Systematic Analysis of Security and Vulnerabilities in Miniapps

Yuyang Han, Xu Ji, Zhiqiang Wang
Beijing Electronic Science and Technology Institute
Beijing, China

Jianyi Zhang*
Beijing Electronic Science and Technology Institute
Beijing, China

ABSTRACT

The past few years have witnessed a boom of miniapps, as lightweight applications, miniapps are of great importance in the mobile internet sector. Consequently, the security of miniapps can directly impact compromising the integrity of sensitive data, posing a potential threat to user privacy. However, after a thorough review of the various research efforts in miniapp security, we found that their actions in researching the safety of miniapp web interfaces are limited. This paper proposes a triad threat model focusing on users, servers and attackers to mitigate the security risk of miniapps. By following the principle of least privilege and the direction of permission consistency, we design a novel analysis framework for the security risk assessment of miniapps by this model. Then, we analyzed the correlation between the security risk assessment and the threat model associated with the miniapp. This analysis led to identifying potential scopes and categorisations with security risks. In the case study, we identify nine major categories of vulnerability issues, such as SQL injection, logical vulnerabilities and cross-site scripting. We also assessed a total of 50,628 security risk hazards and provided specific examples.

CCS CONCEPTS

• **Security and privacy → Web application security**.

KEYWORDS

Miniapps, Security Risk, Vulnerabilities, Least Privilege

ACM Reference Format:
Yuyang Han, Xu Ji, Zhiqiang Wang and Jianyi Zhang. 2023. Systematic Analysis of Security and Vulnerabilities in Miniapps. In *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Superapps (SaTS '23), November 26, 2023, Copenhagen, Denmark*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3605762.3624432

1 INTRODUCTION

According to the Internet Society of China, the miniapp has emerged as a new genre of application that has attracted widespread attention in recent years[21]. As of 2019, the total number of miniapp users in China has reached an impressive 1.4 billion, with 780 million new users and 960 million active users. A miniapp is a sub-application that runs within a mobile application, serving as a lightweight and agile solution; that provides users with fast and convenient functions and services. This type of application has gained remarkable traction on various social networking platforms, especially on WeChat, one of the most widely used communication applications with 1.3 billion monthly active users[11]. Although miniapps are essentially web-based, they provide a seamless and native app-like user experience, allowing users to access various services without installing separate applications. Due to their efficiency and ease of use, these miniapps have gained significant popularity within the mobile Internet ecosystem and have attracted considerable attention from users and businesses.

However, as miniapps continue to flourish, concerns about their security have escalated. As with any web application, ensuring security and protecting user data constitute pivotal aspects that demand careful consideration and proactive measures. This concern has led the official WeChat miniapps to introduce a set of security guidelines[19]. Nevertheless, during their widespread adoption, miniapps have also revealed vulnerabilities and security issues, attracting the attention of hackers and attackers. Consequently, security has emerged as a pressing issue that needs to be addressed within the miniapps ecosystem. According to the Aladdin Research Institute, the major platforms exert guidance and constraints on miniapps safety through two different models. Firstly, the product model, embodied by WeChat and Alipay, employs an official miniapp security testing service to thwart potential threats proactively. Secondly, the rating model, exemplified by Tiktok, Kuaishou, and Baidu, uses scoring to intervene in security events. In addition, third-party markets offer many security products and services to address and prevent various problems[5].

Meanwhile, security experts have delved into various aspects of miniapps in recent years to protect user privacy and data integrity and to facilitate a thriving miniapp ecosystem. Some have performed backward decompilation of miniapps, while others have examined the differences between miniapps in Android and iOS clients or explored the variances across different platforms. Yet, it is noteworthy that research explicitly targeting the security of miniapp web interfaces and their associated vulnerabilities still needs to be explored.

In this paper, we focus on a comprehensive analysis of the security challenges posed by the web interfaces of miniapps. To this end, we propose a powerful triad threat model for holistically assessing and mitigating security threats. The model strictly adheres to the principle of least privilege[7] and the direction of privilege consistency. We also propose a new analytical framework for analyzing and evaluating miniapps. This model can identify, diagnose, and mitigate the security risks inherent in miniapps, strengthening the overall security defences and fostering a more secure and trustworthy miniapp ecosystem. Our research has systematically focused on the web interface security of miniapps, particularly emphasising

*Corresponding authors

SaTS '23, November 26, 2023, Copenhagen, Denmark.

ACM ISBN 979-8-4007-0258-7/23/11…$15.00
https://doi.org/10.1145/3605762.3624432

further investigation and exploration of vulnerabilities in miniapps. During the experiment, we identified nine categories of vulnerability issues and 50,628 security risks and threats. We provided examples for each vulnerability issue and recorded an assessment of the security risks.

Contributions. We make the following contributions:

- We presented a new triad threat model, aiming to gain a more comprehensive insight into the security intricacies of miniapps. Considering the viewpoints of users, the server side, and potential attackers, and under the guidance of least privilege and privilege consistency principles, this model evaluates the security threats and vulnerabilities encountered by miniapps.
- We have constructed a new analysis framework and applied it to traditional vulnerability detection. For the first time, this framework utilizes a unique combination of vulnerability detection, involving AppScan+Proxifier+WeChat(Windows), to detect nearly a thousand well-known mainstream miniapps. This accomplishment represents an essential milestone in miniapp security assessment.
- We conducted a comprehensive evaluation of the novel framework on real-world WeChat miniapps. We meticulously analyzed the detection results, which resulted in identifying nine distinct categories of significant vulnerability issues, along with discovering a total of 50,628 security risks and dangers.

2 RELATED WORK

Recently, the novel concept of miniapps has been delved into across various domains. Regarding the miniapps, they can be used on education[1, 8, 15], online shopping[14], healthcare[18, 25], campus services[6],transportation[2], food service[13]. For instance, Zhang et al.[23] devised mini-crawlers to procure miniapps and conducted extensive measurements. Similarly, Zhang et al.[22] explored the realm of identity obfuscation within webview-based super apps, while Liu et al.[9] fashioned a dynamic analysis framework tailored for WeChat miniapps. Yang et al.[20] also investigated the vulnerability of cross-miniapp redirection among miniapps in WeChat and Baidu. Additionally, Lu et al.[10] thoroughly examined the miniapp paradigm, with a specific focus on access control mechanisms. Recently, Wang et al.[17] delved into the challenge of concealing APIs in mobile super apps. Meanwhile, Zhang et al.[24] began an inquiry regarding AppSecret leakage in miniapps. Furthermore, Wang et al.[16] introduced an innovative approach named TAINTMINI for detecting sensitive data flow in miniapps.

OWASP (Open Web Application Security Project)[12] and WASC (Web Application Security Consortium)[4] are esteemed organizations devoted to fortifying the security of Web applications. They assume a pivotal role in the assessment of web security. In this research study, we adeptly integrate the identification principles of OWASP Top 10 and the security guidelines provided by WASC, delving into exploring vulnerabilities and security evaluation in miniapp security. Our goal is to assess the security of miniapps systematically, precisely pinpoint potential vulnerabilities, and lay a foundation for further in-depth analysis.

3 MOTIVATION AND PROBLEM STATEMENT

During the thriving expansion of miniapps embedded within mobile applications, an increasing number of users and enterprises are harnessing these diminutive wonders to enhance their mobile endeavors. Nonetheless, miniapps face a substantial spectrum of security threats and vulnerabilities. Previous research has primarily focused on isolated security issues, resulting in the need for a holistic perspective and revealing gaps in comprehensive analysis concerning the harmonious convergence of users, servers, and potential attackers.

3.1 Threat Model

We present a comprehensive triad model for improving the security of miniapp web interfaces and proactively addressing vulnerabilities. This model considers the viewpoints of users, servers, and potential attackers, with the primary goal of providing miniapps with robust resilience to mitigate security threats effectively. The model integrates proactive measures to prevent vulnerabilities with passive strategies to mitigate potential attacks. The methodology used in this model is characterized by its systematic and meticulous nature. It is an enlightening guide for developers, enabling them to assess, improve and maintain miniapps' security. The triad model can be meticulously designed to strengthen the security of web interfaces and address the vulnerabilities inherent in miniapps. The model has three primary dimensions:

- `Users`: Representing the clientele of miniapps, users are susceptible to security threats and risks, while their actions can also impact the miniapps overall security posture.
- `Servers`: Serving as pivotal communication hubs between miniapps and backend servers, the security of these nodes directly governs the safeguarding of user data and system dependability.
- `Attackers`: Potentially posing security threats, attackers may employ diverse tactics to assail miniapps, pilfer user data, disrupt servers, tamper with data, and more.

At the same time, the notion of the "system side" is introduced. Within the triad model, the system side encompasses the system layer, which includes the operating system, the host system, and the miniapp system itself. The dynamic interaction between the security aspect (including the three components of the triad model) and the system side, together with the intricate interplay between the system side, the user side, the server side, and the attacker side, collectively create the comprehensive security ecosystem of the miniapp system.

3.2 Key Observations

As shown in Figure 1, we present three potential attack vectors to which the triad model responds.

An attacker directly targets the user: The attacker endeavors to strike directly at the user, aiming to exploit the user's vulnerabilities or missteps to compromise the user's device or extract sensitive information. This may involve employing phishing techniques, deploying malicious links, introducing malware, and more.

The attacker attacks the server side through the user: The attacker adopts an indirect approach to target the server side by manipulating user behavior. By inducing a user to visit a malicious

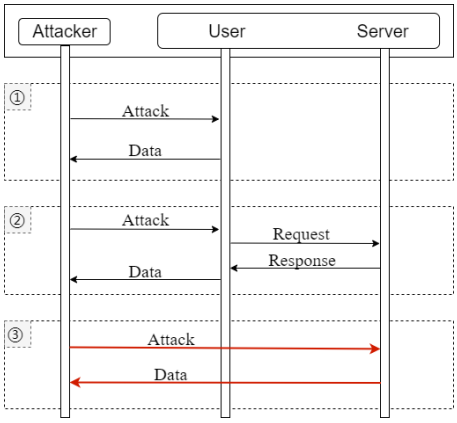

Figure 1: There are three ways of potential attacks.

website or download a file containing malevolent code, the attacker manages to infiltrate the server side with the malicious payload.

The attacker directly attacks the server side: The attacker launches a direct assault on the server side, striving to take control of it or tamper with server side data. This type of attack may encompass SQL injection[3], remote code execution, denial-of-service attacks, and other malicious activities.

Taken together, the triad model provides an integrated security framework that requires attackers to consider the principles of least privilege and privilege consistency, whether they are attacking users directly or exploiting user behavior to target the server side. This approach helps to improve the security of systems and prevent the creation and exploitation of vulnerabilities. In addition, unified attack and defense enable security organizations to gain a deeper understanding of attacker behavior and motivations, enabling more effective mitigation of potential threats.

Meanwhile, each facet of the triad framework assumes a pivotal role in distinct stages of the system side:

- `User-centric security`: User awareness and authentication mechanisms are crucial in empowering users to safeguard their data and privacy during interactions with miniapps and host systems.
- `Server fortification`: Host system providers must ensure the secure development and deployment of their applications, including miniapps, to protect user data and maintain the integrity of the system.
- `Attacker mitigation`: Preventing potential attackers is crucial for the overall system security, safeguarding miniapps and other applications from potential threats.

3.3 Problem Statement and Scope

The triad model exhibits the following exquisite characteristics:

- `Comprehensiveness`: The triad model skilfully encompasses the roles of users, servers, and attackers, converging with the system side to provide a comprehensive security research framework. This methodology facilitates the conduct of multifaceted investigations into the security of miniapps, thereby giving deep insights into the intricate security aspects of the system.
- `Systemic`: Unlike a narrow focus on specific security vulnerabilities, the triad model takes a holistic approach to addressing security concerns throughout the miniapp lifecycle. It proposes more effective security solutions by addressing the root causes of security problems.
- `Extensibility`: Seamlessly integrated with the system side, the triad model offers remarkable flexibility and scalability. This versatility allows it to be applied to different types and sizes of miniapps while allowing for adaptations and enhancements to address evolving technologies and threats.
- `Responsive to Diversity`: The variety and complexity of security issues within miniapps makes traditional one-dimensional research methods inadequate. The introduction of the triad model facilitates a comprehensive exploration of miniapp security from multiple dimensions and perspectives, leading to accurate and effective solutions tailored to various scenarios.

In prior work related to miniapps, the security issues inherent in these applications can be comprehensively categorized from the perspectives of both attackers and victims.

- **`Attacker's perspective`**: From the perspective of potential attackers, their intentions revolve around obtaining sensitive user information, stealing data, disrupting the regular operation of systems, or spreading malicious code. They use methods such as XSS, CSRF, and SQL injection to target the web interfaces of miniapps. This method involves manipulating request parameters, forging requests, and using other mechanisms to carry out their attacks. In addition, they may use techniques such as fuzz testing and penetration testing to uncover vulnerabilities in miniapps, including instances such as unauthorized access and parameter injection, to determine their attack vectors.
- **`Victim's perspective`**: Users could face various risks, ranging from the potential exposure of personal privacy to the risk of account compromise. When using miniapps, users should increase their security vigilance, avoid malicious links and exercise caution when authorizing permissions. Conversely, if servers are compromised, they could face problems such as exposure to sensitive data and system downtime. Therefore, it is vital to strengthen security measures and examine and filter incoming data to prevent attacks.

The triad model provides a new perspective and approach to exploring and solving security challenges within miniapps. Within this model, a holistic approach to security is essential, encompassing both offensive and defensive perspectives to protect user information. Following the principle of least privilege, granting users only the rights they need can reduce the risk of breaches and minimize unauthorized operations and data access. At the same time, maintaining privilege consistency ensures consistent user authentication and validation across the miniapp system, preventing vulnerabilities due to inconsistent auditing privileges.

In addition, the security of miniapps web interfaces requires significant consideration, considering offensive and defensive strategies. Our primary goal goes beyond simply thwarting the malicious

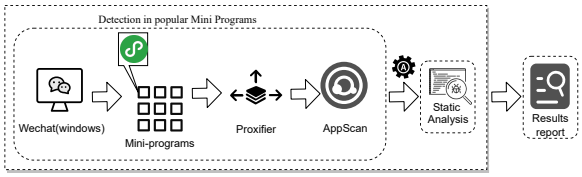

Figure 2: Overview of Analyze Framework

efforts of potential attackers; it includes proactively identifying and remediating security vulnerabilities. In addition, we recognize that implementing robust interface authentication and authorization mechanisms can effectively prevent malicious users and attackers from directly accessing sensitive data or performing dangerous operations via interfaces.

To ensure the security of user information, developers must adopt secure coding practices and perform comprehensive code audits during the development phase. This proactive approach aims to identify potential logical vulnerabilities, control discrepancies, instances of SQL injection, and cross-site scripting attacks. At the same time, a thorough security assessment is essential when integrating third-party technologies to prevent the introduction of latent security risks.

Using the triad model in conjunction with a holistic strategy that includes offensive and defensive measures, the principles of least privilege, and privilege consistency, a comprehensive improvement in the overarching security of miniapp web interfaces is achieved. This effort not only preserves user privacy and data integrity but also strengthens miniapps ability to withstand adversarial intrusions.

4 DESIGN

This section provides the detailed design of the analysis framework, and applies it to traditional vulnerability detection. As described in 3.2, security risks in miniapps occur across three components (i.e., users, servers and Attackers) at various granularity (i.e., event handlers, miniapp pages, miniapp programs) in different manners. To address these challenges, the framework uses a novel combination of vulnerability detection with the pairing of AppScan+Proxifier+WeChat(Windows) to detect mainstream miniapps. The triad threat model guides the design and deployment of detection frameworks. By providing a comprehensive understanding of potential threats and vulnerabilities from the user, server and attacker perspectives, the triad model can increase the effectiveness and comprehensiveness of the detection framework. In particular, for the first time, we employed such a combination of detection methods. We used the Windows version of the WeChat client to access the selection miniapp and subsequently directed the data traffic to AppScan through a Proxifier proxy for the purpose of security risk detection. The structure of this frame is shown in Figure 2:

Scope. In this study, we focus on systematically studying the security of miniapps to understand their root causes, and consequences. Given the security concerns around web interfaces, our goal is to identify vulnerabilities within miniapps that arise from web interface complications. Given WeChat's prominent position as a primary social media platform, coupled with the rapid expansion of its miniapp ecosystem, this research assumes notable importance. Therefore, in this paper, we focus on WeChat miniapp in particular due to its popularity and support for both sensitive data access.

Methodology. As part of the use of the new framework, we are jointly deploying the following applications in the first instance:

- `WeChat platform for Windows`: In the previous research focused on miniapps, the primary research platform was mainly focused on the Android ecosystem. The primary research methodology included reverse engineering and decompilation techniques.
- `Proxifier`: This widely used proxy tool allows applications to connect to the Internet through intermediary servers, facilitating the mediation and management of network traffic.
- `Appscan`: A robust application security scanning tool, Appscan helps to analyze traffic data relayed through proxies. It identifies critical security risks, performs categorization assessments, and assists with remediation. Appscan takes on the task of dissecting the traffic that passes through the Proxifier proxy and provides a comprehensive evaluation of the underlying security concerns.
- `Burp/Fiddler`: While capturing packets for WeChat miniapps, these tools can intercept network requests sent and received by the miniapps. They allow systematic analysis of the information encapsulated in these requests.
- `Airtest`: It, a cross-platform mobile automation testing tool, is used to automate tests and evaluate the performance of mobile applications. It is well suited for automating tests and evaluating performance aspects of mobile applications.

Airtest makes it easy to simulate user actions and data retrieval within WeChat miniapps. When testing WeChat miniapps: Airtest helps us to efficiently simulate user interactions and retrieve data from within WeChat miniapps. When trying WeChat miniapps, Airtest can be used to simulate user actions and, by creating test scripts, locate and retrieve items within the miniapp. While simulating user actions, we set up a Proxifier proxy to route the traffic data of the WeChat miniapp through Proxifier to AppScan. Using AppScan, we analyze the traffic data passed through the Proxifier proxy. Next, the test application generates reports and automatically compiles security risks from the documentation. Finally, we perform a static analysis of various security risks to consolidate the data further.

AppScan meticulously examines the miniapp's source code and binary files during this process, particularly during static analysis. It identifies a range of vulnerabilities, including but not limited to SQL injection and cross-site scripting attacks. It also performs a comprehensive dependency assessment, including third-party libraries and components, to identify potential vulnerabilities or security issues. At the same time, AppScan can trace the intricate paths of data flow within miniapps, facilitating early detection of potential data leakage and identifying insecure data processing techniques.

In prior research, BurpSuite, Android WeChat, Emulator, and Fiddler were amalgamated to perform detection tasks. As the WeChat version has been updated, the effectiveness of the former detection method for miniapp on the Android platform and within simulators has gradually decreased. Therefore, we introduce a novel approach, employing a distinct pairing scheme involving Windows WeChat,

Proxifier, AppScan, and BurpSuite. Through this setup, we analyze the data traffic information of Windows WeChat miniapps using AppScan, detecting potential vulnerabilities. The results are subsequently cross-verified and validated through BurpSuite and other indispensable tools during the testing phase.

5 EVALUATION

This section focuses on examining web interface security vulnerabilities within miniapps. These vulnerabilities are methodically grouped into distinct focus areas, covering both the frontend and backend. Among the many vulnerabilities identified are logical inconsistencies, permission management anomalies, SQL injection susceptibility, cross-site scripting infiltration, arbitrary file uploads, compromised password security, and other intricate security nuances. Each classification is accompanied by specific examples that meticulously illustrate the authentic security challenges faced by miniapps.

5.1 Quantifying the Vulnerabilities Risks

Within this section, these vulnerabilities are systematically categorised into frontend, backend and other relevant dimensions and presented sequentially for comprehensive analysis.

5.1.1 Frontend vulnerabilities. **`Logic Vulnerabilities`**: This vulnerability refers to the presence of defects or errors in the logical flow of an application, resulting in behavior that deviates from expected results or presents potential security risks. Typically, these vulnerabilities do not involve syntax errors within the code, but are related to complications in the program's business logic or permission controls. Because of their ability to bypass conventional security measures and exploit input validation errors to remain undetected, they manipulate the standard logical flow of a miniapp. As a result, they can cause insecure permissions, access control issues, unauthorised data retrieval and business logic errors, among other problems.

`Cross-Site Scripting (XSS)`: This vulnerability attack involves surreptitiously inserting malicious script code into the input fields of miniapp, forcing it to execute within the confines of the user's web browser. It allows malicious actors to perform harmful actions during a user's session, including stealing login credentials, manipulating web page content, and redirecting to malicious online domains.

5.1.2 Backend vulnerabilities. **`Privilege Escalation`**: Inadvertent misconfigurations of permissions can give unauthorised entities access to resources that should be restricted. More dangerously, these misconfigurations could allow such entities to manipulate critical system components. Specific fields such as "id", "uid", and "UserName", in conjunction with their associated counterparts, require scrutiny. Because they can be used as pathways for parameter traversal, putting sensitive information at significant risk.

`SQL Injection`: Insufficient user input validation can lead to malicious SQL query injection to access and modify database data. The SQL injection vulnerability in miniapps means that user input is not fully validated and filtered in the backend code of the miniapp. As a result, an attacker can construct malicious inputs and execute malicious SQL statements to gain unauthorized access to the database or to modify, delete or leak data from the database.

`Arbitrary file uploads`: Owing to the absence of robust file type validation, malicious actors gain the capability to upload pernicious files that pose a grave threat to the security of the system.

`Weak password security`: Within the miniapp, the vulnerability of weak password security resides in the login authentication or backend management system. Should users choose passwords that are overly simplistic, readily guessable, or commonly employed, the security of their accounts diminishes, rendering them susceptible to password guessing or brute-force cracking attacks. Furthermore, weak passwords may also engender password-guessing attacks, brute-force cracking attempts, and instances of multi-account hijacking.

5.1.3 Other Security Issues. **`Leakage of sensitive information`**: Inadequate management of sensitive information within miniapps during their design, development or operational phases can create scenarios where malicious entities can inappropriately access or disclose user-sensitive data. Furthermore, in the miniapps, improper handling of user input data within URLs can also expose sensitive information or facilitate unauthorised access to resources.

`Cross-Site Request Forgery`: This vulnerability presents a scenario where malicious actors impersonate authentic users and trick them into unknowingly executing malicious requests through their web browsers. This manipulation results in the initiation of unauthorised actions within the miniapp. By taking advantage of the user's authenticated session, this vulnerability exploits the limited ability of miniapps to thoroughly validate the source of incoming requests, allowing malicious operations to be covertly executed.

`Integration of Third-party Technologies`: During the development phase of miniapps, the inclusion of third-party technology components, either developed by external vendors or by fellow developers (such as plug-ins, libraries, APIs, etc.), can create security vulnerabilities and potential threats. These include security breaches, data leakage and the propagation of malicious code.

5.2 Case Studies

5.2.1 Frontend vulnerabilities. **Logic Vulnerabilities:** Throughout our experiments, we discovered that a substantial proportion of the vulnerabilities inherent in the miniapp were attributed to logic flaws. Among them, we encountered a category known as "authentication type logic vulnerabilities." These vulnerabilities encompassed a range of security concerns, including login bypass (enabling arbitrary user access), password recovery vulnerabilities, CAPTCHA circumvention, exploitation of mobile phone CAPTCHAs, SMS-based attacks (commonly referred to as SMS bombs), and the masking of mobile phone numbers during registration.

During the experiment, we encountered several noteworthy logical anomalies, which warrant careful attention:

- In our examination of the "Take goods mall" miniapp, we observed that after the login authorization, the miniapp omitted the secondary verification of the user's mobile phone number to ensure data protection. This oversight resulted in direct access to the miniapp without verifying the mobile phone number.

- In our evaluation of the miniapp for the "Haidian Foreign Language Students Service Platform," we input "123," followed by a single quote during the login phase and provided a random password. By capturing the package using tools like BurpSuite, we noticed incorrect account information was returned, yet we could still directly access the system.
- During our assessment of the "Yixianjia Jiadele Life Supermarket Online Store" miniapp, we discovered that entering the verification code "111111" during the registration phase resulted in immediate successful registration without stringent validation of the bound mobile phone number.
- In evaluating the "To Shoot | Milk Delivery Supermarket Recycling Service" miniapp, we observed a lack of robust validation for the verification code during the login phase, allowing unrestricted access using any four-digit code.
- While examining the "Panqinkanghua" miniapp, we attempted to replay the verification code package. We observed multiple SMS messages received on the phone quickly, indicating a vulnerability to SMS bombing.
- Furthermore, while testing the "Yonghui Enterprise Purchase" miniapp, we encountered a four-digit verification code with a five-minute validity period. And upon investigation, we found that the verification code could be traversed to obtain the correct sequence.

Cross-Site Scripting (XSS): In the miniapps "Mom.com Mom's Good Products," "Gourmet Jie Recipe Book," and "Jurong Convenience Supermarket Preferred Life Service Platform," we have strategically inserted prompt messages into the input box, comment section, search box, or any other text fields that users can interact with on the webpage. Our approach strictly adheres to benevolent experimental testing and forbids using malevolent XSS-constructed statements. The attack is deemed valid once the prompt messages can be successfully executed. In the trio mentioned above of miniapps, the prompt statement was entered into the search field, and subsequently, "200 OK" was obtained in BurpSuite.

5.2.2 Backend vulnerabilities. **`Privilege Escalation`**: An authentication vulnerability through parameters in a cookie can be exploited by manipulating the UserId value to gain unauthorized access to user information. When the UserId value is altered to a different value, the system divulges various essential details about the employee, including their employee number, name, mobile phone number, and work status (either "in" or "out" of work). For instance, in the case of Shangbiao Brand Supermarket, by employing Burp Suite to modify the id-data, the corresponding information for different ids can be accessed, thereby revealing mobile phone numbers, order numbers, and other sensitive information.

`SQL Injection`: SQL Injection is a prevalent security vulnerability observed in miniapps. Its occurrence can largely be attributed to the inadequate handling of user input data in the backend code, which results in the direct concatenation of user input into SQL query statements. As a consequence, attackers can manipulate the execution logic of the SQL statement by modifying the input, thus gaining unauthorized access to sensitive information and undermining the integrity of the database.

- During the examination of the "Jiale Source Fresh Supermarket Corner Store" miniapp, we encountered an SQL injection vulnerability. By skillfully constructing and concatenating SQL query statements in the keyword section, we could access sensitive information from the system, including database details.
- In the context of the "Enterprise Management Cloud" miniapp, we identified an SQL injection vulnerability in the login section. By injecting manipulated SQL statements into the account section, we were able to trigger error messages or, at times, inadvertently disclose sensitive information in pop-up windows.
- Similarly, the "Gourmet Jie Recipe Collection" miniapp displayed SQL injection vulnerabilities in our examinations. We confirmed this by inputting specific values into the id section of the SQL query statement.
- Lastly, multiple SQL injections were detected during testing in the "Water Sage Technology Campus Direct Drinking Water" miniapp. For instance, manipulating the uid section allowed access to sensitive information. Should this miniapp, named "Campus to provide direct drinking water," suffer from information leakage, it could lead to severe consequences.

The Figure 3 shows how an attacker exploits a SQL injection vulnerability to gain unauthorised access to sensitive information.

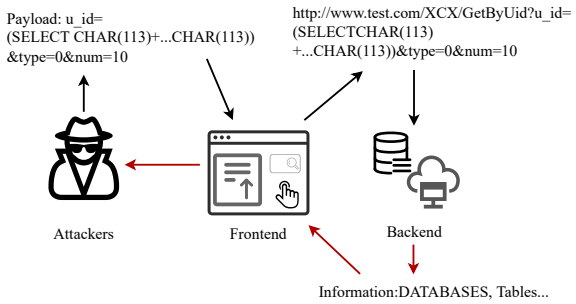

Figure 3: An example of the SQL injection process

`Arbitrary file uploads`: The absence of file type validation permits malevolent files to be uploaded, thus posing a grave threat to system security. Primarily, a burp manipulation of the file extension can exploit any location where an image is uploaded. Secondly, in cases where an ID card is uploaded, if it is done through a camera, the vulnerability can be circumvented by modifying the state in the return packet in conjunction with process validation. For instance, a print miniapp can evade validation and upload an image by altering the file extension via BurpSuite.

`Weak password security`: During the testing process, we shall come across passwords such as "123456," "111111," "888888," "abc123," and other feeble, simplistic account credentials. This situation may arise due to the miniapps' convenient utilization by employees during the testing phase; however, it inevitably gives rise to security vulnerabilities.

5.2.3 Other Security Issues. **`Leakage of sensitive information`**: In the decompiled source code of miniapps, user information is susceptible to leakage. Even in some miniapps where source code maintenance is infrequent or has ceased, highly sensitive information, such as uid and key, might be exposed. Furthermore, in

certain registration and login interfaces, if incorrect or intentional SQL statements are input consecutively, a prompt box will provide feedback based on the input, allowing for the extraction of sensitive information through subsequent statement crafting. During experimentation, we encountered a miniapp that, despite displaying a prompt box indicating discontinued maintenance, still allowed access to numerous pages from which a wealth of private information could be gleaned through decompilation.

`Cross-Site Request Forgery`: Insufficient CSRF protection could allow an attacker to execute actions on behalf of an authenticated user surreptitiously. Once the user logs in and saves their credentials within the miniapp, the authorized information remains accessible for subsequent logins. Subsequently, the attacker can employ a malicious link to deceive the user into clicking on it. Due to the absence of an effective CSRF defense mechanism in miniapps, the attacker can exploit the user's active login session to initiate malevolent requests, all unbeknownst to the user.

`Integration of Third-party Technologies`: During the meticulous testing phase, we encountered many miniapps, totaling more than a dozen, which exhibited a common security flaw -adopting the easily-guessable verification code "11111." Astonishingly, successful registration and login were achieved simply by inputting this rudimentary code. Remarkably, these miniapps were supported by the third-party technological expertise of KM Technology. Regrettably, KM Technology's services vulnerabilities could potentially jeopardize the security and integrity of multiple miniapps affiliated with the company.

5.3 Evaluation Results

After eliminating extraneous factors, the conclusions of the scan are as Table 1. Total number of severity issues included in the report: 50,628.

In miniapps, application data encompasses cookies, JS, parameters, comments, visited URLs, failed requests, and filtered URLs. Amongst this trove of data, a myriad of security vulnerabilities lies in wait, constituting a profound menace to user data's sanctity and system resources' stability. These vulnerabilities can be systematically categorized into risk levels, ranging from high to medium to low.

- High-risk vulnerabilities pose grave repercussions, encompassing the peril of breaching the sanctity of administrative privileges, compromising databases, and executing remote commands on web servers. Moreover, they can potentially unleash debilitating denial-of-service onslaughts, lay bare delicate information like source code and user credentials, and even facilitate illicit transactions.
- Medium-risk vulnerabilities encompass the threat of session hijacking and manipulation, unauthorized access to sensitive resources, and the exposure of confidential data transmitted during encryption. In addition, attackers can subvert authentication mechanisms, infiltrate specific directories, and extract files or configurations from web servers.
- Low-risk vulnerabilities entail gathering sensitive information, luring users into divulging crucial data, and extracting server side script source code. Furthermore, these vulnerabilities may result in information disclosure, downloading of transient script files, and manipulation of client side sessions or cookies to impersonate genuine users.

As is shown in Figure 4, the presented data delineates the distribution of vulnerabilities acquired through framework-based detection, alongside the proportion of vulnerabilities that underwent our comprehensive classification and subsequent meticulous verification encompassing nine prominent vulnerability categories.

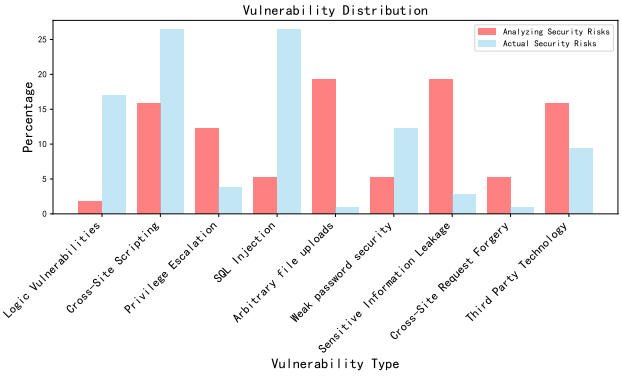

Figure 4: These are nine broad categories of vulnerabilities

The data presented in Figure 4 clearly shows that real security threats such as logical vulnerabilities, cross-site scripting and SQL injection manifest themselves at a much higher frequency than their proportional representation in the security risk analysis. This observation highlights the profound dangers of logical vulnerabilities, XSS and SQL injection to the security of miniapps. At the same time, while injection vulnerabilities dominate the landscape of security risk analysis, our study focuses primarily on the tangible instances of SQL injection vulnerabilities, resulting in their comparatively lower representation.

Conversely, the incidence of privilege escalation, sensitive information leakage and arbitrary file upload vulnerabilities in the actual security risk landscape are significantly lower than their representation in the security risk analysis. This disparity is because these facets are predominantly examined through the lens of code audits. This method is separate from our primary approach within this paper.

6 DISCUSSION

The manuscript above delves into an all-encompassing appraisal of the miniapp, focusing on the esteemed OWASP Top 10 and WESC. It diligently explores the security aspect of the miniapps web interface. It conducts thorough research on vulnerability mining, culminating in the proposition of a triad model that fosters a more systematic and comprehensive approach to future investigations. Furthermore, the paper unveils the vulnerabilities unearthed during the experimentation phase. In light of these security concerns, it is imperative to persistently enhance and fortify the following aspects of miniapp security in the times ahead:

- `Interface Security`: Ensuring the integrity of interfaces is essential to enable seamless communication between miniapps and backend services. These interfaces play a crucial role in the miniapp domain, especially in transferring and managing

Table 1: Security Vulnerabilities in miniapps

Severity Level	A Number	Comments
`High`	5,456	Serious, urgent, need to be fixed immediately.
`Moderate`	4,042	General, higher risk, timely attention.
`Low`	35,543	Slight, small risk, timely attention.
`Informational`	5,587	Recommendations, tips, can be optimized.

sensitive user data. Adopting appropriate authentication and authorisation mechanisms, such as *OAuth* or token-based authentication, is paramount in maintaining the security of interfaces. In addition, implementing rigorous input validation and data filtering is a fundamental safeguard against malicious intrusion, including threats such as SQL injection and XSS attacks.

- `Platform Security`: Platform security, on the other hand, encompasses creating a safe and sheltered ecosystem for miniapps to thrive within. The safe encompasses safeguarding the miniapp distribution platform with an unwavering commitment to data protection. It is imperative to elevate the significance of passwords that correspond to miniapps and diligently update them promptly. Additionally, developing robust user login processes and establishing periodic security reviews and vulnerability remedies serve as steadfast guardians, ensuring users remain shielded from malicious apps and data breaches.
- `Backend Security`: Safeguarding the sanctity of the miniapps backend server stands as an imperative in the realm of backend security. Within this domain, the backend assumes responsibility for storing and processing invaluable user data, hence necessitating rigorous access control and robust data encryption measures. Furthermore, the prohibition of external network access to the backend management platform remains a steadfast fortification, while the timely updating of plug-ins, APIs, and other components bolsters defense walls. Regularly tending to server side scripts and databases by applying updates and patches mitigates the risks posed by potential attacks.
- `Staff Responsibilities`: As the custodians of miniapp development and maintenance, staff have a significant responsibility in ensuring the security of these miniapps. They must undergo comprehensive security training to provide a thorough understanding of security threats and mitigation methods. Adherence to established best practices is a guiding principle in conducting meticulous security assessments that include source code, interfaces, platforms and backend systems. This vigilant approach plays a vital role in the timely identification and remediation of latent security issues, thereby safeguarding the integrity of the miniapps they manage.

A cohesive defense can be strengthened by carefully managing factors such as source code security, interface security, platform security, back-end security, and personnel responsibilities, enabling miniapps to deftly fend off potential security threats and maintain user privacy and data integrity. As we diligently address the security challenges inherent in miniapps, future research could address advanced vulnerability detection techniques, the fusion of artificial intelligence and security protocols, studies of the miniapp ecosystem, vulnerability mitigation and management strategies, security assessment, and authentication mechanisms. These pursuits will foster collaborative efforts. Through the relentless pursuit of knowledge and in-depth research, we will continuously improve the security of miniapps, ensure the protection of user information, and promote the sustainable development of the miniapp industry.

7 CONCLUSION

In this paper, we delve into the realm of miniapp security, with a central focus on discerning and mitigating diverse vulnerabilities that have the potential to jeopardize the sanctity of user data and the integrity of the system. To this end, we propose an innovative triad threat model incorporating users, servers, and potential attackers, upholding the tenets of least privilege and privilege consistency to safeguard user information. This model accentuates the significance of comprehensively considering the interplay between these three entities, thereby creatively designing a novel security analysis framework for miniapps. Furthermore, our exploration encompasses an extensive survey and illustrative examples of the manifold security risks looming in miniapps front-end and backend spheres. By diligently categorizing these issues and furnishing specific instances, we illuminate distinct facets of miniapp security and underscore the criticality of addressing these vulnerabilities. In culmination, we focus on miniapp security and chart a course for future research in this domain.

ACKNOWLEDGMENTS

Supported by the Fundamental Research Funds for Central Universities (328202204).

REFERENCES

[1] Xin Chen, Xi Zhou, Huan Li, Jinlan Li, and Hua Jiang. 2020. The value of WeChat as a source of information on the COVID-19 in China. *Preprint]. Bull World Health Organ* 30 (2020).

[2] Ao Cheng, Gang Ren, Taeho Hong, Kichan Nam, and Chulmo Koo. 2019. An exploratory analysis of travel-related WeChat mini program usage: affordance theory perspective. In *Information and Communication Technologies in Tourism 2019: Proceedings of the International Conference in Nicosia, Cyprus, January 30–February 1, 2019*. Springer, 333–343.

[3] William G Halfond, Jeremy Viegas, Alessandro Orso, et al. 2006. A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE international symposium on secure software engineering*, Vol. 1. IEEE, 13–15.

[4] hcltechsw.com. 2021. WASC Threat Classification v2.0 report. Retrieved May 25, 2025 from https://help.hcltechsw.com/appscan/Enterprise/10.0.1/topics/r_wasc_threat_classifications_report.html

[5] Aladdin Institute. 2023. Mini Program Security Construction and Development Insights. Retrieved July 28, 2023 from https://www.aldzs.com/viewpointarticle?id=16623

[6] Wu Jun, Jiajun Li, Fengning Liu, Zesen Yuan, Yu Han, and Jin Zhang. 2022. A Research on the Construction of Campus Errand Running Service Platform Based on WeChat Mini App. *World Scientific Research Journal* 8, 9 (2022), 501–507.

[7] Maxwell N Krohn, Petros Efstathopoulos, Cliff Frey, M Frans Kaashoek, Eddie Kohler, David Mazieres, Robert Tappan Morris, Michelle Osborne, Steve VanDeBogart, and David Ziegler. 2005. Make Least Privilege a Right (Not a Privilege).. In *HotOS*.

[8] Qinzhen Liang and Chengyang Chang. 2019. Construction of teaching model based on WeChat Mini-Program. *International Journal of Science* 16, 1 (2019), 54–59.

[9] Yi Liu, Jinhui Xie, Jianbo Yang, Shiyu Guo, Yuetang Deng, Shuqing Li, Yechang Wu, and Yepang Liu. 2020. Industry practice of javascript dynamic analysis on wechat mini-programs. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1189–1193.

[10] Haoran Lu, Luyi Xing, Yue Xiao, Yifan Zhang, Xiaojing Liao, XiaoFeng Wang, and Xueqiang Wang. 2020. Demystifying resource management risks in emerging mobile app-in-app ecosystems. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications Security*. 569–585.

[11] AZhongguancun Online. 2023. WeChat's 1.3 Billion Monthly Activities Steadily Sitting on the Throne of the First National APP. Retrieved July 26, 2023 from https://new.qq.com/rain/a/20230326A028AX00

[12] owasp.org. 2021. OWASP Top 10 - 2021. Retrieved May 25, 2022 from https://owasp.org/Top10/

[13] Chen-Kuo Pai, Ze-Tian Wu, Seunghwan Lee, Jaeseok Lee, and Sangguk Kang. 2022. Service Quality of Social Media-Based Self-Service Technology in the Food Service Context. *Sustainability* 14, 20 (2022). https://doi.org/10.3390/su142013483

[14] Qianhui Rao and Eunju Ko. 2021. Impulsive purchasing and luxury brand loyalty in WeChat Mini Program. *Asia Pacific Journal of Marketing and Logistics* 33, 10 (2021), 2054–2071.

[15] Yiling Sui, Tian Wang, and Xiaochun Wang. 2020. The impact of WeChat app-based education and rehabilitation program on anxiety, depression, quality of life, loss of follow-up and survival in non-small cell lung cancer patients who underwent surgical resection. *European Journal of Oncology Nursing* 45 (2020), 101707. https://doi.org/10.1016/j.ejon.2019.101707

[16] Chao Wang, Ronny Ko, Yue Zhang, Yuqing Yang, and Zhiqiang Lin. 2023. Taintmini: Detecting Flow of Sensitive Data in Mini-Programs with Static Taint Analysis. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 932–944. https://doi.org/10.1109/ICSE48619.2023.00086

[17] Chao Wang, Yue Zhang, and Zhiqiang Lin. 2023. Uncovering and Exploiting Hidden APIs in Mobile Super Apps. arXiv:2306.08134 [cs.CR]

[18] Feilong Wang, Lily Dongxia Xiao, Kaifa Wang, Min Li, and Yanni Yang. 2017. Evaluation of a WeChat-based dementia-specific training program for nurses in primary care settings: A randomized controlled trial. *Applied Nursing Research* 38 (2017), 51–59.

[19] Wechat. 2021. WeChat mini programs' Safety guidelines Development Principles and Considerations. Retrieved July 22, 2023 from https://developers.weixin.qq.com/miniprogram/dev/framework/security.html

[20] Yuqing Yang, Yue Zhang, and Zhiqiang Lin. 2022. Cross miniapp request forgery: Root causes, attacks, and vulnerability detection. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3079–3092.

[21] Xian Yun. 2023. Analysis of the development status and development trend of China's small program industry in 2023. Retrieved July 25, 2023 from https://www.sgpjbg.com/info/4a34cfc1716174fa600c82a4662c02a4.html

[22] Lei Zhang, Zhibo Zhang, Ancong Liu, Yinzhi Cao, Xiaohan Zhang, Yanjun Chen, Yuan Zhang, Guangliang Yang, and Min Yang. 2022. Identity Confusion in WebView-based Mobile App-in-app Ecosystems. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1597–1613. https://www.usenix.org/conference/usenixsecurity22/presentation/zhang-lei

[23] Yue Zhang, Bayan Turkistani, Yuqing Yang, Chaoshun Zuo, and Zhiqiang Lin. 2021. A measurement study of wechat mini-apps. *SIGMETRICS21: ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems* 5, 2, Article 14 (2021), 25 pages. https://doi.org/10.1145/3460081

[24] Yue Zhang, Yuqing Yang, and Zhiqiang Lin. 2023. Don't Leak Your Keys: Understanding, Measuring, and Exploiting the AppSecret Leaks in Mini-Programs. arXiv:2306.08151 [cs.CR]

[25] Kaina Zhou, Wen Wang, Wenqian Zhao, Lulu Li, Mengyue Zhang, Pingli Guo, Can Zhou, Minjie Li, Jinghua An, Jin Li, et al. 2020. Benefits of a WeChat-based multimodal nursing program on early rehabilitation in postoperative women with breast cancer: a clinical randomized controlled trial. *International journal of nursing studies* 106 (2020), 103565.