SPC: From browser cache to FIDO/WebAuthn integration

Authors: smcgruer@google.com
Last Modified: 2022/05/19
Status: In-Review
This document: SPC: From browser cache to FIDO/WebAuthn integration

The goal of this document is to lay out the steps needed to go from a browser-cache version of SPC that is not integrated with WebAuthn at the spec level, to proper integration with both FIDO and WebAuthn. Such a shift will enable both 1p SPC scenarios for existing WebAuthn credentials, and also 3p-enabled credentials to be used across different browsers.

This document largely focuses on the specification level and the ideal long-term end state. There will likely be some complications in the implementation level along the way as different platforms move at different speeds.

This is not an official document from any organization, and is simply the author's (who is an SPC spec editor) proposals for the next steps in this space.

Background

Secure Payment Confirmation (SPC) is a Web API that supports streamlined authentication during a payment transaction. It adds the following payment-specific capabilities on top of WebAuthn:

- The ability to register a WebAuthn credential (i.e., call `navigator.credentials.create()`) in a cross-origin iframe.
- The ability to initiate an authentication ceremony (via SPC) from a domain other than the Relying Party's.
  - The browser will only allow this ceremony to happen if (a) the caller can provide RP ID and specific credential IDs from the Relying Party, and (b) if the Relying Party opted into allowing this cross-origin usage at credential creation time.
- A 'Transaction UX' that is shown for SPC authentication flows, informing the user of the payment details that are being verified.
  - Importantly, this UX is shown conditionally based on whether a matching credential is available, and a fallback experience is shown otherwise.
- Payment details embedded in the WebAuthn assertion (by extending CollectedClientData with new payment fields).

Providing these behaviors requires overriding WebAuthn behavior. Currently, SPC defines these overrides in its own specification (§5 WebAuthn Extension - "payment") rather than in WebAuthn.

The implementation of SPC in Chrome also utilizes a local browser cache of known 'SPC-marked' credentials, in order to lookup which credentials are marked as SPC-enabled. This cache is necessary due to lack of authenticator-level support, but has two notable consequences:

1. The 'SPC-enabled' status of a given credential is browser-bound, rather than authenticator-bound.
    a. If one creates an SPC-enabled credential on Chrome and then switches to Edge (with access to the same authenticator device), Edge will not be able to detect the credential as SPC-enabled.
2. It conflates two concepts that are intended to be distinct for SPC:
    a. Whether a given credential is available for the current device (used to determine whether to show the Transaction UX or not)
    b. Whether a given credential is opted into the cross-origin authentication ceremony (as Relying Parties must opt-in to allowing a credential to be used this way).

Ideal End-State

We envisage the ideal end-state as:

1. An SPC specification that doesn't contain any overrides of WebAuthn-owned behavior.
2. An implementation being able to rely on authenticator-provided APIs (rather than a browser cache) in order to:
    a. look up credential existence (without a user gesture)
    b. check if a credential is opted-in for use in a cross-origin payment authentication
3. Any WebAuthn credential being able to be used in an SPC flow on the RP's origin.
4. Only credentials marked as "thirdPartyPayment" being able to be used in an SPC flow from a non-RP origin.
5. SPC works with both platform and roaming authenticators.

The Plan

List of Changes

A (hopefully exhaustive) list of changes that have to happen to achieve our ideal end-state. They are split into creation-time and authentication-time.

Creation (SPC spec section):

- Allowing create() in a cross-origin iframe should be removed from the SPC 'payment' extension and added to WebAuthn.
  - This should continue to require a user-activation, but the required permission policy should change from 'payment' to 'publickey-credentials-create' (to mirror the existing policy in WebAuthn).
- The marking of SPC-enabled credentials should be moved from the browser cache to a CTAP-supported bit in the credential itself.
  - In other words, land the "thirdPartyPayment" extension in FIDO.
  - This also effectively renames the "payment" extension at creation time to "thirdPartyPayment". To avoid breakage, Chrome will continue to support "payment" as an alias - but will deprecate it over time.
- Authenticator devices must support the "thirdPartyPayment" extension.
  - Platform authenticators will need to be updated to allow clients to request that the bit be stored.
  - Roaming authenticators will not support it until they either get a firmware update (very rarely supported in roaming authenticators) or new hardware comes out.
- **TODO**: After the above, there are still additional client processing steps currently defined in SPC:
  - SPC forces authenticatorAttachment=platform, residentKey=required, userVerification=required, in order to ensure that the created credentials work with the current setup.
  - Probably should just be removed and turned into (non-normative?) RP guidance in the SPC spec?
- Once there are no client processing steps left, the "thirdPartyPayment" extension doesn't need to be defined in either SPC or WebAuthn.
  - However the SPC/WebAuthn spec will reference it (see below).

Authentication:
- Authenticator devices must support querying for credential metadata without a user gesture. This is the same API as required for WebAuthn's non-modal UI concept.
  - Some platform authenticators have begun experimenting with this (e.g., `WebAuthNGetPlatformCredentialList` on Windows)
  - Roaming authenticators will require CTAP changes and firmware/hardware upgrades.
    - Per agl@'s comment, and discussion in the FIDO TWG, this may be doable 'today' as long as the credentials are created with credProtect=0 or credProtect=1. Once then does a uv=0 getAssertion, which confirms existence (and, with a CTAP change, the bit existence), without a user interaction.
- Authenticator devices must include the "thirdPartyPayment" extension data as part of the above returned metadata.
  - Platform authenticators will need to be updated for this.

  - Roaming authenticators will require CTAP changes and firmware/hardware upgrades.
- There still needs to be a "payment" extension specified (in either SPC or WebAuthn), which enables the following behavior:
  - Allow cross-origin auth ceremony, **if** (and only if) a matching credential was created with the "thirdPartyPayment" extension.
  - Modify the CollectedClientData with additional payment info.
  - **TODO**: This extension is a bit weird - it is 'internal only' in that a caller is not allowed to specify it in a get() call. They must go through the SPC API.
- [OPTIONAL] We could support roaming authenticators without a listing API, via a change to the SPC UI flow.
  - This could be done by making the 'no matching creds' dialog offer the user the chance to use a roaming authenticator.
    - We may still want a listing API for available credentials, for already-available roaming authenticators
  - Alternatively, WebAuthn/FIDO folks have discussed a sort of platform-caching step when a roaming authenticator becomes available to the platform. Likely much longer term.
- [OPTIONAL] The "payment" extension could be spec'd in CTAP to pass the payment info to the authenticator as well.
  - Suggestion: start with client-only, extend later.

Milestone Breakdown

With some aggressive dates - meant to provoke discussion!

Milestone #1 (Q2 2022)

- FIDO "thirdPartyPayment" extension PR lands.
- SPC spec begins to allow for separating "payment" and "thirdPartyPayment" concepts.
  - One potential solution (TBD):
    - Add a **temporary** creation-time "spc" extension, which triggers the browser to cache the credential for use in SPC but does not mark it as third-party payment enabled.
    - Add support for "thirdPartyPayment", which triggers the browser to mark it as such.
    - For compat, alias the old "payment" extension to do **both** of the above at creation time.
      - In the long-term future, the "payment" extension will eventually become authentication time only and internal, see Q3 2022.
- Allowing create() in a cross-origin iframe moved from SPC to WebAuthn.
  - AI: smcgruer@google.com to re-open issue with justification, & send a draft PR.
- Platform authenticators add support for listing credentials without a user interaction.

Milestone #2 (Q3 2022)

- Platform authenticators start to add support for "thirdPartyPayment" at creation time.
- Platform authenticators start to return the "thirdPartyPayment" extension as part of the credential metadata in their listing credentials API(s).
- Move the "payment" authentication-time extension definition from SPC spec to WebAuthn spec.
    - Reminder - at this point the extension is:
        - authentication only
        - client processing only
        - responsible for allowing a cross-origin authentication ceremony (if thirdPartyPayment set), and for putting payment data into CollectedClientData.

Milestone #3 (Q4 2022 - Q1 2023)

- Once platform authenticators fully support the "thirdPartyPayment" extension:
    - Drop the temporary "spc" extension from the SPC spec (Milestone #1 (Q2 2022))
        - (Only required if we go the route of requiring such an extension)
    - Remove the browser cache.
- CTAP work to support credential-listing APIs without user interaction.
    - (Or some other approach to support roaming authenticators for SPC).

Milestone #4 (Q2 2023+)

- Remove the deprecated "payment" **create()**-time extension from the SPC spec.