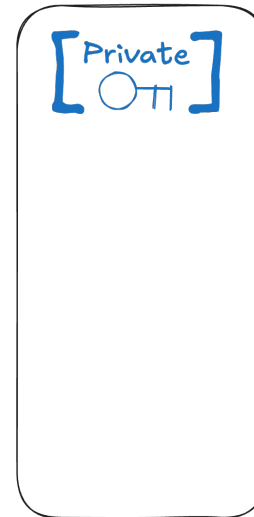


Beyond the Ceremony

The 2026 Passkey Attack Surface

Matteo Giordano

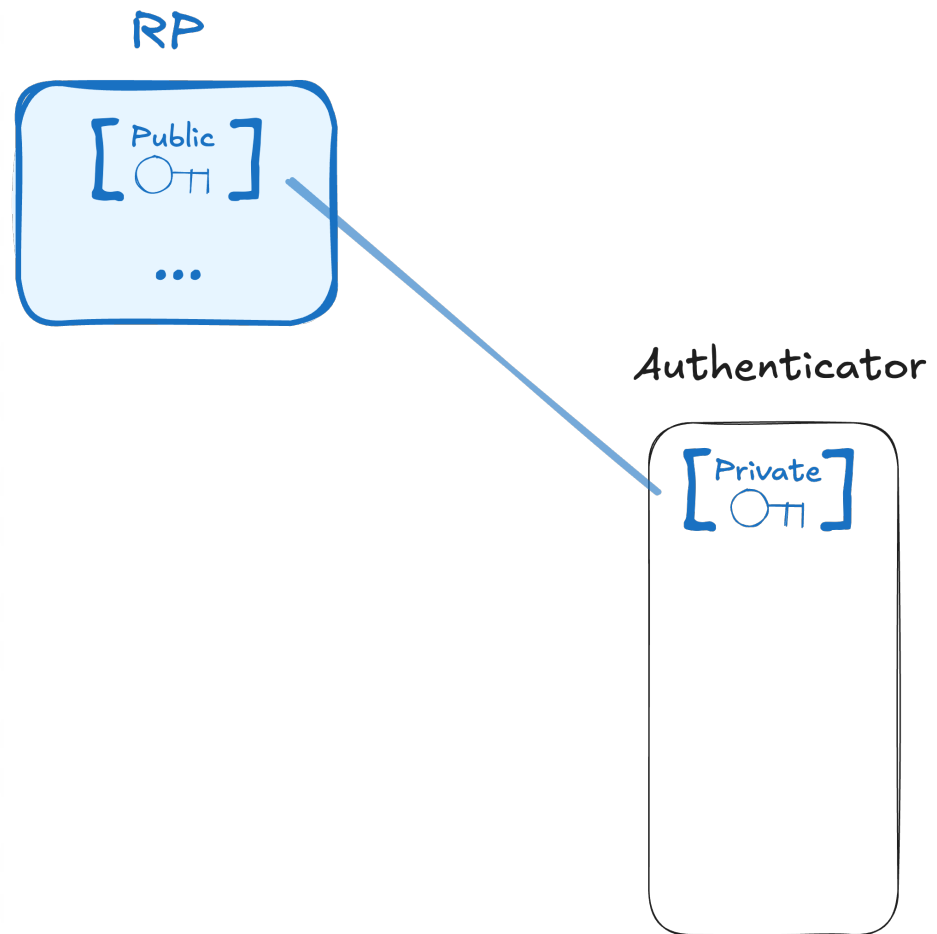
Authenticator

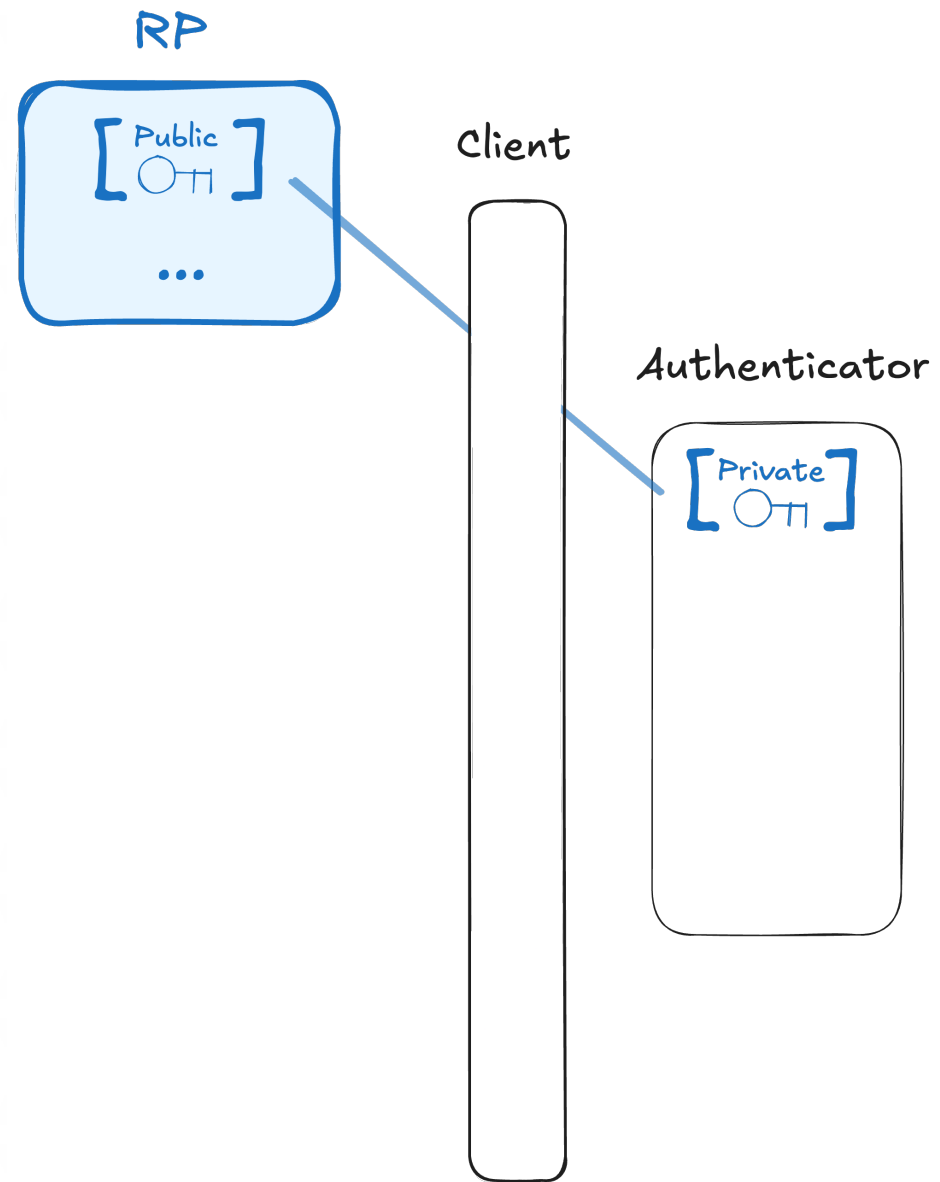


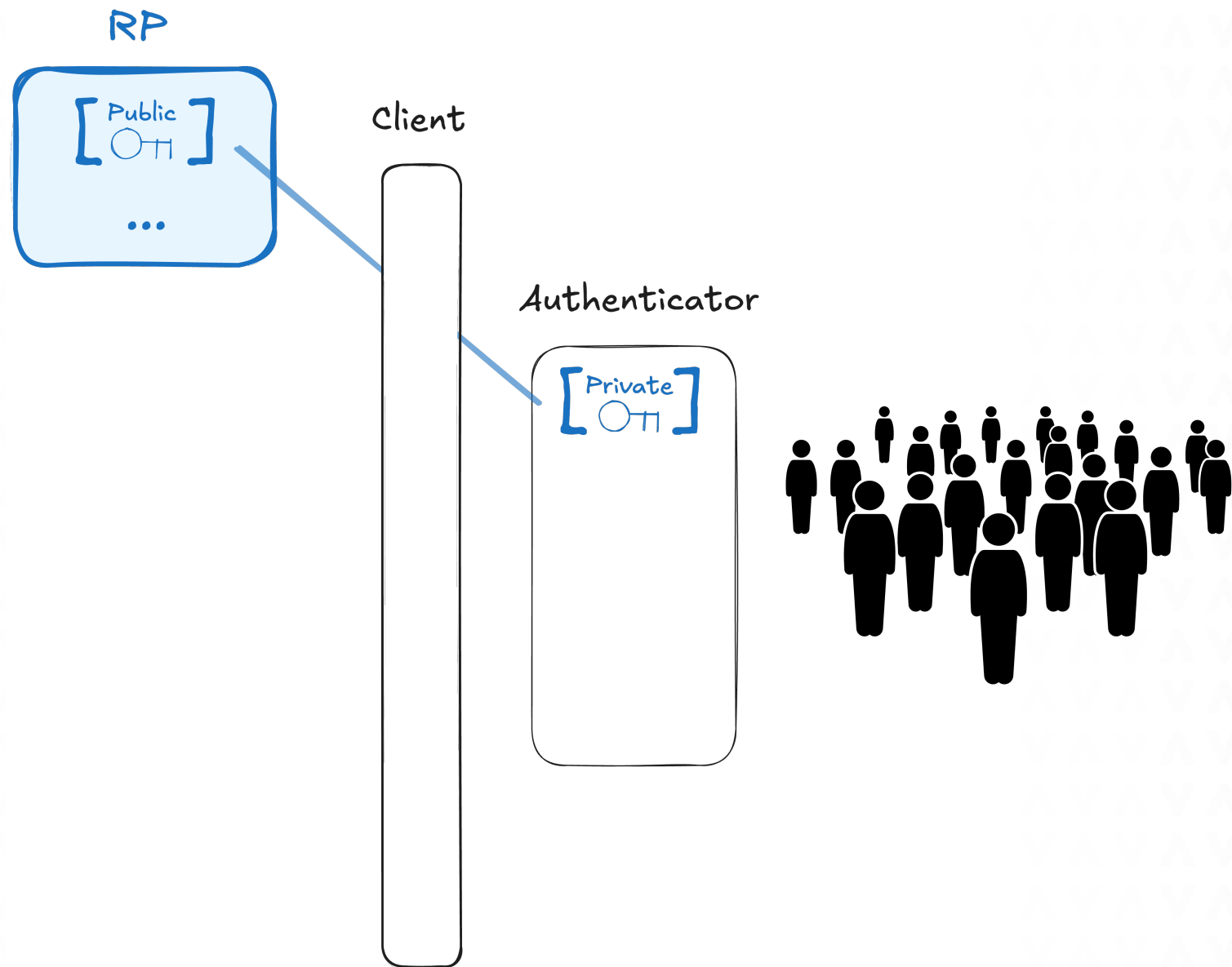
[Public]

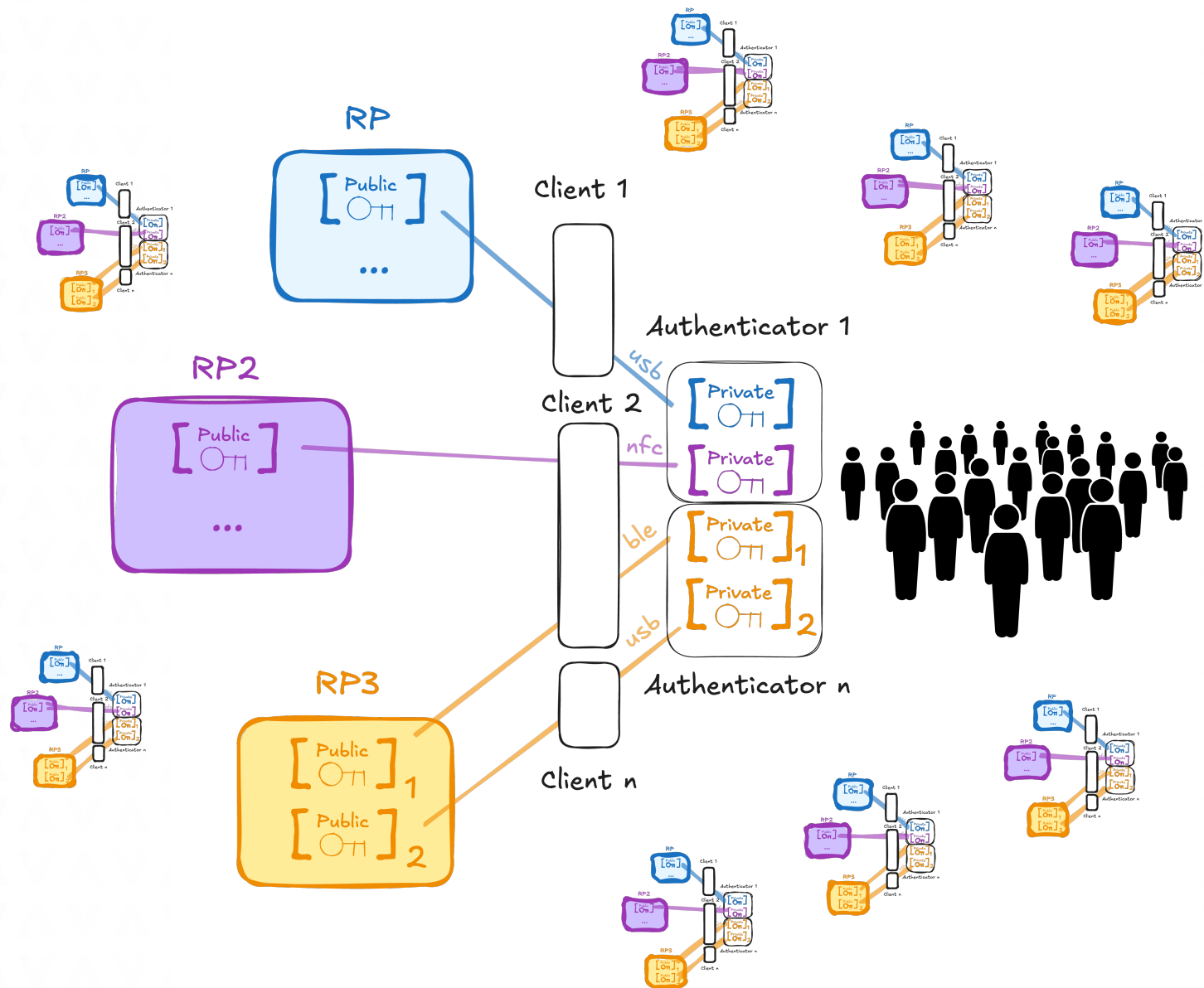
Authenticator

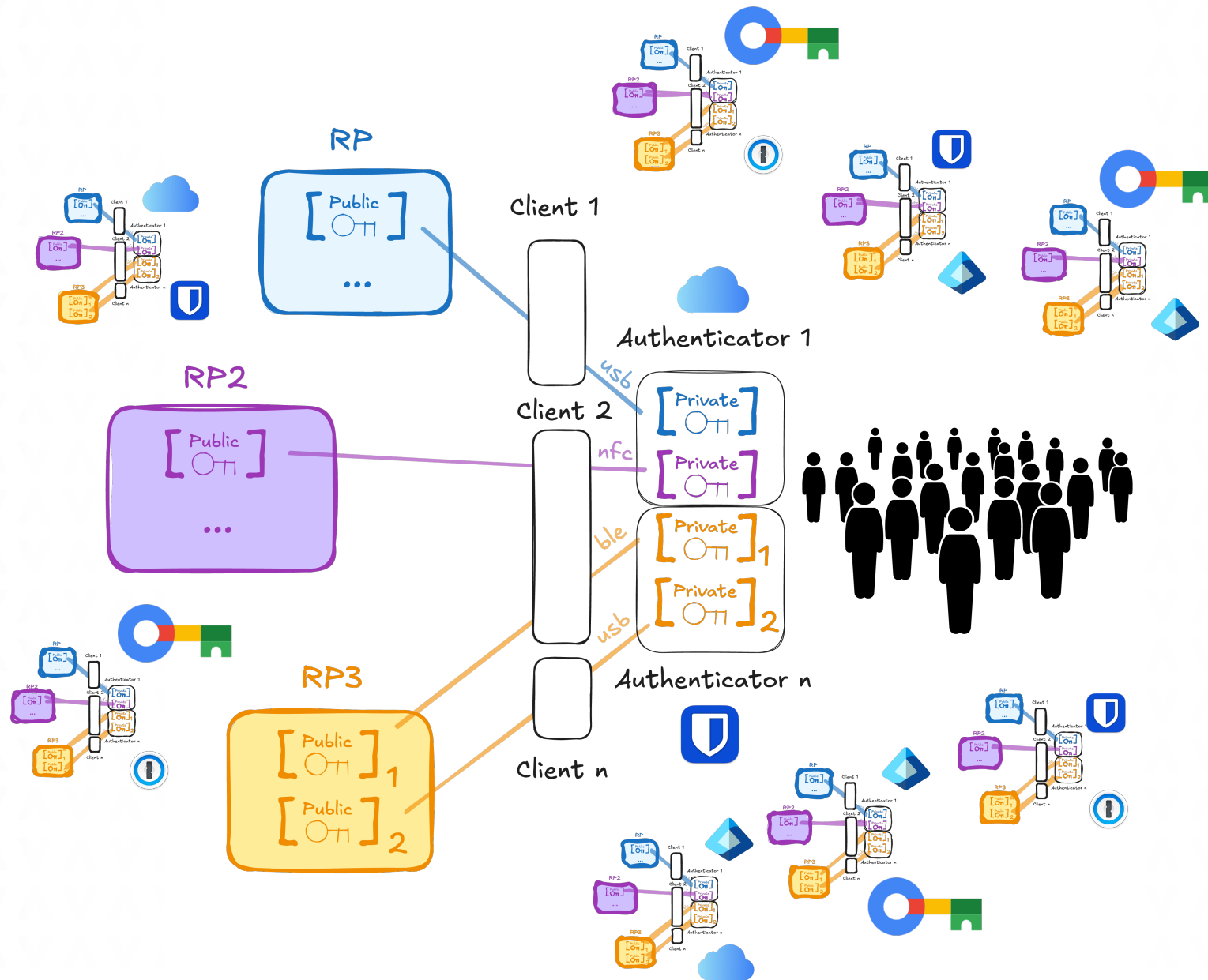


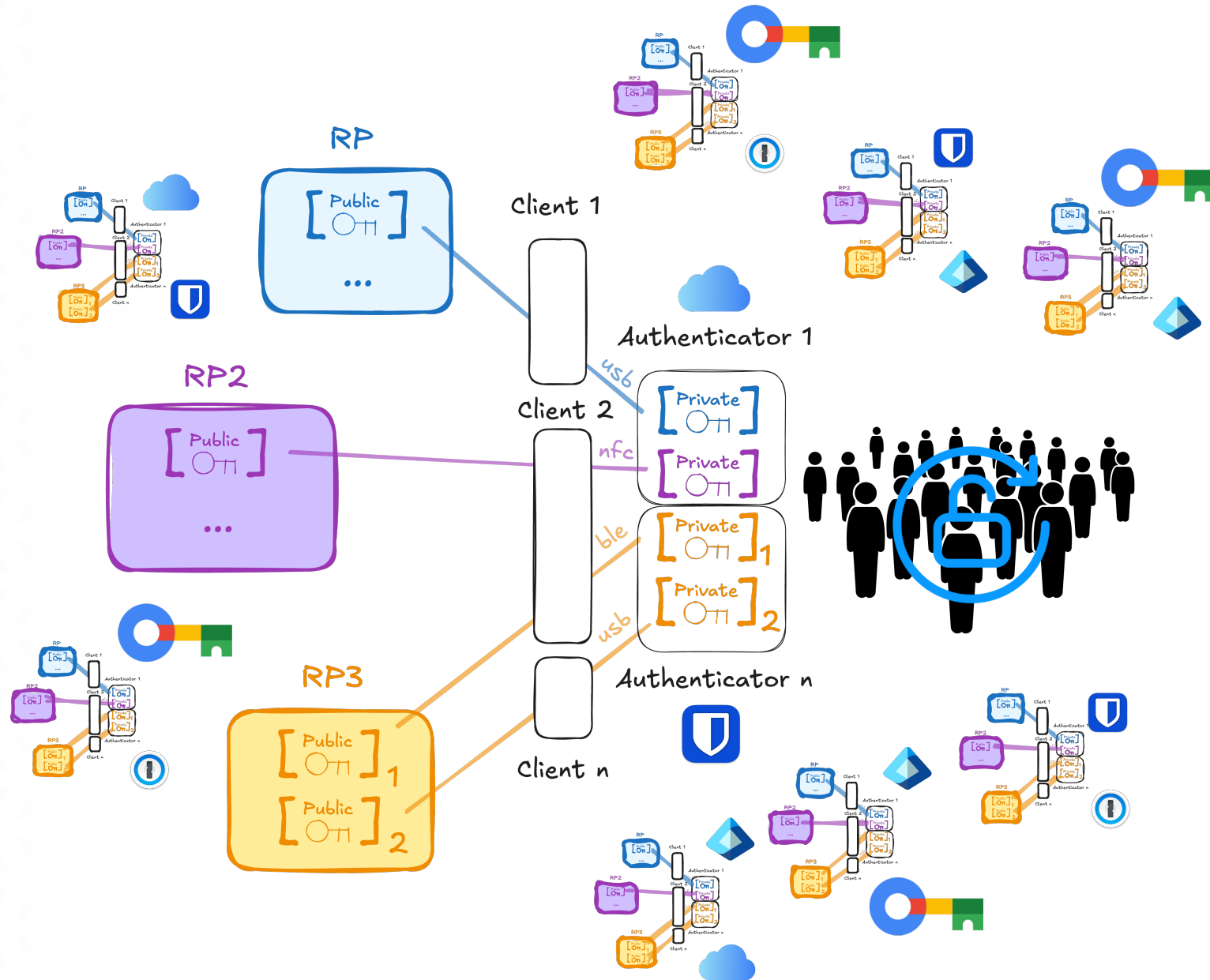


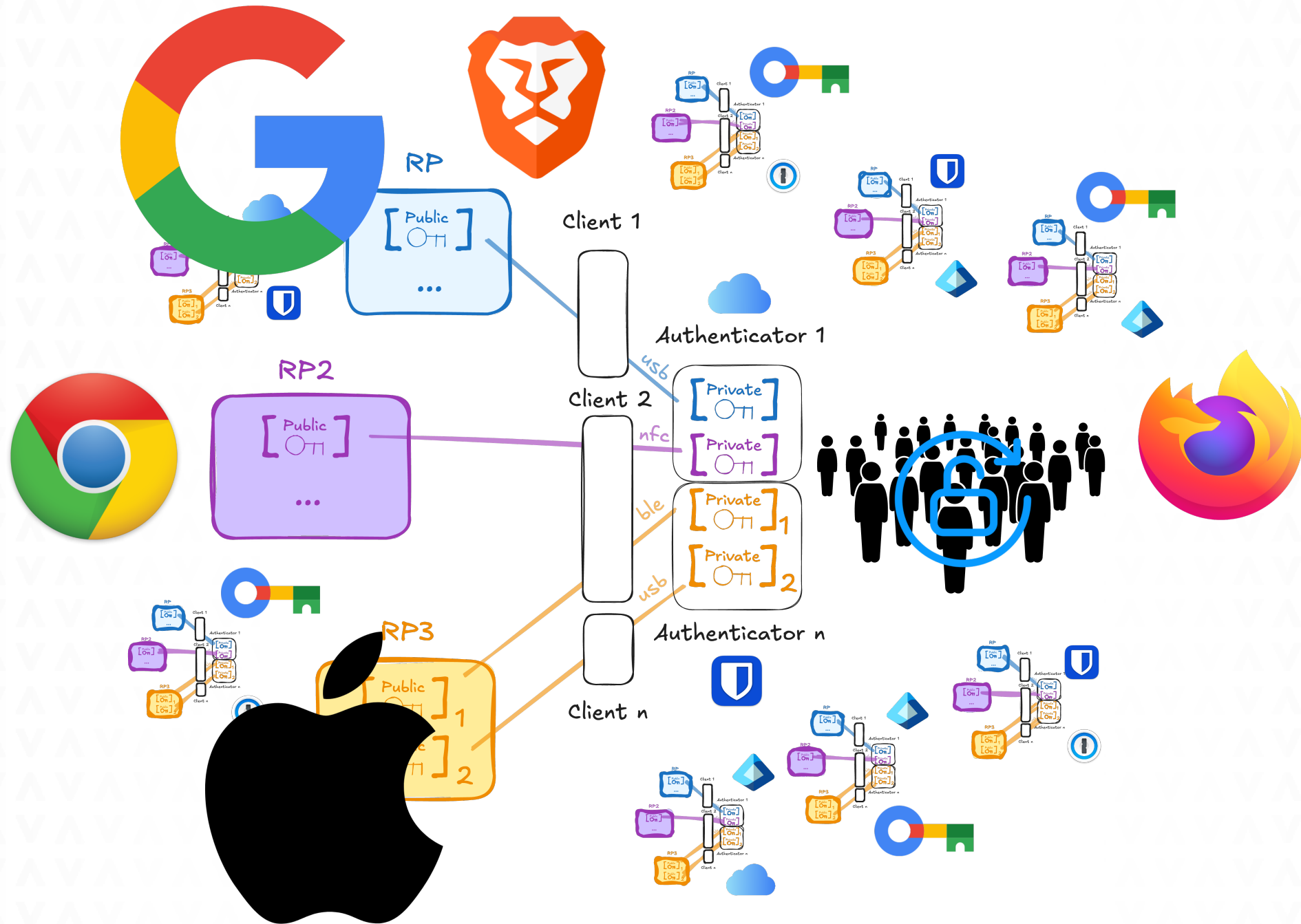






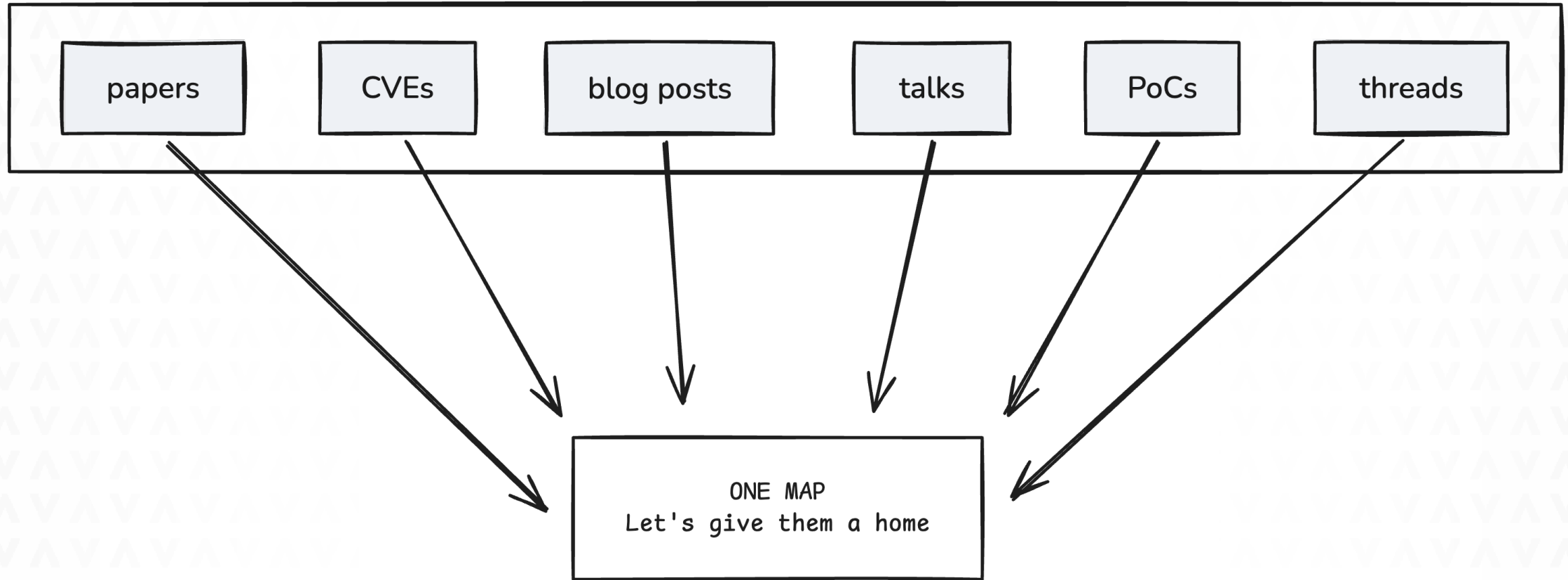






I'm not here with the scariest bug

11



whoami

```
matteo@defcon ~ % whoami --verbose
PublicKeyCredential {
  id: "matteo-giordano",
  rpId: "anvilsecure.com",
  userHandle: "AppSec, offensive research, AI security, Italy",
  authData: {
    flags: {
      UserPresence: true, // I'm here
      UserVerification: true, // trust me, right?
    },
    signCount: 0 // first DEF CON talk.
  },
  attestation: { fmt: "none" }
}
```

Don't trust an identity just because the ceremony looked clean. So don't trust mine.



Passkeys already won-ish

13

♦ 2025

- ♦ over 1B people have activated a passkey
- ♦ ~15B accounts support them
- ♦ [↗ FIDO World Passkey Day 2025](#)

♦ 2026

- ♦ ~5B in active use
- ♦ 68% of orgs deploying passkeys for workforce auth
- ♦ [↗ FIDO State of Passkeys 2026](#)

But they almost **never** run **alone**

14

- ◆ Only ~30% of orgs use passkeys as the **PRIMARY** method
- ◆ ~57% still lean on a **phishable** one

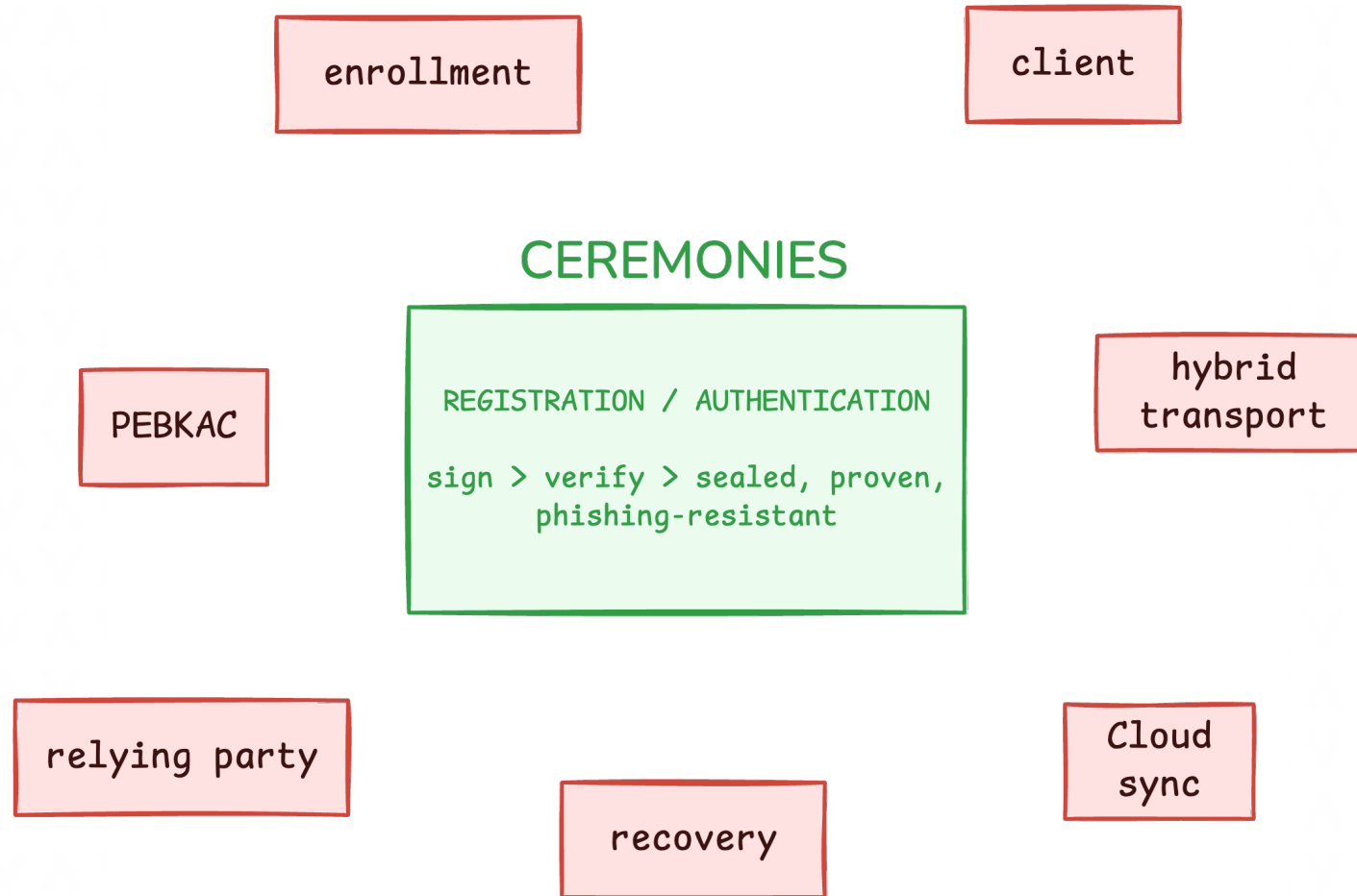


“Passkeys Are Not Broken, The Conversation About Them Often Is.”

[↗ Nishant Kaushik, FIDO CTO, Sept 2025](#)

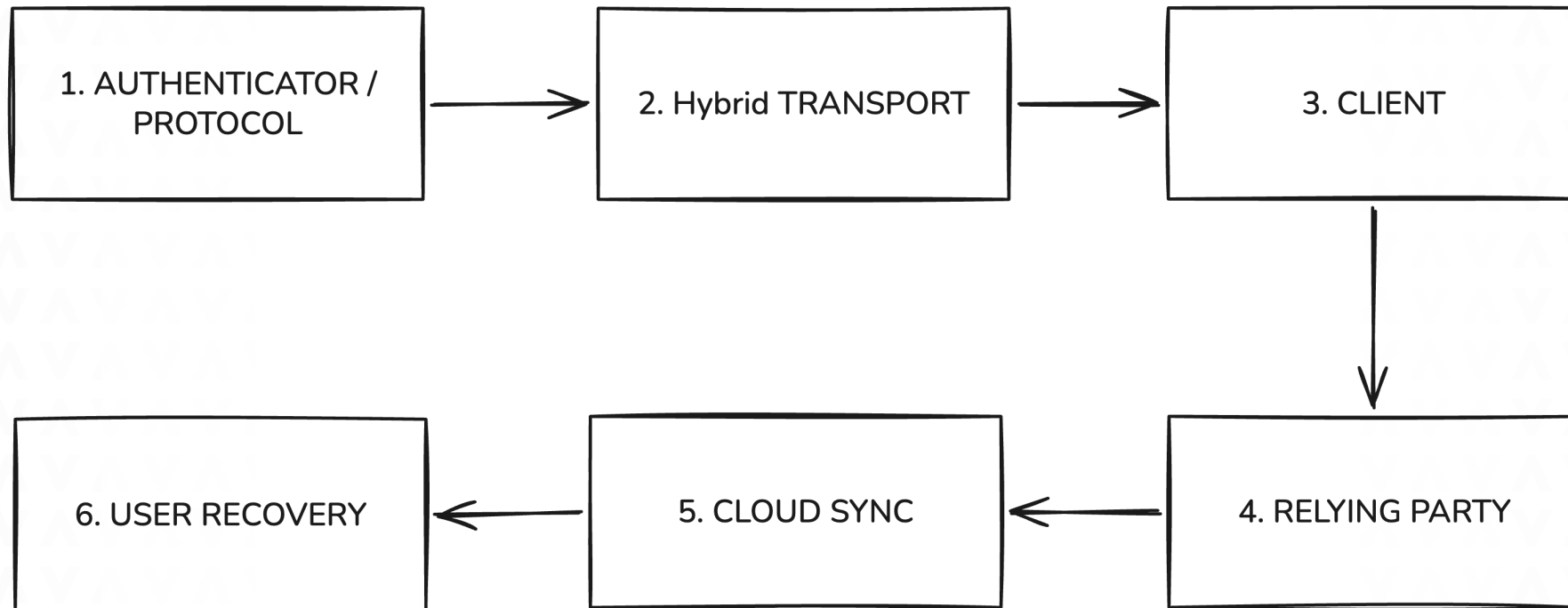
It's everything else

16



What **else**?

17



- ◆ **Six** components, from *meta*/to cloud
- ◆ Every passkey attack lives somewhere on this map

Ceremonies

Quick refresh

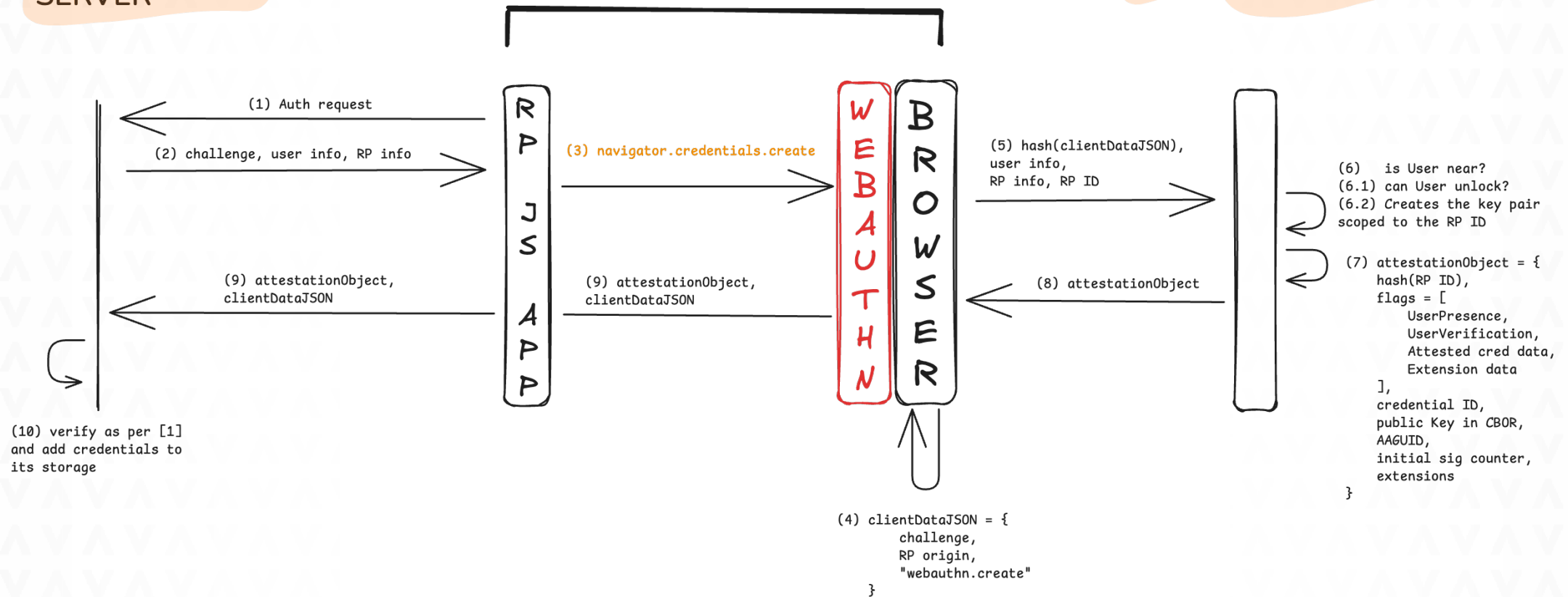
Registration

19

Relying Party
SERVER

Client

Authenticator



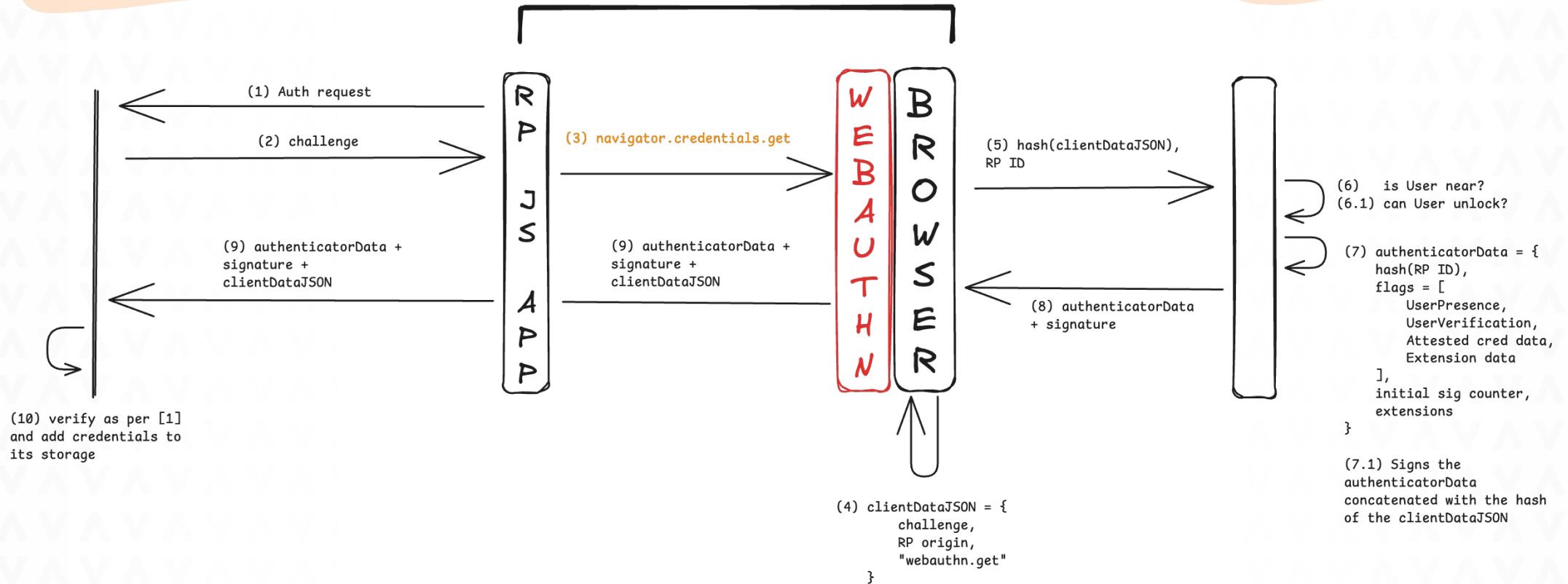
Authentication

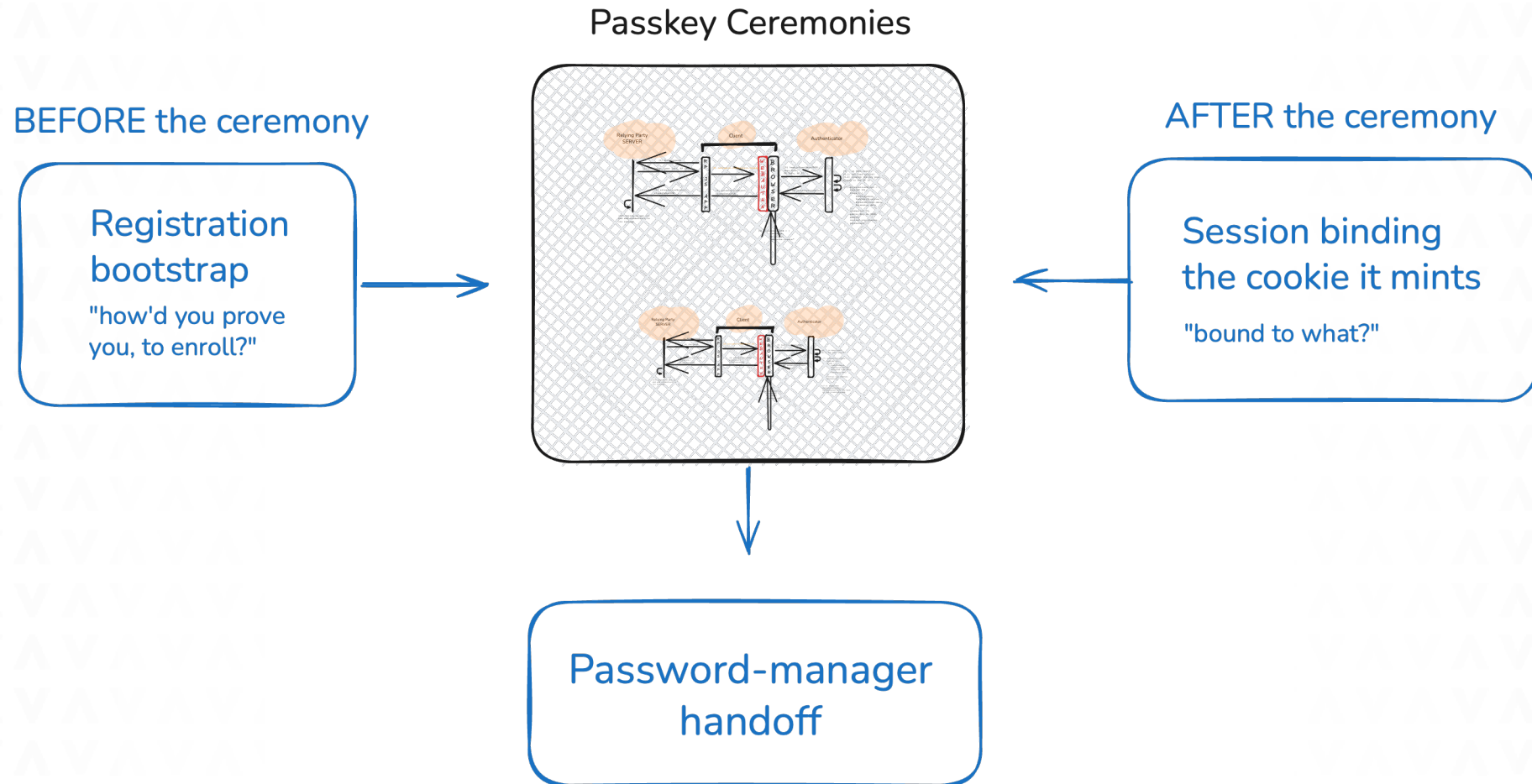
20

Relying Party
SERVER

Client

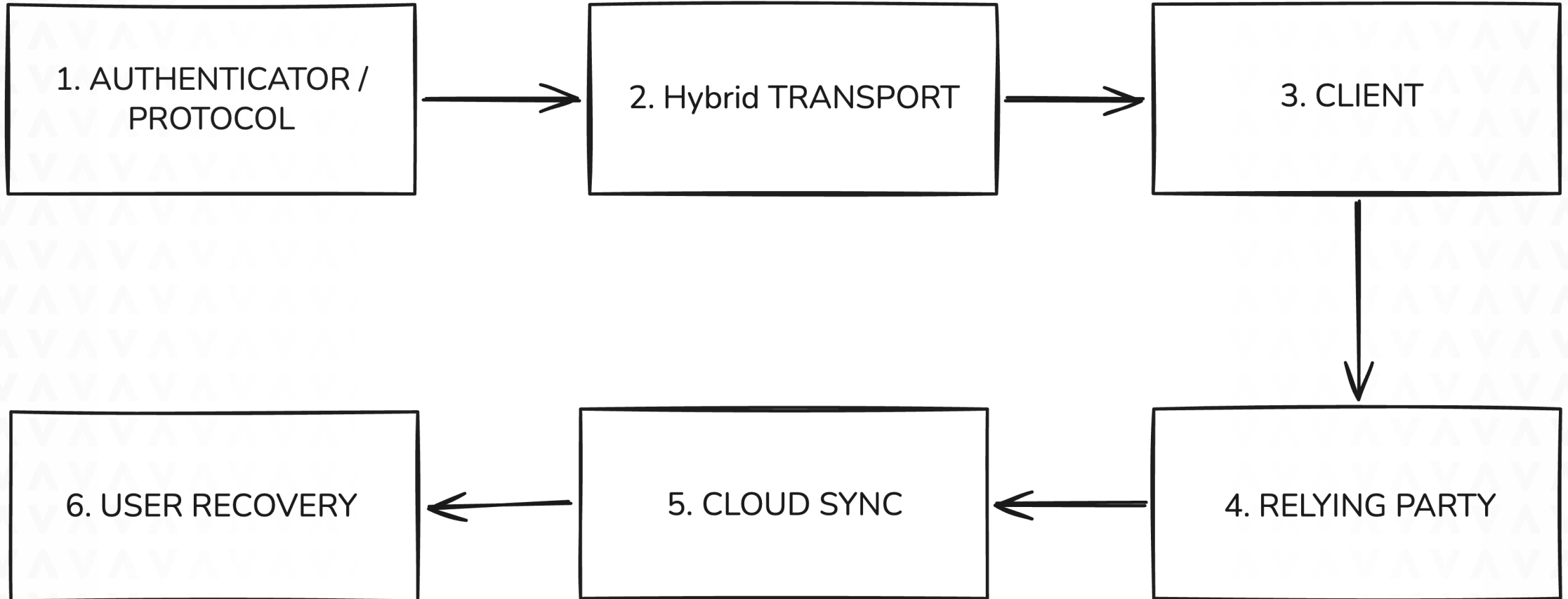
Authenticator





Our map

22



Two planes

23

RESEARCH

- ◆ Pick one actor, go deep.
- ◆ Mint CVEs / Odays and POCs

ENGAGEMENT

- ◆ A pentester or red-teamer.
- ◆ Find out what was actually shipped following a methodology.

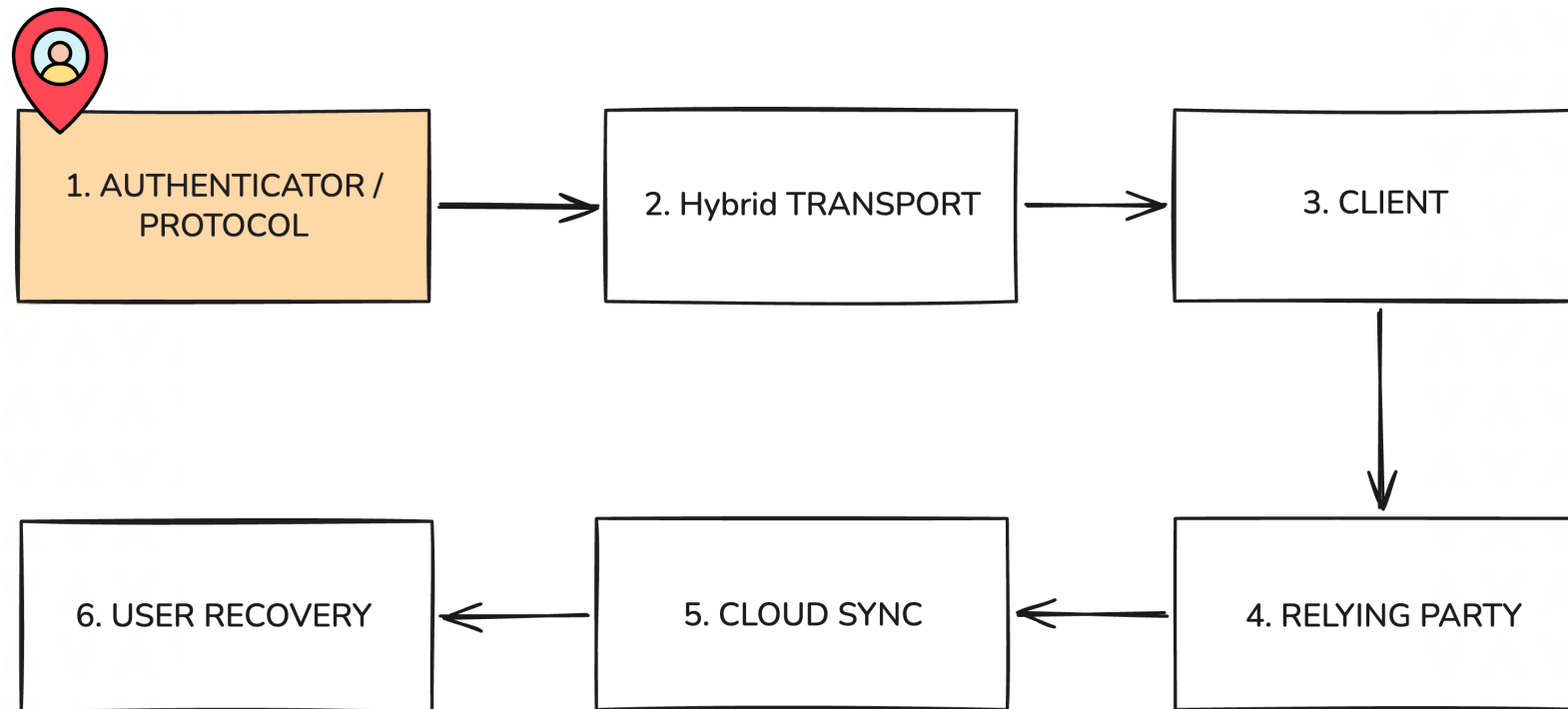
The attack surface

metal to cloud

1. Authenticator and Protocol

Researcher's turf

You are here



Research

Engagement

WebAuthn + CTAP

WHO

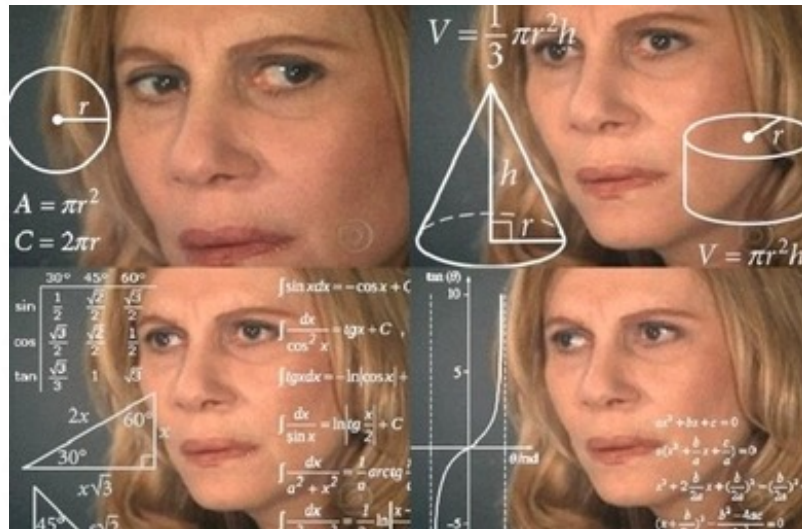
- ◆ protocol designers and academics
- ◆ FIDO Alliance, W3C

WHAT

- ◆ prove the ceremony resists replay, MitM, forgery
- ◆ keep its privacy properties, endpoints assumed honest

HOW

- ◆ symbolic model checkers (ProVerif, Tamarin)
- ◆ hand-built computational proofs
- ◆ grab the specs, and go down the rabbit-holes



The protocol holds

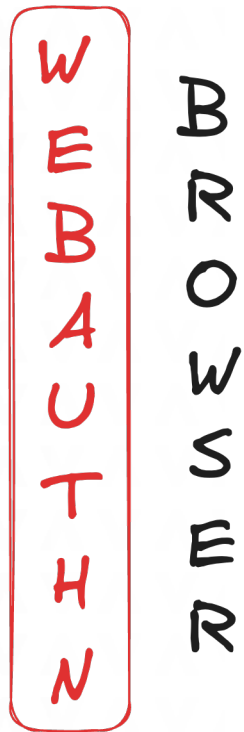
28

- ◆ Computational proofs against forgery and replay
 - ◆ [Barbosa et al. \(CRYPTO 2021\)](#)
 - ◆ [Bindel, Cremers, Zhao \(IEEE S&P 2023\)](#)
 - ◆ Revisited recently as [Barbosa et al. \(PoPETs 2025\)](#)
- ◆ Studies on vertical aspects
 - ◆ Attestation soundness by [Bindel, Gama, Guasch, Ronen, ASIACRYPT 2023](#)



CTAP

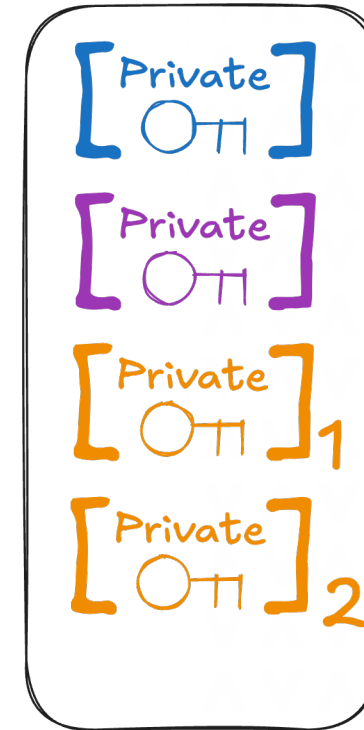
Client



CTAP

USB / NFC / BLE

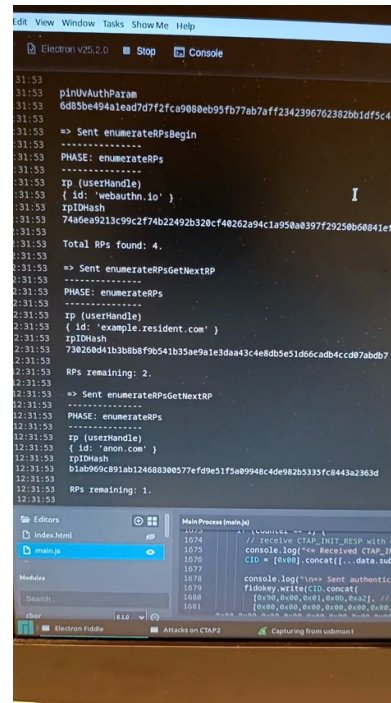
Authenticator



CTRAPS



CI1 - factory reset



AC1 - credential deletion



CI2 - user tracking

- ♦ [CTRAPS paper \(Casagrande and Antonioli, EuroS&P 2025\)](#) - [DEF CON 33 talk](#)
- ♦ Toolkit: github.com/Skiti/CTrAPs

Not all authenticators are equal

	PLATFORM (TPM, Secure Enclave)	ROAMING (key, phone)
FIRST-PARTY	iOS keychain, Windows Hello	Apple / Google phone
THIRD-PARTY	Microsoft Authenticator	YubiKey, Bitwarden

hardware = secure-element backed

software = could fake user-pres / user-verif

Hardware authenticators under attack

WHAT

- ◆ extract the private key
 - ◆ [Ninjalab: Titan](#)
- ◆ clone the authenticator
 - ◆ [Ninjalab: EUCLEAK](#)

HOW

- ◆ side-channel (EM/power to ECDSA nonce)
 - ◆ [EUCLEAK \(ePrint\)](#)
- ◆ fault injection
 - ◆ [NDSS 2024](#)
- ◆ invasive probing (decap, microprobing)
 - ◆ [Ledger Donjon](#)
- ◆ firmware and supply-chain analysis
 - ◆ [NDSS 2024](#)

Software authenticators' anarchy

User Verification

The following list of passkey providers have not implemented [User Verification](#) in a spec-compliant manner.

Provider	Architecture	uv=required	uv=preferred
1Password	Extension	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV
1Password	Native	✓ Performs UV	✓ UV flag accurate
Bitwarden	Extension	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV
KeepassXC	Extension	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV
Okta Personal	Extension	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV
Okta Personal	Native	✓ Performs UV	✓ UV flag accurate
Proton Pass	Extension	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV
Proton Pass	Native	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV
Strongbox	Native	✗ Handles request without performing UV, sets UV true	✗ Sets UV true without performing UV

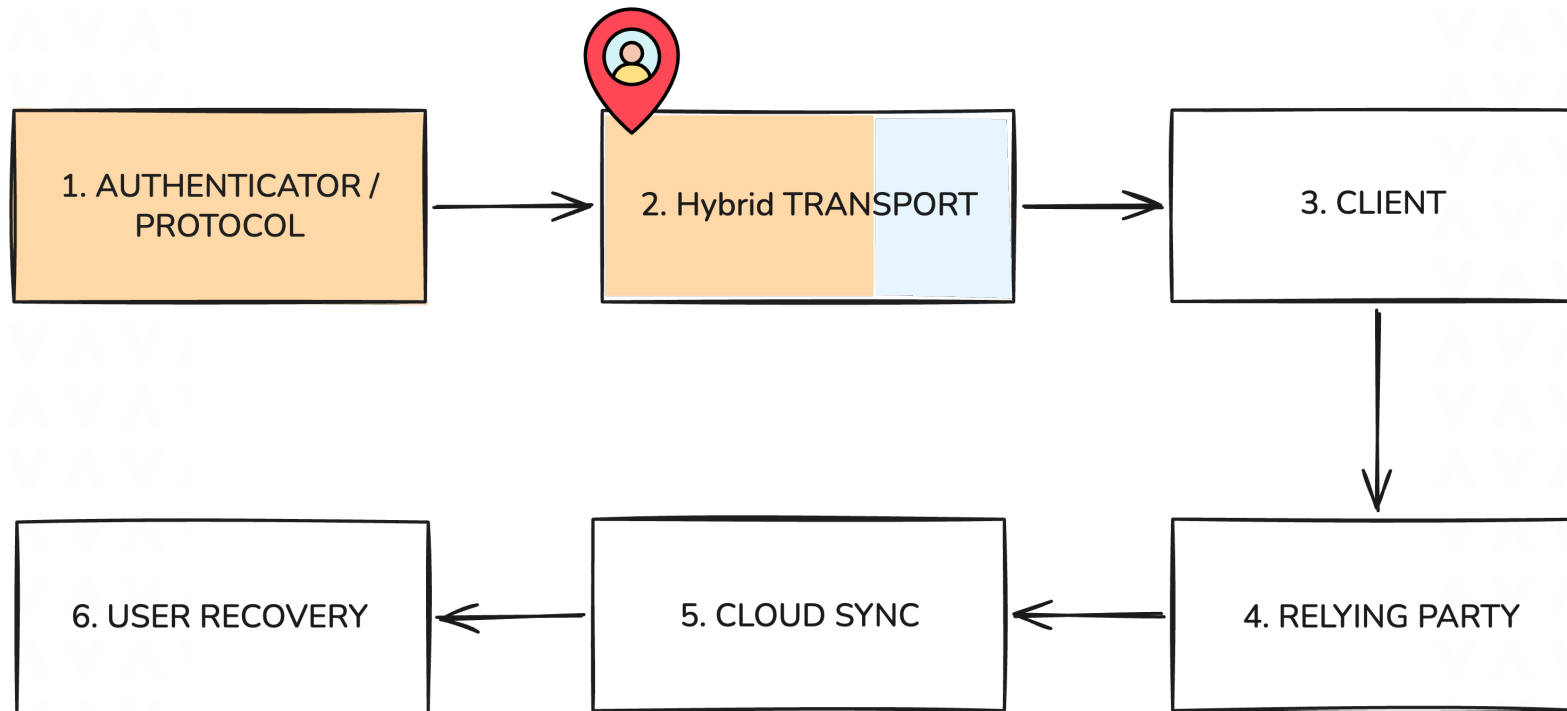
Architecture: **Extension** = web browser extension, **Native** = OS native app using provider APIs

"The following passkey providers have not implemented User Verification in a spec-compliant manner." [↗ passkeys.dev, known issues](#)

2. Hybrid transport

Cross-Device Authentication (CDA)
BLE + WebSocket, or BLE-only

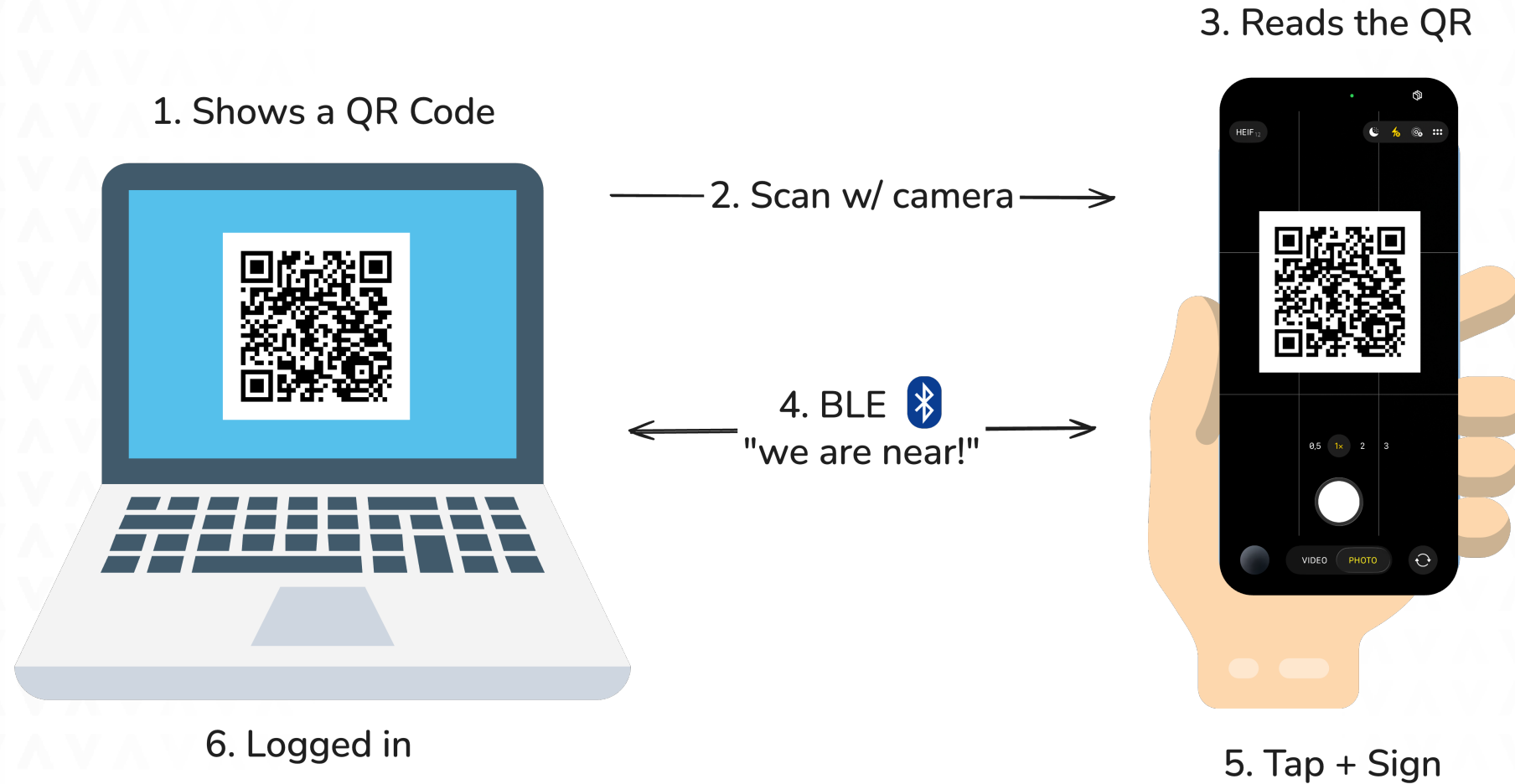
You are here



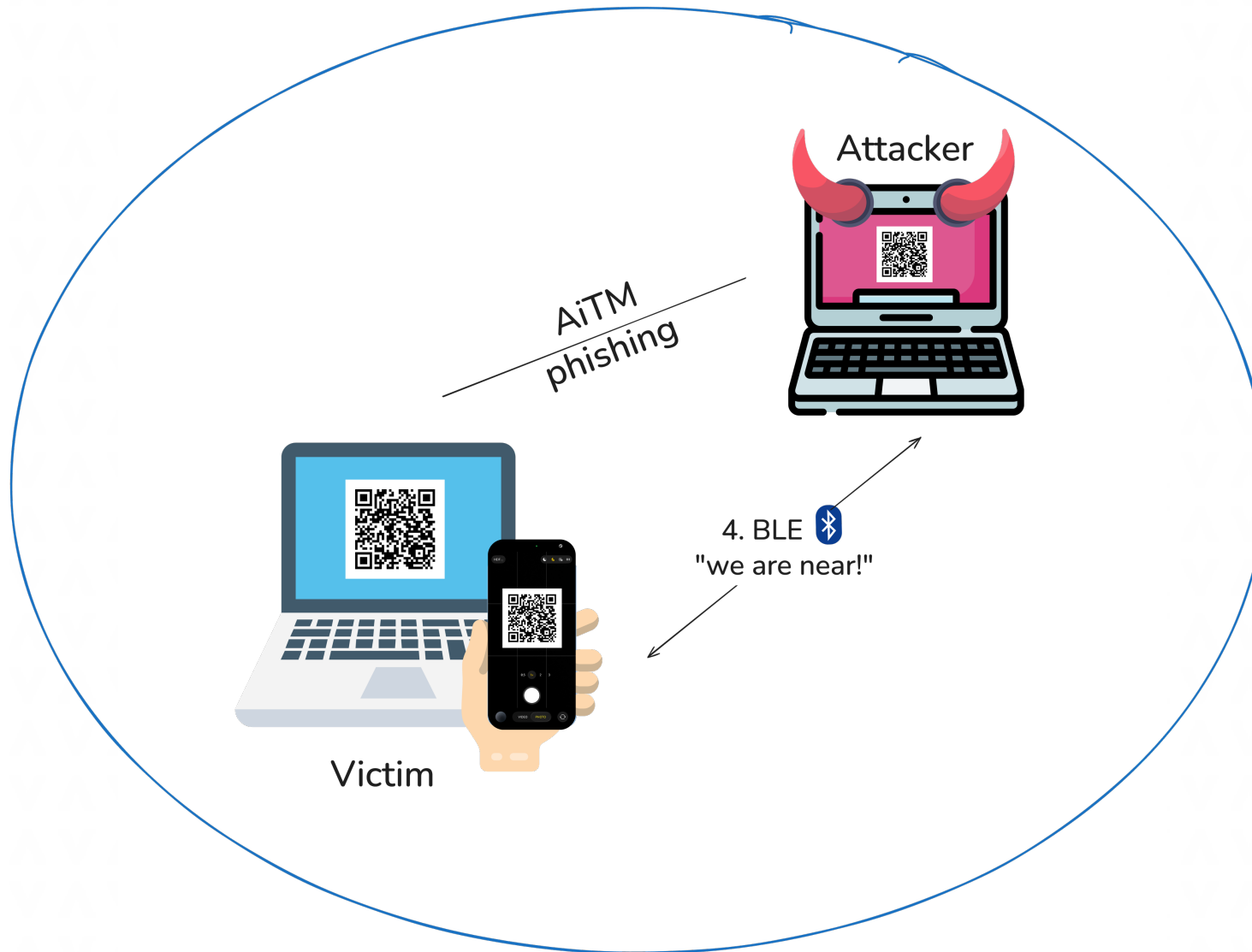
Research

Engagement

Co-location **vs** intent

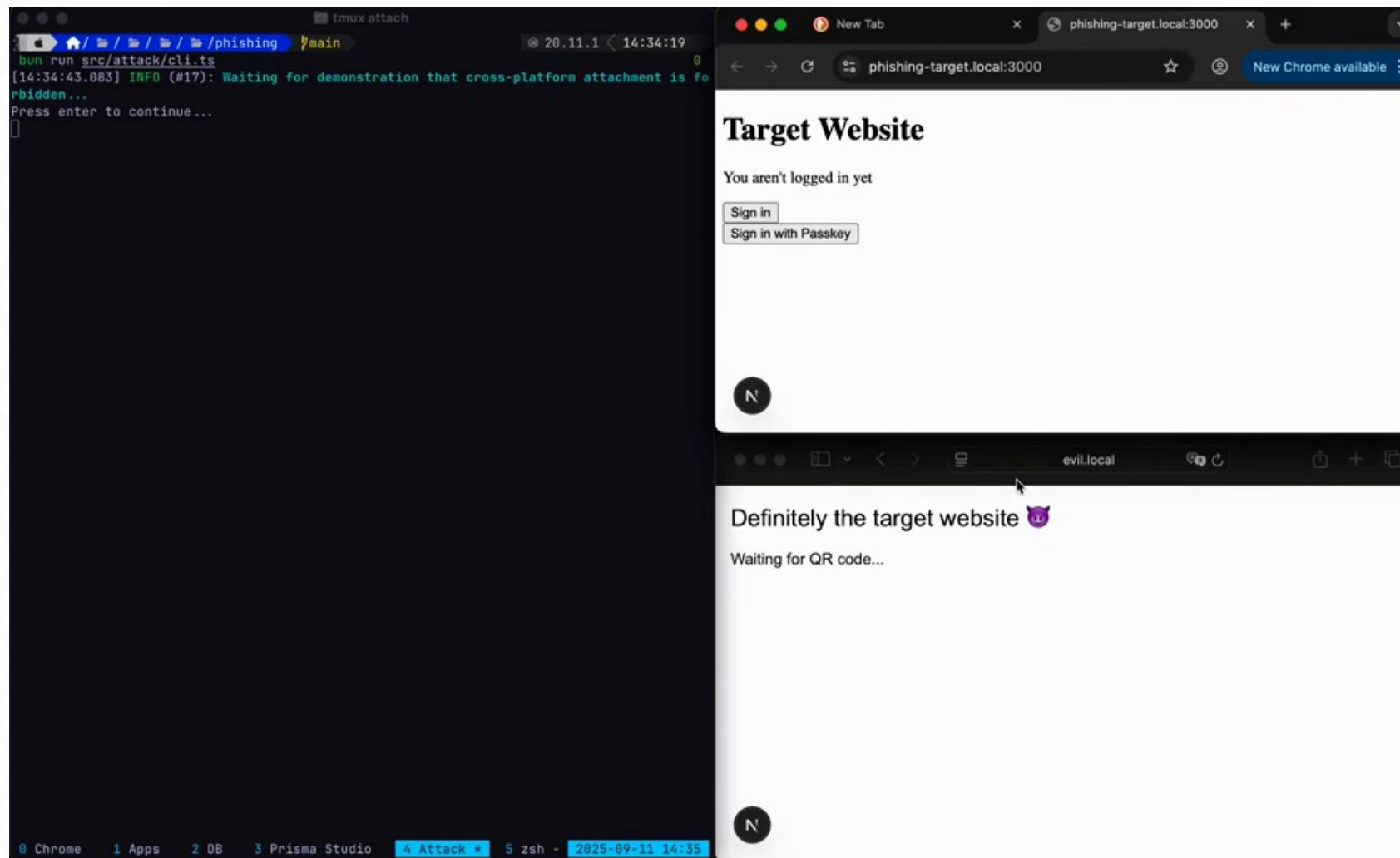


36



BLE distance

PoC



Attacker's laptop



38



Victim's laptop

Demo: "Phishing for Passkeys" - M. Kuckuk, [inovex 2025](#)

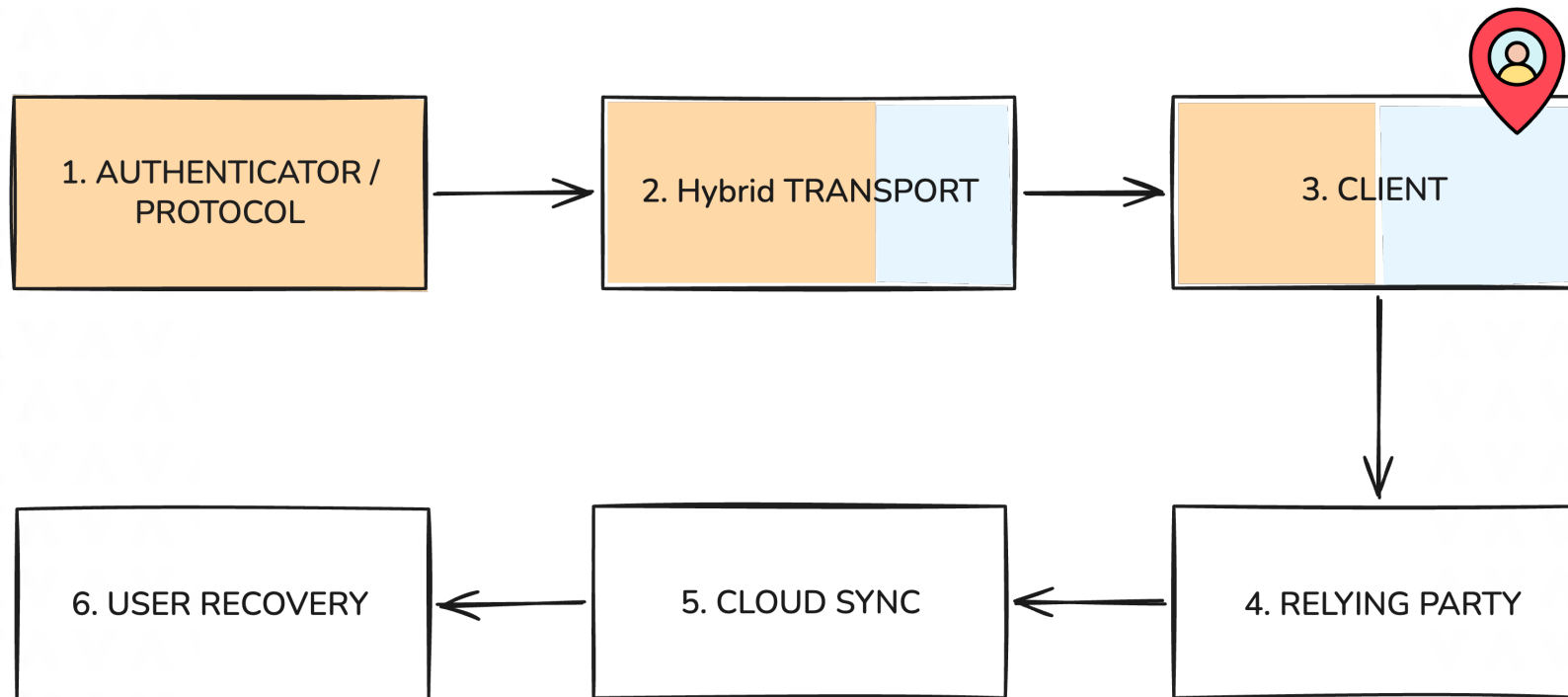
Still open in 2026

- ◆ **HiPass** measured the QR relay: 100% over 300 trials, 65s QR window across 10 major RPs ([↗ Kim et al., IEEE Access 2025](#))
- ◆ **FIDO URI intent injection**: fixed in mobile browsers, but a father of the mobile FIDO-URI attack class ([CVE-2024-9956](#), [↗ Righi 2025](#))
- ◆ Proximity still stops **remote** attackers: PoisonSeed relayed remotely and failed at BLE ([↗ Expel retraction](#))
- ◆ Co-located, it's workable: plant BLE boxes in range, offices/airports/conferences ([↗ Kniep 2025](#))
- ◆ **No RP-side tell**: through CTAP 2.3 (Feb 2026), an RP still can't distinguish a relayed hybrid ceremony from a real one ([↗ FIDO spec](#))

3. Client

and Client-Side attacks

You are here



Research

Engagement

Own the front door

WHO

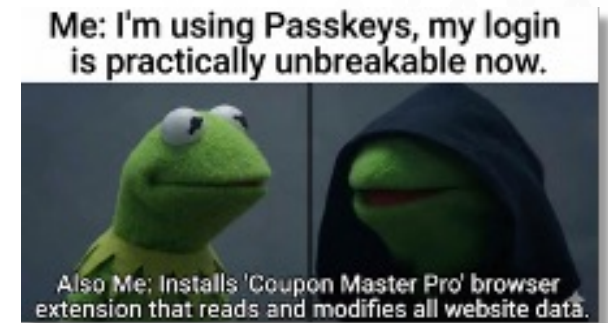
- ◆ browser vendors (Chrome, Firefox, Safari)
- ◆ extension devs (password managers)
- ◆ high-assurance RPs (fintech, gov)

WHAT

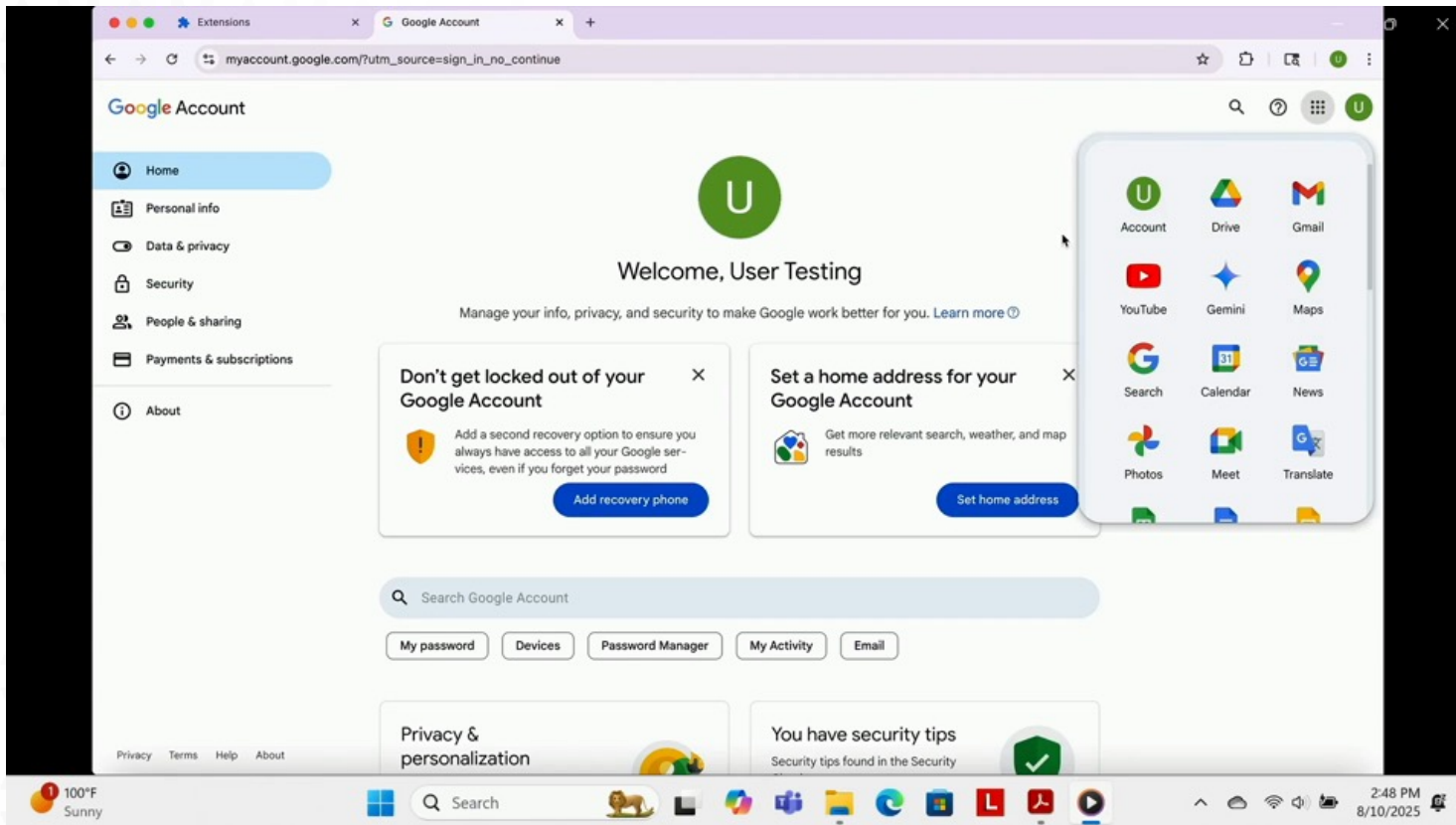
- ◆ UI transparency + user-consent awareness
- ◆ WebAuthn API override (activeTab)
- ◆ piggybacking
- ◆ risk-based-auth bypass resilience

HOW

- ◆ extension fuzzing (PoC malicious extensions)
- ◆ browser instrumentation (hook the API)



Attacker JavaScript forged a live Gmail passkey

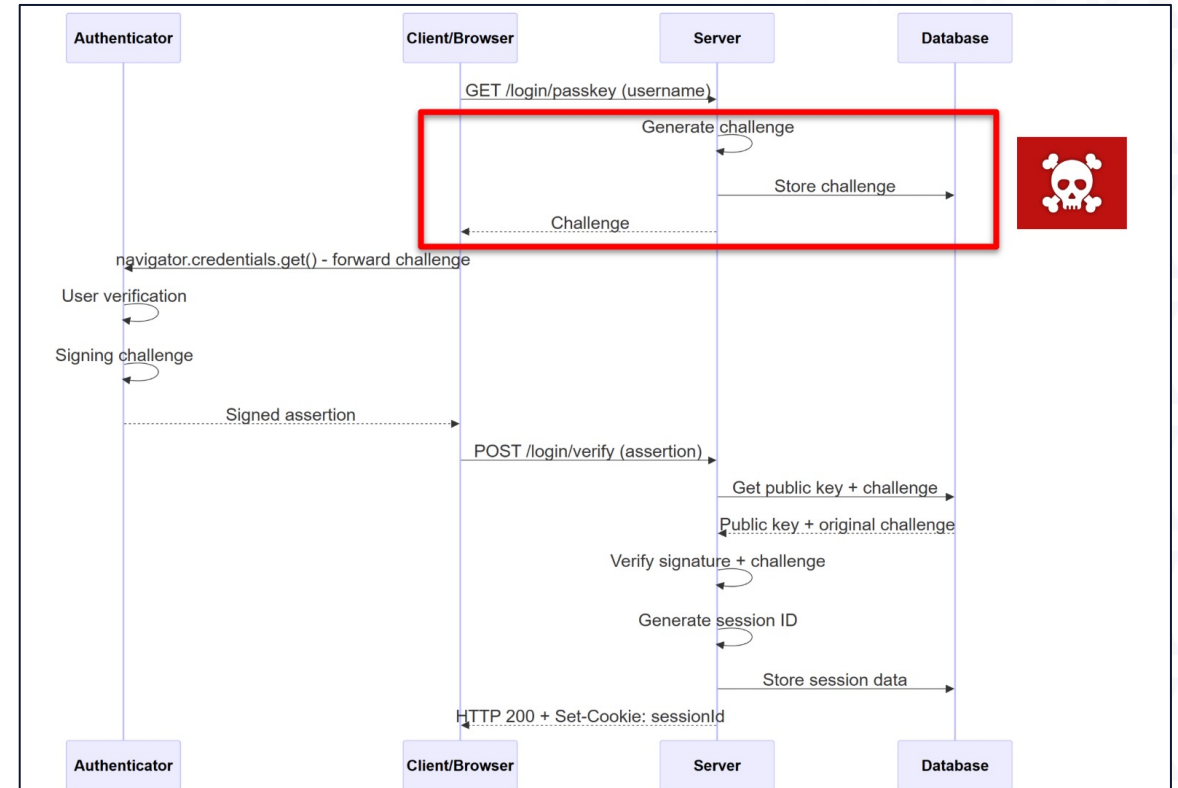
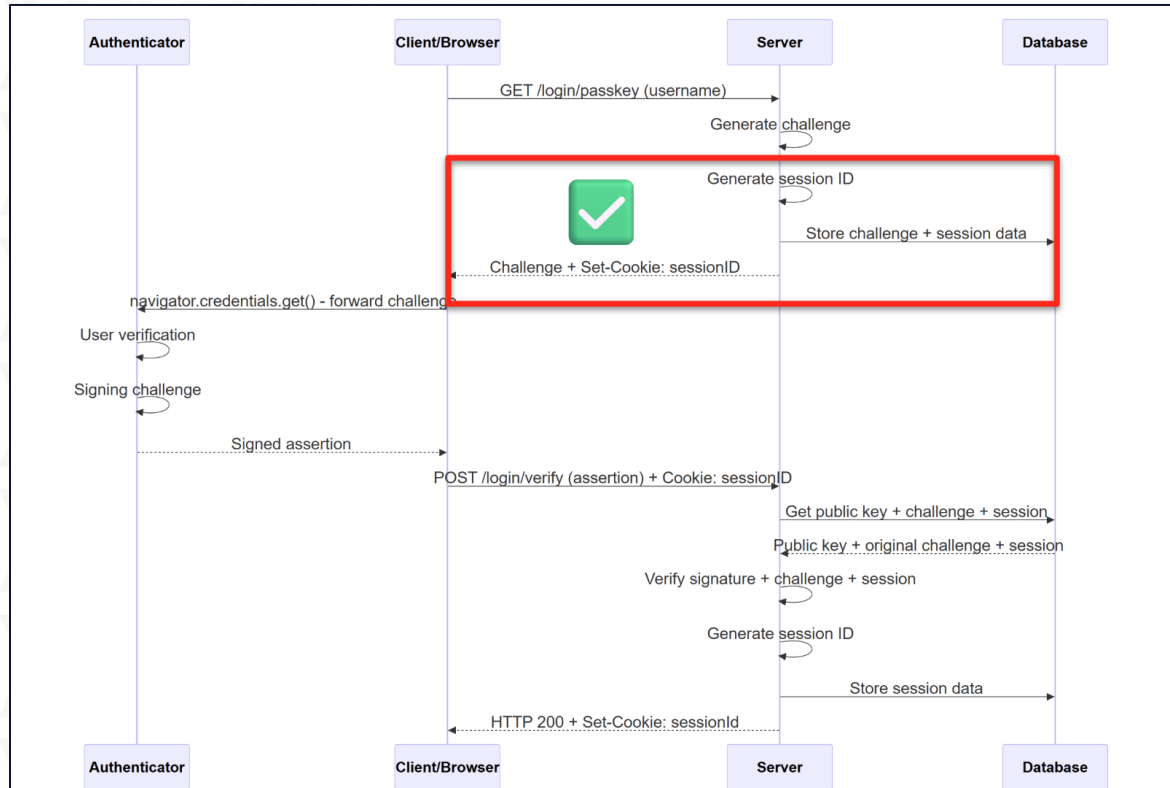


43

- ◆ attacker.passkey.tool
- ◆ [Passkey Raider](#)
- ◆ ...
- ◆ [Passkey Editor](#)

Demo: SquareX, "Passkeys Pwned" - DEF CON 33 2025 (sgrx.com/passkeys-pwned)

Signed Assertion Hijacking



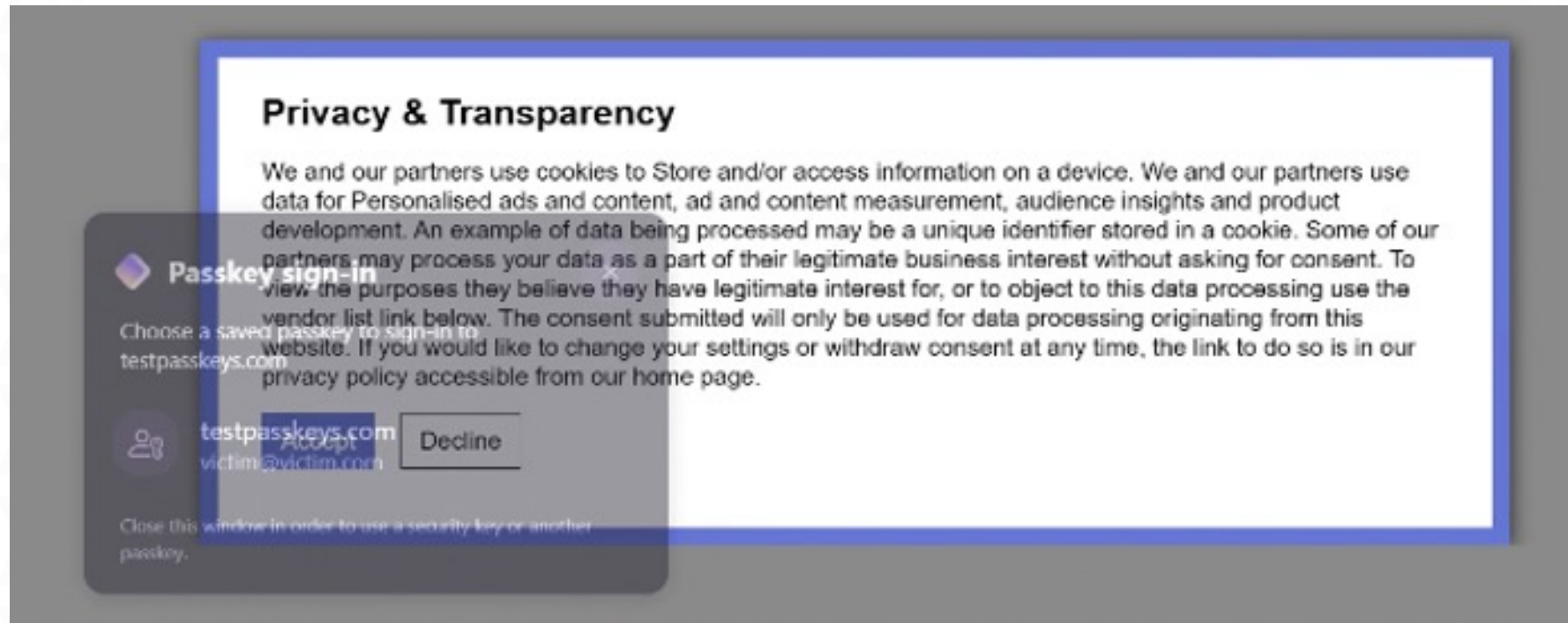
➤ [Marek Toth, "DOM-based Extension Clickjacking", DEF CON 33 2025](#)

DOM-based Extension Clickjacking

1. The attacker **finds XSS** on a server **where passkeys are used**
2. **Inject JS malware**
 1. **Redirect signed assertion** to attacker's controller server

DOM-based Extension Clickjacking

- ◆ **Hide passkey dialog** UI injected by password manager (uses DOM-based extension clickjacking technique)



46

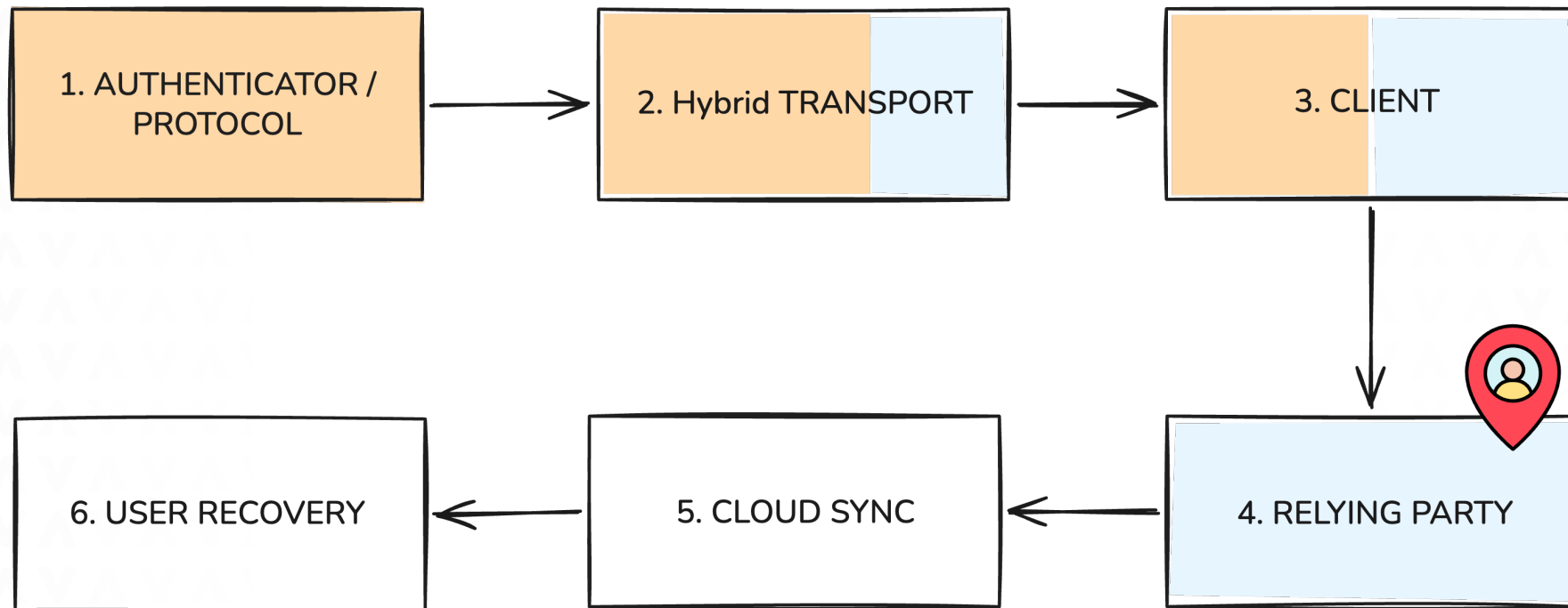
DOM-based Extension Clickjacking

4. The **victim** visits the URL with XSS vulnerability and **clicks once**
5. The attacker obtains the signed challenge
6. The **attacker sends the signed assertion** from their server **to the auth server**

47

4. Relying party

You are here

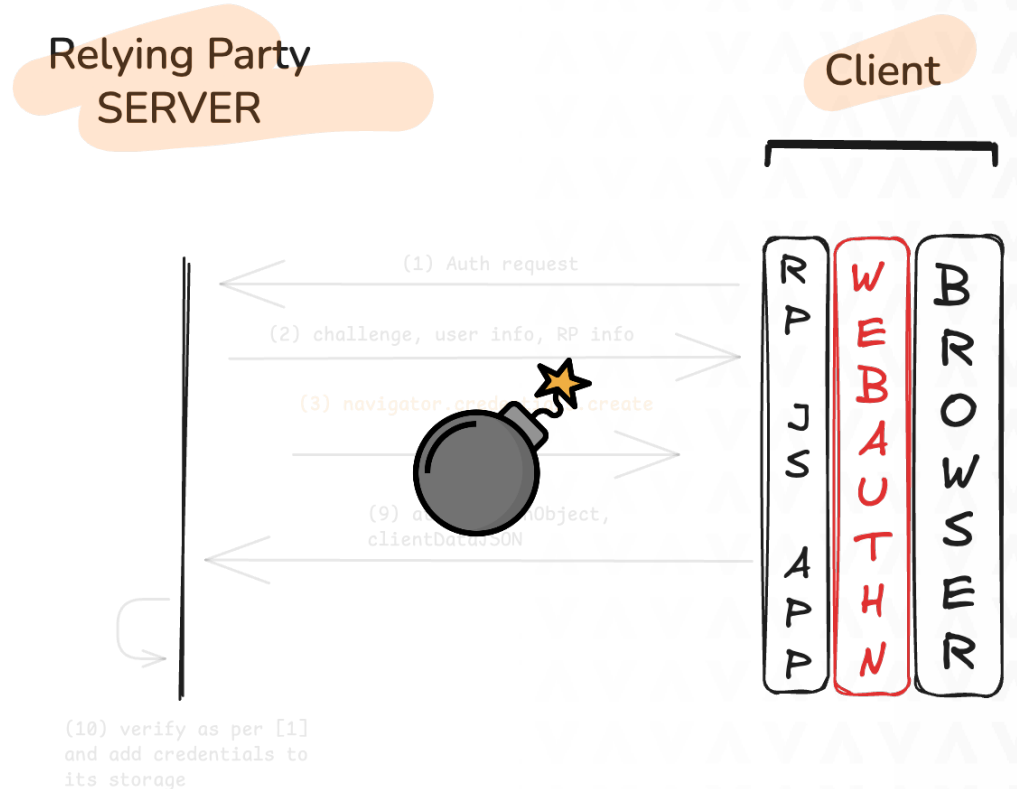


Research

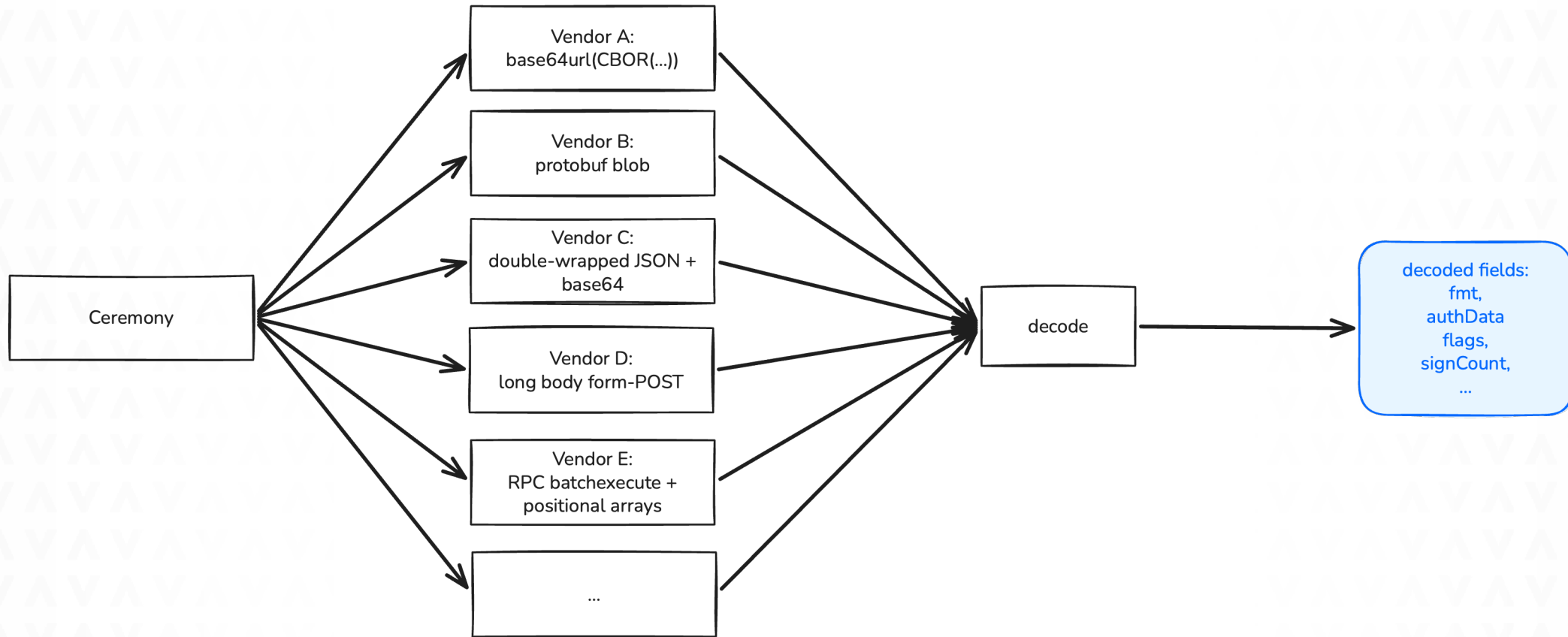
Engagement

On a high level

- ◆ Trigger the ceremonies and analyze.
- ◆ Tampering, tampering, tampering
 - ◆ Fuzzing
 - ◆ Monkey tests, checklists
 - ◆ More nuanced assumptions
- ◆ [Jannett et al., "State of Passkeys," 2026](#)
 - ◆ Actively tested 103 RPs
 - ◆ 103 vulnerable to at least one server-side attack.
 - ◆ 18 critical, 53 high.



Decoding problem



Google SSO 75.2%, navigator.credentials-only 82.3%, known JS lib 18.6% [Census: Bhardwaj & Sastru, PAM 2026](#)

Microsoft

mysignins.microsoft.com

```
POST /api/post/newfido
Content-Type: form-urlencoded

canary=b0f2c1a9...
&clientDataJson=eyJ0eXB1Ijo..
&attestationObject=o2NmbXRkcGfj..
&credentialId=3EHSf9K2mQ..
&credentialDeviceType=multiDevice
&credentialBackedUp=true
&transports=internal,hybrid
&extensions=eyJjcmVkdQ..
```

FLAT FORM FIELDS

position: ~10 flat form params
decode: URL-encoded + Base64URL

GitHub

github.com

```
POST /u2f/trusted_devices
Content-Type: multipart/form-data

-----WebKitFormBoundary...
Content-Disposition: form-data;
  name="response"
{"id":"3EHSf9K2mQ..",
 "type":"public-key",
 "response":{"
   "clientDataJSON":"eyJ0..
   "attestationObject":"o2N..
 },
 "clientExtensionResults":{}}
-----WebKitFormBoundary...--
```

MULTIPART + NESTED JSON

position: spec field names, nested
decode: JSON + Base64URL

Google

myaccount.google.com

```
POST /_/.../batchexecute
Content-Type: form-urlencoded

f.req=[["GtmsU","[null,null,
null,"eyJ0eXB1Ijoi..",
"o2NmbXRkcGfj..",
["internal"],null,1,1]",
null,"generic"]]]
&at=AFehe7k9dQ..

// idx 3 = clientDataJSON
// idx 4 = attestationObject
```

POSITIONAL ARRAYS

position: fields by index in a blob
decode: URL-encoded + Base64URL

52

Decoded fields

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge": "zYJx-8mHw8wK7vC4qRseSJrDCd01yKIfZk_njXE0oeuQD7CuKUoQ2frvV0NB0JiVZSBgjUYy8vGb-0Lq-BS1wA",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "signCount": 42
  },
  "signature": "304402207BC3E1F0A2D4C6980B5E3F1A2C4D6E8F0A1B2C3D4E5F60...DDEEFF02",
  "userHandle": "6D6174746566F2D67696F7264616E6F"
}
```

You want:

- ◆ Traffic detection
 - ◆ across vendor wrappers
- ◆ Every field decoded:
 - ◆ clientDataJSON
 - ◆ authenticatorData
 - ◆ signature
 - ◆ userHandle

53

Passkey Editor

You have:

- ◆ Ceremony traffic detection
 - ◆ Registration/Authentication/Option
 - ◆ Fits every vendor
- ◆ Fields decoding:
 - ◆ clientDataJSON
 - ◆ attestationObject
 - ◆ authData
 - ◆ COSE key
 - ◆ ...

The screenshot shows a web browser window with the URL `https://webauthn.io` and the path `/registration/verification`. The status bar indicates a POST request with a 200 status code. The main content area is titled "Original request" and has tabs for "Pretty", "Raw", "Hex", and "Passkey Editor". The "Passkey Editor" tab is selected, and a "Wrap" checkbox is checked. The JSON data is displayed with syntax highlighting. The JSON structure includes fields for clientDataJSON, attestationObject, authenticatorData, attestedCredentialData, and credentialId.

```
{
  "clientDataJSON": {
    "type": "webauthn.create",
    "challenge": "zYJx-8mHw8wK7vC4qRseSJrDCd01yKIvZk_njXE0oeuQD7CuKUoQ2frvV0NB0JiVZSBgjUYy8vGb-0Lq-BS1wA",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "attestationObject": {
    "attestationStatement": {
      "format": "none"
    },
    "authenticatorData": {
      "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
      "extensions": {},
      "signCount": 1,
      "flags": {
        "userPresent": true,
        "userVerified": true,
        "backupEligible": false,
        "backupState": false,
        "attestedCredentialData": true,
        "extensionDataIncluded": false
      }
    },
    "attestedCredentialData": {
      "aaguid": "01020304-0506-0708-0102-030405060708",
      "coseKey": {
        "keyType": "OKP",
        "algorithm": "EdDSA",
        "curve": "Ed25519",
        "x": "4E7DB02BDD2B21364502D0177930029CEBB39BC8F8BC42CB8997EAB7CDDFEB2D"
      },
      "credentialId": "744D5E363DA57801D203145B4736F9D4653ED2104CE3C4E6C6941C75ED7E0A7F"
    }
  },
  "fmt": "none"
}
```

54

[Profiles](#)
[Guide](#)
[About](#)

- ☐ Default (SimpleWebAuthn / generic) [not active]
- ☒ **webauthn.io (Duo py_webauthn)**
- ☐ passkeys-debugger.io (Next.js action) [not active]
- ☐ passkeys.io (Hanko) [not active]
- ☐ demo.yubico.com [not active]
- ☒ **webauthn.lubu.ch [auto-plant] [re-sign]**

Add

Copy

Delete

Restore built-ins

☒ Enabled ☐ Auto-plant ☐ Auto re-sign

```
id: webauthn.io      name: webauthn.io (Duo py_webauthn)
```

host match: **EXACT**  webauthn.io

default signing alg: EdDSA (-8) plant attestation: None

verify URL: CONTAINS ▾ /registration/verification method: POST

clientDataJSON ☒ path ☐ regex response.response.clientDataJSON

attestationObject ☒ path ☐ regex response.response.attestationObject

authenticatorData ☒ path ☐ regex response.response.authenticatorData

credentialId ☒ path ☐ regex

☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded [webauthn.create](#) decoded ▸

☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded `fmt=none` · Ed25519 key **decoded** ▶

☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded UP UV AT · signCount 1 decoded ▶☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded decoded ▶

verify URL: CONTAINS ▾ /authentication/verification method: POST

clientDataJSON ☒ path ☐ regex response.response.clientDataJSON

authenticatorData ☒ path ☐ regex `response.response.authenticatorData`

signature ☒ path ☐ regex response.response.signature

userHandle ☒ path ☐ regex response.response.userHandle

credentialId ☒ path ☐ regex response.rawId

☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded webauthn.get decoded ▶☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded UP UV · signCount 2 decoded ▶☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded Ed25519 · 64B decoded ▶

☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded 37B · "webauthnio-asdadsada..." **decoded** ▶

☒ Auto ☐ Raw ☐ Base64 ☐ Base64URL ☐ URL encoded 32B · 893025b4e178... decoded ▶

Registration body (paste the reg-verify request body):

```
{
  "username": "defcon34",
  "response": {
    "id": "iTAlt0F4snCezIL1FNoclo0w3Ez4vDkLQIkxrqURt8Q",
    "rawId": "iTAlt0F4snCezIL1FNoclo0w3Ez4vDkLQIkxrqURt8Q",
    "response": {
      "attestationObject": "o2NmbXRxkb9uZwdhdHRTdG10eGhhdxRORGF0YVlBdBkqbKJnC90s1SSsyDPQCylMGpUKA5fyklc2CEHVBFAAAAAQCAwQBgcIAQIDBAUGBgwgAIIkwJBTheLjWnsyC9RTaHJaNMNxM-Lw5C0CJMae6LEbfEpAEBAycgB1FYIP8CuPmFEqE1ebDHTvdIshBJwS2K3FKP2f-eI30HLc_2u",
      "clientDataJSON": "eyJ0eXB1Ijoia2ViYXV0aG4uY3JlYXRlIiwia2VhbGxlbmdlIjoiaSmV2VBETMt5rUnRCOUlsTnJR0lryVmFMZGFNTmJSeUJha2tta3plVEtyVFNR2hCQks5YWwBRSDFDU3NDY1N0RkJaTDNDdFFtY0lmMUE0anlrQVLGZHciLCJvcmlnaW40IjoiaHRwcovL3dlYmF1dGhuLmLiIiwia3Jvc3NPcmIlnaW40ImZhbnlfQ",

```

Authentication body (paste the auth-verify request body):

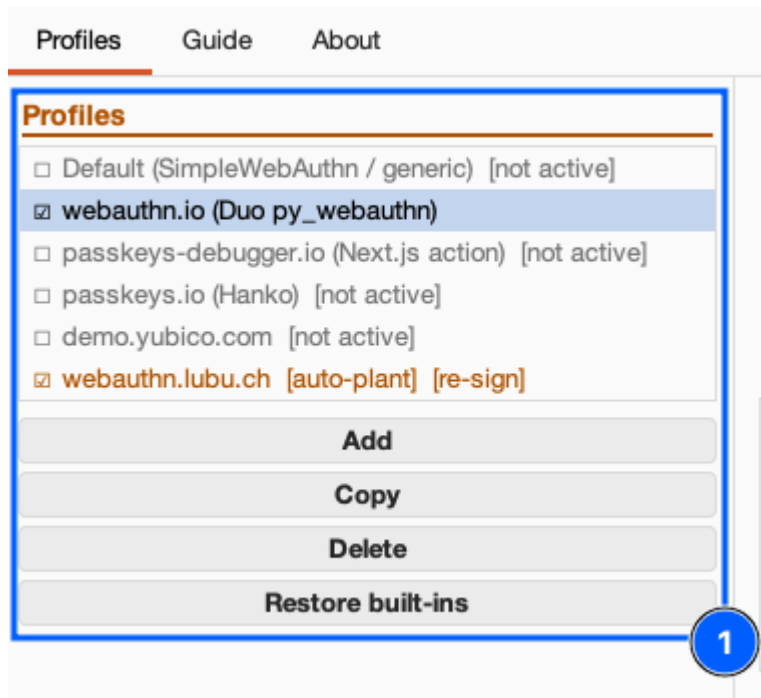
```
{
  "username": "defcon34",
  "response": {
    "id": "iTAlt0F4snCezIL1FNoclo0w3Ez4vDkLQIkxrqURt8Q",
    "rawId": "iTAlt0F4snCezIL1FNoclo0w3Ez4vDkLQIkxrqURt8Q",
    "response": {
      "authenticatorData": "dKbqkhPjNC90sIS5syDPQCYqLMGPUKA5fykLC2CEHvFAAAAAAg",
      "clientDataJSON": "eyJ0eXB1Ijoiaid2ViYXV0aG4uZ2V0IiwiaWY2hhbGxlbmdlLjoiaWR0GRGTR2aE1iVlkySGVwdV9PS0lPYzdCR3BzSkc4Z3ZEdEd2fNwXpWlplN0FNKanfNa1dScGN4RnRkdFNXZ5kZ0FJ0FpWC1ydVRCdm1Kc0JC01kanCIJCjvcmlnaW4iOiJodHRwcovL3dlYmFidGhulmVlIiwiaWY2Z3Vvc3NPcmIlnaW4iOmZhbHNlfQ",
      "signature": "xMqa0xhmJ_68vH4KiUmougIZ1FGnJeVvvFVkuVjRhRwGn9N-jS9Woa99BTY51-V5711UoG5MJ6_0ADJDF8orDQ",
    }
  }
}
```

Check

Prettify JSON

Save profile

all 8 configured field(s) extract cleanly



id: webauthn.io name: webauthn.io (Duo py_we

host match: EXACT webauthn.io

default signing alg: EdDSA (-8) plant attestation: None

2

Registration

verify URL: CONTAINS ▾ /registration/verification method: POST

clientDataJSON	☒ path ☐ regex	response.response.clientDataJSON
attestationObject	☒ path ☐ regex	response.response.attestationObject
authenticatorData	☒ path ☐ regex	response.response.authenticatorData
credentialId	☒ path ☐ regex	

Authentication

3

Authentication

verify URL: CONTAINS method: POST

clientDataJSON	☒ path ☐ regex	response.response.clientDataJSON
authenticatorData	☒ path ☐ regex	response.response.authenticatorData
signature	☒ path ☐ regex	response.response.signature
userHandle	☒ path ☐ regex	response.response.userHandle
credentialId	☒ path ☐ regex	response.rawId

Sample bodies

4

☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	weba
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	fmt=n
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	UP UN
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	der
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	weba
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	UP UN
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	Ed255
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	37B ·
☒ Auto	☐ Raw	☐ Base64	☐ Base64URL	☐ URL encoded	32B ·

5

Tamper + re-sign

- ◆ Edit any field, re-encode in place
- ◆ Any edit breaks the signature
 - ◆ re-plant the key
 - ◆ re-sign on passthrough

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. A table at the top lists intercepted requests, with the first one highlighted: 13:28:50 6 Jul 2026, HTTP → Request, POST, https://webauthn.io/authentication/verification.

Below the table, the 'Request' tab is active, showing the raw JSON of the intercepted request. The JSON is as follows:

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge": "LXi6duZCUvut6pxxch4hKFEuyLMuK7tQH5QfsSs7pts941y_MRf6l-_JI73_y1FB2jJ7CKW-jtsZMmGUpywSog",
    "origin": "https://evil.webauthn.io",
    "crossOrigin": false
  },
  "authenticatorData": {
    "rpIdHash": "22EA1500B25722FD46E8A8653E37107E07805890FCD719A92A9EF5CAD9693AFF",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature": "26C90A162BA1528B08AAAEDEFF454A8A3ACAE7041FAD616A9A52CDAF536D0FC3D6B4BFC5799D3AF4079876C667FD7A1A71EBF4660F3C8D8E84DB270885E3C90F"
}
```

At the bottom of the interface, a 're-signed (EdDSA)' section shows the modifications made:

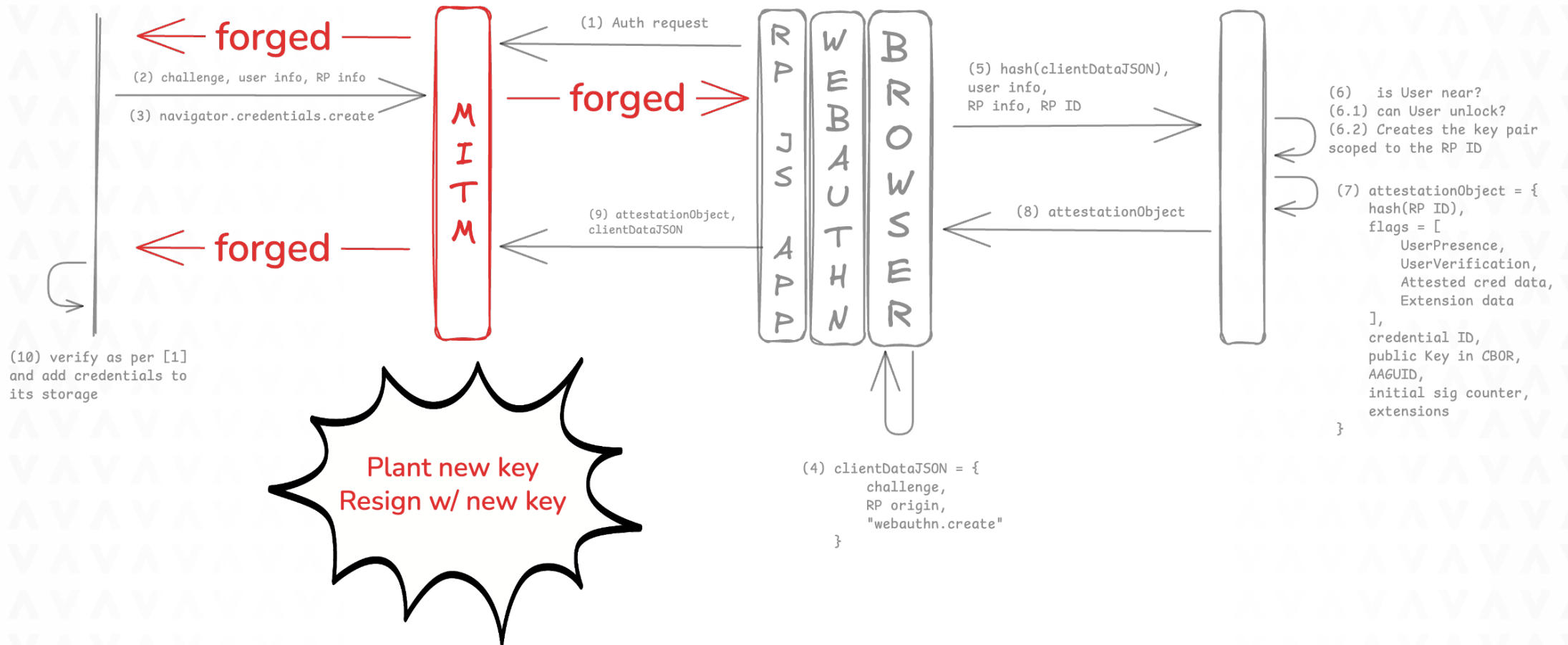
- rpId = <edited>
- origin = https://evil.webauthn.io

Buttons at the bottom right include 'Re-sign with our key', 'Clear edits', and 'Apply edits + re-sign'.

Relying Party SERVER

Client

Authenticator



Not the first tool

- ◆ [↗ webauthn-cbor](#) decodes only, no attacks.
 - ◆ [↗ passkey-scanner](#) passive detection.
 - ◆ [↗ passkey-raider](#) tampers, but by hand.
 - ◆ [↗ Burp_FIDO2](#) handles wrappers, rough on production traffic.
 - ◆ [↗ Grafnetter, Pass-the-Passkey](#) open-sourced at Black Hat this week.
 - ◆ [↗ Passkeys.Tools \(Jannett\)](#) emulates browser AND authenticator, tampers every field at scale
-
- ◆ **Passkey Editor's lane:** Burp-native, attack dropdown live across Intercept, history, and Repeater, auto re-sign on passthrough, decoding paired with tampering.

clientDataJSON

```
POST /authentication/verification HTTP/2
Host: webauthn.io
Cookie: sessionId=cicj8qb34pnwr50xeyuvyt5pgi690q43
Content-Length: 865
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/149.0.0.0 Safari/537.36
Content-Type: application/json
Origin: https://webauthn.io
Referer: https://webauthn.io/

{
  "username": "defcon34_demo",
  "response": {
    "id": "XYMXeBZ1HTwo15ss0IhVTw3K-Cgy_mclKkaE4Jilvq4",
    "rawId": "XYMXeBZ1HTwo15ss0IhVTw3K-Cgy_mclKkaE4Jilvq4",
    "response": {
      "authenticatorData":
        "dKhqkhPjnc90siSSsyDP0CYqLMGpUKA5fykLC2CEHvAFAAAAAg",
      "clientDataJSON":
        "eyJ0eXB1Ijojd2ViYXV0aG4uZ2V0IiwiaWY2hhbGxlbmdlIjoiz19jN0VzdEVPWnBSaDIwS
        DRlUEFTZzJiUTd0M3dJQU90SVNhSWFsNVV3SnVsWUxueUlwQmPY3k5Z1NITlBzZENhajR
        lNfd6dVJRuzBNZ05VLXRSbUEiLCJvcmlnaW4iOiJodHRwczovL3dlYmF1dGhuLmlvIiwiaW
        3Jvc3NPcm1naW4iOiMZhbnHLLCJvdGhlc9rZXlzX2Nhb9IzV9hZGRlZF9oZXJlIjoizG8
        gbm90IGNvbXBhcmUgY2xpZW50RGF0YUpTT04gYWdhaW5zdCBhIHRlbXBsYXRRLiBTZWUga
        HR0cHM6Ly9nb28uZ2wveWF1UGV4In0",
      "signature":
        "IBTQLAKniU6-XLMn1P8_FVAVaRbsmKD5SMoB5kqrAXuYGr03urUvo7e37g6EW_6hLJ7co
        STQ04jMYIO5RrsvCw",
      "userHandle": "d2ViYXV0aG5pby1kZWZjb24zNF9kZW1v"
    }
  },
  "type": "public-key",
  "clientExtensionResults": {
  },
  "authenticatorAttachment": "platform"
}
```

Raw HTTP request

Pretty Raw Hex Passkey Editor

Attacks ☒ Wrap

Flags: ☒ UP ☒ UV ☐ BE ☐ BS

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge":
      "g_c7EstEOZpRh20H4KPASg2bQ7t3wIA0NISaIa15UwJuLYLnyIpBi0cy9gSHNPsdCaj4e4WzuRQ50MgNU--tRmA"
    "origin": "https://webauthn.io",
    "crossOrigin": false,
    "other_keys_can_be_added_here": "do not compare clientDataJSON against a template.
    See https://goo.gl/yabPex"
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    }
  },
  "attestedCredentialData": {},
  "signature":
    "2014D02C02A7894EBE5CB327D4FF3F1550156916EC98A0F948CA01E64AAB017B981ABD37BAB52FA3B7B7EE0E
    845BFEA12C9EDCA124D0D388CC6083B946BB2F0B"
}
```

Passkey Editor

challenge

```
Pretty Raw Hex Passkey Editor
Attacks ▾ ☒ Wrap
Flags: ☒ UP ☒ UV ☐ BE ☐ BS
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge": "evil_challenge",
    "origin": "https://webauthn.io",
    "crossOrigin": false,
    "other_keys_can_be_added_here": "do not compare clientDataJSON against a template.
See https://goo.gl/yabPex"
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature":
"4C0944C43B98FD85B7131C127173B4FE2C6F095C8761068BB1EAC6DC326AF3F7B5B0D4F4867C56BCFE70BBDC
0DB6A00A41BEFD50BF868D4A60018656FF0ECF0F"
}
```

66

origin + rpIdHash

Pretty Raw Hex Passkey Editor

Attacks ▾ ☒ Wrap

Flags: ☒ UP ☒ UV ☐ BE ☐ BS

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge":
    "g_c7EstE0ZpRh20H4KPASg2bQ7t3wIAONISaIal5UwJulYLnYIpBi0cy9gSHNPsdCaj4e4WzuRQS0MgNU-tRmA",
    "origin": "https://evil.defcon34.xyz",
    "crossOrigin": false,
    "other_keys_can_be_added_here": "do not compare clientDataJSON against a template.
    See https://goo.gl/yabPex"
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature":
  "1AF21C1F86B8C216576A09A4575E89BB97A875E4D2A3C4AA2FB4FD20DB6E10DE43938FC12AF1A0FD83DCC704E5401D7E3F3542C19472822BEA251454E6C0960A"
}
```

origin

Pretty Raw Hex Passkey Editor

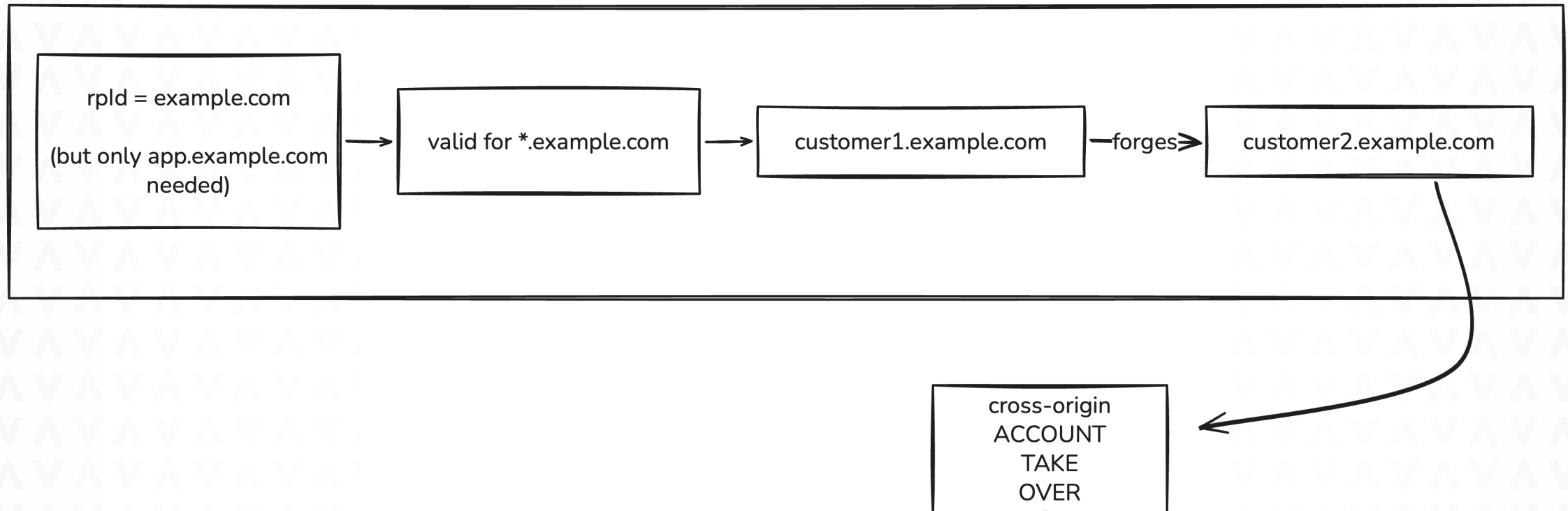
Attacks ▾ ☒ Wrap

Flags: ☒ UP ☒ UV ☐ BE ☐ BS

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge":
    "SnFcY_gy4P0--D3ueYA6PrxBH56EuW_3BcEjBahjtm9tQq8Yk8JNAeDfxN47a4wgp3yNfwPcEySw8axZXhMiqA",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "authenticatorData": {
    "rpIdHash": "00A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature":
  "396EBFC02B229F152413D4F2EBC30AE7815E4EBA939EAB38A2871F52F962153A81E207D9FFB38181B1EE20860736655A547320DC10732961492BA548E8F4C0C"
}
```

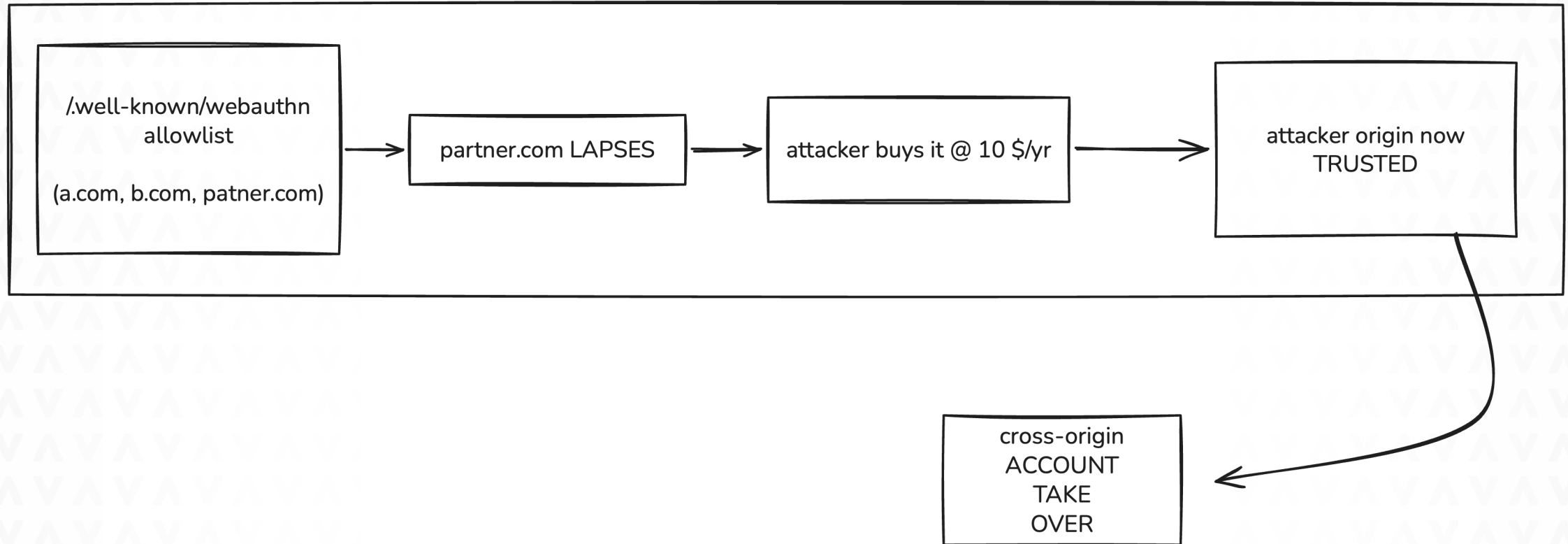
rpIdHash

Over-scoping the RPId



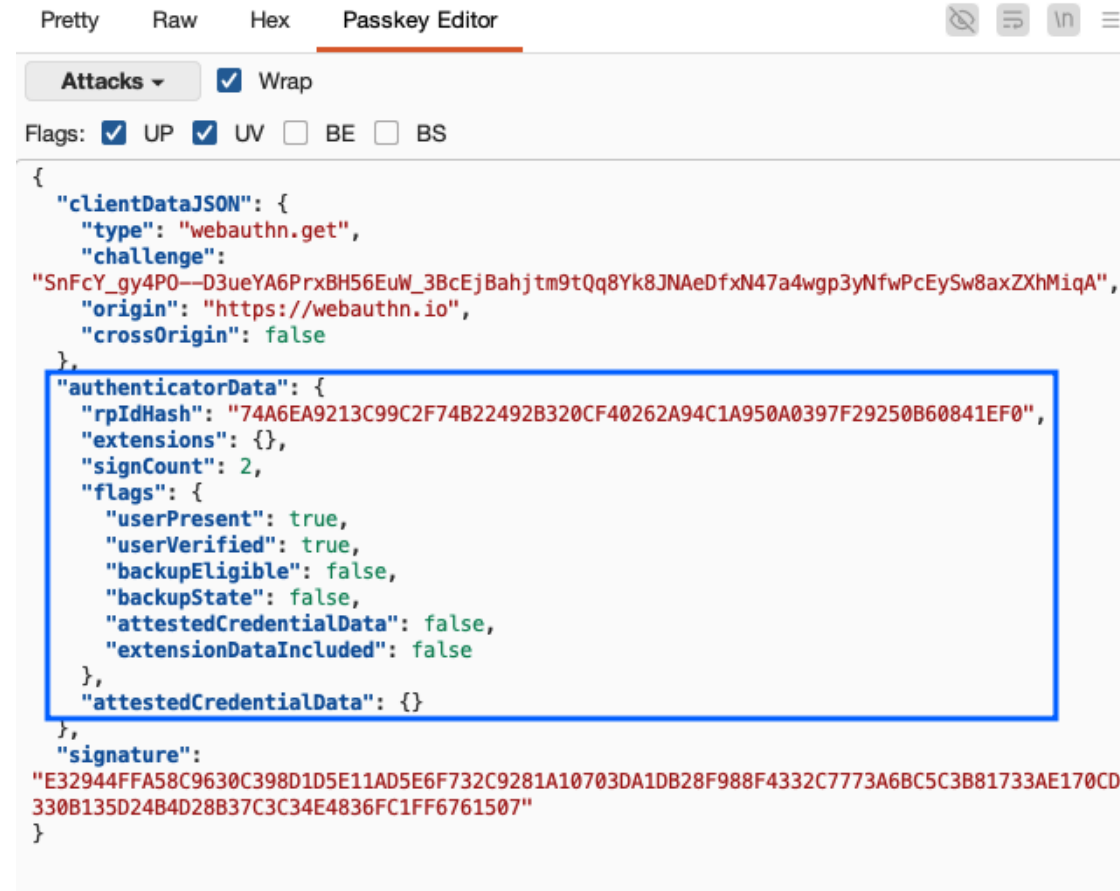
68

Dangling Allowlist Domain



69

attestationObject and authenticatorData

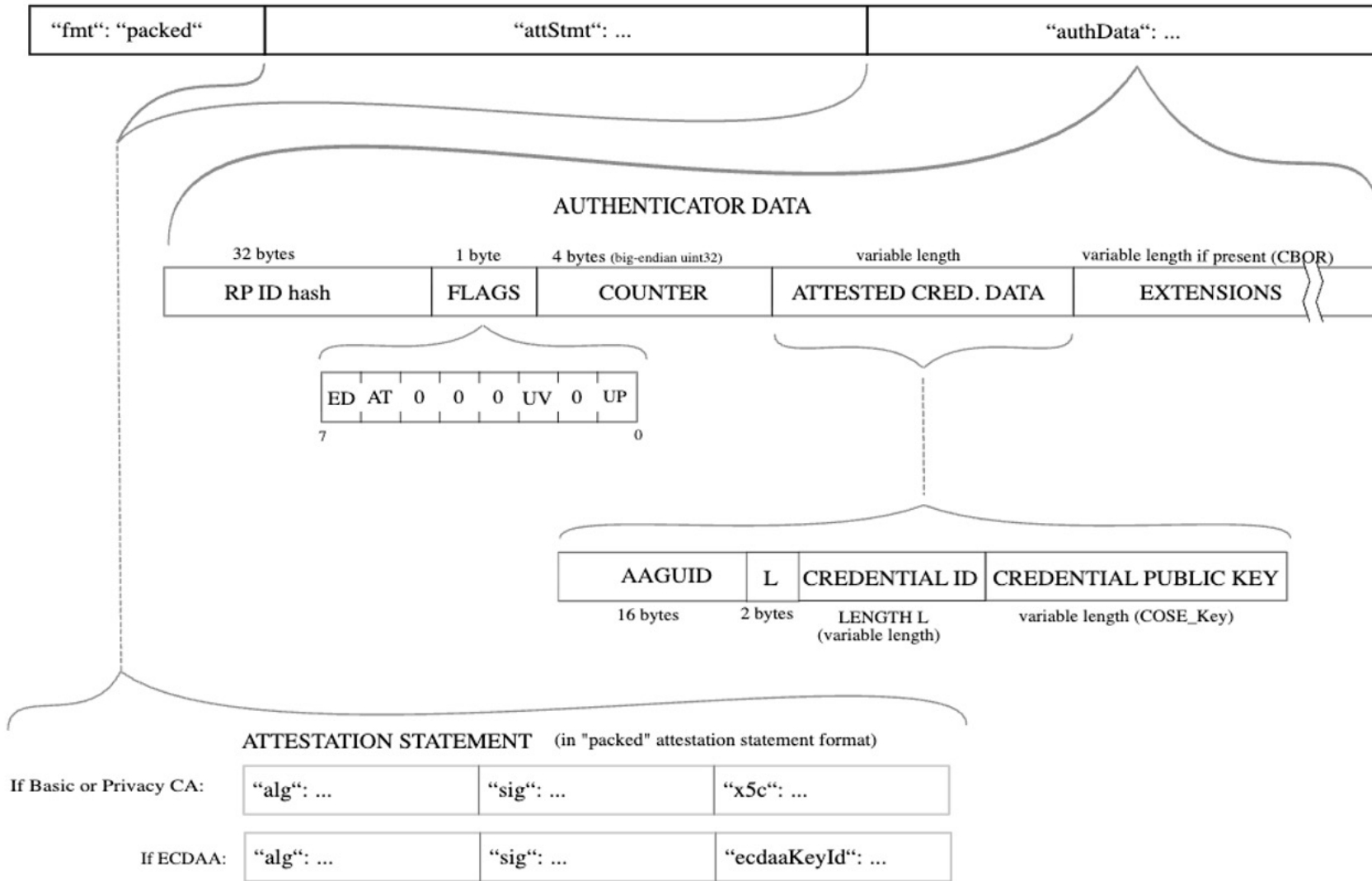


The screenshot shows a web application titled "Passkey Editor" with tabs for "Pretty", "Raw", "Hex", and "Passkey Editor". The "Passkey Editor" tab is active. Below the tabs, there is a section for "Attacks" with a dropdown menu and a checked "Wrap" checkbox. Below that, there are "Flags" with checkboxes for "UP", "UV", "BE", and "BS", where "UP" and "UV" are checked. The main area displays a JSON object representing an attestation object. The "authenticatorData" field is highlighted with a blue rectangular box. The JSON is as follows:

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge": "SnFcY_gy4P0—D3ueYA6PrxBH56EuW_3BcEjBahjtm9tQq8Yk8JNAeDfxN47a4wgp3yNfwPcEySw8axZXhMiqA",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature": "E32944FFA58C9630C398D1D5E11AD5E6F732C9281A10703DA1DB28F988F4332C7773A6BC5C3B81733AE170CD330B135D24B4D28B37C3C34E4836FC1FF6761507"
}
```

70

ATTESTATION OBJECT



ATTESTATION OBJECT

"fmt": "packed"	"attStmt": ...	"authData": ...
-----------------	----------------	-----------------

AUTHENTICATOR DATA

32 bytes	1 byte	4 bytes (big-endian uint32)	variable length	variable length if present (CBOR)
RP ID hash	FLAGS	COUNTER	ATTESTED CRED. DATA	EXTENSIONS

ED	AT	0	0	0	UV	0	UP
7							0

Authentication

AAGUID	L	CREDENTIAL ID	CREDENTIAL PUBLIC KEY
16 bytes	2 bytes	LENGTH L (variable length)	variable length (COSE_Key)

ATTESTATION STATEMENT (in "packed" attestation statement format)

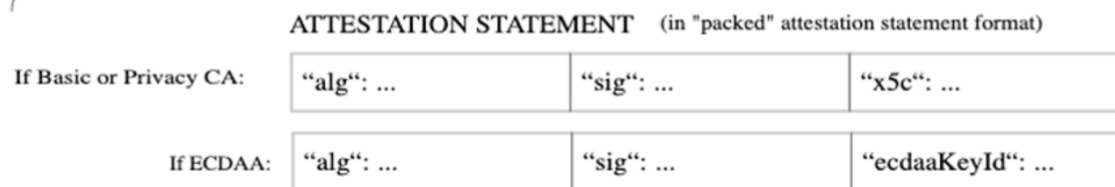
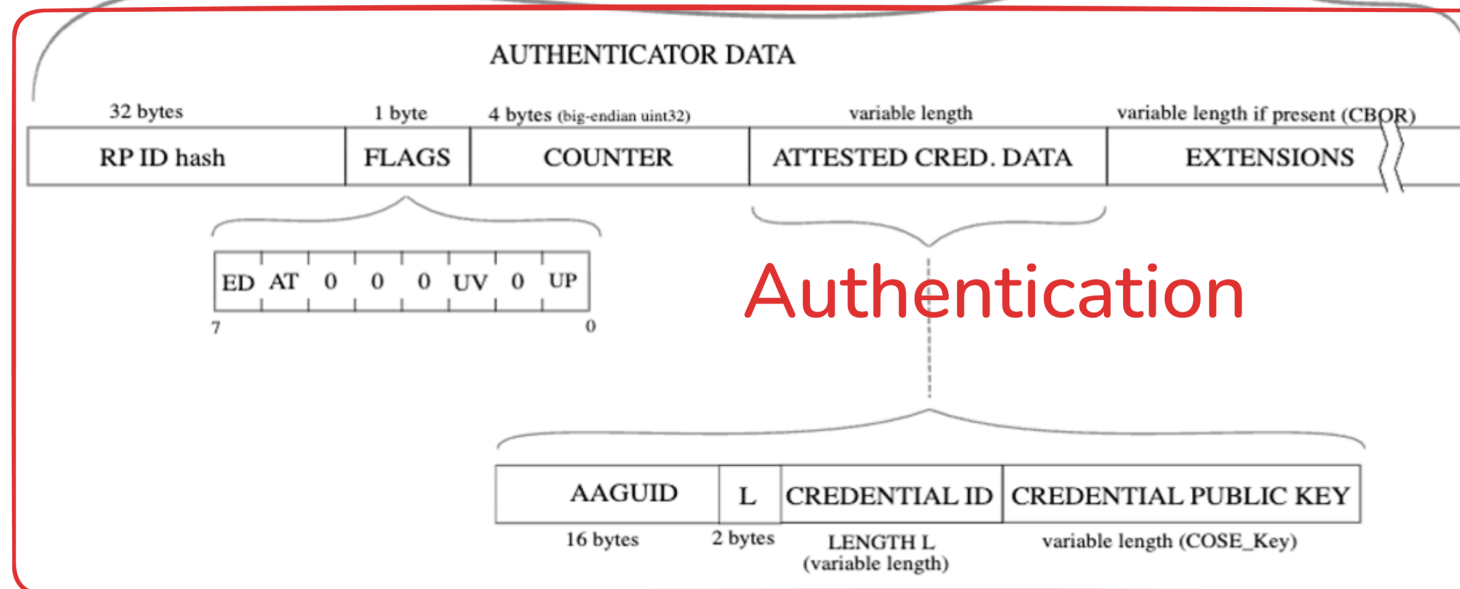
If Basic or Privacy CA:

"alg": ...	"sig": ...	"x5c": ...
------------	------------	------------

If ECDAAs:

"alg": ...	"sig": ...	"ecdaaKeyId": ...
------------	------------	-------------------

Registration



signature

```
Pretty Raw Hex Passkey Editor
Attacks ▾ ☒ Wrap
Flags: ☒ UP ☒ UV ☐ BE ☐ BS
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge":
    "SnFcY_gy4P0—D3ueYA6PrxBH56EuW_3BcEjBahjtm9tQq8Yk8JNAeDfxN47a4wgp3yNfwPcEySw8axZXhMiqA",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 2,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature":
  "585F37FDF8B02D37B69490DF81D684D8951BB65EA5F39E13F5843E9E67C37A5926F8A09902D9FE6E6831BFAD
  951C6145F5BA365188E64E3C348DA3285EE6EE04"
}
```

74

alg

Pretty Raw Hex **Passkey Editor**

Attacks ☒ Wrap

Signing algorithm: RS256 (-257) Attestation: None **RS256 key planted**

Flags: ☒ UP ☒ UV ☐ BE ☐ BS

```
{
  "clientDataJSON": {
    "type": "webauthn.create",
    "challenge": "jdrVAViggN5wTpX3MX6QtnmQJgNoCVxWWlCFIes-6F9ThLq_LFAKw4co1BQ3n_Xqu0mbcA9fsfU_7qGHQVga2w",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "attestationObject": {
    "attestationStatement": {
      "format": "none"
    },
    "authenticatorData": {
      "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
      "extensions": {},
      "signCount": 1,
      "flags": {
        "userPresent": true,
        "userVerified": true,
        "backupEligible": false,
        "backupState": false,
        "attestedCredentialData": true,
        "extensionDataIncluded": false
      },
      "attestedCredentialData": {
        "aaguid": "01020304-0506-0708-0102-030405060708",
        "coseKey": {
          "keyType": "RSA",
          "algorithm": "RS256",
          "raw":
            "A401030339010020590100B104054256CE44CECC357B8A1444DB2A1698C8D37AC992B52ED79B331AF43BED77844651D3959EC15CFE38F6552F
            234AFEDFC5D7AF5F79E067136BFC335C6272FC2FF8A8EA58849E9DF31EA2D27C9F0908D47DDD7FDD781FD2F8CB51ED0BC8058994522A4C13A41
            331B85C037DDBAAED47775A401488E6D5E3823466724D22C6F6BAF274F7B6A8C8EB9660ECCD0BB14530818CE850A553A87A877A2B1C3A88695D7
            A0A319FD6AB26113D7F59CFEEFD2D6725FA9BD3BD2D5CFB0568E70F31FCDFEBCA7FCDD1C2EAF4B46C44FD1D7A8269080D9A4A1DB3FAE649E1073
            4D4729386160C39FC12B660B611765B9C45BC9E9ED548110766221D9C42906A2273CAEF2143010001"
        },
        "credentialId": "755A521FCEF1ABC93EA6C0DDC6B54FAB4CDED7E102DCB4F09B416C29180AA3CD"
      }
    },
    "fmt": "none"
  }
}
```

75

credentialId

Pretty Raw Hex Passkey Editor

Attacks ☒ Wrap

Signing algorithm: RS256 (-257) Attestation: None RS256 key planted

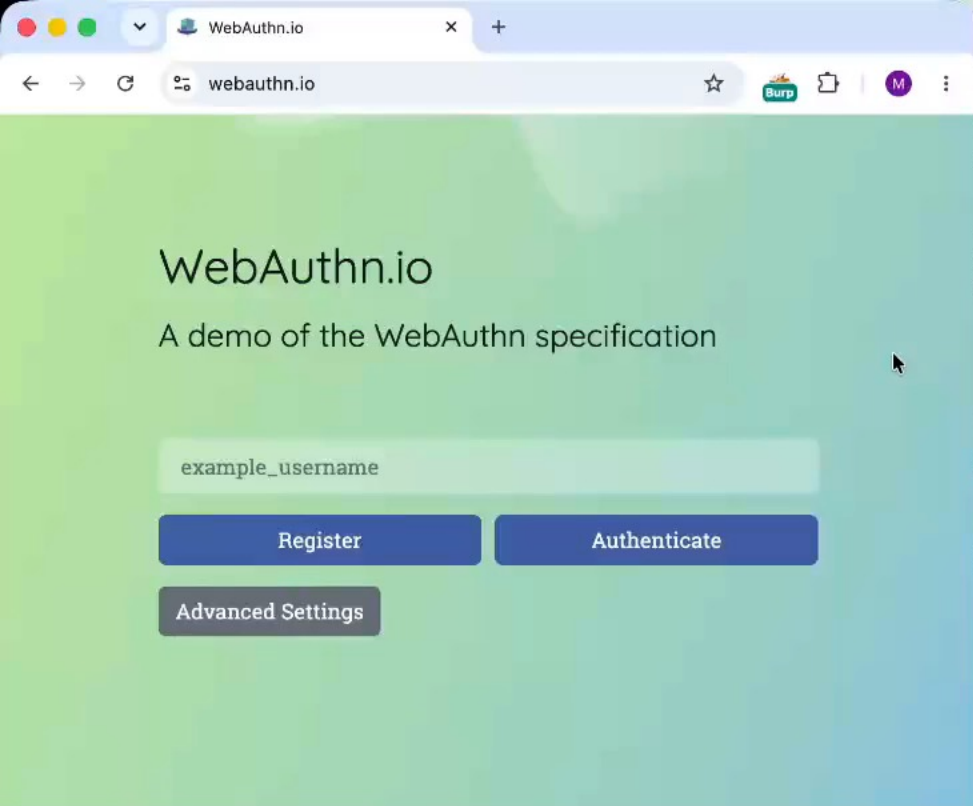
Flags: ☒ UP ☒ UV ☐ BE ☐ BS

```
{
  "clientDataJSON": {
    "type": "webauthn.create",
    "challenge": "jdrVAViggN5wTpX3MX6QtnmQJgNoCVxwWlCFIes-6F9ThLq_LFAKw4co1BQ3n_Xqu0mbcA9fsfU_7qGHQVga2w",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "attestationObject": {
    "attestationStatement": {
      "format": "none"
    },
    "authenticatorData": {
      "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
      "extensions": {},
      "signCount": 1,
      "flags": {
        "userPresent": true,
        "userVerified": true,
        "backupEligible": false,
        "backupState": false,
        "attestedCredentialData": true,
        "extensionDataIncluded": false
      }
    },
    "attestedCredentialData": {
      "aaguid": "01020304-0506-0708-0102-030405060708",
      "coseKey": {
        "keyType": "RSA",
        "algorithm": "RS256",
        "raw":
        "A4010303390100205901009EEAC616434B9A4603B286B10089427B1CA3A4C7C8F599A1909A07D4C08E7FB038E0244894450F5129EFE6E00BA5470862E87DA611255CA0064CB6BAAC891E336FEC6C9D4A04A04F770DBBF7E5831C078B8710B3434F4C5DEA9ACF42F853C81BE1E215FF9ECA1FBA82F730163778E2F6AF5F65D66A20C2CEC58EF6B4E399B2A7CA7856C8F6E6E838C06F8884C5E66966D75CBE6DA49035AE2BA3FC7A88D5F6A93557E1E23E0B13B3C8C0E47CE9A4D9791E0181456177ED790100A7D77723A00A56ECE30FB58A8483B20B1F2230A0A4577DEE35797844C6E0FF3BBDE61C0E27C0F5373EEA0278F2D72147B3D510C22DF8369E0EBFF9E617511934D1DC76BBB3F32143010001"
      }
    },
    "credentialId": "7AF8A5FDCADE7627"
  },
  "fmt": "none"
}
```

signCount

RP	Evil	Auth
12		12
13		13
14		14
15		15
18	18	
		16
		17
		18
19		19
20		
21		

```
Pretty  Raw  Hex  Passkey Editor
Attacks  Wrap
Flags:  UP  UV  BE  BS
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge":
    "SnFcY_gy4P0--D3ueYA6PrxBH56EuW_3BcEjBahjtm9tQq8Yk8JNAeDfxN47a4wgp3yNfwPcEySw8axZXhMiqA",
    "origin": "https://webauthn.io",
    "crossOrigin": false
  },
  "authenticatorData": {
    "rpIdHash": "74A6EA9213C99C2F74B22492B320CF40262A94C1A950A0397F29250B60841EF0",
    "extensions": {},
    "signCount": 13333337,
    "flags": {
      "userPresent": true,
      "userVerified": true,
      "backupEligible": false,
      "backupState": false,
      "attestedCredentialData": false,
      "extensionDataIncluded": false
    },
    "attestedCredentialData": {}
  },
  "signature":
  "D3EB54B0EF8A3A07F8B20A00197DABC9C75F43C02857503AA4E5CE18DF883E96378CD0E6D6FD5FEB5522135E77FDA173BDE16AE256986197BDEA26CE22B6AB0F"
}
```



WebAuthn x

Enable virtual authenticator environment

Authenticator 90e32

Active

Remove

UUID

ec3e8453-ec81-4fee-820c-ff2fe9190e32

Protocol

ctap2

Transport

internal

Supports resident keys

Yes

Supports large blob

No

Supports user verification

Yes

Supports hmac-secret

No

Supports hmac-secret-mc

No

Credentials

ID	Is Resi...	RP ID	User Handle	Signature Cou...	Actions
No credentials. Try calling navigator.credentials.create() from your website.					

New authenticator

ctap2

Match and replace

Proxy settings

Specific extensions

Search

Filter on

	Edited	Status code	Notes	Length	MIME type	Extension	Title	TLS	IP	Cookies
Authentication/verification		200	Authentication - webauthn.io	418	JSON			✓	54.184.242.104	sessionId=1fr4kc...
Authentication/options		200	Options - webauthn.io	611	JSON			✓	54.184.242.104	
Registration/verification		200	Registration - webauthn.io	316	JSON			✓	54.184.242.104	

Response

PrettyRawHexRender

2zvceuo1w

Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, 5;v="150", "Google Chrome";v="150"

1 HTTP/2 200 OK

2 Alt-Svc: h3=":443"; ma=2592000

3 Content-Type: application/json

4 Cross-Origin-Opener-Policy: same-origin

5 Date: Fri, 17 Jul 2026 18:25:11 GMT

6 Referrer-Policy: same-origin

7 Server: Caddy

8 Server: gunicorn

9 X-Content-Type-Options: nosniff

10 X-Frame-Options: DENY

11 Content-Length: 18

12

13 {

14 "verified":true

15 }

Inspector

Notes

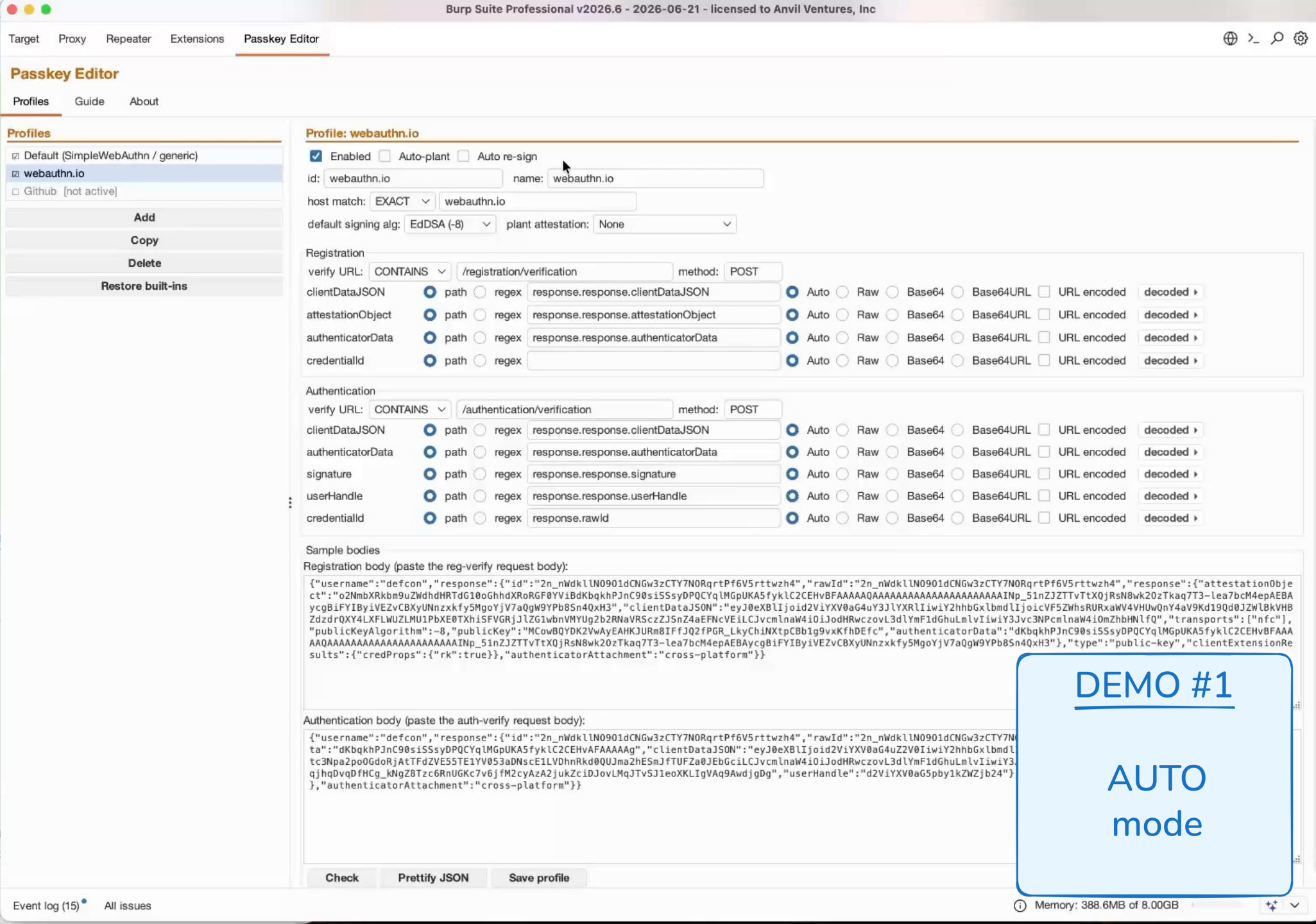
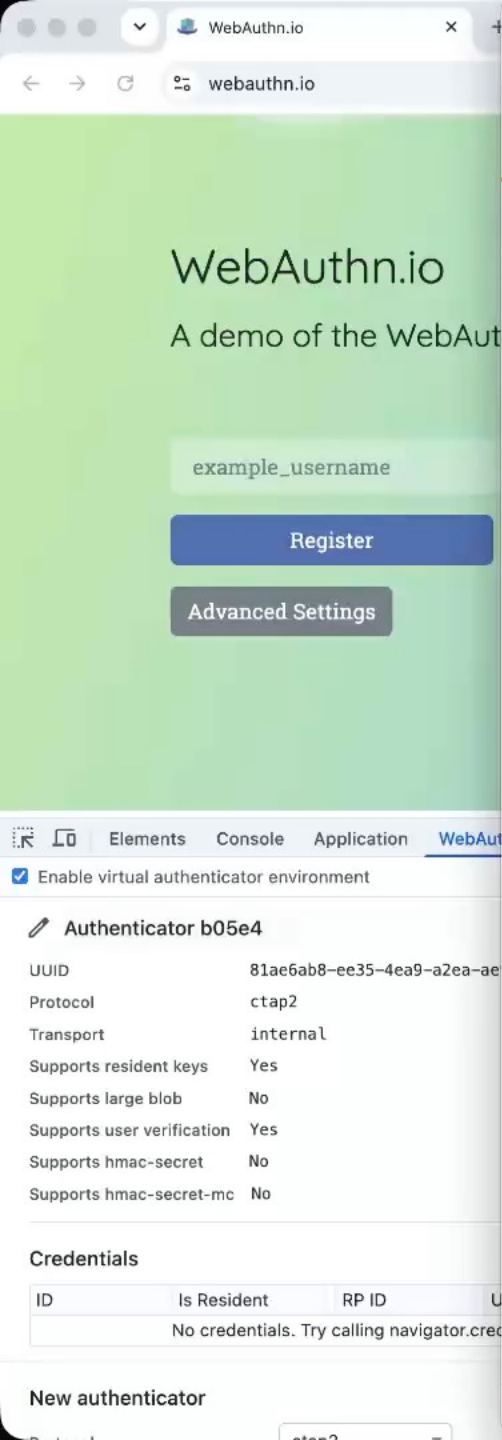
0 highlights

Search

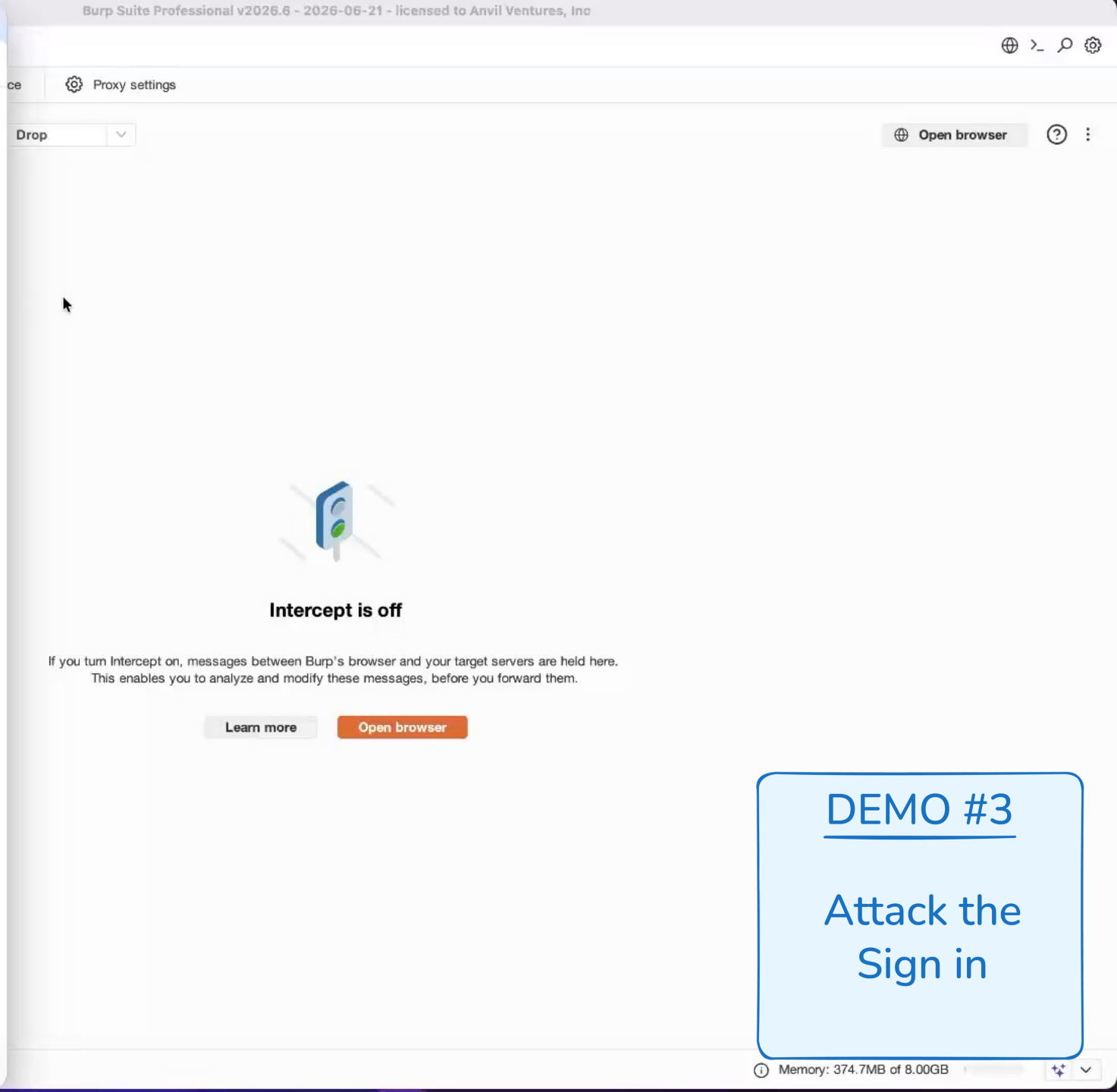
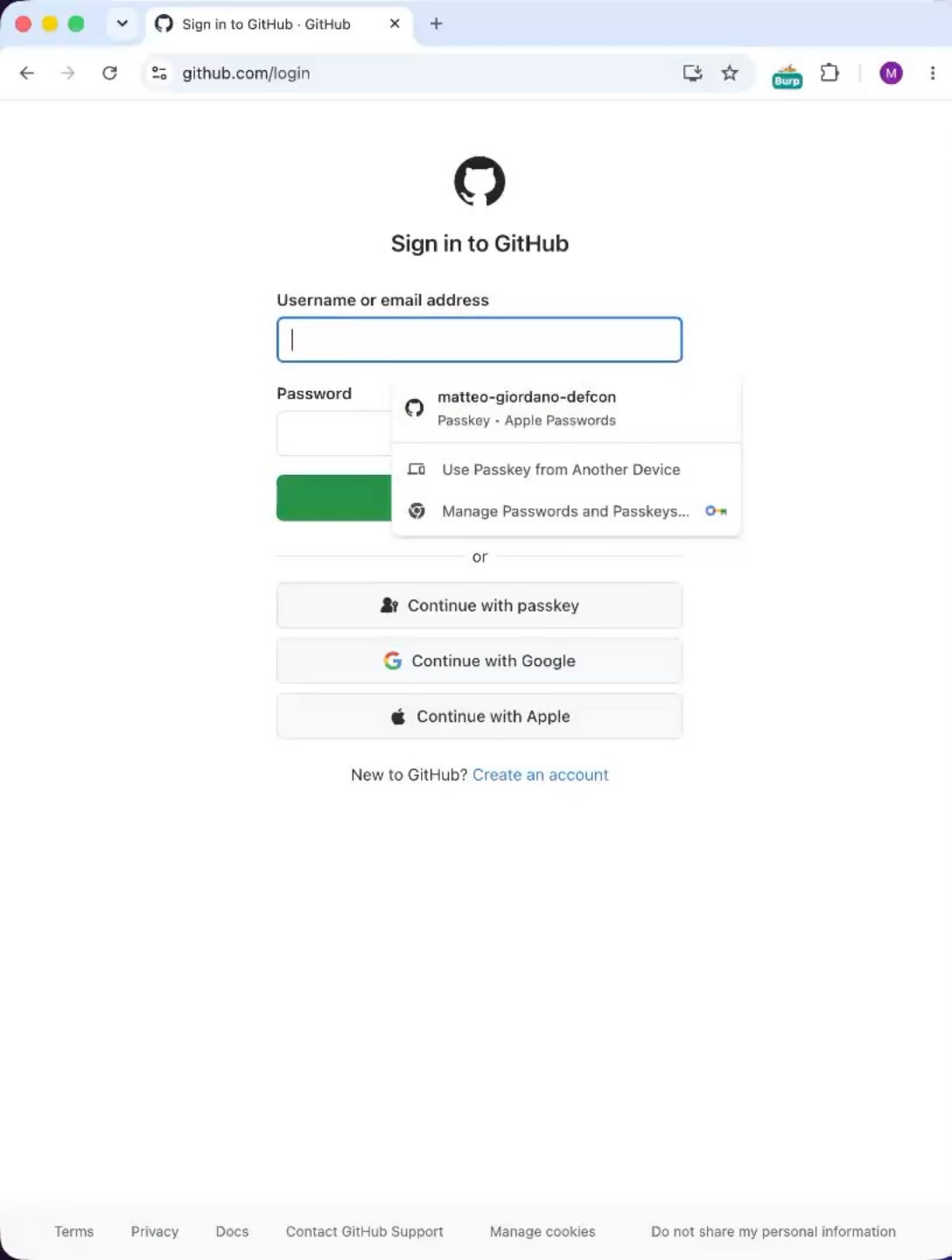
Memory: 194.7MB of 8.00GB

DEMO #1

Decode
Plant
Re-sign



Attack the Registration



DEMO #3

Attack the
Sign in

Even the big players

- ◆ [CVE-2026-46419](#) (Yubico java-webauthn-server, the reference RP library): returns success for a credential owned by a different user in 2FA / non-discoverable flows.
- ◆ [CVE-2025-26788](#) (StrongKey FIDO Server): treats non-discoverable as discoverable and doesn't bind the assertion to the initiating username, so substitute your own credId and sign in as the victim.
- ◆ [CVE-2024-12225](#) (Quarkus, CVSS 9.1): leftover default register/login endpoints stay reachable, yielding a login cookie for any username.
- ◆ [CVE-2025-12150](#) and [CVE-2026-6856](#) (Keycloak): attestation-policy bypass via fmt:none, and an AAGUID-allowlist bypass via packed self-attestation.

Checklist + Creativity

Checklist

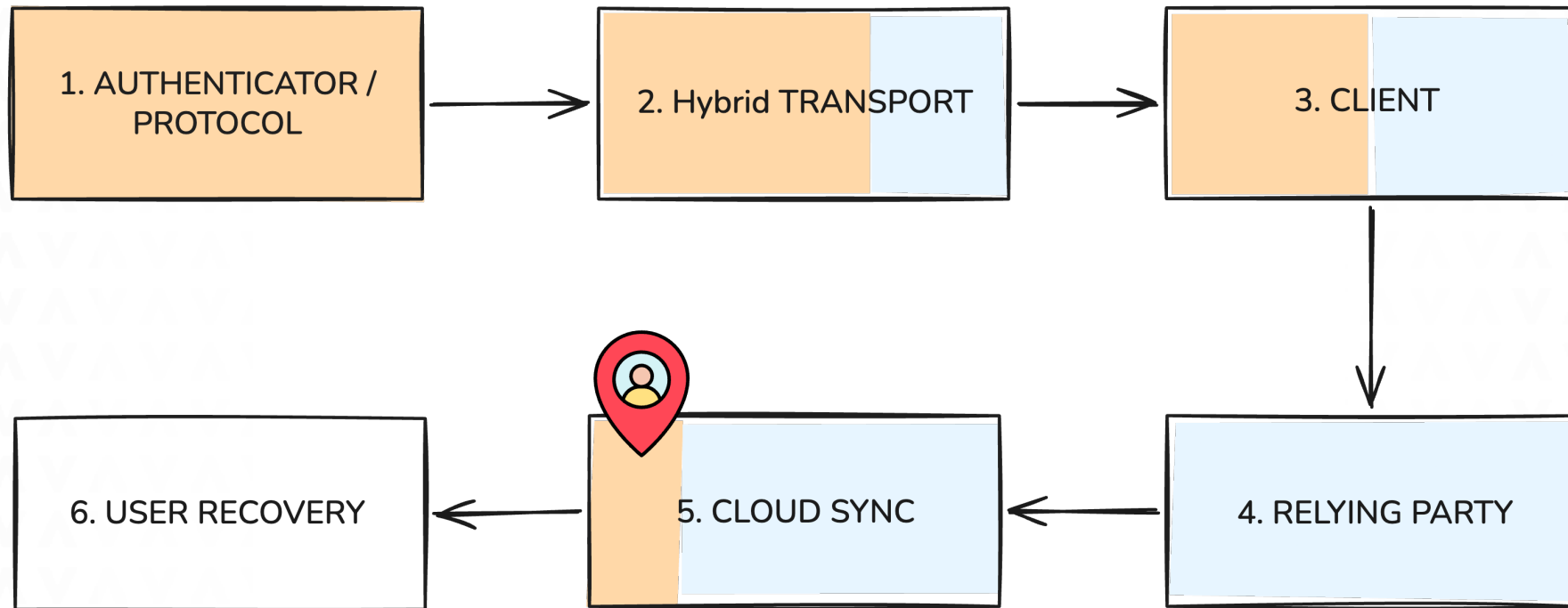
- ♦ signature verified
- ♦ credId bound to one user
- ♦ UV honored
- ♦ UP honored
- ♦ origin allowlist (exact)
- ♦ crossOrigin rejected
- ♦ rpIdHash = sha256(rpId)
- ♦ challenge fresh + session-bound
- ♦ type create vs get
- ♦ COSE alg allowlist
- ♦ userHandle validated
- ♦ credId length 16-1023
- ♦ duplicate credId rejected
- ♦ signCount checked
- ♦ fmt:none empty
- ♦ AAGUID allowlist
- ♦ BE/BS coherent
- ♦ Token Binding rejected

Creativity

- ♦ rpId over-scoping
- ♦ dangling Related-Origin (/well-known/webauthn)
- ♦ clickjacking / framing (register)
- ♦ enrollment step-up
- ♦ CSRF on register
- ♦ admin-API / Entra enrollment
- ♦ post-compromise persistence
- ♦ recovery downgrade
- ♦ mixed-mode fallback (password / TOTP)
- ♦ post-ceremony session binding
- ♦ credential-management authz
- ♦ leftover default endpoints

5. Cloud sync

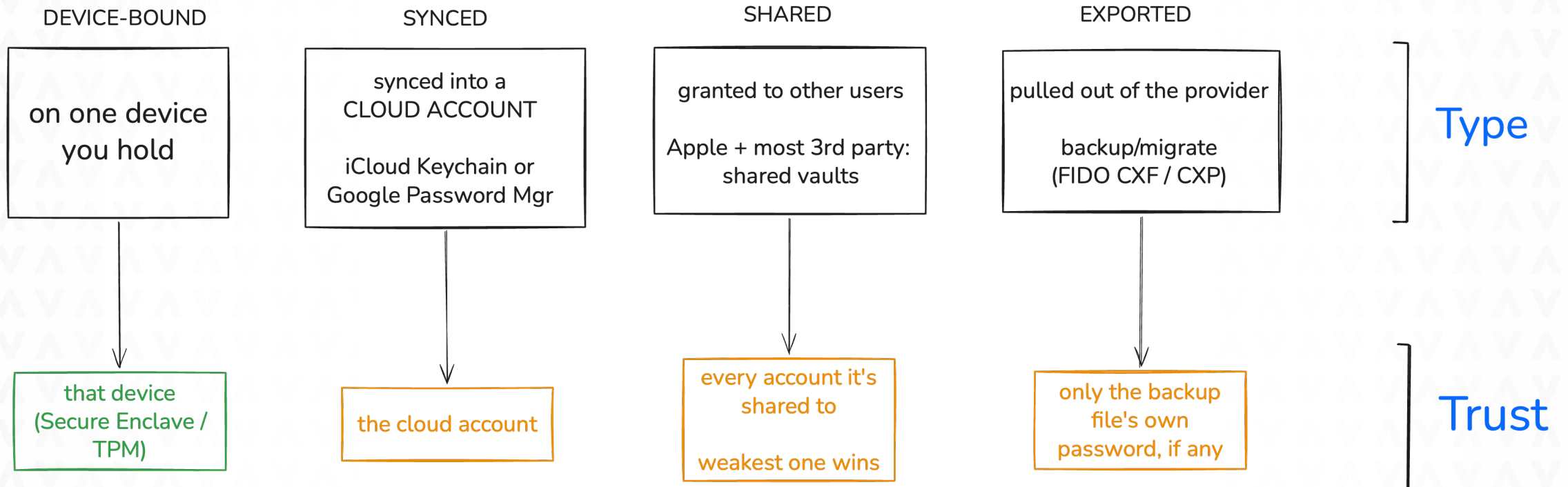
You are here



Research

Engagement

Types of Passkeys



The Cloud options

- ◆ **First-party** (iCloud Keychain, Google)
 - ◆ The vendor owns the hardware, OS, and sync servers
 - ◆ keys stay inside vendor HSMs and never leave hardware.
- ◆ **Third-party** (Bitwarden, LastPass, Dashlane, 1Password)
 - ◆ to stay portable, the key decrypts into ordinary app memory

The account still falls the old ways



Spensky, DEF CON 33 (sync-fabric phishing PoC)

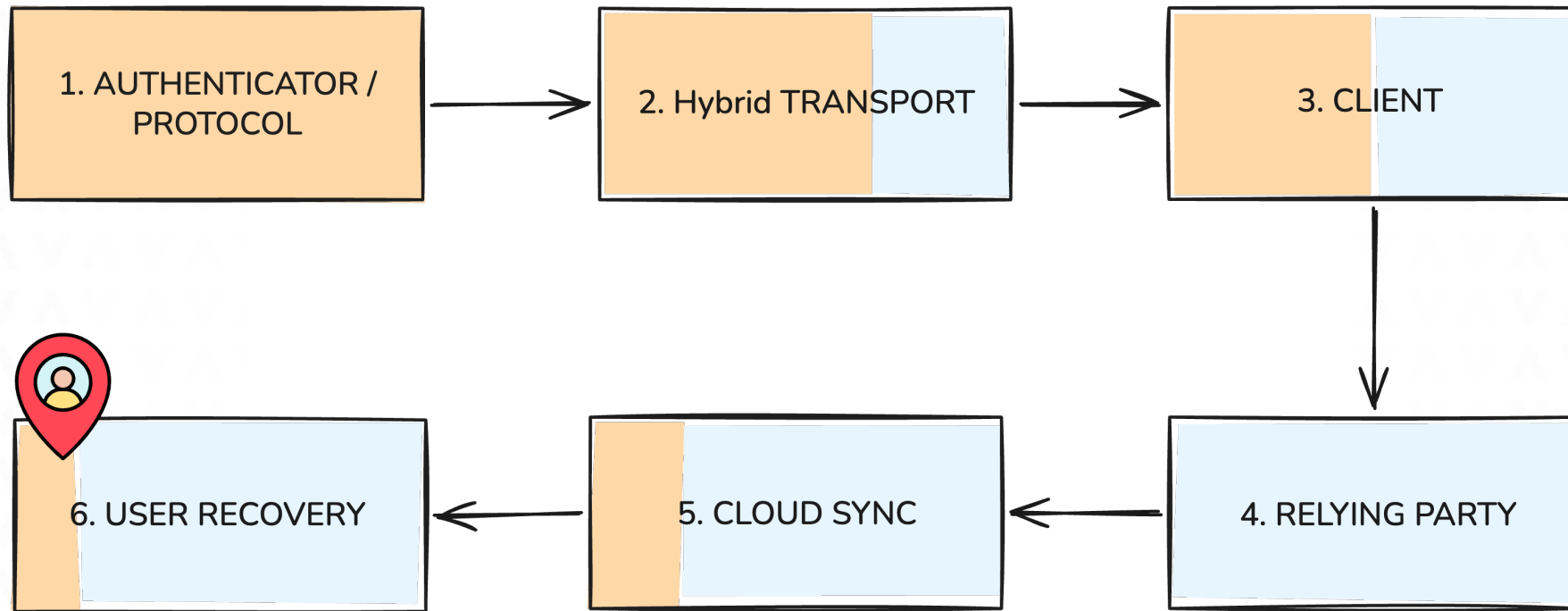
- ◆ SIM swap, ~19% of passkey account-takeover correlated ([Prove](#))
- ◆ Phish the login, drive a real browser as them, walk out with the passkeys ([Spensky, DEF CON 33](#))
- ◆ Or go deeper: VaultJacking phishes the vault PIN for the master key, decrypting every synced passkey at once ([Brazzell, 2026](#))

Decoy passkeys

- ◆ [↗ CASPER \(Islam et al., USENIX Security 2025\)](#)
 - ◆ Hides the real passkey among indistinguishable decoys
 - ◆ a key stolen from a cloud breach trips a decoy at login
 - ◆ so the RP detects the theft.
 - ◆ Detection, not prevention.
- ◆ [↗ Bicakci et al. \(2026\)](#)
 - ◆ Syncs only REAL ciphertext
 - ◆ the decryption key stays in the user's hardware token
 - ◆ the cloud is never trusted
 - ◆ Prevention, not detection

6. User and recovery

You are here



Research

Engagement

Weaker ways in

- ◆ SMS
- ◆ email links
- ◆ security questions
- ◆ lost device flows
- ◆ helpdesk
- ◆ OAuth device-code



Government says it plainly

Attacks

Attackers are seen to **target account recovery** for both tMFA and FIDO2 protected accounts, as the recovery process can often be weaker than the authentication to the account. This is particularly common in two cases:

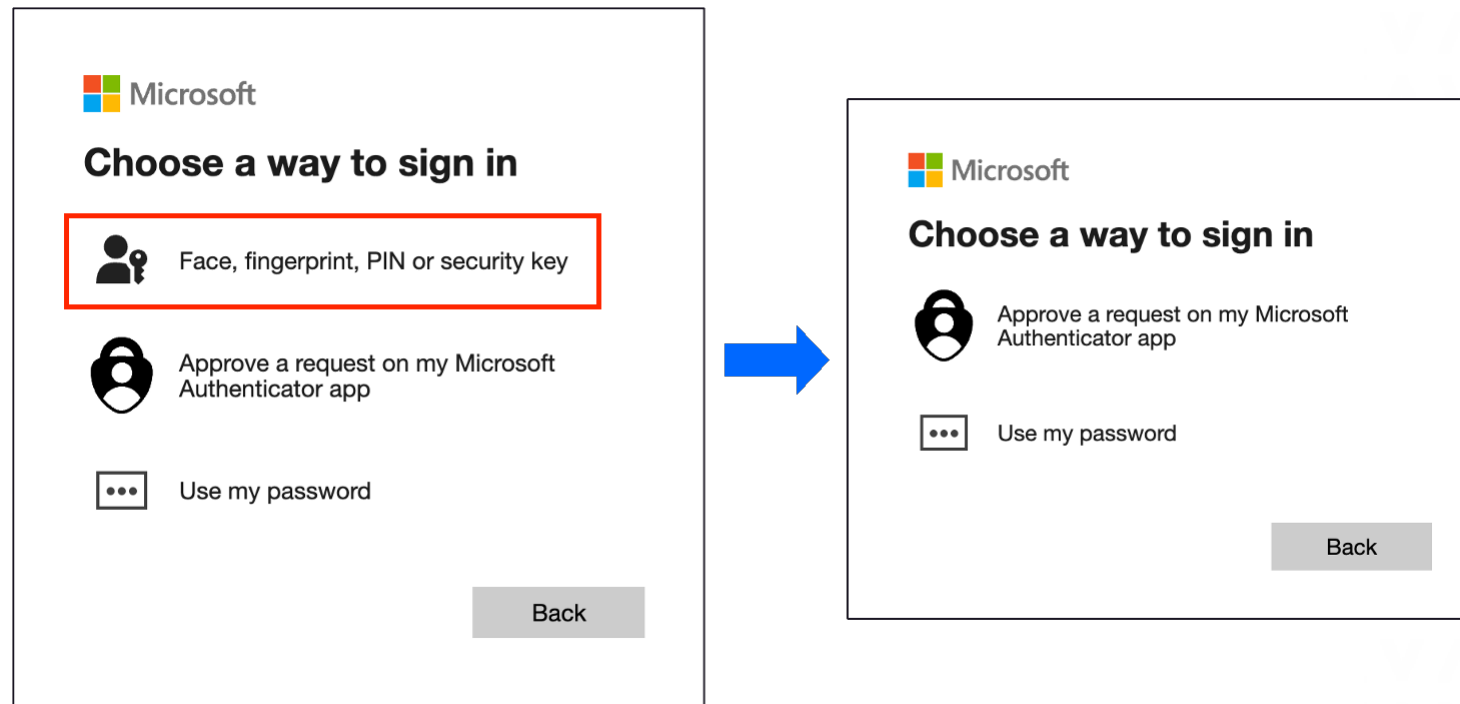
The vast majority of websites and apps will offer a 'forgot password' functionality. This typically leverages a code or link sent to the email address registered with the user's account, thereby **effectively reducing the security of the service to that of the email account**.

credentials, revoking those of the legitimate user. These attacks should be prevented **if only passkeys are used, with no weaker password or tMFA options supported as a 'fallback'**.

Source: UK NCSC, *Traditional and FIDO2 credentials for personal use* (2026). [ncsc.gov.uk](https://www.ncsc.gov.uk)

AiTM still works

- ◆ [Push Security \(2025\)](#) named the class: an AiTM kit **rewrites the method-selection page**, so "passkey OR backup code" becomes just "backup code."
- ◆ [IOActive \(2026\)](#) weaponized it on Cloudflare Workers: flip the FIDO2 isDefault, or **CSS-hide the passkey**.



When the attacker is already inside

- ◆ ↗ [Daffalla et al. \(USENIX Security 2025\)](#)
 - ◆ assume the attacker is **someone in your life**
 - ◆ An intimate partner, a family member
- ◆ **User** verification, not **owner** verification
 - ◆ FIDO2 never binds a passkey to the biometric that enrolled it

Abuse scenario

1. Alex **shares** their phone **PIN** with Billy.
2. One bathroom break later:
 - a. Billy **AirDrops** Alex's TikTok passkey to his own iPhone
 - b. **reads** every message.
3. Alex gets suspicious, **resets the password**, **enrolls** a new passkey.
4. It changes nothing.

Close

**Passkeys didn't remove the attack surface.
They moved it.**

Questions?

- ◆ Mail: matteo.giordano@anvilsecure.com
- ◆ LinkedIn: linkedin.com/in/giordanomatteo/
- ◆ Website: matteogiordano.im
- ◆ GitHub: github.com/anvilsecure/paskey-editor
- ◆ Blogposts:
 - ◆ anvilsecure.com/blog/demystifying-paskeys-under-the-hood-the-protocol.html
 - ◆ anvilsecure.com/blog/demystifying-paskeys-under-the-hood-the-architecture.html
 - ◆ anvilsecure.com/blog/demystifying-paskeys-under-attack.html



Beyond the Ceremony.