authenticate 2024
THE FIDO CONFERENCE

“How do I...?” - Answering common questions from RP devs

Matthew Miller
Technical Lead, Cisco Duo

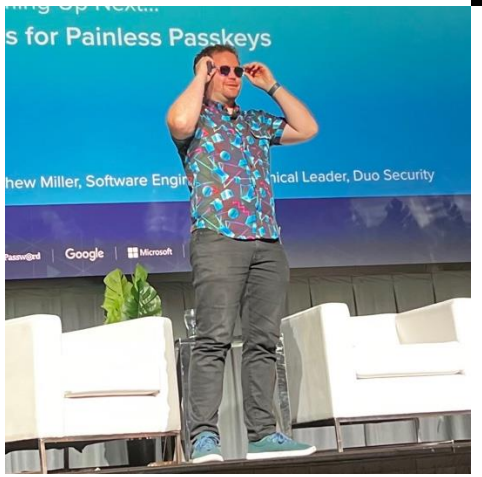
Agenda

- Who am I?
- How do I...
  - ...use passkeys on a site accessed by IP address?
  - ...require only biometrics for UV?
  - ... tell an authenticator when a credential is deleted?
  - ...use passkeys from a cross-domain iframe?
  - ...skip registration when an authenticator is already registered?
  - ...use a passkey on different domains?
- Q & A
- Wrap Up

Who am I?

- **WebAuthn SME** with an eye on the Relying Party developer experience
- Author of **SimpleWebAuthn** and **py_webauthn** libraries
- Current maintainer of **webauthn.io**
- Help drive FIDO2 adoption within the FIDO Alliance, and W3C's WAWG and WACG

Authenticate 2022



Authenticate 2023

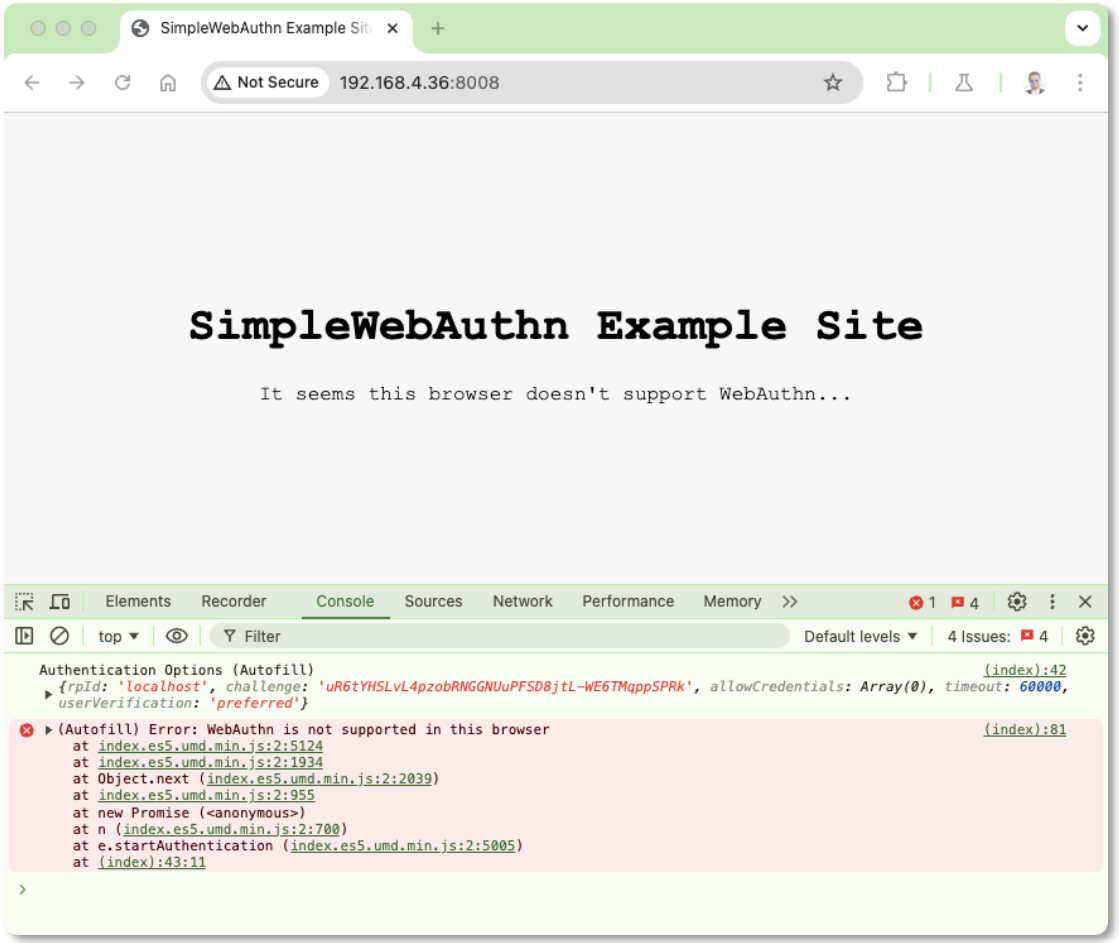
How do I...

...use passkeys on a site accessed by IP address?

WebAuthn requires API calls be made from a “Secure Context”:

- Site must be served over **valid TLS connection**
- **http://*.localhost**, **http://127.0.0.1**, and **file://** are also considered secure origins
- Sites embedded via **<iframe>** must be within a site served over TLS, all the way up the hierarchy

Binding credentials to domains reduces attack vectors by leaning on more stable identifiers than IP addresses that can change on a whim.

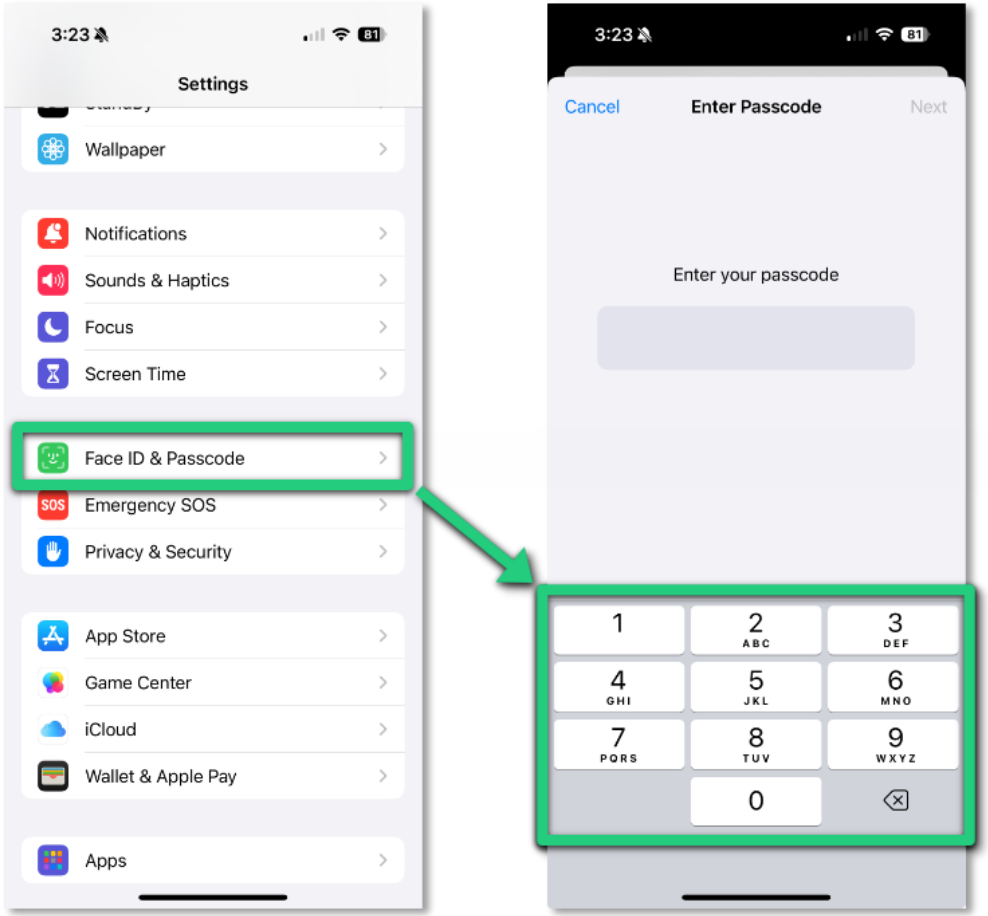


...require only biometrics for UV?

Biometric is a **convenience** over PIN entry!

Attackers that glean a victim's device unlock PIN can easily enroll their own fingerprint / face / etc...

WebAuthn would not become meaningfully more secure if RPs could mandate biometric-only UV.

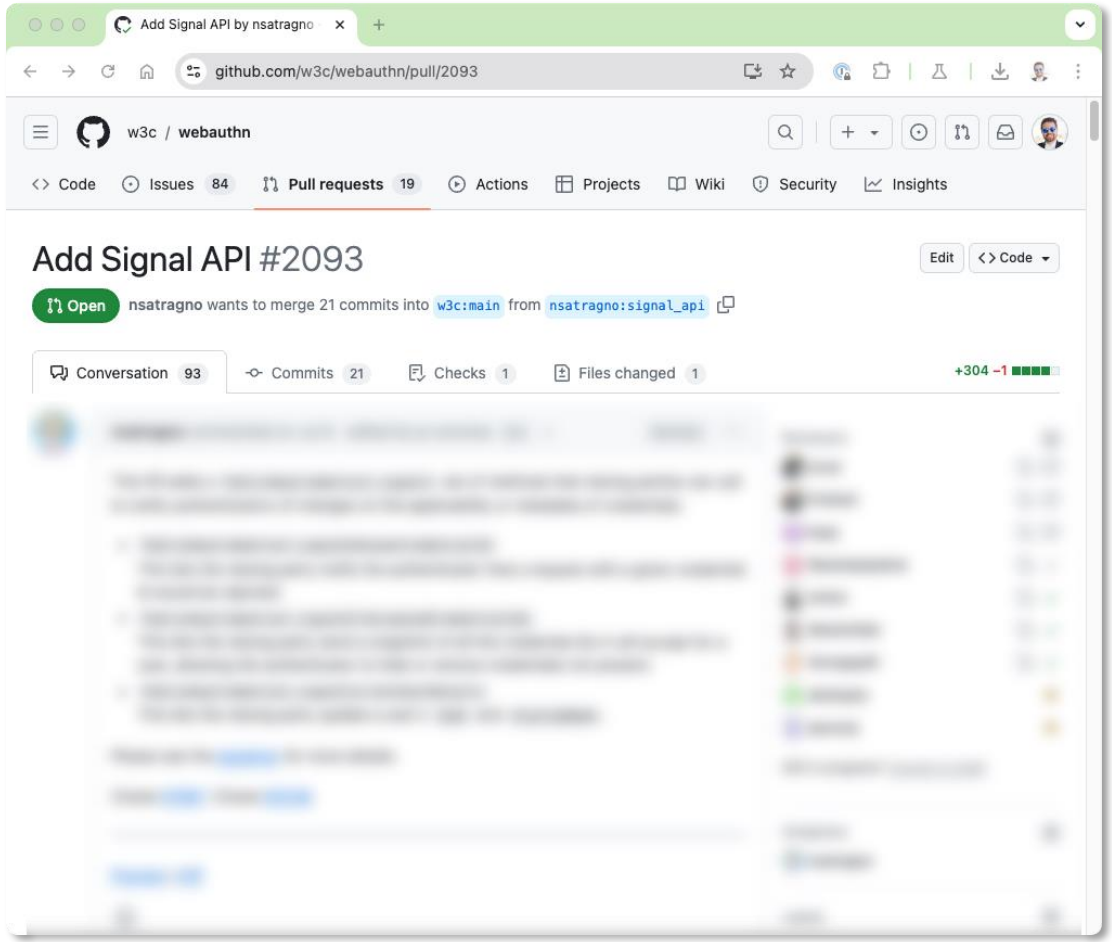


...tell an authenticator when a credential is deleted?

Rejecting or deleting a new passkey public key **leaves the private key on the authenticator.**

Relying Parties cannot meaningfully reach out to the authenticator to say, “we won’t recognize this credential ID for use anymore.”

...but there’s hope! A **new “signal” API** is coming soon to address this! w3c/webauthn#2093



PublicKeyCredential.signalUnknownCredential()

Relying Parties can send this signal after **passkey registration fails** validation for some reason (no attestation statement, unexpected AAGUID, some other policy went unfulfilled...)

Browser will do its best to **eventually** get this information to the corresponding authenticator.

The authenticator is free to hide or delete the corresponding passkey private key.

```
PublicKeyCredential.signalUnknownCredential({
  rpId: 'example.com',
  credentialId: 'dMJnLxztliroTBpko98T4PwV',
});
```

(API subject to change)

Coming soon!

PublicKeyCredential.signalAllAcceptedCredentials()

Relying Parties can send this signal after a user completes some kind of credential management, or even after every successful authentication.

Browser will do its best to **eventually** get this information to the corresponding authenticator.

The authenticator is free to hide or delete the corresponding passkey private key.

```
PublicKeyCredential.signalAllAcceptedCredentials({
  rpId: 'example.com',
  userId: 'fqZpCcoRyUMYzW6D',
  allAcceptedCredentialIds: [
    '42bQFFTGsVB8pEcbiiApotn6',
    '94dIbDRqNFIa9lXdCjpPmU9N',
  ],
});
```

(API subject to change)

Coming soon!

PublicKeyCredential.signalCurrentUserDetails()

Relying Parties can send this signal after a user changes their username/email address/etc... to update credential metadata.

Browser will do its best to **eventually** get this information to the corresponding authenticator

The authenticator can then update credential metadata with up-to-date identifiers.

```
PublicKeyCredential.signalCurrentUserDetails({
  rpId: 'example.com',
  userId: 'fqZpCcoRyUMYzW6D',
  name: 'Matthew Miller',
  displayName: 'Production',
});
```

(API subject to change)

Coming soon!

...use passkeys from a cross-domain iframe?

A coordinated effort between the Relying Party, and the site embedding the Relying Party:

The **Relying Party** should **omit** the X-Frame-Options HTTP header to allow itself to be embedded.

The **site embedding the Relying Party** should embed the site in an <iframe> with the “publickey-credentials-get” value in the allow attribute.

```
<iframe
  src="https://webauthn.io"
  frameborder="1"
  style="width: 1000px; height: 600px;"
  allow="publickey-credentials-get"
></iframe>
```

Showing iframe auth in action

...skip registration when an authenticator is already registered?

Relying Parties cannot **preemptively** fail a WebAuthn call without the user interacting with the registration ceremony.

Prevent authenticator re-registration by including the user's existing credential IDs in **excludeCredentials** when calling **.create()**.

A thrown **InvalidStateError** will signal when the user tried to re-register an authenticator.

```
let credential;
try {
  credential = await navigator.credentials.create({
    publicKey: {
      // ...
      excludeCredentials: [
        { id: new Uint8Array([...]), 'type': 'public-key' },
      ],
    },
  });
} catch (err) {
  if (err.name === 'InvalidStateError') {
    // User tried to re-register an authenticator
  }
}
```

...use a passkey on different domains?

What **rp.id** is set to during **registration** can make or break **authentication**.

"...a valid domain string identifying the WebAuthn Relying Party on whose behalf a given registration or authentication ceremony is being performed."

Most Relying Parties can **keep things simple**:

```
{ 'rp': { 'id': 'example.com' } }
```

Subdomains can be used too:

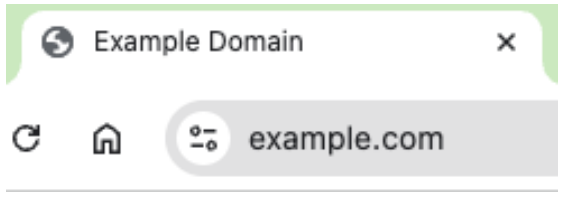
```
{ 'rp': { 'id': 'login.example.com' } }
```

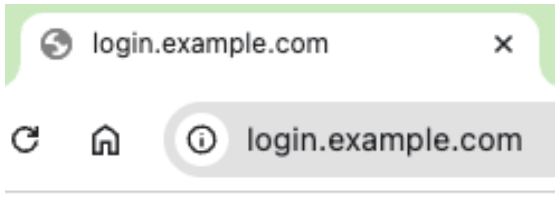
Be mindful of RP ID scope!

Can a passkey registered with...

```
{ 'rp': { 'id': 'example.com' } }
```

...be used to authenticate here?

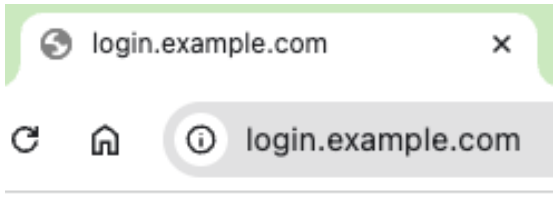




Can a passkey registered with...

```
{ 'rp': { 'id': 'login.example.com' } }
```

...be used to authenticate here?



...use a passkey on related domains?

Organizations that exist across multiple domains (e.g. **locale-specific TLDs**) need new tools to support account sign-in with a single passkey.

Related Origins let RPs specify multiple origins on which a single **rp.id** can be specified during auth.

See https://passkeys.dev/docs/advanced/related-origins for more info.

https://shopping.com/.well-known/webauthn

```
{
  "origins": [
    "https://shopping.com",
    "https://myshoppingrewards.com",
    "https://myshoppingcreditcard.com",
    "https://myshoppingtravel.com",
    "https://shopping.co.uk",
    "https://shopping.co.jp",
    "https://shopping.ie",
    "https://shopping.ca"
  ]
}
```

Coming soon!

Q & A

Where to find me

- Mastodon: @iamkale@infosec.exchange
- W3C WebAuthn Adoption Community Group (WACG): https://www.w3.org/community/webauthn-adoption/
- passkeys.dev: https://passkeys.dev
- Libraries (GitHub):
  - MasterKale/SimpleWebAuthn
  - Duo-Labs/py_webauthn

authenticate 2024

THE FIDO CONFERENCE

Thank you

Signature Sponsors:

cisco | Microsoft | yubico | Google

authenticatecon.com