The Rentable IoT Meltdown: Mass-Scale Hijacking of Shared Mobility and EV-Charging Fleets

Speaker: Hetian Shi
Tsinghua Univerisity

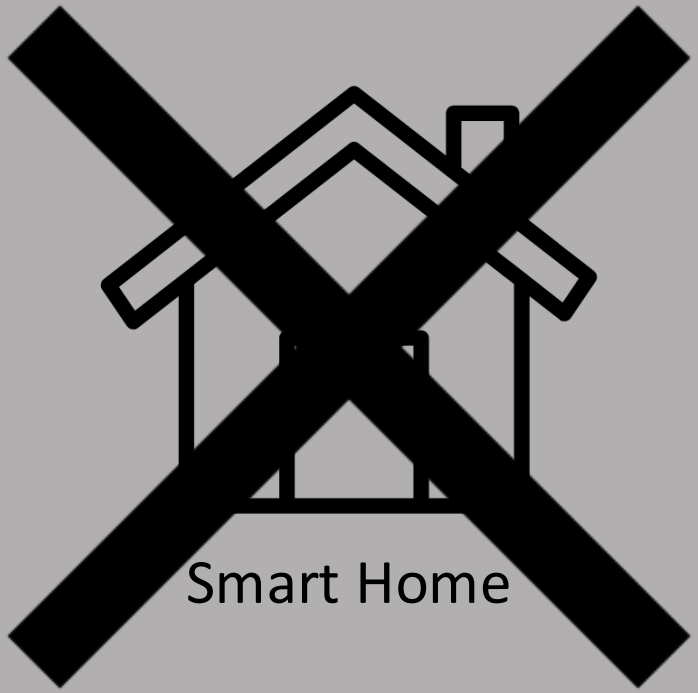
About Me

- Hetian (Mori) Shi
  - Researcher, Tsinghua University
  - Specializing in hardware and IoT security, with a focus on uncovering physical-layer vulnerabilities in IoT devices.
  - Achievements include awards in security competitions, including Tianfu Cup 2021, GeekPwn, and GeekCon 2022 and 2023.
  - Also an invited speaker at industry conferences, such as HITB2024 and CanSecWest 2025.

Agenda

- **Background**: Why Rentable IoT is Different
- Threat Model and Workflow
- **Hardware RE**: Recovering Protocols, Keys
- **Phantom Clients**: Becoming the Device
- **App/API Flaws**: Abusing Users and Devices
- **IDScope**: Turning Local Bugs into Fleet-Scale Attacks
- Live Demonstration
- **Defense Strategies**: What Vendors Must Fix
- Conclusion & Takeaways

What Is IoT Security?



Why Rentable IoT Is a Different Security Problem

shared e-Scooter/e-Bicycles

shared EV chargers

shared umbrella

shared cars

shared power bank

shared wash machine

Public-facing, centrally managed, and physically consequential

These devices are controlled by cellular IoT controllers

These devices are controlled by cellular IoT controllers

Backend Server: send control commands

Base Station: provide cellular network

Cellular IoT Controller: receive commands to control device

Battery Lock: start or stop the e-scooter

Control commands are sent via the cellular network

The smart lock can start or stop the device based on control commands

These devices are controlled by cellular IoT controllers

Backend Server: send control commands

Base Station: provide cellular network

Cellular IoT Controller: receive commands to control device

Battery Lock: start or stop the e-scooter

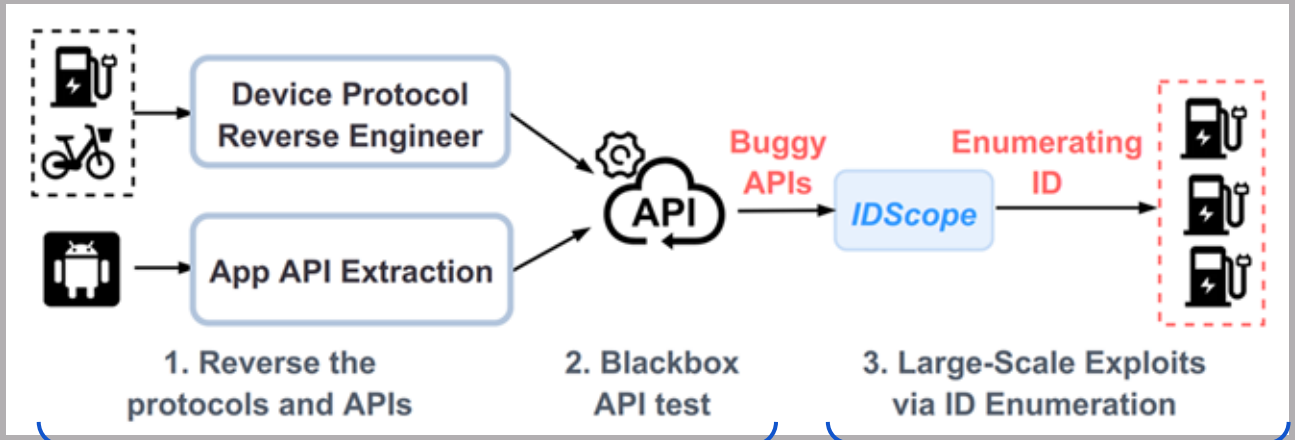
Control commands are sent via the cellular network

The smart lock can start or stop the device based on control commands

User app, Backend server, Cellular IoT controller, and Physical lock/charger

Workflow for analyzing these devices

1. Reverse the device protocols and app APIs
2. Test for access control and authentication flaws
3. Check whether the flaws scale via ID enumeration



Phase-1: identifying vulnerabilities in device-to-server access control and in app-to-server access control

Phase-2: verifying if these vulnerabilities can lead to large-scale exploitations

Weak device identifiers can become an attack vector

1. User scans QR code to rent an e-bike

User

https://e-sooter.rent/d/**970-283**

2. Server finds the right device by device serial number

Server

Device

3. Server unlock device after authenticating user and device

a. User-Server Authenticate: **server** check if user is valid (paid) based on **user ID**

b. Find Device: server find and check the device by **device serial number**

c. Device-Server Authentication: server authenticate the device connection using the **device ID and access key**

Devices are identified by their **device serial number** (in the QR code)

Hardware Vulnerabilities Make RE Easy

Device Type	Device	#Device Deployed	Main MCU	4G MCU	OS	#Func	Encryption Algorithm	Protocol	#CMD/ Traces	Can Debug	Extract Firmware	Phantom Client
EV Charger	Teld	100k+	STM32F207	EC200N	Baremetal	1884	AES	Binary	68/50	✓	✓	✓
	Starcharge	100k+	LPC1778FBD144	ME3630	FreeRTOS	1030	3DES	Binary	66/63	✓	✓	✓
	Potevio	30k+	GD32F407	N58	μC/OS-II	1325	∅	Binary	56/73	✓	✓	✓
Charger	Xlvren (Charger)	100k+	HC32F460	A7670	ZephyrOS	1718	∅	MQTT	34/79	✓	✓	✓
	Lvcc (Charger)	100k+	FM15F366	CUIoT	Baremetal	711	∅	Binary	42/88	✓	✓	✓
	Xlvren (Cabinet)	100k+	STM32F407	SIM-7600CE	eCos	1899	∅	Binary	46/68	✓	✓	✓
	Lvcc (Cabinet)	100k+	FM15F366	EC200N	Baremetal	726	∅	Binary	36/44	✓	✓	✓
	Lvcc (Socket)	100k+	∅	Air720UH	FreeRTOS	-	∅	Binary	18/60	✗	✗	✓
	JieDian	1,000k+	S34ML02G2	N58	Linux	-	TLS	HTTPS	7/288	✗	✗	✗
Mobility	QiXin	50k+	∅	EC200U	-	1645	∅	Binary	24/40	✓	✓	✓
	MeiTuan	10M+	STM32F401	EC200N	FreeRTOS	-	AES	Binary	2/6	✗	✗	✗
Entertainment	YFLe	10k+	∅	EC100N-CN	-	-	∅	Binary	7/25	✗	✗	✓
	BDT	50k+	∅	SIM-A7670C	-	-	∅	JSON	10/32	✗	✗	✓
	WeiPeng	100k+	∅	Air724UG	FreeRTOS	-	∅	Binary	15/48	✓	✗	✓
Tools	AnSheng	10k+	∅	Air780E	FreeRTOS	-	∅	Binary	8/22	✓	✗	✓
	AnKong	10k+	∅	ML307A	-	-	∅	MQTT	8/28	✗	✗	✓
	WZ-Cloud	5k+	∅	Air724UG	FreeRTOS	-	∅	Binary	6/25	✓	✗	✓

Hardware Vulnerabilities Make RE Easy

Encryption Algorithm
AES
3DES
∅
∅
∅
∅
∅
∅
TLS
∅
AES
∅
∅
∅
∅
∅
∅

Protocol	#CMD/ Traces	Can Debug	Extract Firmware
Binary	68/50	✓	✓
Binary	66/63	✓	✓
Binary	56/73	✓	✓
MQTT	34/79	✓	✓
Binary	42/88	✓	✓
Binary	46/68	✓	✓
Binary	36/44	✓	✓
Binary	18/60	✗	✗
HTTPS	7/288	✗	✗
Binary	24/40	✓	✓
Binary	2/6	✗	✗
Binary	7/25	✗	✗
JSON	10/32	✗	✗
Binary	15/48	✓	✗
Binary	8/22	✓	✗
MQTT	8/28	✗	✗
Binary	6/25	✓	✗

Most devices do not encrypt their network messages

The debug ports on many devices are not protected, allowing attackers to extract firmware and network traffic logs

Hardware Vulnerabilities Make RE Easy

Encryption Algorithm
AES
3DES
∅
∅
∅
∅
∅
∅
TLS
∅
AES
∅
∅
∅
∅
∅
∅

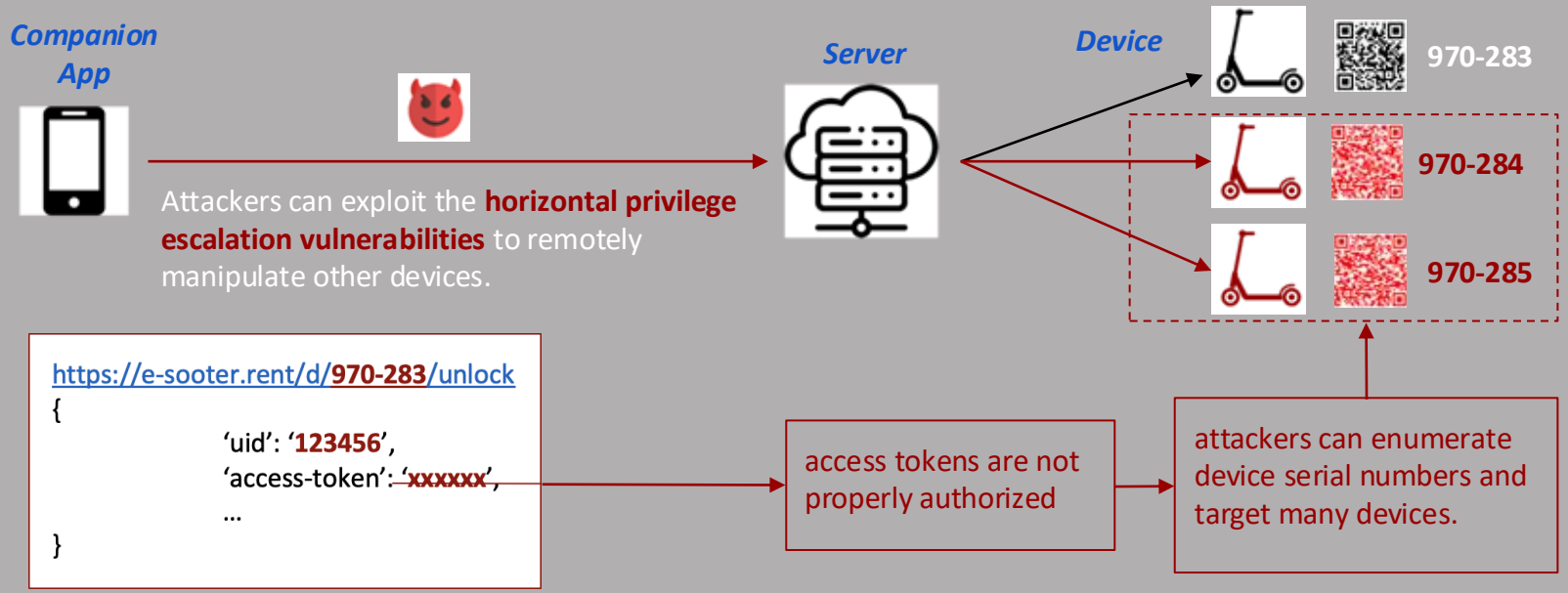
Protocol	#CMD/ Traces	Can Debug	Extract Firmware
Binary	68/50	✓	✓
Binary	66/63	✓	✓
Binary	56/73	✓	✓
MQTT	34/79	✓	✓
Binary	42/88	✓	✓
Binary	46/68	✓	✓
Binary	36/44	✓	✓
Binary	18/60	✗	✗
HTTPS	7/288	✗	✗
Binary	24/40	✓	✓
Binary	2/6	✗	✗
Binary	7/25	✗	✗
JSON	10/32	✗	✗
Binary	15/48	✓	✗
Binary	8/22	✓	✗
MQTT	8/28	✗	✗
Binary	6/25	✓	✗

Most devices do not encrypt their network messages

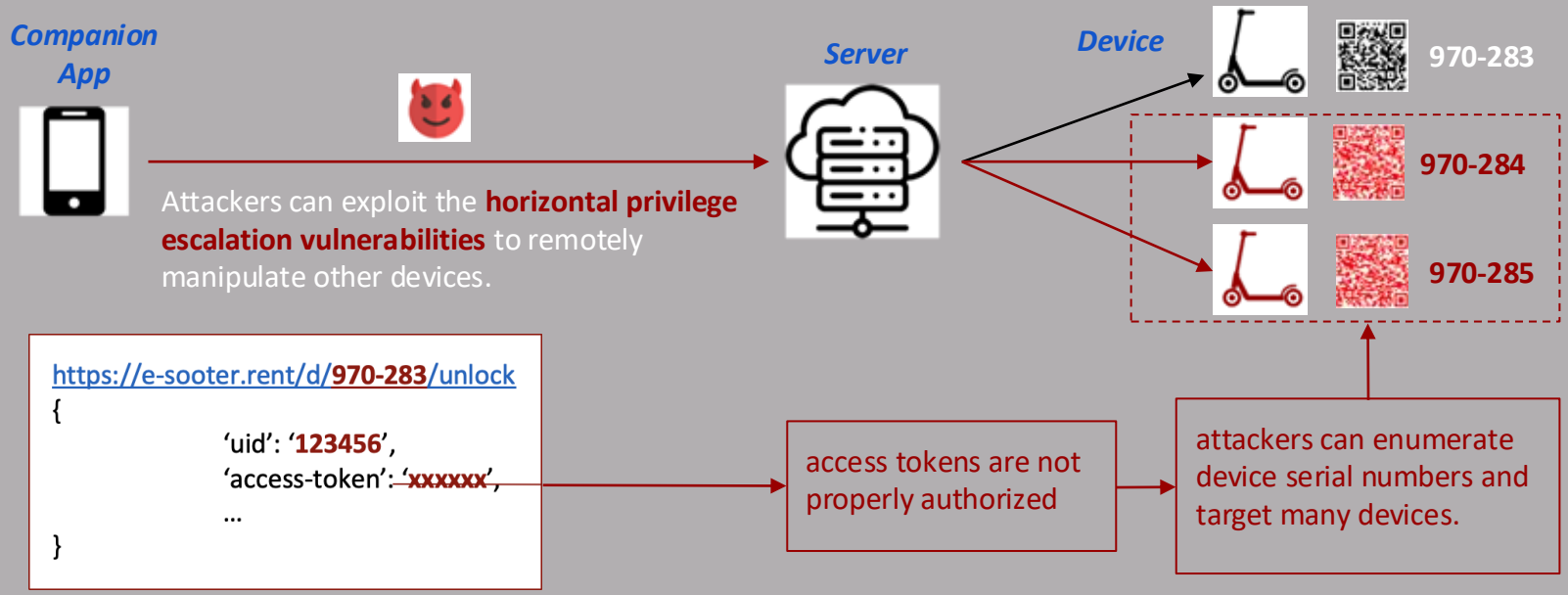
The debug ports on many devices are not protected, allowing attackers to extract firmware and network traffic logs

Summary: most devices have no hardware protections, and attackers can easily analyze their firmware and network traces. Moreover, a significant number of devices **hardcode a common key** into their firmware for authentication.

Why Enumeration Changes the Game



Why Enumeration Changes the Game



Attackers can exploit vulnerable app-server APIs to manipulate victim devices by **guessing their IDs** (device serial numbers).

Study Scope

1. Finding real-world products as research targets

17 physical devices

- leading EV-Chargers
- Chargers: E-bike/mobile phone charging outlets
- Tools: washing machine
- Mobility: shared bicycle (e.g., Meituan)

92 mobile apps

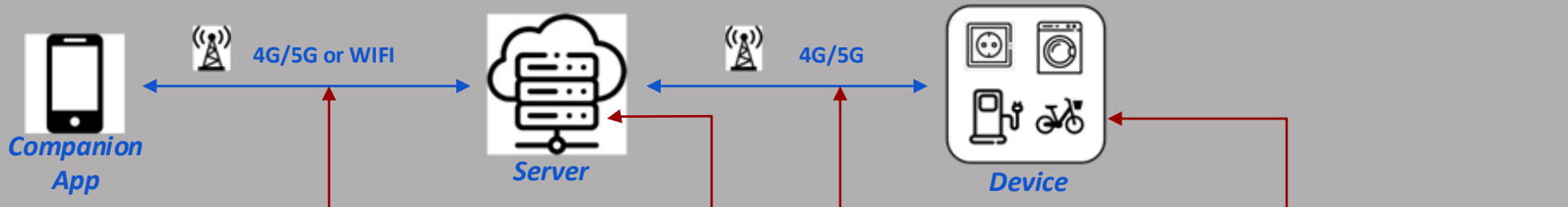
- the apps/WeChat mini-programs of 17 devices in China
- 64 WeChat mini-programs (or Android apps) in China
- 11 iOS apps for e-scooters in Europe

2. Checking if these devices are vulnerable to large-scale remote attacks

Step-1: Identify vulnerabilities in apps and devices APIs.

Step-2: Verify if these vulnerabilities can be exploited on a large scale by enumerating the IDs.

Threat Model



Exploiting the app-server interaction: attackers can exploit the app APIs to send unauthenticated (or unauthorizated) requests to harm other users or devices.

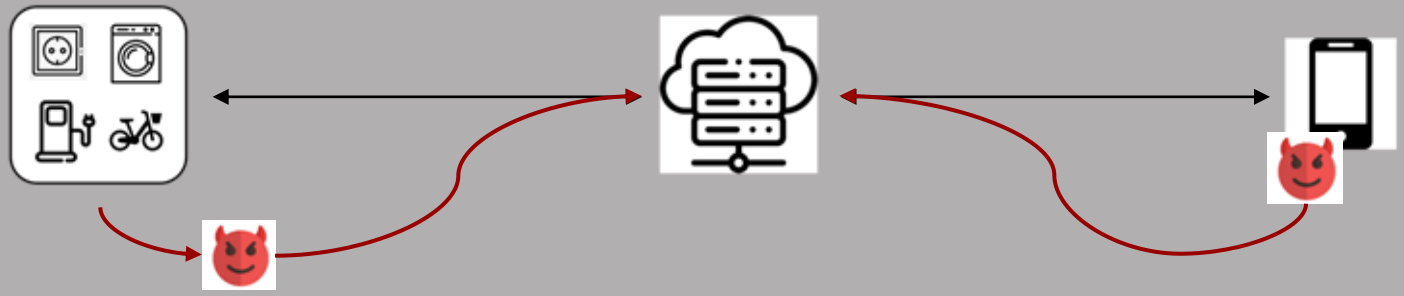
Exploiting the device-server interaction: attackers can impersonate the device to connect to the server and manipulate device states or inject malicious messages.

Exploiting the cellular network: attackers can intercept the cellular network communication between devices and server to inject malicious messages and manipulate devices.

Exploiting the local devices: attackers can physically tamper with the local devices' hardware (e.g., plant hardware trojan) to abuse them

Threat Model: attackers can only access the **mobile apps** and **limited devices**

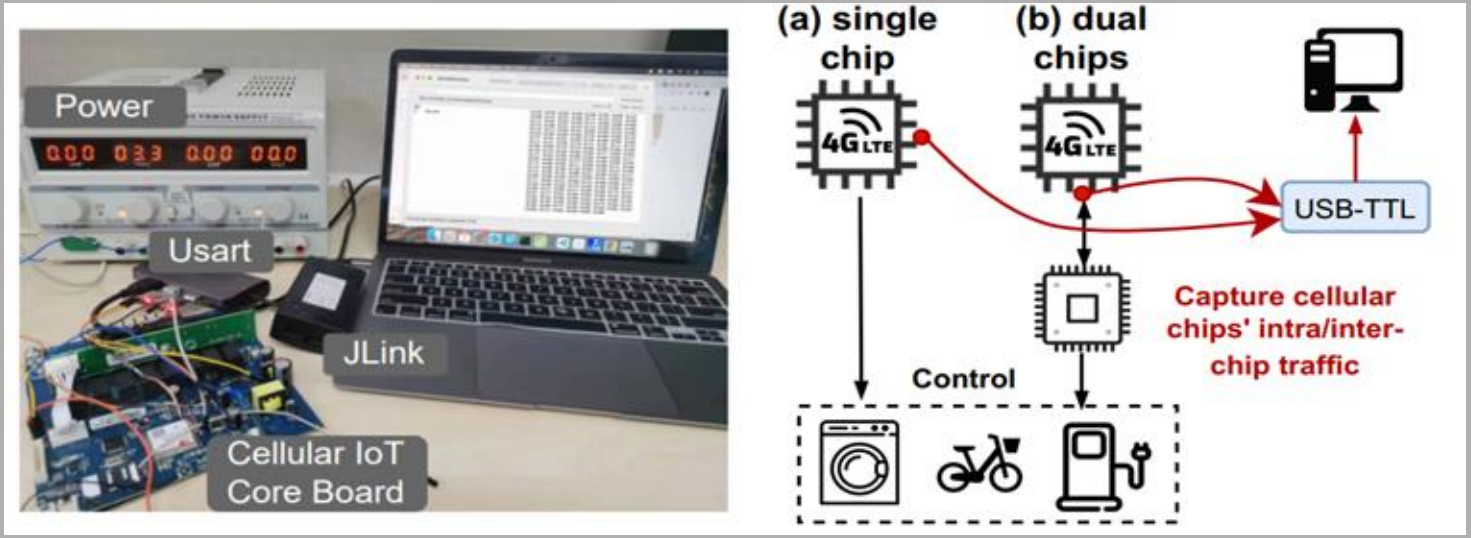
Phase-1: Identifying vulnerabilities in devices and apps



Device side: forge device-to-server messages

App side: forge app-to-server messages

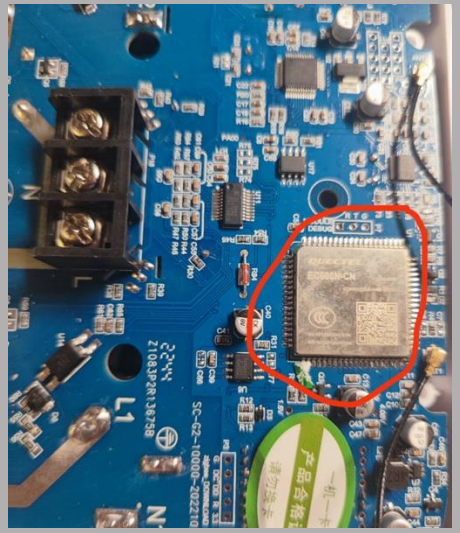
Hardware RE



Step-1: gain hardware access, extract firmware, and capture cellular traffic

Even without exposed debug headers, the device can often still be instrumented through chip-level access.

Hardware RE - No Debug Port, Still Debuggable



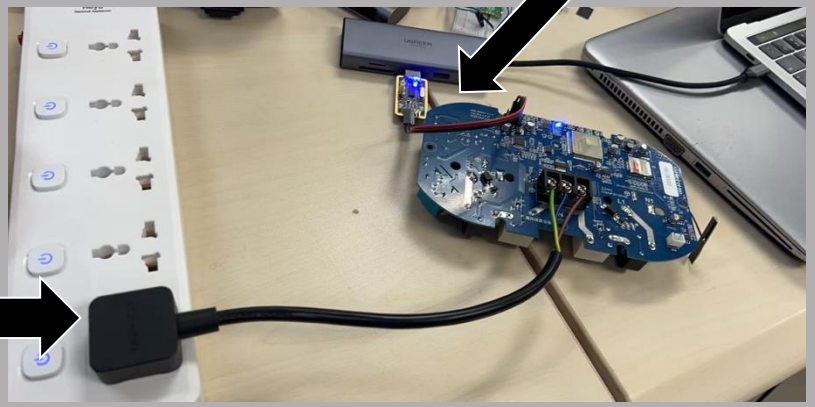
Firmware

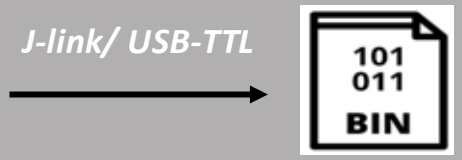
Network Trace

→ **Fly-wire with JTAG / SWD / UART**
→ **Live access to firmware and boot process**

Many products ship without an obvious debug connector, but that does not mean the device is no longer debuggable.

Hardware RE – Power-isolated Live Debugging



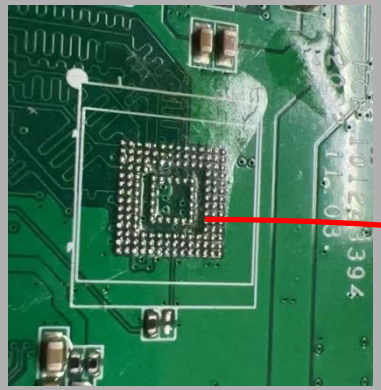


We do not need to power the full system.

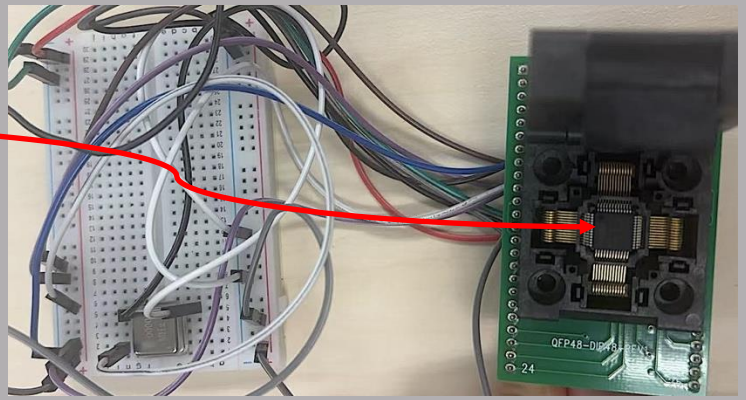
Benefits

- avoids high-voltage risk
- improves stability for boot tracing
- enables live debugging and runtime extraction

Hardware RE – Chip-off and Minimal Runnable Fixtures



If SWD/Jtag is disabled or read-protected !

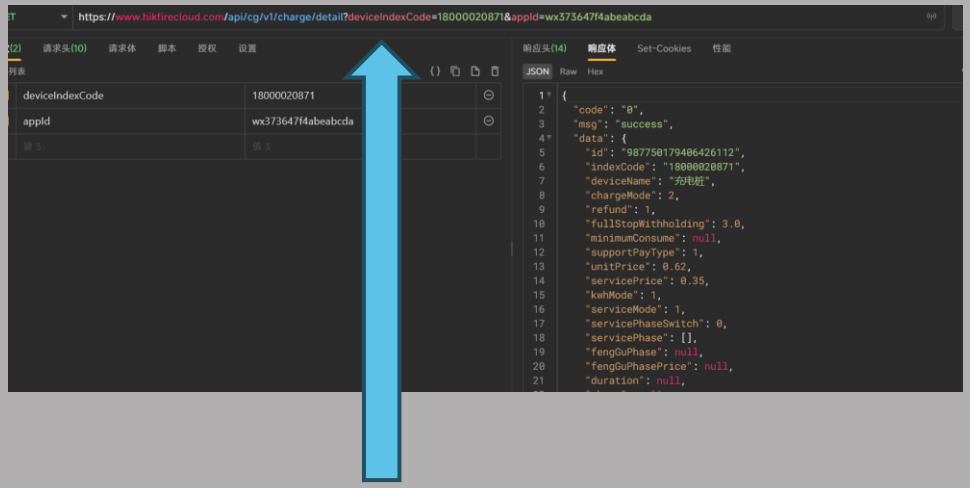


Minimal Runnable Fixture

The real barrier is not the missing header, but whether the chip can still be physically reached and electrically reconstructed.

Hardware RE – Recovered Firmware Exposes Protocol & Cloud Trust Material

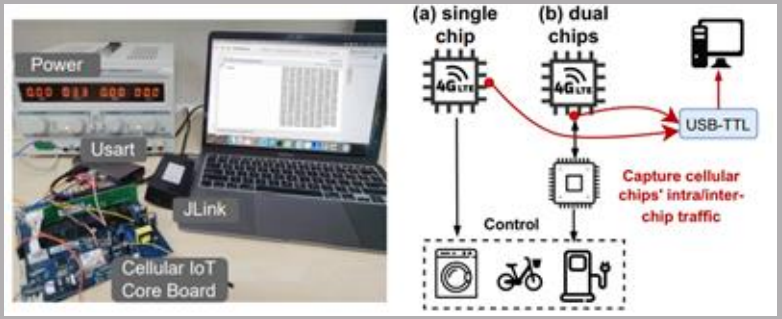
Firmware artifacts



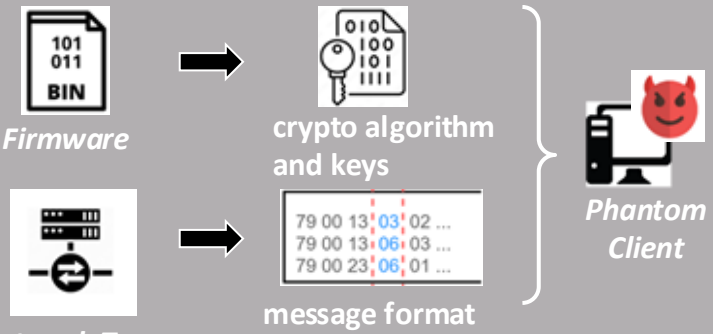
Recover message format, device IDs, key secrets and backend point

Hardware reverse engineering becomes the entry point to cloud reverse engineering.

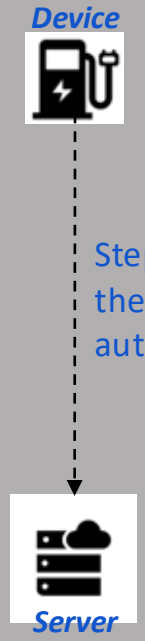
Hardware RE



Step-1: extract firmware and capture cellular traffic

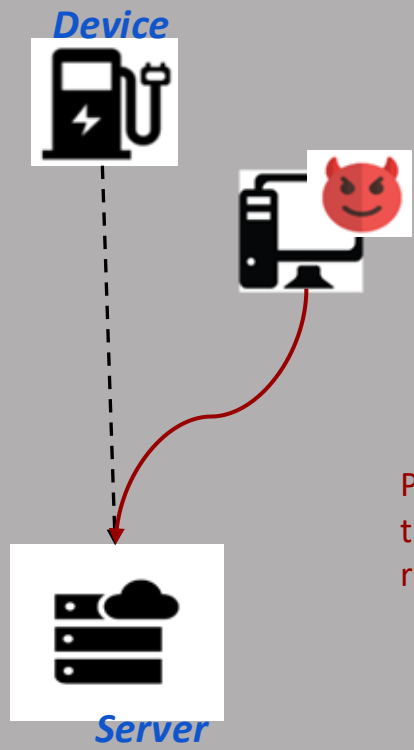


Step-2: reverse the protocol to reimplement a phantom client that can replicate device's requests



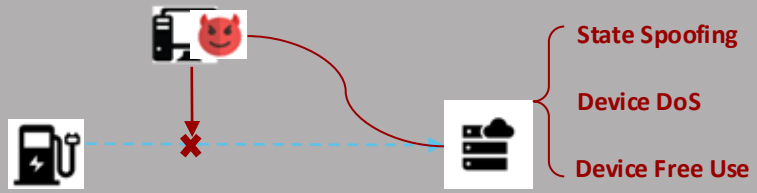
Step-3: use phantom clients to test if the server correctly authenticates and authorizes the devices' requests

Phantom Clients: Becoming the Device



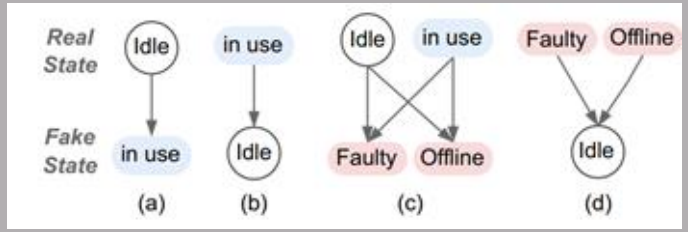
Phantom clients reconnect to the server and send forged requests to test the server

Hazards of these vulnerabilities



Phantom clients can reuse real devices' IDs and keys to send forged requests and launch various attacks

1. **State spoofing**: attackers can use phantom clients to manipulate the state of devices, hindering both users and managers.



2. **Device DoS:** attackers can use phantom clients to reconnect to the server and make the real devices **disconnect** from the server and become unusable.

3. **Free Use:** attackers can use phantom clients to stop billing while the real devices remain unlocked.

Hazards of these vulnerabilities

Device Type	State Spoofing (14)	DoS (5)	Free Use (4)
EV Charger	Teld, Starcharge Potevio	Teld, Starcharge Potevio	Teld
Charger	Lvcc (Socket) Lvcc (Charger), Xlvren (Charger) Lvcc (Cabinet), Xlvren (Cabinet)	Xlvren (Cabinet)	
Mobility	QiXin	QiXin	QiXin, MeiTuan
Entertainment	BDT, WeiPeng		
Tools	AnKong, AnSheng WZ-Cloud		AnKong

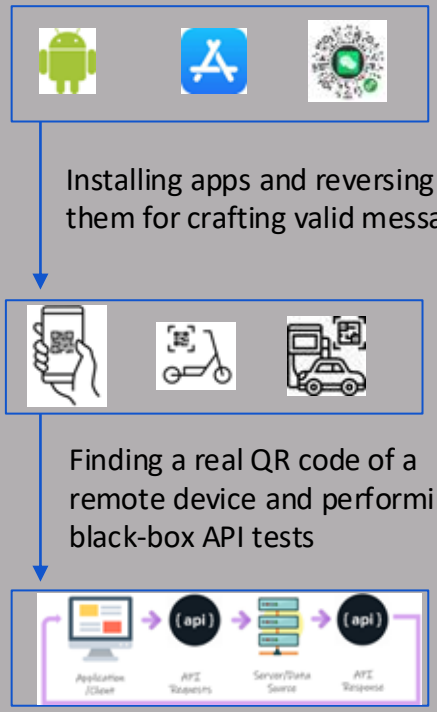
Summary of vulnerable devices

Identifying vulnerabilities in companion apps

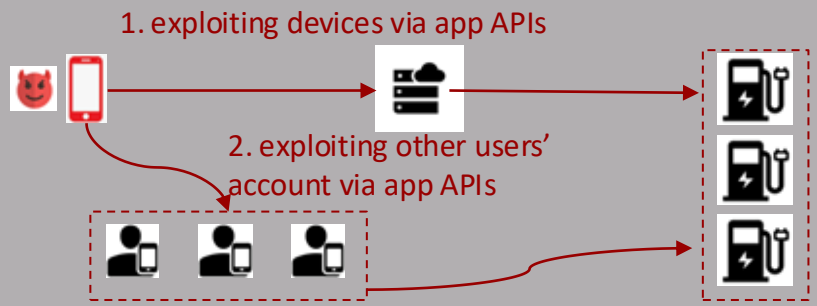
We collected 81 Chinese apps (including 6 Android apps and 75 WeChat mini-programs) and 11 European iOS apps.

We reversed the apps and performed black-box tests.

Device Type	Vendor
EV Chager	**Teld**(特来电) **Potevio**(普天新能源) **StarCharge**(星星充电) EVChargeNetwork(电动车充电网) Sunmue(尚e充电)
Charger	ShanKaiCharge(闪开来电) UUCharge(UU充电) MamCharge(猛犸充电) YunAn(世纪云安) NBLinks(涌鑫充电) EnergyMonster(怪兽充电） Dian(小电充电) DailyCharge(天天充电) BlueCatCharge(蓝猫共享充电) PisenPower(闪葱共享充电宝) QuQingTingCharge(趣蜻蜓共享充电) HeiQingTingCharge(黑蜻蜓充电) XiaoXunCharge(小巡共享充电) LuLuChong(路路充) TowerEnergy(铁塔充电) FlyPower(飞天鹰扫码充电器) QQCharging(千牛充电) DingDingCD(叮叮充电) Lvcc(驴充充) JieDian(街电) Xlvren(小绿人充电) BaJieCharge(八戒充电) DuDuBox(嘟嘟充电) BeiDian(倍电科技) ZhouDian(昼电共享)
Mobility	**Lyft Lime Bird Spin Dott Voi TIER Helbiz Neuron GO-ON Bolt**
	XiaoYuCX(小雨出行) KVCOOGO(快趣出行) XiaoMaCX(小玛电单) DiDiCX(滴滴出行) **HelloBike**(哈罗单车) **MeiTuanBike**(美团) FeiYueTu(飞跃兔出行) BossGo(Boss行) LeGeCX(乐哥出行) Liubike(小遛出行) LetFunGo(雷风出行) BaQiCX(巴骑出行) SunlightGo(阳光共享智行)MiMaDD(觅马出行QIQI(骑骑共享) XiaoHuangYaCX(小黄鸭共享) GongChi(共驰电单车) XiaoBeiCX(小呗畅行) XianLvGo(闲驴出行) **Hozonauto**(哪吒新能源) **DFPV**(东风新能源) QiXin(齐信共享单车) XiaoBinBike(小彬科技) ModaCX(摩达出行) YueHuoCX(月火出行) KeNaDianCX(克哪点出行) GXRongYi(共享荣熠惠行) YueShiJi(粤世纪共享)
Tools	Penguin Technology(企鹅共享) XiaoLianHB(智慧笑联) XiaoIZhiNengXiYi(小I智能洗衣) Smile-Iot(思迈尔智能) HQJL(华清捷利) Xiao-V(小v共享设备) AiPei(爱陪共享) WZ-Cloud(微众云) AnSheng(安圣科技) AnKong(安控)
Entertainment	AnMoJianKangGo(按摩健康GO) LeMoBar(乐摩吧) QSMX(骑思妙想) WeiPeng(微鹏) YFLe(易付乐) BDT(便电通)



Vulnerabilities in IoT companion apps



1. **Exploiting devices:** attackers can use vulnerable app APIs (e.g., vertical/horizontal privilege escalation) to exploit remote devices
   - bypass payment to freely use devices
   - abusing vulnerable device control APIs to unauthorized manipulate devices

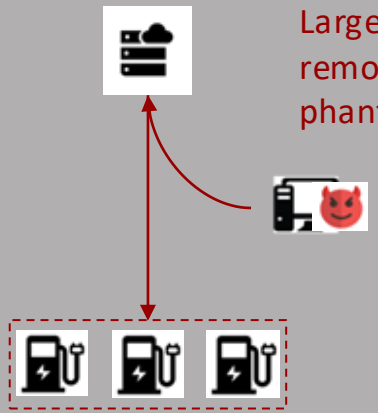
2. **Exploiting other users:** attackers can use vulnerable app APIs to access other users' resources (e.g., account, profile, devices)
   - steal other users privacy data
   - hijack other users' accounts to rent/return devices

Vulnerabilities in IoT companion apps

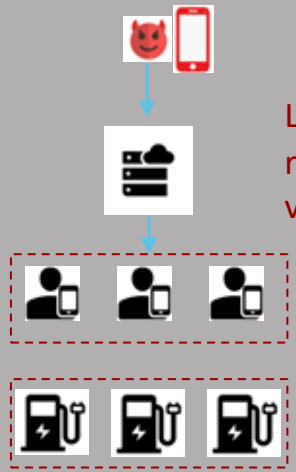
Device Type	Attacking Users		Abusing Devices	
	Privacy Leakage	Account Hijack	Payment Bypass	Manipulate Device
EV Charger	Sunmue, Potevio	Teld		
Charger	LuLuChong, NBLinks ZhouDian			LuLuChong, NBLinks
Mobility	YueHuoCX, YueShiJi GXRongYi	YueHuoCX, YueShiJi GXRongYi,DFPV	QiXin, GO-ON	YueHuoCX, YueShiJi GXRongYi, Hozonauto DFPV, Lime, Helbiz
Tools		AnSheng	HQJL	
Entertainment	BDT		QSMX, Dadaball	

Summary of vulnerable products

Phase-2: Can These Bugs scale?



Large-scale affecting remote device via phantom clients



Large-scale affecting remote device and users via companion app

Why short IDs can lead to large-scale attacks?

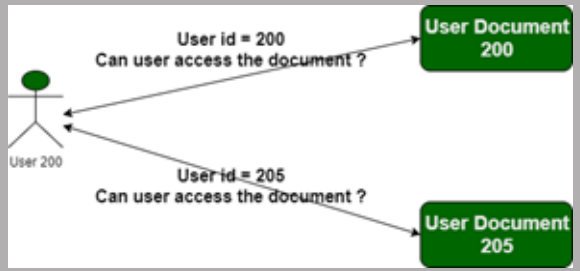
1. Weak device authentication exists in real products.

dev.login(server_ip, **dev_id**, **key**)

dev_id: can be enumerated

key: are shared by all devices

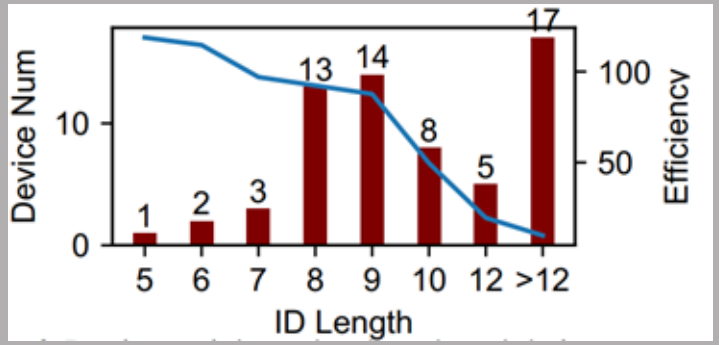
2. Insecure Direct Object References (IDOR)-style vulnerabilities are hard to avoid in app/backend services.



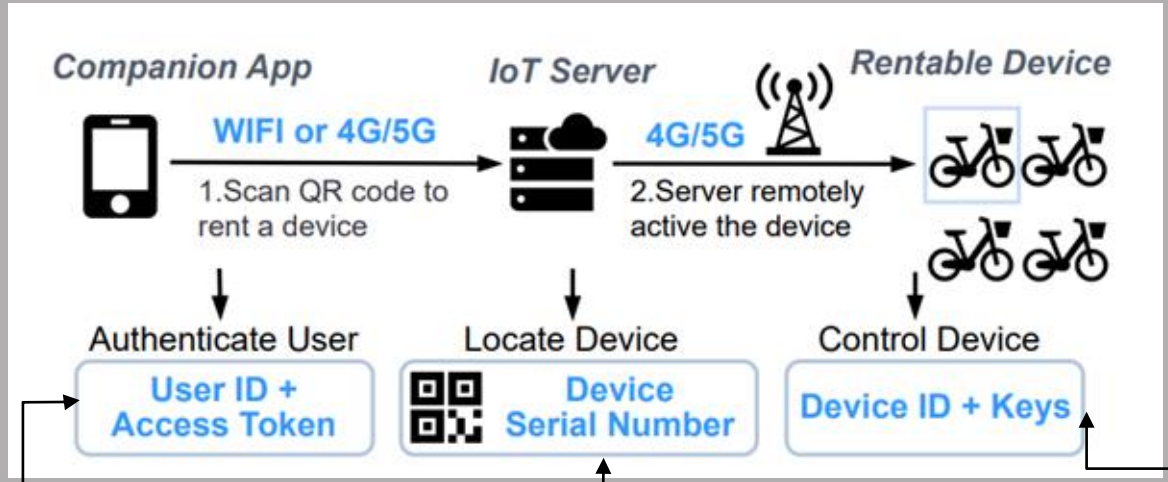
3. Large product fleets make numeric identifiers easy to enumerate.

#App	Service Area	Device Serial Number		User ID	
		#Enumerable	#Pattern	#Enumerable	#Pattern
81	China	48 (59.2%)	32 (39.5%)	14 (17.3%)	7 (8.6%)
11	Europe/US	8 (72.7%)	6 (54.5%)	- *	1 (9%)

* Fail to enumerate IDs due to not finding ID verification APIs.



Three types of identifiers used in rentable IoT products



1. **User ID** is used to authenticate and distinguish different users. Many products leak UIDs or use sequential numbers as UIDs

2. **Device Serial Number** is used for the QR code of a device and sometimes users need to type in this number to rent a specific device

3. **Device ID** is used by the device firmware to connect and register to the server. Server can then use it to authenticate different devices

IDScope: Weak IDs Turn Local Bugs into Fleet-scale Impact

Across real products, weak identifiers amplified device-side and app-side flaws into scalable abuse.

Device Type	Device	Obtain ID	State Spoofing	DoS	Free Use
EV Charger	Teld	A1 A2	✓	✓	✓
	Starcharge	A1 A2	✓	✓	
	Potevio	A2	✓	✓	
Charger	Xlvren(Charger)	A1 A2	✓		
	Lvcc(Charger)	A1 A2	✓		
	Xlvren(Cabinet)	A1 A2	✓	✓	
	Lvcc(Cabinet)	A1 A2	✓		
	Lvcc(Socket)	A1 A2	✓		
Mobility	QiXin	A2	✓	✓	✓
Entertainment	BDT	A1	✓		
Tools	AnKong	A1	✓		✓
	AnSheng	A1	✓		
	WZ-Cloud	A1	✓		

Device Type	Vendor	ID Emueration		Large-Scale Privacy Leakage	Large-Scale Device Free Usage		
		Uid (Len)	SN (Len)		Account Hijack	Payment Bypass	Manipulate Devices
EV Charger	Teld	✗(14)	✗(13)		✓[1]		
	Sunnue	✗(15)	✓(10)	✓[1]			
Charger	LuLuChong	✓(11)	✓(5)	✓			✓
	NBLinks	✓(9)	✓(9)	✓			✓
	ZhouDian	✓(7)	✓(13)	✓			
Mobility	YueHuoCX	✓(7)	✓(10)	✓	✓		✓
	GXRongYi	✗(24)	✓(9)				✓
	YueShiJi	✗(24)	✓(9)				✓
	Hozonauto	✗(11)	✓(17)				✓
	DFPV	✗(11)	✓(17)				✓
	QiXin	✓(7)	✗(12)			✓[2]	
	GO-ON	✗(16)	✓(6)			✓	
Tools	HQJL	✓(7)	✓(8)			✓	
	Ansheng	✓(9)	✓(9)		✓		
	Dadaball	✓(5)	✓(8)			✓	
Entertainment	QSMX	✓(32)	✓(11)			✓	
	BDT	✓(6)	✗(16)	✓			

EV Charging ：

- **Free charging**
- **Fleet-scale** denial of servisse
- Unauthorized charger manipulation

Shared Mobility:

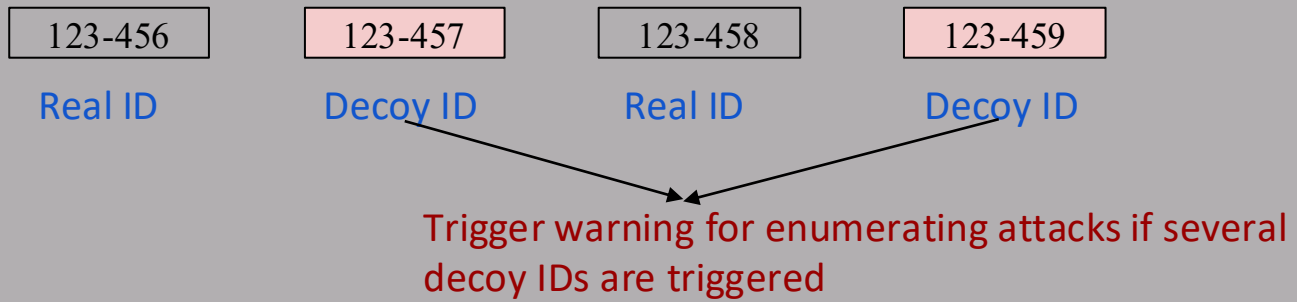
- Privacy leakage
- Account hijacking
- Unauthorized device control
- Free rides at scale

Live Demo I: Unauthorized charger activation or free-use

Live Demo II: Persistent DoS / forcing devices offline

Mitigation and Responsible Disclosure

- **Responsible Disclosure**
  - 23 vulnerabilities across 14 physical devices
  - 34 vulnerabilities within 23 apps
  - 18 CVE/CNNVD/NVDB
- **Mitigating Vulnerabilities**

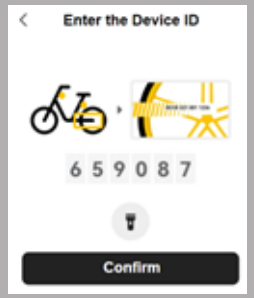


Ethical Considerations

- We only studied devices that we could lawfully obtain, either by purchase or through direct vendor cooperation.
- All experiments were conducted in controlled settings, without targeting real users or live deployments.
- We followed responsible disclosure practices and coordinated with affected vendors.

Discussion – Why Weak IDs and Common Keys Presist

- **Root Causes: Convenience Over Trust**
  - Short, human-friendly IDs were chosen for usability, not security.
  - Common keys were selected to simplify **device provisioning and backend coordination.**
  - At fleet scale, those convenience decisions became security liabilities.



Short IDs exist because humans need **to read/ type** them

Device ID

Device credentials must exist in multiple places, which pushes vendors toward simplified key management

Conclusion

- Rentable IoT is not just app–connected IoT—it is centrally controllable physical infrastructure.
- The real risk is scale: weak identifiers turn local bugs into fleet-wide attacks.
- Securing rentable IoT requires fixing identifiers, device identity, and backend authorization together.

Takeaways

1. Rentable IoT can enable end-to-end compromise, where device-side weaknesses, backend flaws, and weak identifiers combine into fleet-wide abuse.
2. Predictable identifiers are the key scale amplifier that turns local bugs into fleet-wide attacks.
3. Real defense requires fixing identifiers, device identity, backend authorization, and abuse detection together.
4. For IoT researchers, the real story starts after device compromise: how recovered trust material, weak IDs, and backend flaws turn one broken device into many.

black hat ASIA 2026

Tsinghua University

Thanks for listening & Questions?

Scan QR code and know more about our work