Misbinding Attacks on Secure Device Pairing

**Tuomas Aura**, Aalto University, Finland

joint work with **Mohit Sethi**, Ericsson, and **Aleksi Peltonen**, Aalto University

IETF 104, SAAG, Prague
Outline

1. Background:
   misbinding in authenticated key exchange

2. Misbinding in device pairing
   (Bluetooth)

3. Misbinding in connecting devices to cloud
   (EAP-NOOB)
Background: misbinding in authenticated key exchange
Misbinding in key exchange

• A thinks it is authenticating to E, but it is actually authenticating to B
• E is dishonest. B can be honest

• Known since 1992 (STS, Diffie et al. 1992) and motivated the SIGMA protocols (IKEv1, IKEv2)
• Named unknown key-share, misbinding, cuckoo
Misbinding of responder:
A thinks it is connected to E. In fact, A and B are connected.

Misbinding of initiator:
B thinks it is connected to E. In fact, A and B are connected.
Solution to misbinding: be explicit about identities

ISO 9798-3

SIGMA
(slightly better protection in case of an incompetent CA)
Detecting misbinding of responder

A, $g^x$ → E, $g^y$, $\text{SIG}_E(g^x, g^y, A)$

$\text{SIG}_A(g^x, g^y, E)$ → B, $g^y$, $\text{SIG}_B(g^x, g^y, A)$

$\text{SIG}_A(g^x, g^y, E)$ → E ≠ B

Detecting misbinding of initiator

E ≠ A

A, $g^x$ → E, $g^x$

B, $g^y$, $\text{Sig}_B(g^x, g^y, E)$

$\text{SIG}_A(g^x, g^y, B)$ → B, $g^y$, $\text{SIG}_B(g^x, g^y, E)$

B, $g^y$, $\text{SIG}_B(g^x, g^y, E)$ → $\text{SIG}_E(g^x, g^y, B)$
How serious is it? (1)

• Seriousness difficult to grasp:
  • **No failure of confidentiality.** Victim wants to talk with the malicious party E, and thus attacker would get all the secrets even without misbinding
  • **Problem related to data authentication.** Victim is confused about to who it is at the other end of the secure connection

• Attack scenarios in literature are artificial:
  • A is commander, E and B fighter jets. E has been compromised by the enemy. A tells E to self-destruct, but the command goes to B  
    [Hugo Krawczyk]
  • A connects to bank B and, over the secure session, deposits an electronic cheque. Bank B thinks the cheque was deposited by E  
    [Diffie et al.]
How serious is it? (2)

• Well-defined problem in formal verification: failure of a correspondence property:

  If $A$ and $B$ share session key $K$,
  $A$ should think it shares the key $K$ with $B$.

• Easy to prevent in most protocols: bind endpoint identifiers to the key

• However, must have authenticated identifiers (e.g. certificates) and the other endpoint must know what id to expect
Misbinding in device pairing
Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK ➔ Paired!

A

Pair with: > B

B
Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK ➔ Paired!
Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK ➔ Paired!
Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK → Paired!
Misbinding in Bluetooth

A
Pair with: > B

B
Device B is compromised (malicious app)
Misbinding in Bluetooth

Attacker has another device named “B”

Device B is compromised (malicious app)
Misbinding in Bluetooth

Attacker has another device named “B”

Device B is compromised (malicious app)

Key exchange between wrong devices

“A” Pair with: > B

“B”
Misbinding in Bluetooth

Key exchange between wrong devices

Attacker has another device named “B”

Device B is compromised (malicious app)
Misbinding in Bluetooth

Key exchange between wrong devices

Attacker relays 6-digit code

Attacker has another device named “B”

Device B is compromised (malicious app)

Malicious app spoofs UI
Misbinding in Bluetooth

Attacker has another device named “B”

Device B is compromised (malicious app)

Malicious app spoofs UI

User clicks OK
Misbinding in Bluetooth

Wrong devices paired!
Why does Bluetooth not detect misbinding?

Could it?
Why does Bluetooth not detect misbinding?

Could it?

Devices have no verifiable identifiers!

Authentication based only on physical access
Formal modeling

- Previous security analysis of Bluetooth had not detected misbinding
- We modeled Bluetooth numeric comparison and other pairing protocols with ProVerif
  - Physical channel defines device identity
  - Check correspondence between user intention and completed pairing

→ Can detect misbinding

- Analysis yielded a new double-misbinding case:
Lessons

• All device-pairing protocols are vulnerable if devices have no verifiable identifiers and authentication is based only on physical access

• Trusted path issue: attacker can spoof the pairing UI on the compromised device
  • Trusted path (e.g. hard-wired reset button) would prevent malicious apps from spoofing the critical UI
  • Device UIs are difficult to standardize, and attacker could still replace or modify the hardware
Misbinding in connecting devices to cloud (EAP-NOOB)
EAP-NOOB

• **EAP method** for bootstrapping devices out-of-the-box without professional administration and without pre-established device credentials or identifiers

• **User-assisted out-of-band (OOB) authentication**
  • One OOB message in one direction between peer and server, e.g. scanning a dynamic QR code or NFC tag

• OOB authentication registers a new peer device. Once registered, reauthentication without user interaction

[link: draft-aura-eap-noob]
EAP-NOOB architecture

- Remote AAA
- Local AAA
- AP
- New device

UI or API

EAP in-band

OOB output (or input)

User-assisted OOB channel
EAP-NOOB protocol

1. Unauthenticated ECDHE key exchange
2. User-assisted out-of-band message
3. Key confirmation

Remote AAA → New device
UI or API → OOB (or input)
 OOBB channel

Local AP
Misbinding in EAP-NOOB

Remote AAA  Local AAA  AP  New device

UI or API  EAP in-band  Device UI compromised

User Alice  OOB output (or input)
Misbinding in EAP-NOOB

Attacker has access to another peer device

Remote AAA

Local AAA

AP

New device

Device UI compromised

EAP in-band

UI or API

OOB output (or input)

User Alice
Misbinding in EAP-NOOB

Attacker has access to another peer device

Attacker relays OOB message

Attacker relays OOB message replayed to user

Remote AAA

Local AAA

AP

EAP in-band

UI or API

User Alice

New device

User delivers wrong OOB message

OOB output (or input)
Wrong device registered to user Alice’s account in the Remote AAA server
Why misbinding in EAP-NOOB?

- **User physically identifies the peer device; no other authentication**

- **Not a flaw in this specific protocol:** Inherent weakness in pairing-like protocols that rely on user’s physical access for authentication

- **Misbinding of the server not possible because typical OOB channels use web certificates, and user or app checks the server name**
Misbinding and trusted execution

• Misbinding-like cuckoo attacks are known in trusted-computing

• Cryptographic authentication of TPM/TEE does not prove that the secure execution takes place inside a
  the user-chosen physical device
  • Compromised device with fake number plate or fake UI
can cause misbinding

• Relevant to two IETF WGs:
  • Remote ATtestation ProcedureS (rats)
  • Trusted Execution Environment Provisioning (teep)
Mitigation and summary
Mitigating misbinding

• Cryptographically bind session keys to context data
  • Persistent non-modifiable device identifiers and hw info
  • Channel binding to wireless MAC addresses
    → Harder to trick user, and attacker may be forced to modify hardware or perform active MitM in the access network

• Preventing software-based UI spoofing
  • Specify a trusted path for the devices (e.g. reset button)

• Knowing your devices
  • Device certificates to attest make, model, serial number
  • Asset tracking: user or admin has prior knowledge of the devices, identifiers and intended deployment
Summary

• All device-pairing and bootstrapping protocols are vulnerable to misbinding if
  • Device authentication is based on physical access
  • Device identity not cryptographically authenticated, or if the verifier does not know which identifier is correct

• Several ways to mitigate the threat, but complete prevention will require redefining the assumptions (or goals) of device pairing and registration

Discussion question: Should we now tell everyone that Bluetooth pairing is inherently insecure, or similarly for TPM/TEE provisioning?