant," RFC 8628, Aug. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8628

[14] K. Drakonakis, S. Ioannidis, and J. Polakis, "The cookie hunter: Automated black-box auditing for web authentication and authorization flaws," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1953–1970.

[15] M. Eiband, M. Khamis, E. Von Zezschwitz, H. Hussmann, and F. Alt, "Understanding shoulder surfing in the wild: Stories from users and observers," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 4254–4265.

[16] G. Fernandez, F. Walter, A. Nennker, D. Tonge, and B. Campbell, "Openid connect client-initiated backchannel authentication flow - core 1.0," https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html, 2021.

[17] M. A. Ferrag, L. Maglaras, A. Derhab, and H. Janicke, "Authentication schemes for smart mobile devices: Threat models, countermeasures, and open research issues," *Telecommunication Systems*, vol. 73, no. 2, pp. 317–348, 2020.

[18] M. Ghasemisharif, A. Ramesh, S. Checkoway, C. Kanich, and J. Polakis, "O single Sign-Off, where art thou? an empirical analysis of single Sign-On account hijacking and session management on the web," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1475–1492. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/ghasemisharif

[19] X. Han, Y. Zhang, X. Zhang, Z. Chen, M. Wang, Y. Zhang, S. Ma, Y. Yu, E. Bertino, and J. Li, "Medusa attack: Exploring security hazards of {In-App}{QR} code scanning," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4607–4624.

[20] F. He, Y. Jia, J. Zhao, Y. Fang, J. Wang, M. Feng, P. Liu, and Y. Zhang, "Maginot line: Assessing a new cross-app threat to pii-as-factor authentication in chinese mobile apps–ndss symposium," in *NDDS Symposium*, 2024.

[21] W. Jin, X. Ji, R. He, Z. Zhuang, W. Xu, and Y. Tian, "Sms goes nuclear: Fortifying sms-based mfa in online account ecosystem," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2021, pp. 7–14.

[22] A. Kharraz, E. Kirda, W. Robertson, D. Balzarotti, and A. Francillon, "Optical delusions: A study of malicious qr codes in the wild," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 192–203.

[23] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Journal of Information Security and applications*, vol. 22, pp. 113–122, 2015.

[24] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, ser. NDSS 2019, Feb. 2019.

[25] K. Lee, B. Kaiser, J. Mayer, and A. Narayanan, "An empirical study of wireless carrier authentication for {SIM} swaps," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, 2020, pp. 61–79.

[26] Z. Lei, Y. Nan, Y. Fratantonio, and A. Bianchi, "On the insecurity of sms one-time password messages against local attackers in modern mobile devices," in *Network and Distributed Systems Security (NDSS) Symposium 2021*, 2021.

[27] Y. Li, Y.-C. Chen, X. Ji, H. Pan, L. Yang, G. Xue, and J. Yu, "Screenid: Enhancing qrcode security by fingerprinting screens," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[28] Z. Li, W. He, D. Akhawe, and D. Song, "The {Emperor's} new password manager: Security analysis of web-based password managers," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 465–479.

[29] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 113–131.

[30] S. G. Lyastani, M. Schilling, S. Fahl, M. Backes, and S. Bugiel, "Better managed than memorized? studying the impact of managers on password strength and reuse," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 203–220.

[31] S. Ma, J. Li, H. Kim, E. Bertino, S. Nepal, D. Ostry, and C. Sun, "Fine with "1234"? an analysis of sms one-time password randomness in android apps," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1671–1682.

[32] P. Markert, D. V. Bailey, M. Golla, M. Dürmuth, and A. J. Aviv, "This pin can be easily guessed: Analyzing the security of smartphone unlock pins," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 286–303.

[33] V. Mavroeidis and M. Nicho, "Quick response code secure: a cryptographically secure anti-phishing tool for qr code attacks," in *Computer Network Security: 7th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2017, Warsaw, Poland, August 28-30, 2017, Proceedings 7*. Springer, 2017, pp. 313–324.

[34] Microsoft, "Microsoft playwright python," https://github.com/microsoft/playwright-python, 2024.

[35] S. Oesch and S. Ruoti, "That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers," in *Proceedings of the 29th USENIX Conference on Security Symposium*, 2020, pp. 2165–2182.

[36] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 417–434.

[37] H. Pan, Y.-C. Chen, L. Yang, G. Xue, C.-W. You, and X. Ji, "mqrcode: Secure qr code using nonlinearity of spatial frequency in light," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–18.

[38] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's go in for a closer look: Observing passwords in their natural habitat," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 295–310.

[39] Pypi.org, "googletrans 3.0.0," https://pypi.org/project/googletrans/, 2024.

[40] G. Smith, T. Yadav, J. Dutson, S. Ruoti, and K. Seamons, "" if i could do this, i feel anyone {could:}" the design and evaluation of a secondary authentication factor manager," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 499–515.

[41] M. Squarcina, P. Adão, L. Veronese, and M. Maffei, "Cookie crumbles: breaking and fixing web session integrity," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5539–5556.

[42] I. Stylios, S. Kokolakis, O. Thanou, and S. Chatzis, "Behavioral biometrics & continuous user authentication on mobile devices: A survey," *Information Fusion*, vol. 66, pp. 76–99, 2021.

[43] C. Wang, Y. Wang, Y. Chen, H. Liu, and J. Liu, "User authentication on mobile devices: Approaches, threats and trends," *Computer Networks*, vol. 170, p. 107118, 2020.

[44] X. Zhang, H. Ye, Z. Huang, X. Ye, Y. Cao, Y. Zhang, and M. Yang, "Understanding the (in) security of cross-side face verification systems in mobile apps: a system perspective," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 934–950.

[45] Y. Zhang, C. Xu, H. Li, K. Yang, N. Cheng, and X. Shen, "Protect: Efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2297–2312, 2020.

A More Details of the QRLogin Website Collection Process

Our approach to detect whether the website has QRLogin contains four major steps:

1) Identifying the login page. Given a domain name, we first load it into a web browser as the *landing page*. Then starting from the landing page, we search for the semantics of "login" in the HTML elements and try to navigate to the login page. We refer to a prior work [9], which trains an SVM binary classifier to evaluate the login forms within the HTML content, to confirm the login page. Also, we enhance this work by supporting websites that may use other elements including input elements than login forms on their login pages.

2) Detecting QR code semantics. After determining the login page, we need to determine whether the page contains features related to QRLogin. For this purpose, we employ a method of keyword-based heuristics. Specifically, we begin by standardizing the language of the texts in the HTML. Then, we ascertain whether the page texts include QRLogin-related keywords, including "QR code", "QR login", "scan the code", and their grammatical variants. We notice that some websites may provide a QR code for users to download resources, so we exclude those with the semantics of resource downloading using QR codes.

3) Finding the QRLogin app. This step is to associate the detected QRLogin feature with its corresponding mobile app. This is achieved by analyzing prompts or links provided on the login page that guide users on where and how to download or use the appropriate scanning app, for example, "Using the provided app to scan the QR code". We notice that not all websites provide such hints, thus we also search the mobile app on app markets using the domain name of the websites as a supplement. Then manual efforts are required to further confirm the correctness.

4) Manual confirmation. Finally, two researchers are asked to independently verify the collected QRLogin websites. They cross-check the login pages to confirm the presence and functionality of QR codes specifically used for login. This process involves not only visiting and reviewing the page content but also testing the QRLogin process to ensure that it links to the associated mobile app and functions as expected. The researchers then compare their findings and resolve any discrepancies to finalize the list of valid QRLogin deployments, ensuring the accuracy and reliability of the collected data.

Implementation details. We implement the crawler based on *Playwright browser automation* tool [34] to drive an instance of Chromium Web browser. To ensure the full loading of each web page, we wait until the network activity of the page has reached an idle state, with a timeout limit of 10 seconds, based on our observation that most websites can load one page within 10 seconds. Besides, we use the *googletrans* Python module [39] to translate the language of websites from various countries to English to ease semantic matching.

B User Perception Analysis

B.1 Design of the User Study

In this user study to investigate the users' perceptions on QRLogin security, we aim to answer the following research questions:

- RQ1: How do users engage with QRLogin?
- RQ2: What is the likelihood of QR code leakage in QRLogin?
- RQ3: How do users perceive the potential security risks in QRLogin?

Based on the research questions, we design a questionnaire with 13 questions. Specifically, we show an example of a real-world website's QRLogin to clarify to the participants what the QRLogin discussed in this study is to avoid any misunderstanding. Then, to ensure that the participants have indeed used QRLogin before, they are asked to provide any websites/apps they have used that offer QRLogin functionality, other than the one mentioned in the initial example. We will verify the website/app provided for validity.

After that, participants are required to answer 7 questions concerning our research questions. Lastly, the questionnaire collects demographic information from participants, including age, gender, education, and occupation, ensuring the anonymity and privacy of participants. The complete questions in the questionnaire and the answers collected are listed in §B.3.

Recruitment and Demographics. We recruited 200 participants via Wenjuanxing [6] (one of the most popular crowdsourcing platforms in China) in March 2024. The responses that failed in our attention test question were rejected to ensure the quality of the data collected for analysis. On average, our questionnaire took 2.3 minutes according to the responses, and participants who completed the survey received 2 CNY based on the local income.

In our study, we did not place any additional restrictions on participants to avoid bias, other than having used QRLogin. According to the demographic information collected in the study, 47.8% participants are from 18 to 30 years old while 45.6% are from 31 to 45 years old. Female participants account for 63.3%. Besides, the majority (83.9%) of participants have a college degree or above, and 52.8% have a professional background of STEM (Science, Technology, Engineer, Mathematics).

B.2 User Perceptions to QRLogin

In the responses from 200 recruited participants having passed the attention test, 180 participants answered at least one valid website/app with QRLogin, while the other 20 responses are excluded as these participants may not have used QRLogin and their perceptions are not informative.

As for the overall result, QRLogin plays an important role in the user's daily life. However, QR code leakage has substantial risks, and users have insufficient security awareness regarding QRLogin. Additionally, we also asked about users' opinions on scanning others' QR codes and found that users do not perceive a significant difference in the level of danger between the two. We discuss the detailed findings as follows.

- **Finding-1: Users frequently engage in QRLogin in daily activities, and the categories of used websites are relatively sensitive.** According to the responses of Q1 and Q2, the majority of participants use QRLogin at a high frequency. Specifically, 39.44% participants utilize QRLogin every day and 43.89% use it several times per week. Additionally, we collected and analyzed the websites/apps answered in Q1 and found that their categories are mostly sensitive, such as social, financial, and governmental. This finding further highlights the importance of the security of QRLogin.
- **Finding-2: Many users face the risk of QR code leakage when performing QRLogin.** As for RQ2, we collect answers from two aspects, i.e., leaking the QR code passively and actively during QRLogin. First, the responses of Q5 show that 41.67% of participants never or occasionally take measures to prevent others from obtaining their QR code. This implies that these users are likely to be passively targeted by attackers who may steal the QR code through methods such as taking photos. On the other hand, 12.78% participants think it reasonable to provide their QR codes when asked by others (Q6), and 72.78% participants are uncertain about the reasonableness of providing their own QR codes, thinking that it may be reasonable. This attitude may expose users to social engineering attacks by attackers, such as impersonating the website's employees and requesting users to provide QR codes, resulting in leaking QR codes actively.
- **Finding-3: Users have insufficient understanding and awareness of the risks associated with QRLogin.** In the responses of Q4, participants give an average rating of 4.82 (on a scale of 1-7, with 7 indicating very dangerous). This score approaching the midpoint suggests that users lack awareness of the dangers of QR code exposure, which will cause severe consequences like account takeover illustrated in this paper. In addition, according to the responses of Q7 and Q8, only 7.22% of participants answer correctly about the information contained in the QR code for QRLogin (without selecting any other incorrect options). Only 15.00% answer correctly about the risk of leaking QR codes. Moreover, only 2 (1.11%) participants answer both questions right. The results indicate that users have a lack of understanding of QRLogin and the associated risks, even though 83.9% of the participants have a college degree or above.

B.3 Questionare for our User Study

Q1. Please provide specific examples of apps/websites that have the QR code login feature, as evidence of having used the QR code login. (Except for the provided example of 12306). If no examples are provided or the provided apps/websites do not have the QR code login feature, the response will be considered invalid and no payment will be made. [Fill in the blank]
Answer: WeChat, QQ, etc.
Q2. How often do you utilize the QR code login feature?

(A) Almost daily (71/180, **39.44%**)

(B) Several times a week (79/180, **43.89%**)

(C) Several times a month (25/180, **13.89%**)

(D) Other lower frequency (5/180, **2.78%**)

Q3. In your opinion, what is the level of risk associated with scanning QR codes provided by others in daily life?

(A) Not dangerous at all (6/180, **3.33%**)

(B) 2 (13/180, **7.22%**)

(C) 3 (14/180, **7.78%**)

(D) 4 (38/180, **21.11%**)

(E) 5 (48/180, **26.67%**)

(F) 6 (44/180, **24.44%**)

(G) Extremely dangerous (17/180, **9.44%**)

Q4. In your opinion, what is the level of risk associated with showing your own QR code to others during QR code login?

(A) Not dangerous at all (7/180, **3.89%**)

(B) 2 (9/180, **5%**)

(C) 3 (18/180, **10%**)

(D) 4 (35/180, **19.44%**)

(E) 5 (48/180, **26.67%**)

(F) 6 (33/180, **18.33%**)

(G) Extremely dangerous (30/180, **16.67%**)

Q5. Do you take measures to prevent others from obtaining your QR code (such as preventing others from taking photos of your QR code) when performing QR code login?

(A) Always (47/180, **26.11%**)

(B) Often (58/180, **32.22%**)

(C) Occasionally (58/180, **32.22%**)

(D) Never (17/180, **9.44%**)

Q6. Do you consider it reasonable if an app or a personnel from a service requests you to provide a QR code?

(A) Completely reasonable (23/180, **12.78%**)

(B) Possibly reasonable, uncertain (131/180, **72.78%**)

(C) Completely unreasonable (26/180, **14.44%**)

Q7. What kind of information do you believe is included in the QR codes used for QR code login? [Multiple-choice]

(A) The QR code is a display of system (software) information and does not include my personal information. (81/180, **45%**)

(B) Personal information such as name, ID card. (125/180, **69.44%**)

(C) Information such as username, nickname, and avatar. (149/180, **82.78%**)

(D) Other sensitive information (if any, please list) (8/180, **4.44%**)

Q8. In your opinion, what risks do you believe may arise if a stranger obtains the QR code you use for QR code login? [Multiple-choice]

(A) The risk of the phone being infected with viruses or malware by others. (116/180, **64.44%**)

(B) The risk of the geographical location of the phone being obtained by others and tracked. (125/180, **69.44%**)

(C) The risk of the account being logged in by others, leading to the leakage of relevant information within the account. (156/180, **86.67%**)

(D) There are no such risks. (6/180, **3.33%**)

Q9. Please select the option below that represents an animal. [This is a test question, providing an incorrect answer will result in no reward.]

(A) Ice cream (0/180, **0%**)

(B) Refrigerator (0/180, **0%**)

(C) Panda (180/180, **100%**)

(D) Sunflower (0/180, **0%**)

Q10. What is your age range?

(A) <18 (0/180, **0%**)

(B) 18-30 (86/180, **47.78%**)

(C) 31-45 (82/180, **45.56%**)

(D) 46-60 (11/180, **6.11%**)

(E) >61 (1/180, **0.56%**)

Q11. What is your gender?

(A) Male (66/180, **36.67%**)

(B) Female (114/180, **63.33%**)

(C) Decline to answer (0/180, **0%**)

Q12. What is the highest level of education you have completed?

(A) No high school/Some high school/High school graduate (3/180, **1.67%**)

(B) Some college – No degree (26/180, **14.44%**)

(C) Associates (2-year degree) /Bachelor (4-year degree) (133/180, **73.89%**)

(D) Graduate degree – Master, PhD, professional, medicine, etc (18/180, **10%**)

Q13. Which describes best regarding your professional background?

(A) STEM (Science, Technology, Engineer, Mathematics) (95/180, **52.78%**)

(B) Liberal arts (83/180, **14.44%**)

(C) Associates (2-year degree) /Bachelor (4-year degree) (133/180, **46.11%**)

(D) Other (2/180, **1.11%**)