E-Spoofer: Attacking and Defending Xiaomi Electric Scooter Ecosystem

WiSec 2023, Guildford, Surrey (UK), May 29 - June 01

Marco Casagrande*, Riccardo Cestaro, Eleonora Losiouk, Mauro Conti, and Daniele Antonioli*
Motivation

- **E-scooters** are a critical *wireless* attack surface
  - Security (theft), privacy (data leak), safety (break)
- We know little about their security mechanisms
  - Proprietary, undocumented, untestable
- Millions of e-scooters and users
  - Controlled by a couple of companies (e.g., **Xiaomi**)
- One attack on Xiaomi has a huge impact
  - E.g., 2019 Zimperium remote braking system exploit ([ref](#))
Contributions

- RE all **Xiaomi e-scooter protocols** since 2016
  - Pairing and Session phases
- Uncover critical **protocol-level vulnerabilities**
  - E.g., unauthorized pairing, no password enforcement
- **Proximity** and **remote** wireless attacks
  - Malicious Pairing (MP), Session Downgrade (SD)
- **E-Spoofer** open-source toolkit
  - Reproduce the attacks, tamper with protocols
- **Countermeasures** and **disclosure** to Xiaomi
System Model

Xiaomi E-scooter

Xiaomi Protocols
Over BLE

Mi Home
(Android, iOS)
Password-protected

User Phone

Software-lock
Attacker Models

**Proximity Attacker**

- Xiaomi Protocols Over BLE
- E-scooter

**Remote Attacker (Android app)**

- User Phone
- Xiaomi Protocols Over BLE
Attacker Goals

**Spoof** Mi Home to the e-scooter. Send arbitrary and unauthorized **read** and **write** commands without user consent and notice.
Xiaomi E-Scooter Protocols Introduction

- **P1, P2, P3, P4 (since 2016)**
  - Application-layer Pairing and Session phases
  - No BLE link-layer security

- **Pairing** phase
  - Devices agree on a **Pairing Key** *(PK)*

- **Session** phase
  - Devices compute a **Session Key** *(SK)* from PK
  - Devices use SK to establish a secure channel
P1: No Security Mechanisms

NO PAIRING

NO SESSION

BLE Connection Req
BLE Connection Resp
Read/Write Req
Read/Write Resp

Proprietary BLE GATT characteristics and binary data over GATT
P1: Proximity/Remote Attacks

Spoofed BLE Connection Req → BLE Connection Resp → Arbitrary Read/Write Req → Read/Write Resp

NO PAIRING
NO SESSION

Read all data: password, mileage.
Write all data: lock/unlock, password, max speed.
P2: Public PK and XOR Obfuscation

BLE Connection Req
BLE Connection Resp
Read pk_characteristic
PK [24 B]

Read/Write Req XOR PK
Read/Write Resp XOR PK

Characteristic exposes new PK every boot.
Obfuscation by XORing PK.
P2: Proximity/Remote Attacks

PAIRING

Spoofed BLE Connection Req

BLE Connection Resp

Read pk_characteristic

PK [24 B]

SESSION

Arbitrary Read/Write Req XOR PK

Read/Write Resp XOR PK

PK exposed publicly.

No encryption but XOR obfuscation.
P2: Proximity/Remote Attacks

MALICIOUS PAIRING

Spoofed BLE Connection Req

BLE Connection Resp

Read pk_characteristic

PK [24 B]

SESSION

Arbitrary Read/Write Req XOR PK

Read/Write Resp XOR PK

PK exposed publicly.

No encryption but XOR obfuscation

E-Spoof: Attacking and Defending Xiaomi Electric Scooter Ecosystem
P2: Proximity/Remote Attacks

- Spoofed BLE Connection Req
- BLE Connection Resp
- `p2_session_downgrade_cmd`
- Arbitrary Read/Write Req
- Read/Write Resp

Session downgraded to P1 (more insecure).
P2: Proximity/Remote Attacks

SESSION DOWNGRADE

Spoofed BLE Connection Req
BLE Connection Resp
p2_session_downgrade_cmd
Arbitrary Read/Write Req
Read/Write Resp

Session downgraded to P1 (more insecure).
P3: Const PK and XOR Obfuscation

PK = AES-ECB(key=const, inp=esc_name)

PK derived from esc_name, and const from Mi Home.

E-Spoof: Attacking and Defending Xiaomi Electric Scooter Ecosystem
P3: Proximity/Remote Attacks

PAIRING

esc_name

Spoofed BLE Connection Req

BLE Connection Resp

PK = AES-ECB(key=const, inp=esc_name)

SESSION

Arbitrary Read/Write Req XOR PK

Read/Write Resp XOR PK

PK derived from publicly known data.

RE const from Mi Home.
Recap: P1, P2, P3 insecurity

- **P1, P2, P3** are **insecure by design**
  - Security through obscurity (XOR, public seeds, binary data)
  - **Proximity/remotef impersonation is trivial**
- **P4** to the rescue?
  - **NOT** really
**P4: Pairing (ECDH, AES-CCM)**

**PAIRING**

- $\text{chal}, \text{PUB}_e$
- $\text{PUB}_a$

**SS** = ECDH($\text{PUB}_e, \text{PRV}_a$ or $\text{PUB}_a, \text{PRV}_e$)

**PK||OTK** = HKDF(key=SS, inp=const)

**resp** = AES-CCM(key=OTK, inp=chal)

ECDH to generate PK from SS.

App auth using OTK.
Unauthorized and non authenticated pairing.

\[
\text{SS} = \text{ECDH}(\text{PUB}_e, \text{PRV}_a \text{ or } \text{PUB}_a, \text{PRV}_e)
\]
\[
\text{PK}||\text{OTK} = \text{HKDF}(\text{key=SS,inp=const})
\]
\[
\text{resp} = \text{AES-CCM}(\text{key=OTK,inp=chal})
\]
P4: Proximity/Remote Attacks

Malicious Pairing

\[ SS = \text{ECDH}(PUB_e, PRV_a \text{ or } PUB_a, PRV_e) \]
\[ PK || OTK = \text{HKDF}(key=SS, inp=\text{const}) \]
\[ \text{resp} = \text{AES-CCM}(key=OTK, inp=chal) \]

Unauthorized and non authenticated pairing.
P4: Session (HKDF, AES-CCM) (1)

Random challs.
Directional SKs and IVs.

$SK_e || SK_a || IV_e || IV_a = HKDF(key=PK, inp=const, salt=rand_e || rand_a)$
P4: Session (HKDF, AES-CCM) (2)

**resp_a** = HKDF(key=SK_a, inp=rand_a||rand_e)

**resp_e** = HKDF(key=SK_e, inp=rand_e||rand_a)

AES-CCM(key=SK_a, IV=IV_a||cnt, inp=Read/Write Req)

AES-CCM(key=SK_e, IV=IV_e||cnt, inp=Read/Write Resp)

Mutual session auth.

Encrypted and integrity protected comms.
P4: Proximity/Remote Attacks

Session downgraded to P3 (more insecure).

PK = AES-ECB(key=const, inp=esc_name)

Arbitrary Read/Write Req XOR PK

Read/Write Resp XOR PK
P4: Proximity/Remote Attacks

Session downgraded to P3 (more insecure).

$PK = \text{AES-ECB}(key=\text{const}, inp=\text{esc\_name})$
Implementing the attacks: E-Spoofe

- **E-Spoofe** is open-source
  - Automated Proximity MP ([link](#))
  - Automated Remote SD ([link](#))
- **Reversed BLE firmware** on Ghidra
- Xiaomi protocol **dissectors**
- **Frida hooks** for Mi Home crypto calls
- WiSec Artifact approval
Evaluation Setup

- 5 BLE boards (M365, Pro 1, Pro 2, Essential, Mi 3)
- 8 BLE firmware (P1, P2, P3, P4)
## Evaluation Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M365</td>
<td>M365</td>
<td>072</td>
<td>P1</td>
<td>RE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M365</td>
<td>M365</td>
<td>081</td>
<td>P2</td>
<td>RE, MP, SD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M365</td>
<td>Pro 1</td>
<td>090</td>
<td>P3</td>
<td>RE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M365</td>
<td>M365</td>
<td>122</td>
<td>P4v1</td>
<td>RE, MP, SD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M365</td>
<td>Pro 2</td>
<td>129</td>
<td>P4v1</td>
<td>RE, MP, SD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Essential</td>
<td>Essential</td>
<td>152</td>
<td>P4v1</td>
<td>RE, MP, SD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mi 3</td>
<td>Mi 3</td>
<td>153</td>
<td>P4v1</td>
<td>RE, MP, SD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mi 3</td>
<td>Mi 3</td>
<td>157</td>
<td>P4v2</td>
<td>RE, MP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Countermeasures

- Update firmware via Mi Home
  - From P1, P2, P3 to P4v1 or P4v2

- **Password-protected and authorized Pairing**
  - Addresses MP on P4v1 and P4v2
  - More details in Section 8.1

- **Anti-downgrade patching script for BLE fw**
  - Addresses SD on P4v1
  - Evaluated on a real M365
  - More details in Section 8.2
Conclusion and Q&A

- RE all **Xiaomi e-scooter protocols** since 2016
  - Pairing and Session phases
- Uncover critical **protocol-level vulnerabilities**
  - Unwanted pairing, weak authentication
- **Proximity** and **remote** wireless attacks
  - Malicious pairing, session downgrade
- **E-Spoofer** open-source toolkit
  - Reproduce the attacks, tamper with protocols
- **Countermeasures** and **disclosure** to Xiaomi