MiniBLE: Exploring Insecure BLE API Usages in Mini-Programs

Zidong Zhang
Simon Fraser University; Shandong University
Burnaby, Canada
zza323@sfu.ca

Jianqi Du
Shandong University
Qingdao, China
dujianqi@mail.sdu.edu.cn

Wenrui Diao*
Shandong University
Qingdao, China
diaowenrui@link.cuhk.edu.hk

Jianliang Wu*
Simon Fraser University
Burnaby, Canada
wujl@sfu.ca

Abstract

As the ecosystem of emerging mini-programs rapidly develops, an increasing number of IoT devices, particularly BLE devices, are providing accompanying mini-programs as a convenient means for end users to control these IoT devices. The associated security issues have also attracted researchers' interest recently. However, existing research on mini-program security primarily focuses on their permissions, common development bugs, and cross-mini-program communications with little attention on interactions between mini-program and IoT devices. We propose MiniBLE, a static taint analysis tool for analyzing BLE mini-programs, focusing on detecting insecure BLE pairing issues. MiniBLE was used to analyze 41,276 real-world mini-programs, demonstrating its effectiveness in identifying potential vulnerabilities. We show MiniBLE with some preliminary results and a real-world case study in this work-in-progress (WIP) paper.

CCS Concepts

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

Keywords

Mini-program Security; Program Analysis; BLE Security

ACM Reference Format:
Zidong Zhang, Jianqi Du, Wenrui Diao, and Jianliang Wu. 2024. MiniBLE: Exploring Insecure BLE API Usages in Mini-Programs. In *Proceedings of the ACM Workshop on Secure and Trustworthy Superapps (SaTS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3689941.3695774

1 Introduction

Mini-programs, which are lightweight applications within a super or host app, have become significant in mobile computing due to their convenience and functionality. The worldwide popularity of mini-programs, hosted by platforms such as WeChat [5], Baidu [3], Alipay [1], and TikTok [6], highlights the increasing concerns regarding their security and privacy.

Among those platforms, the WeChat mini-program platform, an integral feature of the WeChat ecosystem developed by Tencent, is the most popular. The mini-programs on WeChat, which are built on a JavaScript-based WebView engine and operate within the super app (i.e., WeChat), enable users to access various services, including e-commerce, gaming, and utility tools. In addition, WeChat mini-programs can possess hardware connectivity features, such as Bluetooth Low Energy (BLE) [11], Near Field Communication (NFC) [18], and WiFi [19]. Among these features, the BLE implementation in WeChat mini-programs presents notable security and privacy challenges, particularly regarding interaction protocols with IoT devices due to the universal application of BLE in IoT devices.

Existing research has primarily assessed the security of mini-programs from the perspective of mobile apps, like permissions policies [34], common development bugs [31], and cross-mini-program communications [33]. Wang et al. developed TaintMini [29] and Li et al. [25] created MiniTracker, both for tracking sensitive data flow in mini-programs, while Wang et al. also developed APIDIFF [30] was designed to detect API execution differences across platforms. However, security issues related to the BLE capabilities of mini-programs have received little attention in existing research.

In this work, we focus on the insecure BLE API usage in WeChat mini-programs, highlighting the security risks of IoT devices within this ecosystem. We designed an automated analysis framework, MiniBLE, to detect potential BLE security issues in WeChat mini-programs. MiniBLE is designed to automatically collect and filter BLE-related mini-programs and analyze BLE security issues. Thus, we conducted a preliminary large-scale analysis of 41,276 collected mini-programs. Among those mini-programs, we filtered 1,316 BLE mini-programs and found that 1,099/1,316 (83.5%) mini-programs had potential BLE security issues.

2 Background

2.1 WeChat Mini-Program Architecture

WeChat offers a versatile platform for delivering mini-programs without installation. These mini-programs enable users to access various services, including e-commerce, gaming, and utility tools. The architecture of a WeChat mini-program is bifurcated into two primary components, as delineated in Figure 1: (1) the front-end,

*Corresponding authors.

SaTS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1237-1/24/10
https://doi.org/10.1145/3689941.3695774

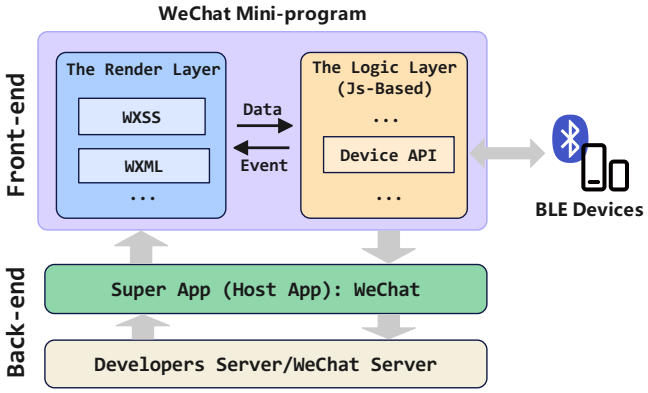

Figure 1: WeChat mini-program architecture.

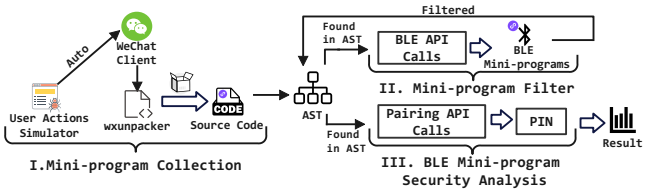

Figure 2: The Workflow of MiniBLE.

which operates within the super app, facilitating user interaction and access to system services; and (2) the back-end, which provides the runtime environment and manages server-side operations.

A WeChat mini-programs source code [4] is composed of JavaScript, WXML (WeiXin Markup Language) [8]. The front-end of the mini-program is further subdivided into the render layer and the logic layer [7]. The render layer employs WXML templates and WXSS for structuring and styling the user interface, while the logic layer utilizes JavaScript for handling application logic. These layers are orchestrated by a WebView thread for the render layer and a JavaScript thread for the logic layer. To utilize BLE to communicate with the IoT device, the mini-program invokes WeChat BLE APIs in back-end Js files, which further trigger WeChat to invoke BLE APIs of the operating system.

2.2 BLE Mini-program Application

A mini-program utilizes the interfaces encapsulated uniformly by WeChat to use the hardware's Bluetooth capabilities. Currently, WeChat mini-programs only support Bluetooth Low Energy (BLE) with host mode, peripheral mode, and BLE Beacons. The following steps must be followed to use Bluetooth capabilities in a WeChat mini-program:

- **Initialize Bluetooth Adapter**: A mini-program uses `wx.openBluetoothAdapter` [11] to enable Bluetooth first (other Bluetooth-related APIs must be called after this API is invoked).
- **Discover and Connect to Peripheral Devices**: The API `wx.startBluetoothDevicesDiscovery` [17] is used to search for nearby Bluetooth peripheral devices (AdBP). After starting the discovery process, the mini-programs can find target devices from all discovered devices using `wx.onBluetoothDeviceFound` [16]. Subsequently, the mini-program stops the discovery process with `wx.stopBluetoothDevicesDiscovery` [20] and establishes a connection to the identified BLE device using `wx.createBLEConnection` [11].
- **Interact with Peripheral Devices**: After connecting to the device, the mini-program can retrieve all services of the BLE device using `wx.getBLEDeviceServices` [13], obtain service characteristics using `wx.getBLEDeviceCharacteristics` [12], subscribe to and notify characteristics using `wx.notifyBLECharacteristicValueChange` [15], or write to characteristics using `wx.writeBLECharacteristicValue` [21].
- **Close Bluetooth Adapter**: The Bluetooth adapter can be closed by calling `wx.closeBluetoothAdapter` [10] or when the mini-program is destroyed.

Comparison with Android BLE APIs. On Android, BLE APIs of the WeChat mini-program platform are built on encapsulating the underlying operating system's BLE functionality. This encapsulation simplifies the operation of BLE devices within the mini-program, allowing developers to implement complex BLE communication through simpler mini-program APIs. However, this encapsulation restricts the flexibility of BLE operations in the mini-program. The restriction also involves certain BLE security settings. For example, Table 1 shows the BLE capabilities in the mini-program and Android apps. Compared to Android apps, these mini-program BLE APIs only support basic readable, writable, and encrypted forms of data transmission, which might cause potential MITM attacks.

2.3 Insecure BLE Pairing in Mini-programs

In the BLE connection process, pairing plays a crucial role and is the foundation for secure communication between the host and peripheral devices. In mini-programs utilizing BLE, device IDs are obtained through the Discovery process before initiating a connection using the API `wx.createBLEConnection`. The mini-program and its operating system (OS) automatically handle the subsequent pairing process. Besides using this API for connection and pairing, developers can also use `wx.makeBluetoothPair` [14]. However, the official WeChat documentation describes this API: "This interface should be used only when the developer does not want the user to enter the pin code manually, and real machine verification confirmation can be used in normal working conditions." In other words, this API allows developers to use a hard-coded PIN for the Passkey Entry pairing method. Although the API is designed to provide a convenient pairing method, it essentially functions similarly to the insecure method discovered by Sivakumaran et al. [27] in Android: developers use the `setPin` API to input hardcoded PINs, allowing users to bypass the manual PIN entry step during secure pairing, thus introducing a security downgrade risk.

In conclusion, the relatively simple implementation of BLE-related configurations and APIs in mini-programs means that the security of associated IoT devices depends on the developer's security awareness. Developers may need to employ additional security mechanisms to ensure secure interactions.

Table 1: Comparison of BLE Capabilities between WeChat Mini-Program and Android.

No.	WeChat Mini-programs	Description	Android Apps	Description
1	readable	Readable	PERMISSION_READ	Readable
2	writeable	Writable	PERMISSION_WRITE	Writable
3	readEncryptionRequired	Encryption Read Request	PERMISSION_READ_ENCRYPTED	Read/Encrypted
4	writeEncryptionRequired	Encryption Write Request	PERMISSION_WRITE_ENCRYPTED	Write/Encrypted
5	-	-	PERMISSION_READ_ENCRYPTED_MITM	Read/Encrypted with MITM protection
6	-	-	PERMISSION_WRITE_ENCRYPTED_MITM	Write/Encrypted with MITM protection
7	-	-	PERMISSION_WRITE_SIGNED	Write/Signed
8	-	-	PERMISSION_WRITE_SIGNED_MITM	Write/Signed with MITM protection

3 MiniBLE Design

We designed an automated analysis framework, MiniBLE, to detect potential BLE security issues in WeChat mini-programs. In this section, we first discuss our threat model and elaborate on the three phases of MiniBLE: mini-program collection, BLE mini-program filter, and BLE security analysis, as shown in Figure 2.

Threat Model. In this paper, we assume that the attacker possesses the following capabilities: the attacker can intercept, modify, and eavesdrop on all BLE packets transmitted over the air. The attacker can also have physical access to the BLE device and obtain the latest version of the BLE device's mini-programs source code by decompiling the mini-programs package. Lastly, when attacking, the attacker is within the range of Bluetooth.

Mini-program Collection.: The first part of MiniBLE focuses on collecting WeChat mini-programs. Crawling these mini-programs is challenging due to the lack of official or third-party app markets like Google Play for Android. Instead, users typically access mini-programs through QR codes or searches within the WeChat client.

Similar to the previous work MiniCAT [37], we developed an automated crawler that simulates user actions on the WeChat Windows client. This crawler utilizes Natural Language Processing (NLP) techniques to construct a keyword dictionary for searching mini-programs and leverages metadata obtained via the WeChat API to enhance the search process. The crawler then retrieves and unpacks the mini-programs for further analysis, providing valuable input for subsequent security evaluations. Furthermore, the crawler retrieves mini-program packages from the user profile directory and unpacks them into source code using *wxappUnpacker* [9] for further analysis.

BLE Mini-programs Filter. To conduct a security analysis of BLE mini-programs, MiniBLE first needs to extract BLE mini-programs from the collected mini-programs. We found that in previous measurement work targeting BLE companion mobile apps, Du et al. [22, 23] identified BLE companion apps based on the presence of Bluetooth permission declarations and Bluetooth API calls within the apps. This inspired us to filter BLE mini-programs similarly by searching for common characteristics.

As described earlier, a mini-program needs to call `wx.openBluetoothAdapter` to enable the Bluetooth module, which is an essential step to use BLE. MiniBLE uses the static analysis tool CodeQL to generate an abstract syntax tree (AST) of the source code for each crawled mini-program. It then searches for a callee node named `wx.openBluetoothAdapter` within the AST and filters out BLE mini-programs without this node.

```
/* From: WeChat 8.0.40 for Android */
...
k2.j(jVar.f295994r, "PAIRING_VARIANT_PIN", new Object[0]);
if (jVar.f295991o == null) {
    jVar.p(sR0.m.f307918v);
    jVar.m();
    return;
}
k2.j(jVar.f295994r, "setPin", new Object[0]);
if (!bluetoothDevice.setPin(jVar.f295991o) || jVar.f295993q) {
    return;
}
...
```

Listing 1: Implementation of wx.makeBluetoothPair from Decompile WeChat Android App.

BLE Security Analysis. At this phase, MiniBLE primarily focuses on the security issues related to the BLE pairing API mentioned earlier. To verify whether the `wx.makeBluetoothPair` API causes a security downgrade in IoT devices, we conducted a reverse engineering to the WeChat Android APK. Listing 1 shows the underlying implementation of this API. We found that this API also called the Android system-level API `Bluetooth.setPIN`, which is used to set the PIN in Base64 encoding. Thus, any IoT device using this API in a WeChat mini-program will downgrade to the insecure Just-Works pairing method.

Therefore, this phase of MiniBLE can be translated into whether the mini-program 1) calls the `wx.makeBluetoothPair` API, and 2) can obtain its mandatory parameter PIN. Specifically, MiniBLE searches for a callee node `named wx.makeBluetoothPair` in the AST. If the node is found, MiniBLE then extracts the value of its parameter PIN. MiniBLE considers a mini-program vulnerable if the PIN can be extracted.

4 Evaluations

In this section, we discuss the detection results of MiniBLE, and present a case study with a real-world example.

Experiment Setup. The mini-program collection crawler of MiniBLE was deployed on a Windows 10 laptop (i7-9750H/32 GB RAM) with Python 3.8.10. The static analysis of MiniBLE is performed on a server running Ubuntu 20.04 with 32 CPU cores and 256 GB memory, utilizing 10 threads to analyze all mini-programs.

```
/* Pairing with device */
wx.makeBluetoothPair({
    deviceId: deviceId,
    pin: wx.arrayBufferToBase64(new Uint8Array([1, 2,
      3, 4, 5, 6])),
    success: (res) => {
    console.log("makePair-success:", res)
    },
    fail: (err) => {
        console.log("makePair-fail", err)
    },
    complete: (res) => {
        console.log("makePair-complete", res)
    }
})
```

Listing 2: Source code of a BLE Smart Lock Companion Mini-program.

Dataset. We use the collected 44,273 WeChat mini-programs as the dataset for our experiment, occupying a storage space of 126.38 GB.

Result Overview. After excluding non-unpackable mini-programs, 44,273 mini-programs were successfully unpacked into the source code. Among those collected mini-programs, MiniBLE successfully analyzed 41,726 (94.2%) of them, identifying 1,316 (3.2%) mini-programs using BLE APIs (i.e., BLE mini-programs).

Among those BLE mini-programs, MiniBLE found that 1,099 (83.5%) mini-programs used `wx.makeBluetoothPair` for pairing. This reflects the widespread use of `wx.makeBluetoothPair` in BLE mini-programs and highlights that many mini-programs may have adopted simplified pairing mechanisms, potentially introducing security risks.

Real-World Case Study. To validate the effectiveness of MiniBLE, we manually analyzed a companion mini-program for a smart lock based on the analysis results. Listing 2 shows the code snippet of this mini-program calling `wx.makeBluetoothPair` and using "123456" as the PIN, encoded in Base64 (using `wx.arrayBufferToBase64`). This degrades the PassKey Entry method to Just Work, putting end users at risk of MITM attacks.

During the BLE pairing process, an attacker can exploit an MITM attack using a static and predictable PIN ("123456") encoded in Base64. To conduct such an attack, the attacker positions themselves within the Bluetooth range of the victim's smart lock and its companion mini-program. Then, the attacker can easily eavesdrop on the BLE communication by BLE dongles or sniffers. The static PIN allows them to extract sensitive information, such as pairing credentials, without complex techniques or tools.

5 Discussions

Limitations. Currently, MiniBLE has only detected insecure BLE pairing issues in mini-programs. We will further discuss and investigate other potential BLE security issues. Additionally, MiniBLE currently supports automated analysis of WeChat mini-programs only. However, we have identified that other mini-program platforms, such as Alipay mini-programs [2], also support BLE devices. Since mini-programs on these platforms are WebView-based applications developed using JavaScript, MiniBLE can be easily extended to analyze these platforms as well.

Ethical Considerations. To ensure ethical research practices, we conducted proof-of-concept attacks only on our accounts, devices, and mini-programs. When crawling mini-programs or using the metadata API, we set a reasonable rate limit (i.e., 20 requests per minute) to prevent server disruptions.

6 Related Work

Mini-Program Security. Zhang et al. [36] presented MiniCrawler to analyze WeChat mini-programs resource consumption, API usage, and obfuscation rate. Lu et al. [26] identified security flaws in app-in-app systems related to resource management. Wang et al. [31] developed WeDetector to find bug patterns, uncovering 11 new bugs in 25 mini-programs. Zhang et al. [34] examined identity confusion vulnerabilities in 47 super apps, finding all were susceptible. Yang et al. [33] introduced CMRFScanner, revealing that 50,281 WeChat mini-programs and 493 Baidu mini-programs are vulnerable to Cross Miniapp Request Forgery (CMRF) attacks.

In contrast to the aforementioned research, our work focuses on the BLE security issues in mini-programs. We developed MiniBLE to evaluate the prevalence of these BLE security issues on a large scale.

BLE Security. With the advancement of Bluetooth technology, more security research efforts are focusing on BLE security vulnerabilities. Wu et al. [32] targeted the BLE link-layer authentication mechanism's reconnection procedure, proposing a BLE spoofing attack where an attacker can supply spoofed data to a previously paired BLE client device by masquerading as a BLE server device. Tschirschnitz et al. [28] exposed a design flaw in the inconsistent association model of the BLE pairing process, enabling method confusion attacks to easily achieve an MITM position between two BLE devices. Additionally, there is growing research on BLE privacy leakage, including identity tracking attacks that exploit static UUIDs [38], MAC addresses [35], and advertised packets [24].

7 Conclusion

Mini-programs have gained worldwide popularity due to their convenience and functionality. Despite their benefits, they present significant security and privacy concerns, particularly with advanced features like Bluetooth Low Energy (BLE). In this work, we specifically targeted BLE security issues in WeChat mini-programs. We developed MiniBLE, an automated analysis framework designed to detect potential BLE security issues in WeChat mini-programs. Through a large-scale preliminary analysis of 41,276 collected mini-programs, we identified 1,316 BLE mini-programs and found that 83.5% had potential BLE insecure pairing issues, demonstrating the effectiveness of our approach.

Acknowledgements

This work was partially supported by the SFU Start-up Grant (Grant No. N001305), Taishan Young Scholar Program of Shandong Province, China (Grant No. tsqn202211001), Shandong Provincial Natural Science Foundation (Grant No. ZR2023MF043), and Xiaomi Young Talents Program.

References

[1] Accessed: 2024-07-15. *Alipay Mini-Program.* https://global.alipay.com/platform/site/product/mini-program

[2] Accessed: 2024-07-15. *Alipay Mini-program: Bluetooth API Overview.* https://miniprogram.alipay.com/docs/miniprogram/mpdev/api_device_bluetooth_bluetoothapioverview

[3] Accessed: 2024-07-15. *Baidu Smart Program.* https://smartprogram.baidu.com

[4] Accessed: 2024-07-15. *Code Composition of a WeChat Mini Program.* https://developers.weixin.qq.com/miniprogram/en/dev/framework/quickstart/code.html

[5] Accessed: 2024-07-15. *Tencent WeChat.* https://www.wechat.com/en/

[6] Accessed: 2024-07-15. *TikTok Mini-programs.* https://www.tiktok.com/discover/mini-programs

[7] Accessed: 2024-07-15. *WeChat Mini Program Host Environment.* https://developers.weixin.qq.com/miniprogram/en/dev/framework/quickstart/framework.html

[8] Accessed: 2024-07-15. *WeChat Style Sheets (WXSS).* https://developers.weixin.qq.com/miniprogram/en/dev/framework/view/wxss.html

[9] Accessed: 2024-07-15. *wxappUnpacker.* https://github.com/system-cpu/wxappUnpacker

[10] Accessed: 2024-07-15. *wx.closeBluetoothAdapter.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.closeBluetoothAdapter.html

[11] Accessed: 2024-07-15. *wx.createBLEConnection.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth-ble/wx.createBLEConnection.html

[12] Accessed: 2024-07-15. *wx.getBLEDeviceCharacteristics.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.getBLEDeviceCharacteristics.html

[13] Accessed: 2024-07-15. *wx.getBLEDeviceServices.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.getBLEDeviceServices.html

[14] Accessed: 2024-07-15. *wx.makeBluetoothPair.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.makeBluetoothPair.html

[15] Accessed: 2024-07-15. *wx.notifyBLECharacteristicValueChange.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.notifyBLECharacteristicValueChange.html

[16] Accessed: 2024-07-15. *wx.onBluetoothDeviceFound.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.onBluetoothDeviceFound.html

[17] Accessed: 2024-07-15. *wx.startBluetoothDevicesDiscovery.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.startBluetoothDevicesDiscovery.html

[18] Accessed: 2024-07-15. *wx.startHCE.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/nfc/wx.startHCE.html

[19] Accessed: 2024-07-15. *wx.startWifi.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/wifi/wx.startWifi.html

[20] Accessed: 2024-07-15. *wx.stopBluetoothDevicesDiscovery.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.stopBluetoothDevicesDiscovery.html

[21] Accessed: 2024-07-15. *wx.writeBLECharacteristicValue.* https://developers.weixin.qq.com/miniprogram/en/dev/api/device/bluetooth/wx.writeBLECharacteristicValue.html

[22] Jianqi Du, Fenghao Xu, Chennan Zhang, Zidong Zhang, Xiaoyin Liu, Pengcheng Ren, Wenrui Diao, Shanqing Guo, and Kehuan Zhang. 2022. Identifying the ble misconfigurations of iot devices through companion mobile apps. In *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON).*

[23] Jianqi Du, Zidong Zhang, Fenghao Xu, and Wenrui Diao. 2023. Living in the Past: Analyzing BLE IoT Devices Based on Mobile Companion Apps in Old Versions. In *19th International Conference on Mobility, Sensing and Networking, MSN 2023, Nanjing, China, December 14-16, 2023.*

[24] Aleksandra Korolova and Vinod Sharma. 2018. Cross-App Tracking via Nearby Bluetooth Low Energy Devices. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy (CODASPY), Tempe, AZ, USA, March 19-21, 2018.*

[25] Wei Li, Borui Yang, Hangyu Ye, Liyao Xiang, Qingxiao Tao, Xinbing Wang, and Chenghu Zhou. 2023. MiniTracker: Large-Scale Sensitive Information Tracking in Mini Apps. *IEEE Transactions on Dependable and Secure Computing* (2023).

[26] 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 (CCS), Virtual Event, USA, November 9-13, 2020.*

[27] Pallavi Sivakumaran and Jorge Blasco. 2019. A Study of the Feasibility of Co-located App Attacks against {BLE} and a {Large-Scale} Analysis of the Current {Application-Layer} Security Landscape. In *28th USENIX Security Symposium (USENIX Security 19).*

[28] Maximilian Von Tschirschnitz, Ludwig Peuckert, Fabian Franzen, and Jens Grossklags. 2021. Method confusion attack on bluetooth pairing. In *2021 IEEE symposium on security and privacy (SP).*

[29] 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 *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE), Melbourne, Australia, May 14-20, 2023.*

[30] Chao Wang, Yue Zhang, and Zhiqiang Lin. 2023. One Size Does Not Fit All: Uncovering and Exploiting Cross Platform Discrepant APIs in WeChat. In *Proceedings of the 32nd USENIX Security Symposium (USENIX-Sec), Anaheim, CA, USA, August 9-11, 2023.*

[31] Tao Wang, Qingxin Xu, Xiaoning Chang, Wensheng Dou, Jiaxin Zhu, Jinhui Xie, Yuetang Deng, Jianbo Yang, Jiaheng Yang, Jun Wei, and Tao Huang. 2022. Characterizing and Detecting Bugs in WeChat Mini-Programs. In *Proceedings of the 44th IEEE/ACM 44th International Conference on Software Engineering (ICSE), Pittsburgh, PA, USA, May 25-27, 2022.*

[32] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Dave Jing Tian, Antonio Bianchi, Mathias Payer, and Dongyan Xu. 2020. {BLESA}: Spoofing attacks against reconnections in Bluetooth low energy. In *14th USENIX Workshop on Offensive Technologies (WOOT 20).*

[33] 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 (CCS), Los Angeles, CA, USA, November 7-11, 2022.*

[34] 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 *Proceedings of the 31st USENIX Security Symposium (USENIX-Sec), Boston, MA, USA, August 10-12, 2022.*

[35] Yue Zhang and Zhiqiang Lin. 2022. When Good Becomes Evil: Tracking Bluetooth Low Energy Devices via Allowlist-based Side Channel and Its Countermeasure. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.).

[36] Yue Zhang, Bayan Turkistani, Allen Yuqing Yang, Chaoshun Zuo, and Zhiqiang Lin. 2021. A Measurement Study of Wechat Mini-Apps. In *Proceedings of the 2021 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Virtual Event, China, June 14-18, 2021.*

[37] Zidong Zhang, Qingsheng Hou, Lingyun Ying, Wenrui Diao, Yacong Gu, Rui Li, Shanqing Guo, and Haixin Duan. 2024. MiniCAT: Understanding and Detecting Cross-Page Request Forgery Vulnerabilities in Mini-Programs. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, Salt Lake City, UT, USA. October 14-18, 2024.*

[38] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. 2019. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019.*