ChoiceJacking

Compromising Mobile Devices through Malicious Chargers like a Decade ago

SCIENCE
PASSION
TECHNOLOGY

Florian Draschbacher, Lukas Maar, Mathias Oberhuber, Stefan Mangard
Graz University of Technology

USENIX Security 2025

JuiceJacking (~2011)

Stealthy data extraction through **manipulated phone chargers**

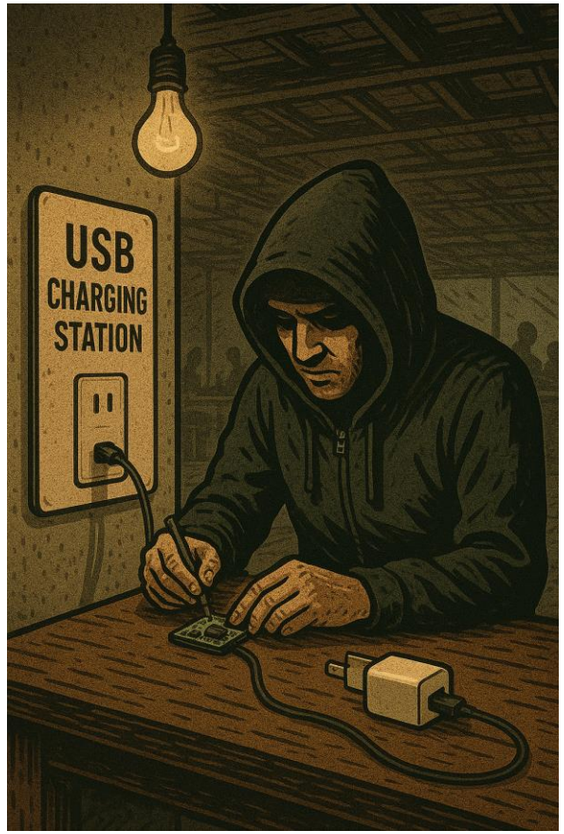
Airports, Train Stations, Museums, Hotel Rooms, Rental Cars, Power Banks, …

JuiceJacking (~2011)

Stealthy data extraction through **manipulated phone chargers**

Airports, Train Stations, Museums, Hotel Rooms, Rental Cars, Power Banks, …

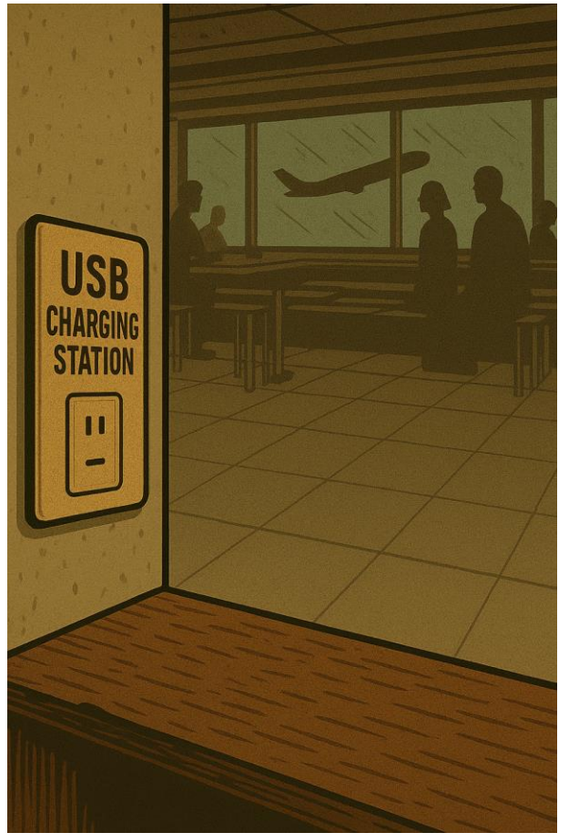
JuiceJacking (~2011)

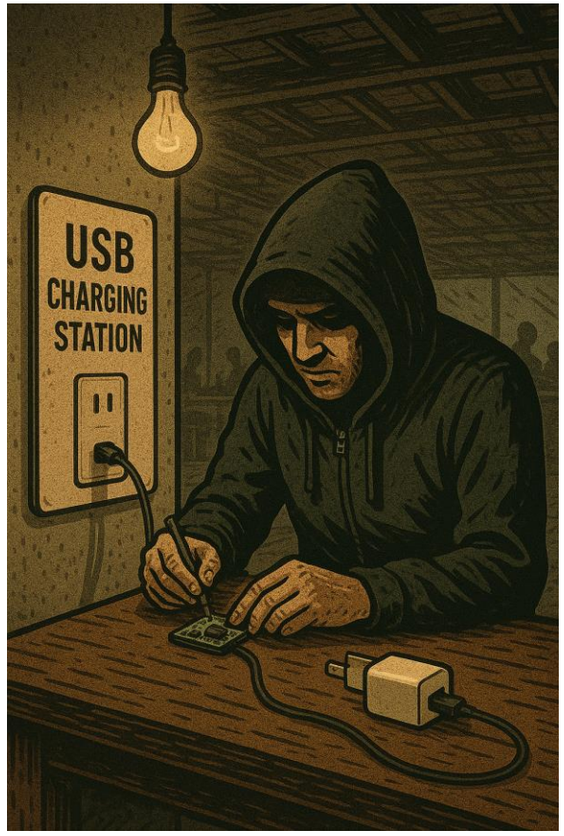


Stealthy data extraction through **manipulated phone chargers**

Airports, Train Stations, Museums, Hotel Rooms, Rental Cars, Power Banks, …

JuiceJacking (~2011)

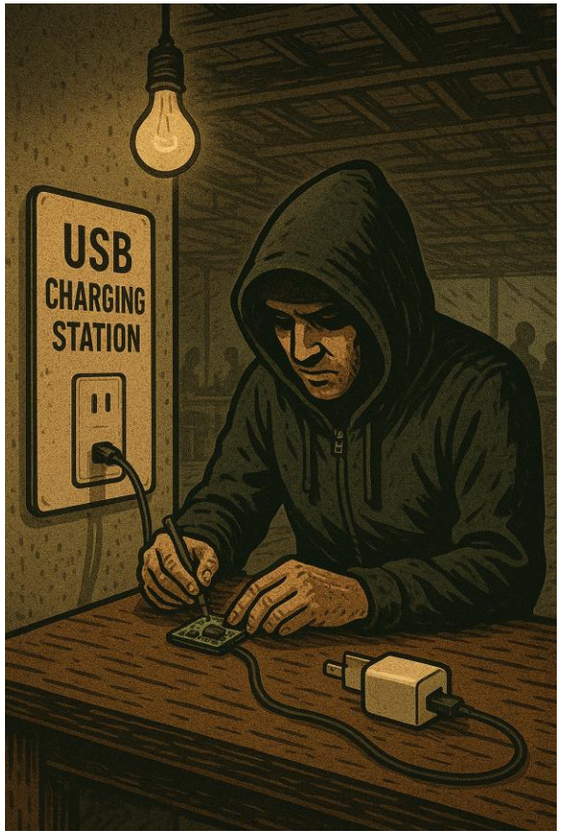


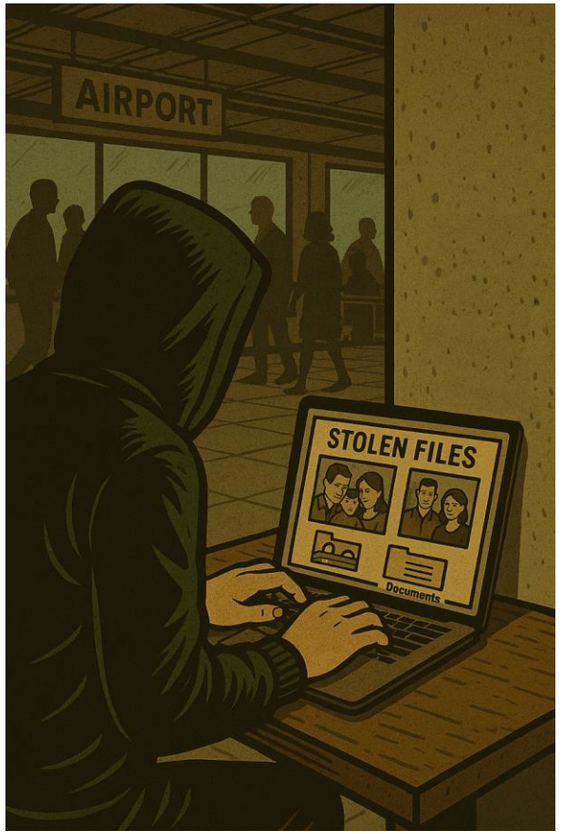


Stealthy data extraction through **manipulated phone chargers**

Airports, Train Stations, Museums, Hotel Rooms, Rental Cars, Power Banks, …

JuiceJacking (~2011)

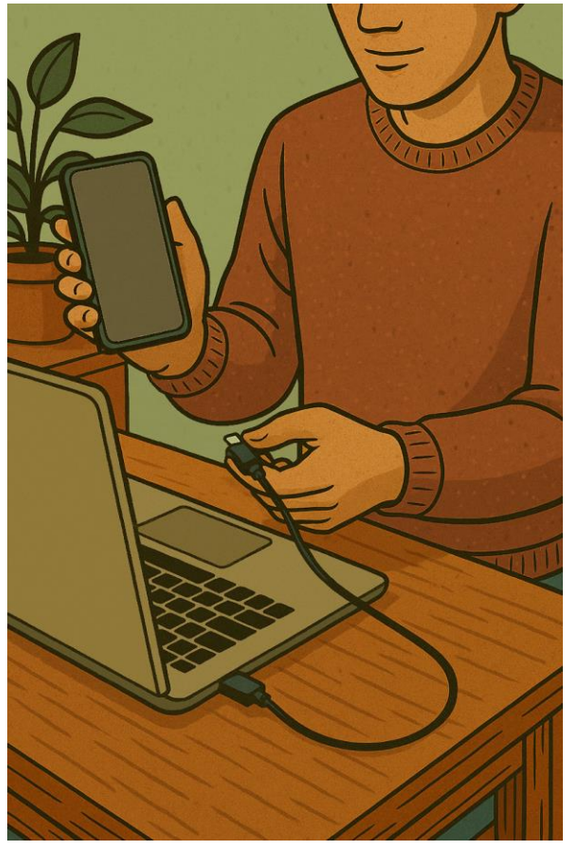


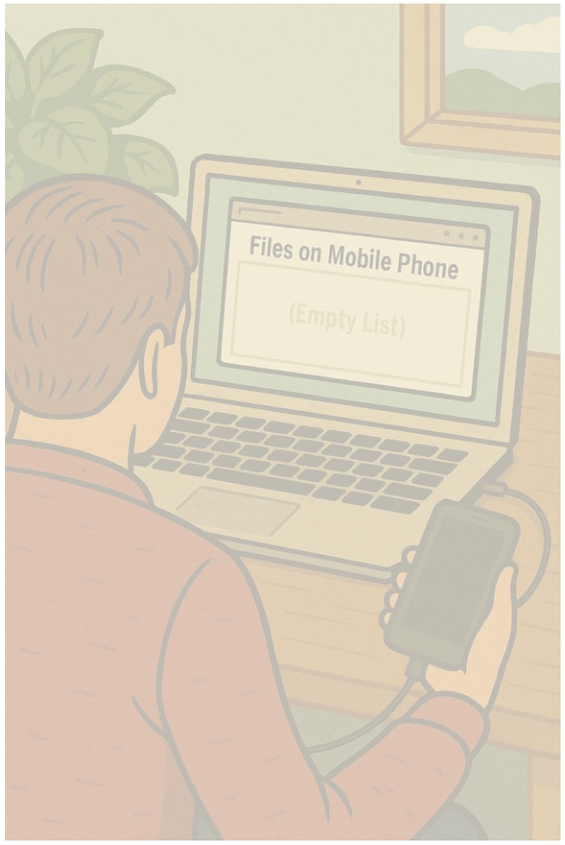


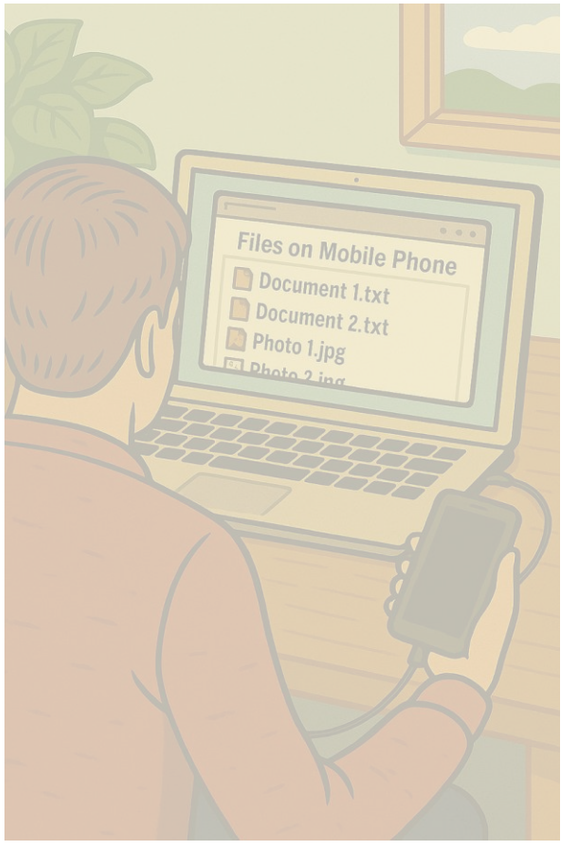
Stealthy data extraction through **manipulated phone chargers**

Airports, Train Stations, Museums, Hotel Rooms, Rental Cars, Power Banks, …

JuiceJacking Mitigations (2013-)

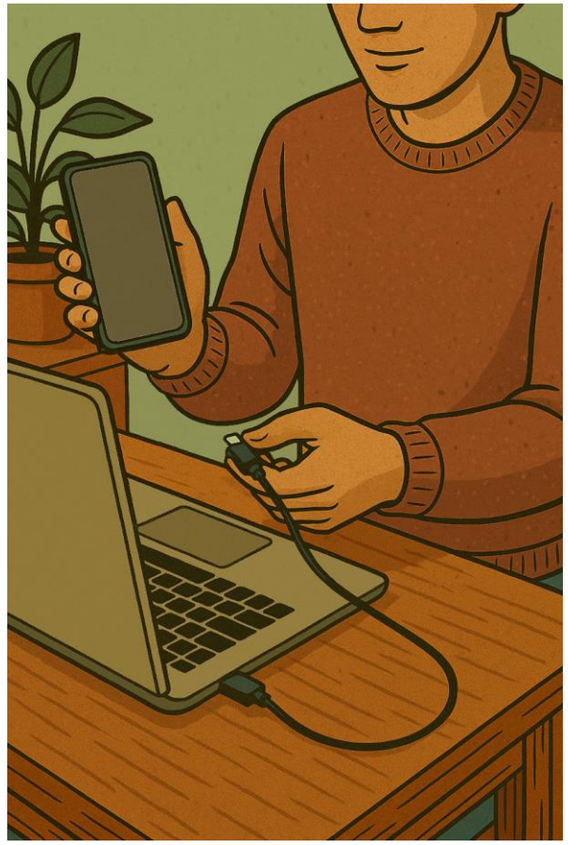


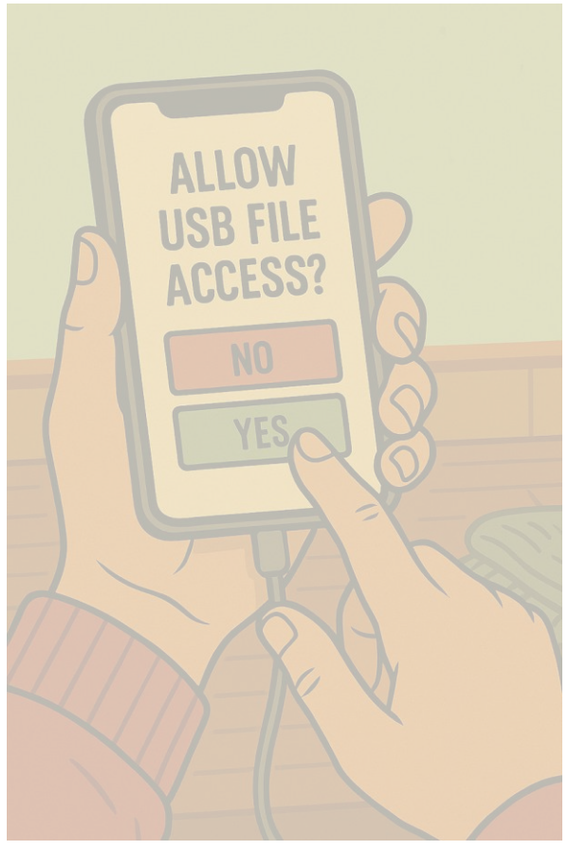


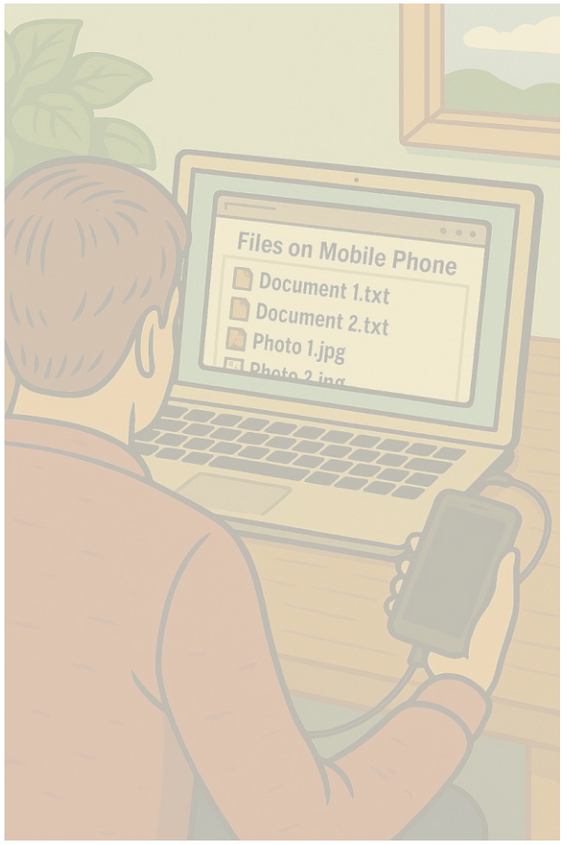


iOS and Android require **user consent** for USB file access

JuiceJacking Mitigations (2013-)

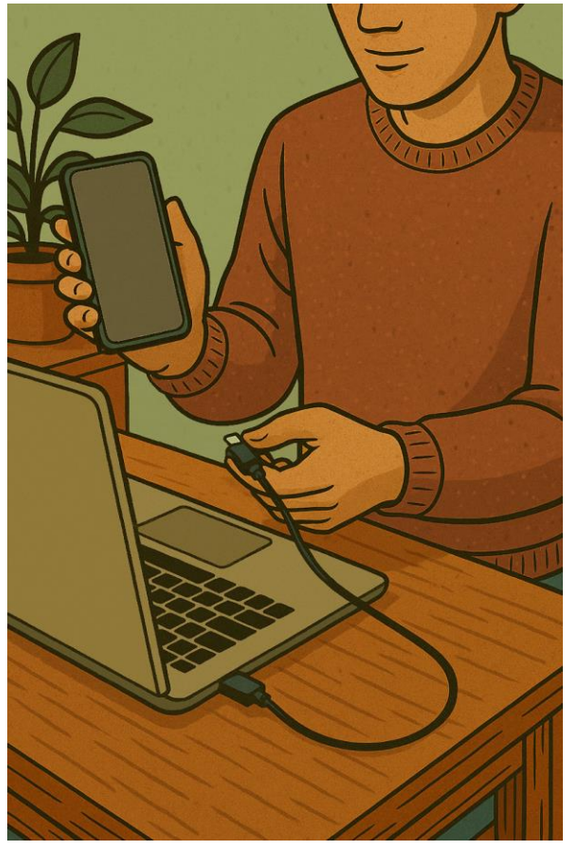


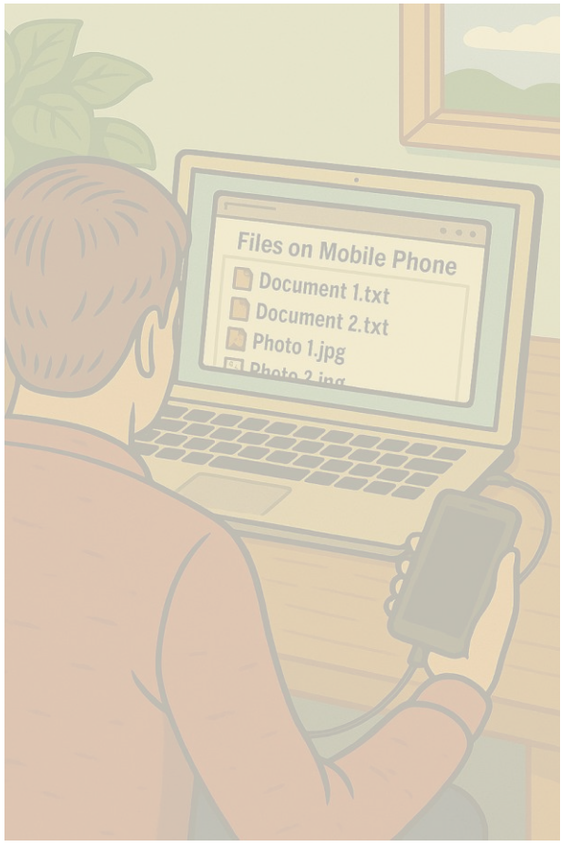




iOS and Android require **user consent** for USB file access

JuiceJacking Mitigations (2013-)





iOS and Android require **user consent** for USB file access

JuiceJacking Mitigations (2013-)

iOS and Android require **user consent** for USB file access

Motivation

- 2011: JuiceJacking allowed **USB-based extraction of user data**
  - Pictures, Videos, Documents, …
- 2013: Mobile OSs added **user consent** for USB data connections
  - To mitigate JuiceJacking attacks
- Since then, no USB-based attack with similar impact has been found
  - Mitigation widely **believed to be effective**
- **However:** No-one actually challenged this mitigation

Contributions

- We uncover **fundamental flaws** in JuiceJacking mitigations
  - Across Android and iOS
- **Novel USB attacks** that combine host-based and device-based attacks
  - Three concrete attack techniques for bypassing USB data consent
- Evaluation on **11 devices across 8 vendors**
  - All susceptible to our attacks
- Power Line Side-Channel for identifying **suitable attack time**

Background: USB on Mobile

Mobile devices use multi-function USB-C ports

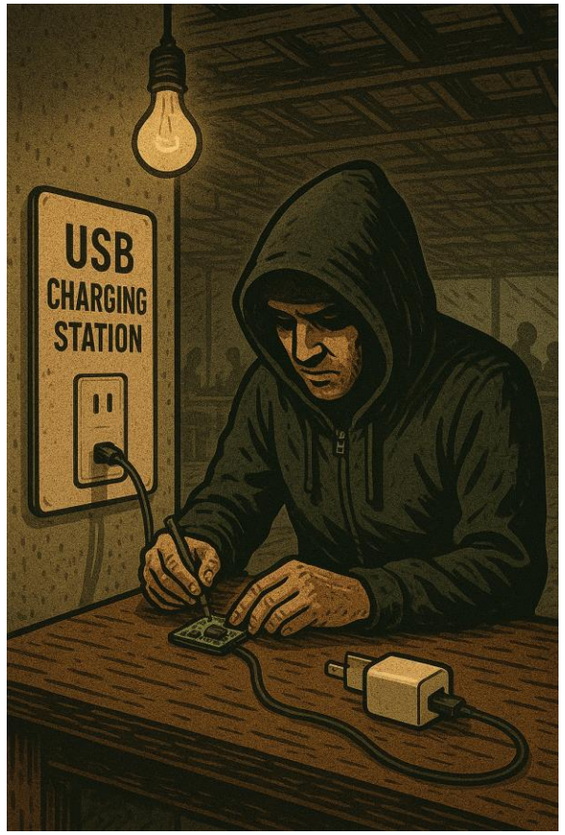
Image: Kyu3 / CC BY-SA 4.0 / wikimedia

- **Power**
  Charge phone or supply peripherals
- **USB Power Delivery**
  Negotiation of power and data roles
- **USB Host**
  Data exchange with peripherals
- **USB Device**
  Data exchange with computer

Important: A USB port **either** acts as USB host or USB device at a given time

Threat Model

- Victim uses **manipulated phone charger**
  - Reflashed firmware, replaced electronics, …
- Attacks are **untargeted**
  - Collect data from as many victims as possible
- Some attacks require unlocked screen
- **Attacker Goal:** Access **sensitive user data**
  - Pictures, Videos, Documents, App-specific data



Key Observations on JuiceJacking Mitigations

- **Goal:** Ensure user consciously enables USB file access

1. **Require Screen Unlock**
   Idea: Subsequent actions executed by legitimate owner
   Observation: Users routinely unlock screen while charging

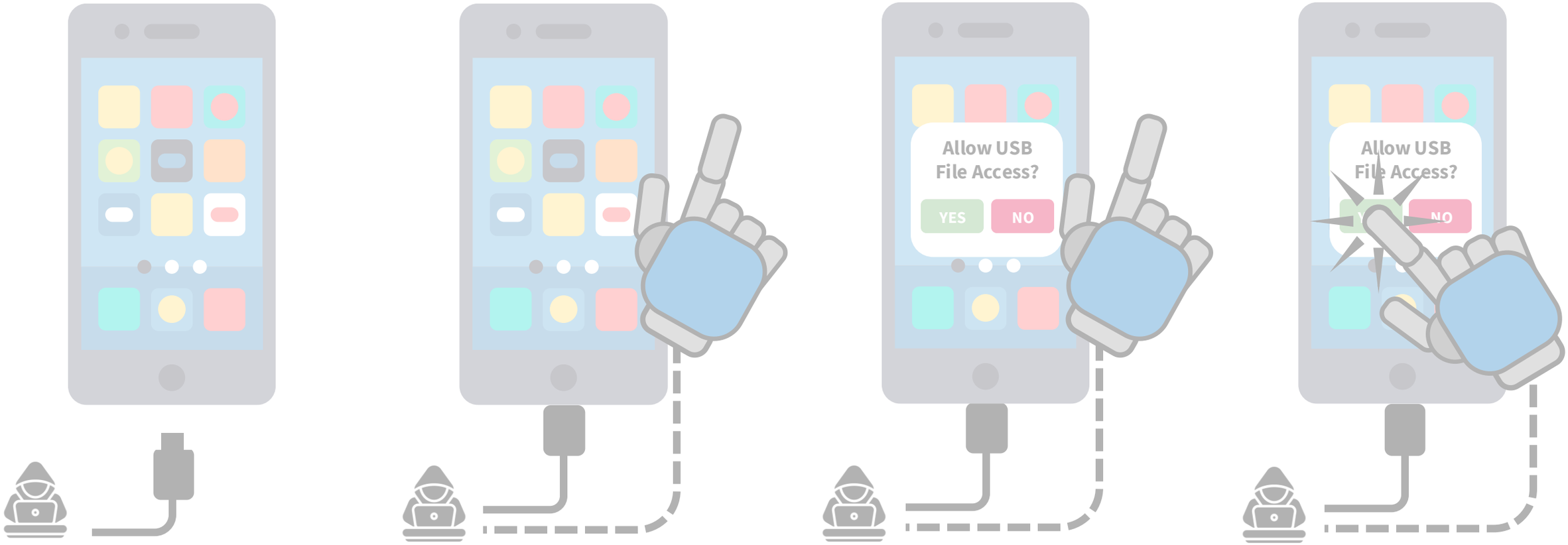
Key Observations on JuiceJacking Mitigations

- **Goal:** Ensure user consciously enables USB file access

1. **Require Screen Unlock**
   Idea: Subsequent actions executed by legitimate owner
   Observation: Users routinely unlock screen while charging

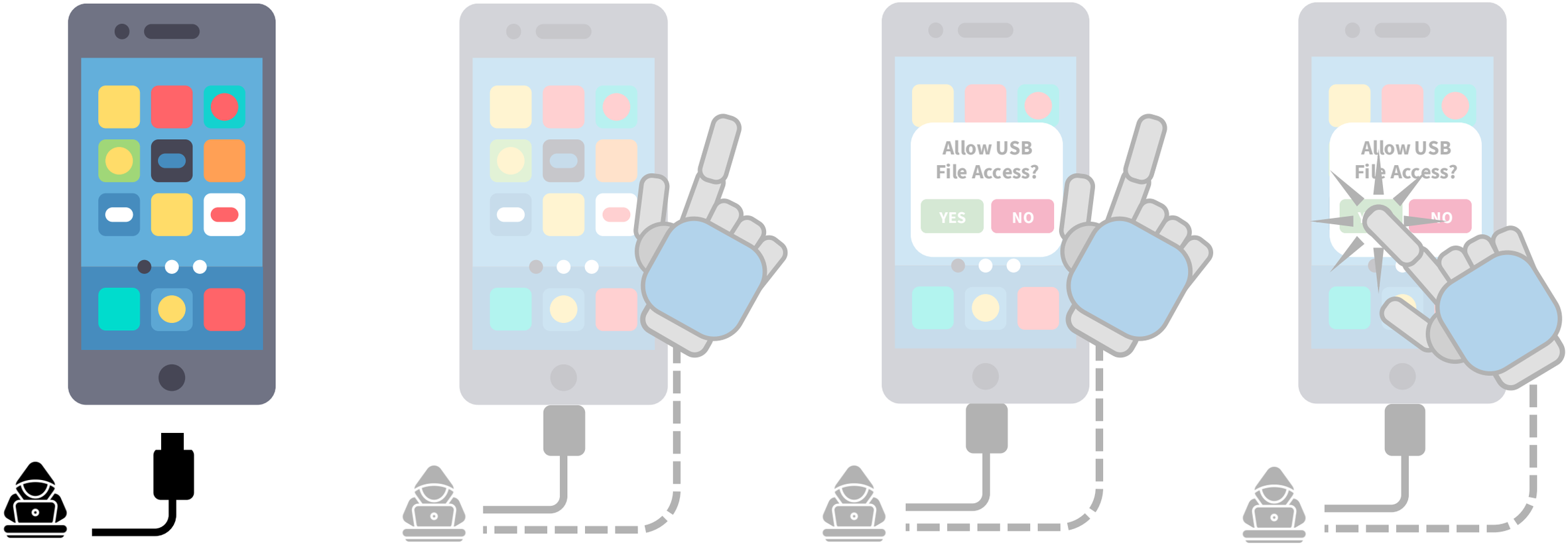
2. **Require User Consent**
   Idea: USB partner cannot establish MTP and inject input at same time
   Flaw: Impossible by USB specification, but possible in practice!

ChoiceJacking: Spoofing User Consent for USB Access



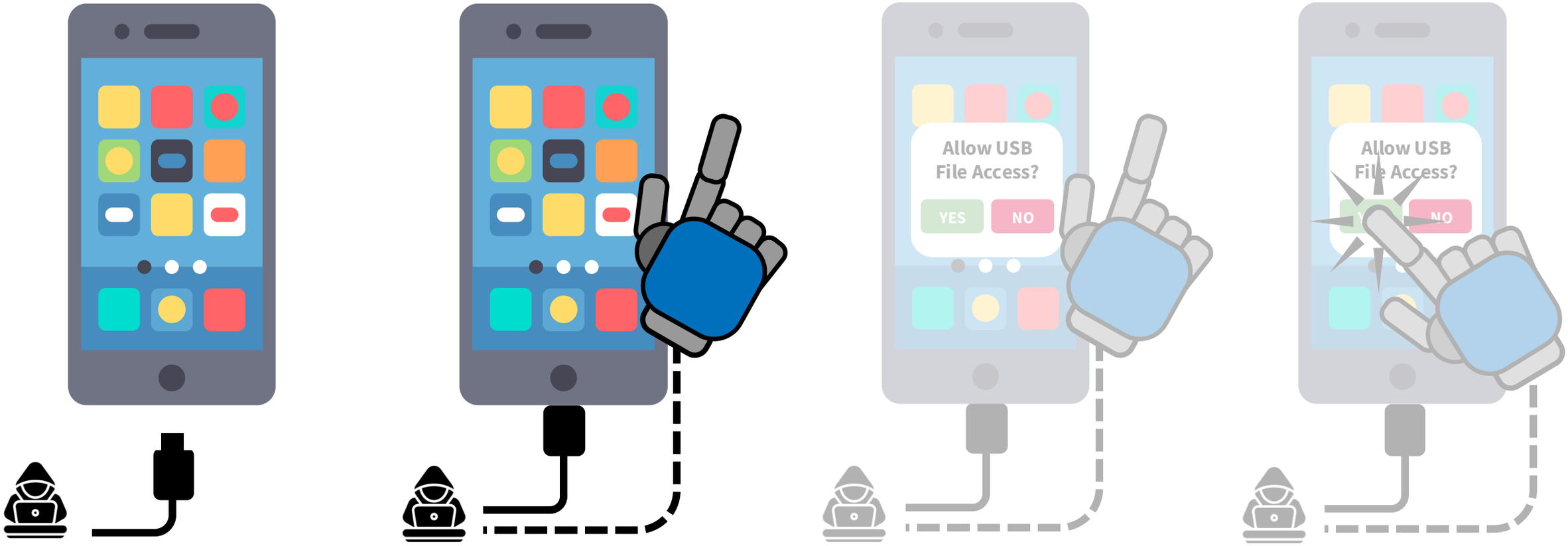
Core Principle: Establish input event channel in addition to file channel

ChoiceJacking: Spoofing User Consent for USB Access



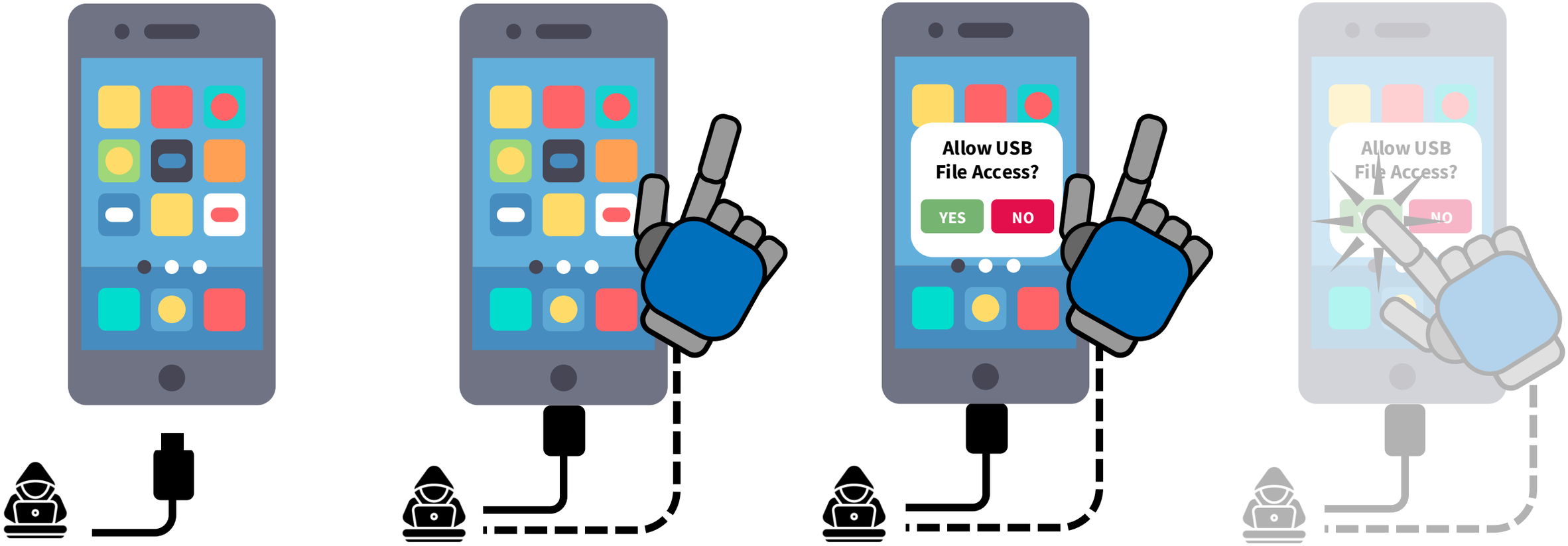
Core Principle: Establish input event channel in addition to file channel

ChoiceJacking: Spoofing User Consent for USB Access



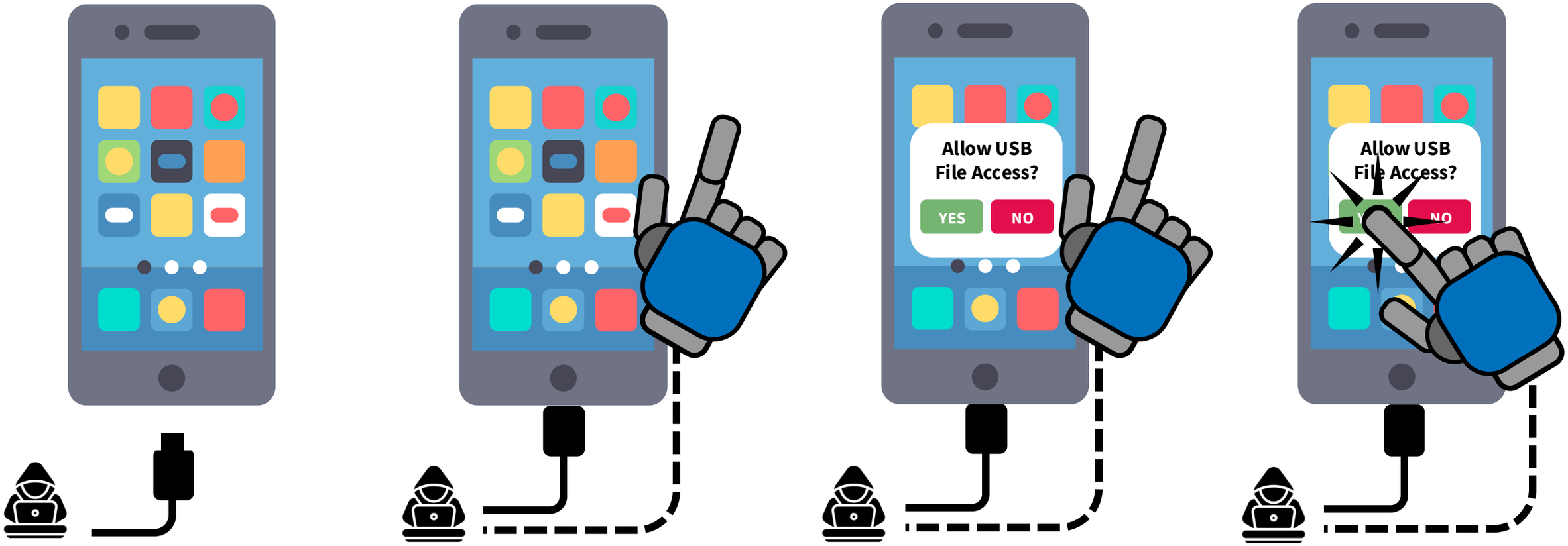
Core Principle: Establish input event channel in addition to file channel

ChoiceJacking: Spoofing User Consent for USB Access



Core Principle: Establish input event channel in addition to file channel

ChoiceJacking: Spoofing User Consent for USB Access



Core Principle: Establish input event channel in addition to file channel

Concrete Attack Techniques

Each technique **independently** allows bypassing USB data user consent

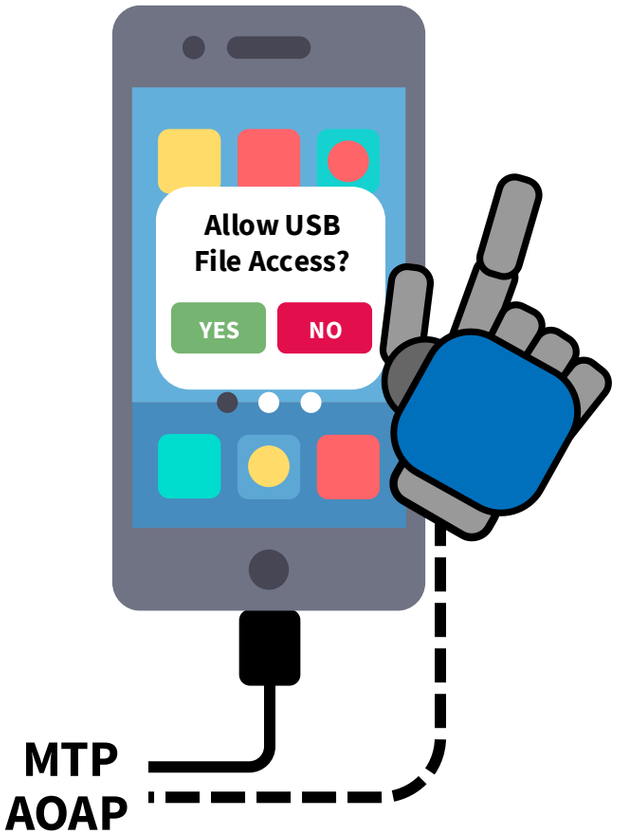
1. **Input Accessory**
   Exploits implementation flaw in Android's accessory protocol

2. **Flooding Input Events**
   Exploits race condition in Android's input dispatcher

3. **Pivoting via Bluetooth HID**
   Exploits flaw in trust model across iOS and Android

Example: Technique 1 - Input Accessory

- Android Open Accessory Protocol (AOAP)
  - Input accessories for devices lacking USB host hardware
- USB host sends HID events as special USB requests
  - Normally only USB device (e.g. mouse) can send HID events
- AOAP Specification: Accessory mode is exclusive
  - No parallel MTP connection possible
- **However: All implementations violate specification**



Evaluation

- 11 devices across **Android and iOS**
  - **Samsung, Xiaomi, Vivo, Oppo, Huawei, Honor, Google, Apple**
  - Most recent OS version for each device

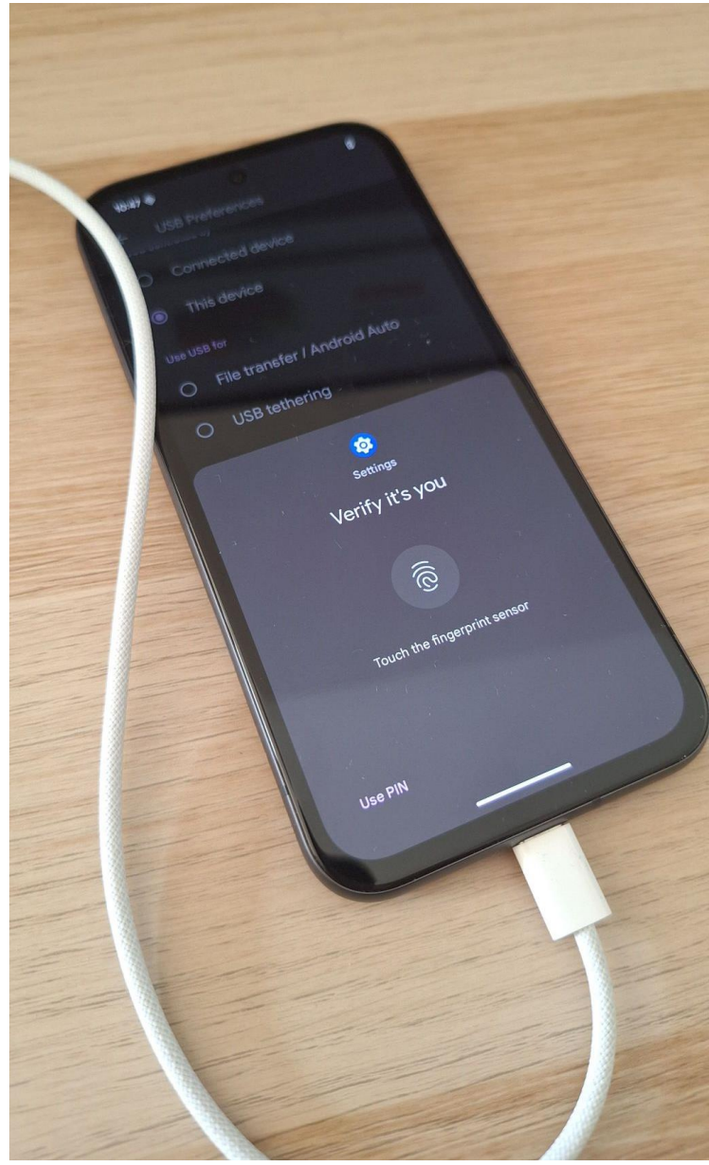
- All susceptible to at least 1 technique
  - **Android:** Access to any file in public storage and shell
  - **iOS:** Access to pictures and videos

- Some attacks faster than a human blink
  - **Samsung:** ~130 milliseconds

- Even worked on locked screens for some vendors

Disclosure

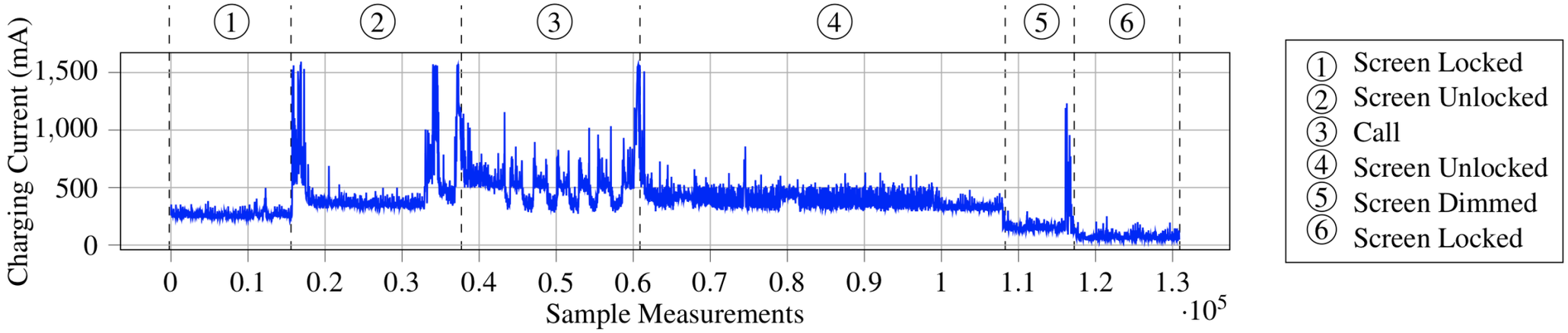
- Responsibly disclosed 21 specific security issues
  - 16 vendor-specific, 4 upstream Android, 1 iOS issue
- Several CVEs, among them:
  - **Google:** CVE-2024-43085
  - **Apple:** CVE-2025-24193
- Fixes in Android 15 and iOS 18.4
  - Require authentication for USB file access
- Some Android attacks still possible
  - Vendors who customized Android OS
  - Bypassing USB debugging prompt



Suitable Attack Moments

- Some attacks require unlocked but unattended screen
  - **Suitable moment:** User is in a phone call

- Such moments can be detected from power consumption



- We implemented a PLSC based on one-dimensional CNN classifier

Conclusion

- First analysis of **JuiceJacking mitigations**
  - **Fundamental flaws** across the industry
- Novel USB attacks called **ChoiceJacking**
  - Three different attack techniques
- Attacks succeeded on **all tested devices**
  - **Faster** than human blink for some
- **Disclosed all findings** to affected vendors
  - Most vulnerabilities fixed now

Scan for paper PDF