Micro-Mobility Security: A Holistic Approach via Mobile App Analysis

Marco Balossini[1,*,†], Michele Carminati[1,†], Stefano Zanero[1,†] and Stefano Longari[1,†]

[1]*Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy*

Abstract
The growth of urbanization and technology has reshaped transportation, bringing about shared micro-mobility services like e-scooters and e-bikes as sustainable solutions for short urban trips, mitigating traffic and pollution issues. However, the rapid expansion of this market has outpaced the development of standardized security measures. Our research aims to analyze theoretically and empirically the security landscape of shared micro-mobility services, focusing on client components. We present a reference architecture and a detailed threat model to identify and address the most pressing security challenges. Additionally, we design a set of experiments to evaluate existing services and assess their security implementations. Finally, we propose guidelines and best practices to enhance the security of shared micro-mobility services, highlighting the need for a standardized security approach and offering a foundation for future research.

Keywords
Micro-mobility, e-scooters, mobile security, threat modeling

1. Introduction

Micro-mobility has recently emerged as a significant trend in urban transportation, characterized by the use of lightweight electric vehicles such as e-scooters and e-bikes [1]. Sharing services offer a sustainable solution for short trips in cities, alleviating urban traffic congestion and lowering carbon emissions. These services distribute vehicles throughout the city, allowing users to rent them via a mobile application, use them for their trips, and drop them off almost anywhere when finished [2].

Despite their benefits, the rapid growth of shared micro-mobility services has introduced new security challenges, mostly due to integration with technologies such as GPS, GSM, and Bluetooth, along with their dependence on mobile applications, all of which significantly expand the attack surface. Additionally, the sharing framework requires the vehicles to be constantly accessible to the public, making it difficult to protect them from external threats.

While existing research has addressed general mobile or IoT security, it has largely overlooked the micro-mobility context. Although these fields are akin to the main micro-mobility aspects, few studies – none of them beyond the theoretical level – tackle the security challenges of the entire system. To the best of our knowledge, the most comprehensive threat model available in the literature is [3], which defines a reference architecture for the system and consolidates insights from the few academic studies and the most relevant online resources.

This landscape lacks both a holistic study evaluating the system as a whole – rather than focusing on its individual components – and an experimental evaluation of the system and its real-world industry implementations. These gaps have resulted in the lack of domain-specific security guidelines designed for shared micro-mobility services.

In this work, we begin by slightly refining the reference architecture in [3] and using the STRIDE framework to identify the cyber and physical threats. Sequentially, we evaluate the most popular shared

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
*Corresponding author.
†These authors contributed equally.
marco.balossini@polimi.it (M. Balossini); michele.carminati@polimi.it (M. Carminati); stefano.zanero@polimi.it (S. Zanero); stefano.longari@polimi.it (S. Longari)
0009-0000-1644-136X (M. Balossini); 0000-0001-8284-6074 (M. Carminati); 0000-0003-4710-5283 (S. Zanero); 0000-0002-7533-4510 (S. Longari)

micro-mobility services, focusing on the use case of shared e-scooter services – the most common in the micro-mobility framework [4]. Our methodology consists of three steps. First, we conduct an exploratory analysis of the sample, using automated tools to detect the most common security weaknesses and malpractices. Next, we perform manual reverse engineering of the applications to gain insights into their backend mechanisms. Finally, we examine the real-world behavior of a subsample of these applications, designing targeted tests to verify the presence of the cyber threats we modeled before. The analysis results allow us to propose a set of security guidelines and best practices to support the development of shared micro-mobility services.

With this research in the shared micro-mobility field, our contributions are:

- The creation of a comprehensive threat model for the system, considering both cyber and physical threats
- The evaluation of the security measures and architectural patterns currently implemented in the industry
- The formulation of tailored guidelines aimed at enhancing the system's security

2. Related Works

This section provides an overview of the existing challenges and explores solutions in the two key areas relevant to this study: micro-mobility security and mobile application security.

Micro-mobility security area is far from being thoroughly explored, however, Vinayaga-Sureshkanth et al. [3] proposed a threat model in 2020, identifying the challenges. They considered six attack categories and, for each category, provided examples.

The first category is **vehicle tampering**, which could lead to firmware alterations [5], battery drainage, and physical tampering [6]. **Eavesdropping** is a feasible attack due to the insecure communication mediums and lack of encryption [7], and it could also lead to full **MITM and replay** attacks, potentially taking control of the vehicle [8]. Through **fuzzing and DoS**, an attacker could find new vulnerabilities in the system and exploit them or disrupt the service [9]. Studies show that very few companies are protected against **GPS spoofing** [10], even though attackers could easily alter the vehicle's position [11]. Finally, **data inference** can be used to track users' movements and infer sensitive information [12, 13, 14, 10].

Since our work focuses on the mobile ecosystem as the attack surface, we deem it necessary to present some research in the areas of our interest. As opposed to micro-mobility security, mobile application security is a well-studied field with many established guidelines and practical studies. OWASP [15] provides a list of the top 10 mobile risks, which include insecure data storage, insecure communication, and insufficient cryptography. Ghafari et al. [16] identified the most common vulnerabilities in Android applications, highlighting XSS-like injections as the most common. While this study focused only on proprietary code, Zhan et al. [17] analyzed TPLs as a source of vulnerabilities.

Our work differentiates from prior studies in shared micro-mobility security by employing a hands-on analysis methodology. While previous research, such as that by Vinayaga-Sureshkanth et al., proposed countermeasures for threats identified by theoretical studies, our approach starts with the observation and evaluation of the current security measures to retain effective practices and propose solutions for the identified weaknesses.

3. Identified Threat Model

Building upon the existing studies, our threat model focuses on vulnerabilities imported into the system by the presence of a mobile application. The first step (in Section 3.1) is to identify the components and their way of communicating. Section 3.2 identifies assets and threat agents; subsequently Section 3.3 categorizes the threats using the STRIDE framework [18].

3.1. Reference Architecture

The shared micro-mobility system's architecture contains three components: the electric vehicle, mobile application, and API server. A schematized version is presented in Figure 1.

Vehicle. Usually an e-scooter or an e-bike with a removable battery. It contains motion sensor, GSM, and GNSS modules. Some also communicate over short distances using BLE.

Mobile Application. The interface between the user and the system. It allows users to locate available vehicles, initiate and close rental sessions, and manage payment.

Server. Coordinates all the system's operations and stores all the service's data. It processes the clients' requests and may act as an intermediary between the application and the vehicle.

3.1.1. Communication

With three components, only three communication channels are possible: server-app, server-vehicle, and app-vehicle. While observations show that the server always communicates with the other components, the application and vehicle do not communicate directly in all cases.

The communication between client and server uses HTTPS protocol to prevent MITM attacks. The vehicle usually employs a GSM module to report its position and receive commands. Any discussions regarding the content of this communication are hypothetical since our research focuses on the client application, leaving the direct analysis of this channel beyond our scope.

The presence of a communication channel between client and vehicle distinguishes the two architectural approaches we observed in micro-mobility services: centralized and decentralized services. Even though Vinayaga-Sureshkanth et al. [3] considered only decentralized services, we observed many more centralized applications, likely due to the evolution of micro-mobility.

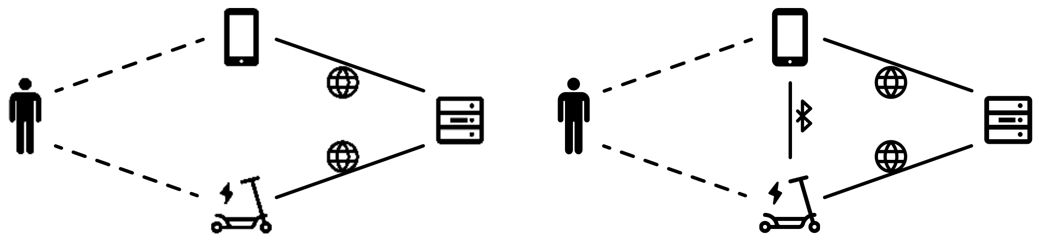

(a) Centralized Architecture. (b) Decentralized Architecture.

Figure 1: Possible Shared Micro-mobility System Architectures.

The difference between these two variations lies in the server's role. In centralized architectures, the server acts as a broker while, in decentralized architectures, the server registers and responds to the client's requests, then commands are relayed to the vehicle by the client itself.

3.2. Assets, Threat Agents

Micro-mobility systems present valuable assets that can become targets for various security threats. Key assets include the **vehicle** itself, susceptible to vandalism, theft, or tampering. **User safety** is at risk, especially if an attacker interferes with vehicle operations, for example, by uploading malicious firmware or damaging the vehicle itself [6]. **Service revenue** can be undermined by unauthorized use or fare evasion, while **service reputation** can be damaged by reducing availability or reliability. **Personal Identifiable Information (PII)** and **tracking data** are at risk of being stolen for malicious activities or user profiling.

Various threat agents can target these assets, each with specific motivations and capabilities. **Cybercriminals** may seek to steal data for ransom or resale and/or disrupt service operations. **Competitors** might try to damage reputation and service reliability with the goal of drawing users away. **Illegitimate users** bypass payment mechanisms, leading to revenue loss, while **legitimate users** may breach terms by misusing vehicles. **Service providers** might exploit collected data beyond its intended use while **company employees**, especially those with privileged access, pose an inside threat of data manipulation or tampering with the system's behavior. **Vandals**, without specific motives, threaten the physical integrity of vehicles.

3.3. Threat Categorization

In our threat model, we use the STRIDE framework [18] to classify the threats into 6 categories: Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, and Elevation of privilege. Moreover, for the sake of comprehensibility, we separately consider physical threats and cyber threats. Physical threats, such as those posed by legitimate users and vandals, require direct access to vehicles. Instead, cyber threats require the ability to influence the system's behavior, often through exploiting vulnerabilities in server, communications, or clients. A table summarizing these threats is provided in the appendices for clearer reference.

3.3.1. Physical Threats

The vehicle is a key component of the micro-mobility system and presents important security challenges since anyone can physically access the target. While it is possible to mitigate some of the problems associated with this exposure, some threats cannot be completely mitigated.

The threats include potential spoofing attacks for the vehicle (**PS1**) or its GPS location (**PS2**). Tampering poses further risks, with attackers altering firmware (**PT1**), disrupting communication with the server (**PT2**) or the app (**PT3**), or modifying the vehicle's hardware (**PT4**). Finally, DoS attacks like battery draining (**PD1**) or vehicle breakdowns (**PD2**) can cause the interruption of the service.

3.3.2. Cyber Threats

Cyber threats in micro-mobility systems present significant security challenges due to the widespread use of mobile applications and their inherent vulnerabilities. Spoofing can allow attackers to manipulate users' GPS locations (**CS1**). Tampering risks involve malicious activities such as APK repackaging (**CT1**) or exploiting application vulnerabilities stemming from outdated libraries, weak cryptographic practices, or insecure code (**CT2**). Repudiation is another concern, as users may alter logs (**CR1**) or deny actions like ride creation (**CR2**). Information disclosure threats include a cyber-criminal extracting PII or tracking data from a device's memory (**CI1**) as well as the provider company monitoring users even when the app is inactive (**CI2**). Denial of Service attacks can be performed by tricking the server into misinterpreting the status of vehicles' battery (**CD1**), leading to service disruption. Lastly, elevation of privilege threats could enable attackers to access higher-level functions, such as unlocking vehicles without payment (**CE1**), thereby bypassing standard security and payment protocols.

4. Security Analysis Methodology

Our approach to the analysis is composed of three steps: **preliminary analysis**, **manual static analysis**, and **dynamic analysis**. The preliminary analysis explores the sample of applications and employs automatic tools to identify the most common weaknesses and bad practices. The static analysis involves the manual inspection of the applications to detect different patterns in the service's structure. Finally, dynamic analysis monitored a subsample applications' runtime behavior and interaction with the other components of the system.

Our study focused on Android applications for several reasons. Android's open-source nature, and the possibility to install a rooted OS, make it particularly convenient for in-depth analysis. It is also the most appealing target for cyber-criminals [19] because of its wide diffusion – 70% of the market share [20] – and open nature. Apps typically behave consistently across Android and iOS platforms, often sharing the same codebase with only minor implementation differences, which is not an issue as our primary focus is on identifying design errors.

4.1. Preliminary Analysis

First, we selected a sample of highly downloaded micro-mobility rental applications from the Google Store. The initial exploration involved identifying each app's main features and implementation technology.

We unpacked and decompiled APKs using JADX [21] to analyze their structure and content, focusing on the `AndroidManifest.xml` file. We flagged applications with excessive permissions or unnecessary exported Activities, as these significantly expand the attack surface. Then, we used two widely renowned static analysis tools, such as MobSF [22] and Ostorlab [23] to detect common vulnerabilities and insecure coding practices, such as weak cryptographic practices and the presence of anti-root or anti-repackaging mechanisms. By employing this method, we aimed to balance thoroughness and efficiency, avoiding the prohibitive time costs associated with manual testing while still addressing critical vulnerabilities.

4.2. Manual Static Analysis

As outlined in Section 3.1, shared micro-mobility services can follow two distinct system architectures. This step aims to identify their average implementation structure and understand the differences through manual static analysis.

The analysis starts by decompiling the native apps – twenty out of the original sample. This step mainly focuses on the model and controller components, which define the application's internal logic. Moreover, we prioritized analyzing less obfuscated apps for a clearer understanding of the general micro-mobility apps' structure, before diving into the most complex ones.

The next phase involves the examination of the API endpoints called by the client application. In doing so, we seek to understand the differences in the server's role, in both the different architectural designs, analyzing the set of functions offered by the server.

Finally, for decentralized applications, the analysis concluded with the examination of the BLE communication between the app and the vehicle, to completely evaluate the whole communication chain that relays the commands from the server to the vehicles.

4.3. Dynamic Analysis and Integration Tests

The sample consisted of real-world micro-mobility applications characterized by extensive features, sophisticated security measures, and detailed graphical interfaces. In 60% of the cases, this results in obfuscated codebases containing thousands of Java classes, many dedicated solely to graphical presentation. Therefore, static analysis alone is insufficient to fully comprehend the system, especially since the application has only partial knowledge of the system's state. To address these limitations, we incorporated dynamic analysis techniques and integration testing.

Firstly, we monitor the applications' communications and reconstruct the key sequences for the most common use case: purchasing a ride. This approach proved particularly valuable for highly obfuscated applications, where static analysis was less effective.

Finally, we conducted targeted field tests to address the verifiable threats outlined in Section 3.3 – more on this in Section 5.4. Given resource constraints, it was not feasible to examine all physical and digital threats. Instead, we prioritize the most critical threats centered on the client application.

5. Security Evaluation

To respect the boundaries of the applications in the sample, we restricted our testing to only the cyber threats **CS1**, **CT1**, **CT2**, **CI1**, **CI2**, and partially **CE1**. Moreover, we present the results in an anonymous form to avoid potential harm to the applications or their users.

5.1. Sample Selection

The recent proliferation of micro-mobility applications provides a wide range of options for our study. We prioritized the most downloaded services in Europe and North America focusing on e-scooter shared mobility, which is the most prevalent market. The final sample includes Beam, Bird, Bit, Bolt, Bounce, Dott, Ginger, Goon, Helbiz, Jet, Lime, MVGO, Neuron, Ridemovi, Ryde, Skip, Spin, Superpedestrian, Tier, Voi, Whee, Wind, Yango, Yoio, and Zwings.

Our selection results in a sample of twenty-four applications, all offering e-scooter rental, alongside other services (e.g., e-bike rental, ride-hailing, and food delivery). Out of the twenty-four analyzed applications, twenty are native Android apps, three are developed with Flutter, and one with React Native.

5.2. Preliminary Analysis

In the preliminary phase of the analysis, we test the whole sample for the threats **CT1** and **CT2**. The most significant identified issue is the use of outdated TPL in numerous applications, many of which are affected by active CVEs. On the other hand, several applications have root detection capabilities as well as anti-repackaging measures. We also identified a few custom OAuth implementation schemes, which could introduce vulnerabilities like account takeovers [24]. Some applications also use weak encryption methods, even in critical functionalities such as vehicle locking, or maintain world-writeable shared preferences, allowing unauthorized access by other applications.

The results of the preliminary analysis are presented in Figure 2.

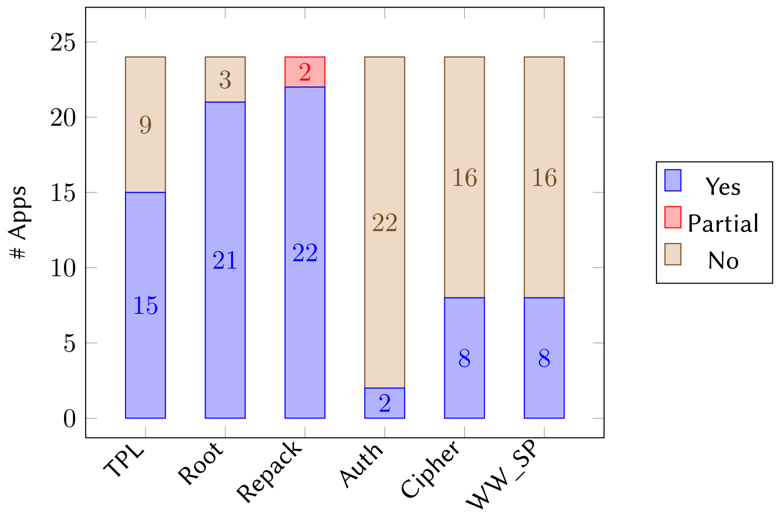


Figure 2: Automatic Analysis Results

5.3. Static Analysis

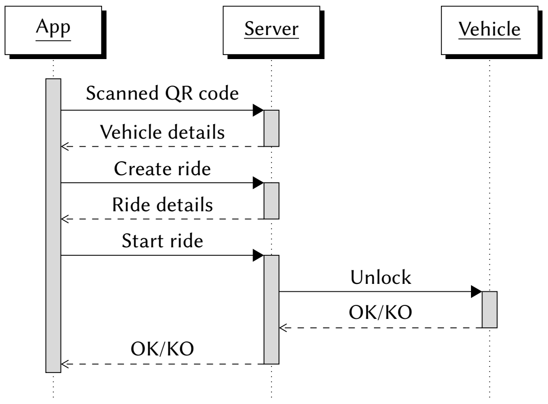
The static analysis examined the subsample of native applications. We observe that decentralized applications – implemented exclusively via BLE – are generally more obfuscated, likely due to the client being more central in this architecture. The API endpoints analysis reveals substantial variations at the implementation level but no distinct patterns can be observed between the two architectural types. At a high level, all applications exhibit similar functionalities.

We identified seven decentralized applications and, in each, note a common process: the client registers a ride on the server, pairs with the vehicle, requests a token to unlock it, and forwards the token to the vehicle over BLE. Unfortunately, we could not observe the token generation mechanism, plausible methods include static tokens, OTP, or distribution by the server. Since the use of static tokens would be a serious vulnerability, we evaluate the possibility in Section 5.4.

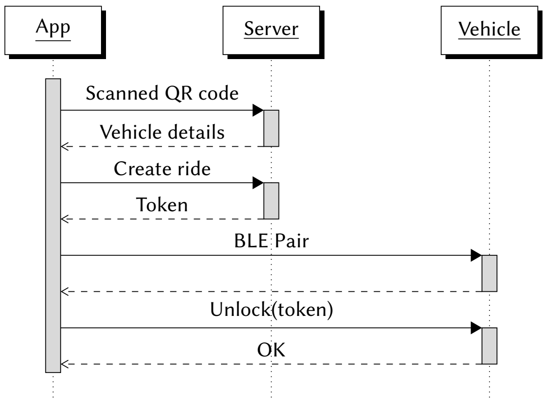
5.4. Dynamic Analysis and Field Testing

For the dynamic analysis phase, we developed a setup using a rooted Nexus 5X smartphone running Android 8.1.0. During the tests, the device is connected to a laptop using the USB debugging feature while the Frida server is running. We monitor the communications with Burp (for internet protocols) and the Bluetooth HCI Snoop log feature functionality (for Bluetooth). All the applications are launched with Frida [25] instrumentation to disable SSL pinning.

The first step is to dump the client communication sequences of the six centralized and decentralized services available in our region. The observed sequences show significant similarity for services with the same architecture; hereafter we present a generalization in figure 3.



(a) Sequence of centralized applications



(b) Sequence of decentralized applications

Figure 3: Vehicle unlock sequences in both architectures

Since the manual static analysis is not enough to fully assess the threats **CS1**, **CI1**, **CI2**, and **CE1**, we design a set of specific tests covering each of these threats. In the following sections, we present the tests designed for each threat and the corresponding results. Additional procedural details are provided in the Appendices.

CS1 - Smartphone GPS Spoofing. During our analysis, we observe that many applications share the user's position with the server at a very high frequency, which produces two questions:

- Do the micro-mobility services rely **completely** on the client's position for vehicle tracking? (instead on the vehicle's position)
- Is it possible to misguide the vehicle's behavior using the smartphone's location? (Either during the ride or after it ends).

To verify the first question, we move the vehicles without starting a ride to check if their position remains up to date on the app. As expected, all the vehicles in the subsample contain a GPS tracker and keep their position updated. However, we notice that some services take as long as 30 minutes to update the position.

Since this delay is substantial, we suspect that the system prefers using the smartphone's position during a ride to save the vehicle's energy. This approach allows a user to exit the geographical boundaries or ignore the speed limits in specific areas (e.g., pedestrian zones).

Therefore, we conduct new tests by spoofing the smartphone's GPS position and evaluate the server's behavior both during the ride and immediately at its end. Both tests yield negative results for the entire subsample, proving that servers use the vehicle's location—which is more reliable—instead of the smartphone's.

CI1 - Memory Analysis. With the amount of data handled by shared mobility applications, the threat of data leakage remains significant. This test verifies whether the information stored in the phone's memory is accessible to anyone with physical access to the device.

After using the application, we scan the device's public and private memory and inspect the files connected to the application for any PII and tracking data. The results reveal that one-third of the subsample does not expose any relevant data, another third exposes incomplete tracking data, and the final third exposes both PII and partial tracking information.

CI2 - User Tracking. Service providers may misuse their permissions to constantly track users if the permissions are inaccurately configured. To verify this, we test whether the application continues transmitting the user's position to the server even when inactive and running in the background. The results show that only one of the six tested applications keeps tracking the user. While it may be used for profiling, the purpose of the tracking remains unclear.

CE1 - Illicit Unlock in Decentralized Architectures. In decentralized applications, the user's smartphone—the most untrusted component of the system—directly sends the vehicle the unlock command (see Figure 3b). We do not fully investigate the specifics of token generation nor attempt to heavily attack the system to gain a deeper understanding of this process. However, we test whether the token changes or remains the same.

Only one of the six field-tested services employs a decentralized architecture, restricting this test to a single application. However, we confirm that the token consistently changes between different vehicles and between separate rides of the same vehicle.

While this test does not conclusively verify sound token generation in decentralized services, it suggests that any vulnerability in this domain is more likely due to an implementation flaw rather than poor design.

6. Proposed Security Guidelines

In this section, we present a set of guidelines and best practices for the development of shared micro-mobility services. These recommendations are derived from our analysis and informed by existing industry solutions. While companies have implemented various strategies to address such challenges, these approaches often lack formal evaluation.

6.1. Physical Threats

Physical threats target the system's most exposed component: the vehicle. The vehicle faces constant attack risk, making it particularly difficult to defend. Companies cannot fully eliminate these risks but must aim to mitigate them. In some cases (e.g., vandalism), not even threat mitigation is feasible. Our guidelines address various threats to the vehicle, emphasizing protection mechanisms that reduce the risk of successful attacks.

Threats PS2, PT2, PT3. Weak encryption in communications could allow an attacker to perform MITM attacks and tamper with data exchanges, while the absence of GPS spoofing detection permits fake location data to compromise system operations.

Threats PT1, PT4, PD1. Protecting the vehicle's board and battery requires both software and physical security measures. Firmware should be secured by disabling unused ports on the motherboard, signing the updates, and authenticating their source. On the hardware side, locking the motherboard and battery with tamper-resistant lids and custom keys can reduce the risk of physical tampering.

Threat PS1. Implementing robust authentication protocols for vehicles is essential to prevent unauthorized entities from impersonating legitimate ones or accessing sensitive network data. Such measures mitigate various attack vectors, including MITM and replay attacks, ensuring the security and integrity of the vehicle's communications.

6.2. Cyber Threats

Cyber threats leverage vulnerabilities in the micro-mobility system's mobile components to compromise either the vehicle or the server. Addressing these threats effectively requires a multi-layered approach that encompasses data validation, secure development practices, robust logging mechanisms, and a strong commitment to user privacy.

Threats CS1, CD1. Ensuring data originating from the client application, such as GPS coordinates or battery status, are always validated against data generated by the vehicle is critical. Since the vehicle's data is harder to tamper with, this approach helps mitigate risks such as unauthorized movements outside designated zones or bypassing speed limits.

Threats CT1, CT2. Securing the application code by employing secure signature mechanisms ensures its integrity by preventing unauthorized modifications or insertions. Regular updates to third-party libraries address known vulnerabilities, while code obfuscation, although not a definitive solution, helps deter reverse engineering. Avoiding storing sensitive data directly within the application limits the attack surface.

Threat CR1. Implementing a robust logging mechanism in an immutable format ensures tamper-evident records of actions. Strong access controls restrict access so that only authorized personnel can view or manage these logs, while integrity checks verify logs are unaltered.

Threat CI1. Sensitive information stored on the mobile device must be encrypted, ensuring its security even in cases of theft or loss. Private memory should be preferred over shared memory and all communications between the client and server must utilize secure protocols.

Threat CI2. Service providers should limit data collection to essential data needed to deliver the service, avoiding misuse of tracking information. Moreover, explicit consent must always be sought before collecting data.

6.3. Considerations

Cyber threat CE1 is particularly complex and cannot be directly addressed by the guidelines provided earlier. This privilege escalation threat relies on exploiting an existing vulnerability and is strongly influenced by the system's architecture. There are two possible attack vectors: tricking the server into believing a ride has started or forging new commands for the vehicle.

In the first scenario, an attacker would need to directly exploit the server, which is out of our scope. The second scenario is more relevant to our study but applies specifically to decentralized services, where the smartphone relays commands to the vehicle.

Our recommendation would be to use a centralized architecture, which reduces the system complexity and minimizes the attack surface. However, decentralized architecture is not inherently insecure, provided it includes measures to prevent MITM and brute-force attacks.

One drawback of centralized architecture is its reliance on constant vehicle connectivity. This limitation can be mitigated by carefully restricting the service area to avoid locations with weak connectivity. Additionally, vehicles in decentralized services use BLE instead of GSM, offering higher energy efficiency and reducing the strain on the vehicle's battery.

In conclusion, while centralized architecture is simpler and has reduced risk, the choice should ultimately depend on the specific goals of the service.

7. Conclusion

In this study, we developed a comprehensive shared micro-mobility threat model, grounded in experimental observations of the most popular services. Through the analysis of a representative sample of mobile applications, we identified existing industrial security measures and potential vulnerabilities. Finally, we proposed practical guidelines and best practices to support the design and implementation of safe shared micro-mobility platforms. Our analysis revealed several common issues, such as the use of outdated TPLs and weak ciphers. However, we also observed adequate security in internet communications – handled with proper protocols – and verified that neither centralized nor decentralized architectures are inherently vulnerable by design. Building on industry practices, our guidelines introduce targeted recommendations to address specific vulnerabilities we identified. Furthermore, we address security considerations at an architectural level, highlighting the strengths and weaknesses of both centralized and decentralized solutions.

Future research could involve collaboration with industry partners to evaluate the proposed threat model across the whole system, including physical threats to the vehicle and attacks on the server. Additionally, investigating the risks posed by cyber-criminals and insiders, aiming to steal data stored on servers, would provide valuable insights into protecting sensitive information.

In conclusion, while shared micro-mobility systems show reasonable protection against mobile-specific threats, our research highlights the need for a security assessment of the complete system. Such evaluations are essential for designing standardized security practices to safeguard user privacy, ensure safety, and protect the provider's business.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4o and Google Gemini in order to: paraphrase, reword, and improve the writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] NABSA, Shared Micromobility State of the Industry Report, nabsa.net/about/industry/?mc_cid=344470a2b0, 2022.

[2] D. Yanocha, M. Allan, The Electric Assist: Leveraging E-bikes and E-scooters for More Livable Cities, 2019. URL: https://www.itdp.org/wp-content/uploads/2019/12/ITDP_The-Electric-Assist_-Leveraging-E-bikes-and-E-scooters-for-More-Livable-Cities.pdf.

[3] N. Vinayaga-Sureshkanth, R. Wijewickrama, A. Maiti, M. Jadliwala, Security and Privacy Challenges in Upcoming Intelligent Urban Micromobility Transportation Systems, in: Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security, AutoSec '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 31–35. URL: https://doi.org/10.1145/3375706.3380559. doi:10.1145/3375706.3380559.

[4] Statista, Statista - The Statistics Portal, https://www.statista.com/, 2025.

[5] L. Catuogno, C. Galdi, Secure Firmware Update: Challenges and Solutions, Cryptography 7 (2023) 30. URL: https://www.mdpi.com/2410-387X/7/2/30. doi:10.3390/cryptography7020030, number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[6] A. J. Hawkins, Florida man arrested for cutting the brakes on over 100 electric scooters, https://www.theverge.com/2019/10/2/20895028/scooter-brake-cut-vandalism-fort-lauderdale-fla, 2019.

[7] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, S. Uluagac, Peek-a-boo: i see your smart home activities, even encrypted!, in: Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 207–218. URL: https://doi.org/10.1145/3395351.3399421. doi:10.1145/3395351.3399421.

[8] M. Casagrande, R. Cestaro, E. Losiouk, M. Conti, D. Antonioli, E-Spoofer: Attacking and Defending Xiaomi Electric Scooter Ecosystem, in: Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks, ACM, Guildford United Kingdom, 2023, pp. 85–95. URL: https://dl.acm.org/doi/10.1145/3558482.3590176. doi:10.1145/3558482.3590176.

[9] J. Thai, C. Yuan, A. M. Bayen, Resiliency of Mobility-as-a-Service Systems to Denial-of-Service Attacks, IEEE Trans. Control Netw. Syst. 5 (2018) 370–382. URL: http://ieeexplore.ieee.org/document/7574290/. doi:10.1109/TCNS.2016.2612828.

[10] Q. Zhao, C. Zuo, G. Pellegrino, L. Zhiqiang, Geo-locating Drivers: A Study of Sensitive Data Leakage in Ride-Hailing Services., in: Annual Network and Distributed System Security symposium, February 2019 (NDSS 2019), 2019. URL: https://publications.cispa.saarland/2757/.

[11] K. C. Zeng, Y. Shu, S. Liu, Y. Dou, Y. Yang, A Practical GPS Location Spoofing Attack in Road Navigation Scenario, in: Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications, HotMobile '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 85–90. URL: https://doi.org/10.1145/3032970.3032983. doi:10.1145/3032970.3032983.

[12] J. Li, F. Zeng, Z. Xiao, Z. Zheng, H. Jiang, Z. Li, Social Relationship Inference Over Private Vehicle Mobility Data, IEEE Trans. Veh. Technol. 70 (2021) 5221–5233. URL: https://ieeexplore.ieee.org/document/9360507/. doi:10.1109/TVT.2021.3060787.

[13] N. Vinayaga-Sureshkanth, R. Wijewickrama, A. Maiti, M. Jadliwala, An Investigative Study on the Privacy Implications of Mobile E-scooter Rental Apps, in: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 125–139. URL: https://doi.org/10.1145/3507657.3528551. doi:10.1145/3507657.3528551.

[14] Y. Xu, X. Yan, V. P. Sisiopiku, L. A. Merlin, F. Xing, X. Zhao, Micromobility Trip Origin and Destination Inference Using General Bikeshare Feed Specification Data, Transportation Research Record 2676 (2022) 223–238. URL: https://doi.org/10.1177/03611981221092005. doi:10.1177/03611981221092005, publisher: SAGE Publications Inc.

[15] C. Holguera, S. Schleier, B. Mueller, J. Willemsen, OWASP Mobile Application Security Testing Guide, 2022. URL: https://github.com/OWASP/owasp-mastg/.

[16] M. Ghafari, P. Gadient, O. Nierstrasz, Security Smells in Android, in: 2017 IEEE 17th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2017, pp. 121–130. URL: https://ieeexplore.ieee.org/document/8090145. doi:10.1109/SCAM.2017.24, iSSN: 2470-6892.

[17] X. Zhan, T. Liu, L. Fan, L. Li, S. Chen, X. Luo, Y. Liu, Research on Third-Party Libraries in AndroidApps: A Taxonomy and Systematic LiteratureReview, 2021. URL: http://arxiv.org/abs/2108.03787, arXiv:2108.03787 [cs].

[18] A. Shostack, Threat Modeling: Designing for Security, John Wiley & Sons, 2014. Google-Books-ID: YiHcAgAAQBAJ.

[19] S. Garg, N. Baliyan, Comparative analysis of Android and iOS from security viewpoint, Computer Science Review 40 (2021) 100372. URL: https://www.sciencedirect.com/science/article/pii/S1574013721000125. doi:10.1016/j.cosrev.2021.100372.

[20] StatCounter, Mobile Operating System Market Share Worldwide, https://gs.statcounter.com/os-market-share/mobile/worldwide, ????

[21] Skylot, Jadx, https://github.com/skylot/jadx, 2023.

[22] MobSF, https://github.com/MobSF/Mobile-Security-Framework-MobSF, 2024.

[23] Ostorlab, Ostorlab mobile application security scanner, https://report.ostorlab.co/dashboard/posture, 2023.

[24] OAuth, One Scheme to Rule Them All: OAuth Account Takeover | Ostorlab: Mobile App Security Testing for Android and iOS, https://blog.ostorlab.co/one-scheme-to-rule-them-all.html, 2023.

[25] Frida, Frida, https://github.com/frida/frida, 2024.

A. Threats Overview

This appendix provides a structured overview of the identified threats to the shared micro-mobility systems, categorized into physical threats and cyber threats. Table 1 outlines physical threats, that involve direct interaction with the vehicle, while Table 2 details cyber threats targeting the mobile application, backend systems, and user data. The assigned IDs identify each threat macro-category (Physical/Cyber) and category mnemonic in the STRIDE security framework.

Table 1
Physical Threats Overview

Physical Threats		
	Threat	**Description**
PS1	Vehicle Spoofing	Spoofing attacks targeting the vehicle itself.
PS2	GPS Spoofing	Manipulating the GPS location of the vehicle.
PT1	Firmware Tampering	Altering the vehicle's firmware to gain unauthorized access or cause malfunctions.
PT2	Communication Disruption (Server)	Disrupting communication between the vehicle and the server.
PT3	Communication Disruption (App)	Interfering with communication between the vehicle and the mobile app.
PT4	Hardware Tampering	Physically modifying the vehicle's hardware to disrupt functionality.
PD1	Battery Draining	Draining the vehicle's battery to cause a denial of service.
PD2	Vehicle Breakdown	Causing breakdowns to interrupt the service.

Table 2
Cyber Threats Overview

Cyber Threats		
	Threat	**Description**
CS1	GPS Spoofing	Manipulating users' GPS locations via the mobile application.
CT1	APK Repackaging	Modifying and redistributing the mobile application for malicious purposes.
CT2	Exploiting Vulnerabilities	Leveraging outdated libraries, weak cryptography, or insecure code to compromise the system.
CR1	Log Alteration	Tampering with logs to falsify records.
CR2	Ride Repudiation	Denying actions such as ride creation.
CI1	PII Extraction	Extracting Personal Identifiable Information or tracking data from a device's memory.
CI2	Excessive User Tracking	Monitoring users even when the app is unused in background.
CD1	Battery Info Manipulation	Tricking the server into misinterpreting vehicle battery status to disrupt the service.
CE1	Privilege Escalation	Gaining unauthorized access to high-level functions, such as unlocking vehicles without payment.

B. Detailed Field Test Procedures

This appendix provides the detailed description of the field tests we performed. For all tests involving active smartphone use, we consistently disabled SSL pinning using Frida and monitored the communications with Burp and Bluetooth HCI Snoop Log, even if not explicitly mentioned each time.

Table 3
Detailed Procedure Test GPS Spoofing #1

Smartphone GPS Spoofing #1	
Goal	Understand whether or not the services rely exclusively on the GPS information given by the user's smartphone.
Procedure	• Open the app and choose a vehicle • Move the vehicle while locked to another location • Check whether the position on the map changes or not
Interpretation	If the position changes, it means that the vehicle uses its own GPS tracker module. If not, it means that the service relies only on the locations provided by the client device.
Results	Every vehicle independently updated its location.
Observations	Almost every vehicle had an update time of 2-10 minutes. One app updated the position after about 30 minutes.

Table 4
Detailed Procedure Test GPS Spoofing #2

Smartphone GPS Spoofing #2	
Goal	Understand if the server acts according to fake data fed from the smartphone to the server during the ride.
Procedure	• Spoof the location of the smartphone onto the actual position • Open the app • Select and unlock a vehicle • Move slowly the false position outside the pedestrian zone (We did this with a plausible speed to avoid some plausible checks against an abrupt change of location) • Try the vehicle inside and outside the pedestrian zone
Interpretation	If the speed can go higher than 6km/h inside the pedestrian zone, then the service relies on fake data. Otherwise, the service uses the correct GPS data coming from the vehicle.
Results	Every target service maintained the correct behavior and enforced the correct speed limit.
Observations	In all the services, at the end of the ride, the application shows the correct route, even though the device position was elsewhere. We can deduce that the tested services completely ignore the client's position, at least in important matters.

Table 5
Detailed Procedure Test GPS Spoofing #3

Smartphone GPS Spoofing #3	
Goal	Understand if the server acts according to fake data fed from the smartphone to the server at the end of the ride.
Procedure	• Spoof the location of the smartphone on the actual position • Open the app • Select and unlock a vehicle • Move the vehicle inside a no-parking zone • Try to lock the vehicle
Interpretation	If it's possible to lock the vehicle, then the service trusts the fake data coming from the client. Otherwise, it uses the GPS data coming from the vehicle.
Results	Every target service maintained the correct behavior and refused to lock the vehicle in a forbidden area.

Table 6
Detailed Procedure Test Memory Analysis

Memory Analysis	
Goal	Verify whether or not the client application exposes personal information or tracking data in the phone memory.
Procedure	• Download the application • Register to the service • Use the service at least once, to create tracking data • Snapshot the memory and explore it
Results	One-third of the applications did not expose any relevant data, one-third exposed partial tracking information, and the remaining exposed PII and partial tracking information.
Observations	All the target applications did not save important information on public memory, so an attacker cannot access them from a malicious application. The attacker would need to infect the OS of the target device or have physical access to the data.

Table 7
Detailed Procedure Test User Tracking

User Tracking	
Goal	Understand whether or not the target applications trace the user even when the application is running in the background and has no active rides.
Procedure	• Open the app and ensure it is sending real-time location to the server • Move the app to the background • Check whether it keeps sending real-time position to the server or not
Results	Only one application kept sharing the user's location with the server.

Table 8
Detailed Procedure Test Illicit Unlock #1

Illicit Unlock in Decentralized Architectures #1	
Goal	Verify whether the unlock token is fixed.
Procedure	• Open the app • Choose a vehicle, start a ride, and end it • Choose another vehicle, start a ride, and end it • Compare the tokens
Results	As expected, the token changes for different vehicles.

Table 9
Detailed Procedure Test Illicit Unlock #2

Illicit Unlock in Decentralized Architectures #2	
Goal	Verify whether the unlock token is fixed for the vehicles.
Procedure	• Open the app and choose a vehicle • Start and end a ride • Start and end a second ride • Compare the tokens
Results	The token changes at each usage.