WebOTP in cross-origin iframes

THIS DOCUMENT IS PUBLIC

Status: Final
Authors: yigu@chromium.org
Last Updated: 2021-02-09

tl;dr

We plan to support WebOTP in cross-origin iframes[1] with a new feature policy `otp-credentials`. The hostname of the iframe should be appended[2] to the last line of the SMS. e.g. `@top.com #1234 @iframe.com`. A typical flow looks like this:

Figure 1. Example of using WebOTP in a cross-origin iframe. When the SMS with the expected format arrives, a prompt pops up with both origin information. Upon clicking "Allow" the code automatically fills in.

[1] The caller iframe cannot have more than one unique origin in its ancestor chain. See below for details.
[2] The SMS format only needs to change when using WebOTP in an cross-origin iframe.

Background

In the initial launch of the WebOTP API, we deliberately ignored the **cross-origin iframe** support because we didn't find enough demand that outweighs the risk / complexity. Post launch, we believe that there are three pillars to consider when reprioritizing this work: partner signal, interoperability and cost of execution.

Motivations

Partner signal

We have been seeing stronger signals from some major partners. e.g. Shopify has been experimenting with the WebOTP API and the first results are promising. They use Shop Pay to authenticate users across multiple Shopify stores (**more than a million**). There are **two equally important** use cases:

1. Authenticating on the first party Shop Pay domain `shop.app/pay`
2. Authenticating on a merchant domain and using an iframe to `shop.app` to pop the SMS confirmation dialog

Unfortunately WebOTP does not work in the second use case due to the lack of support in cross-origin iframes.

This could be a common pattern on the web because many real-world websites delegate authentication to use auth widgets from other origins. e.g. some of our partners (Twitter, Facebook etc.) support embedded contents on third party websites. WebOTP could serve as a 2-step verification for identification purposes to access certain contents.

In addition to the third party authentication use case, some cross-origin iframe could be from a separate service run by the same party. One such example is icloud.com, which uses apple.com for authentication. However, it's not clear whether this is a common pattern.

Interoperability

Safari did mention that they had received requests from several customers/clients and they have implemented the iframe support in iOS 14. We should engage to ensure that we agree on an interoperable implementation.

Cost of execution

After the initial investigation, we believe that the implementation should be straightforward. See below for details.

SMS format

WebOTP complies with the SMS text format defined in "Origin-bound one-time codes delivered via SMS" specification. The existing format for origin-bound OTC is:

```
@example.com #123456
```

Although Safari does not support WebOTP API at the moment, they shipped in iOS 14 the `autocomplete="one-time-code"` (OTC) to enable the OS to detect the one time password from an SMS. It also complies with the origin-bound OTC format.

As mentioned above, the iframe support is in high demand for Apple due to the “icloud.com and apple.com” separation. Therefore they shipped the support with

```
@topframe.com #123456 %iframe.com
```

In the pull requests (1, 2), we brought up some concerns regarding the format. In particular,

1. Prefer @iframe.com to avoid the use of a new token
    a. special characters introduce problems with SMS aggregators. e.g. `@` is misencoded as `¡` in some non-English regions.
    b. both top frame and iframes are frames and we should use the same token for consistency
2. this format conflates the audience with the content metadata. i.e. the origins are separated by the code

Note that **Spotify has adopted this format** on their Webkit implementation and hopes we do the same.

After some discussions with Safari engineers, we believe that we could find convergence with the following format:

```
@topframe.com #123456 @iframe.com
```

which addresses concern #1 above while preserving backwards compatibility.

Nested iframes

From the extensibility’s point of view, we should support an arbitrary level of nested iframes in the long run. In theory, we should include origins from all the intermediate frames to reduce ambiguity. For example, consider the following artificial example:

```
store.com -> productA.com -> ShopPay
```

```
store.com -> productB.com -> ShopPay
```

On the random merchandise website store.com there are two products from two different origins. However both products use ShopPay to verify the payment info. In this case, when ShopPay sends out the SMS, it must include the intermediate origin otherwise the user will receive two SMSes with the same format:

```
SMS 1: @store.com #1234 @shop.app
SMS 2: @store.com #5678 @shop.app
```

However there are some practical concerns:

- Including all intermediate origins may easily break the 160 character limit for SMS (and 67 char for UCS-2 encoding)
    - Note: this could be mitigated in verification via email
- Including all intermediate origins may confuse users
- There are security concerns for the innermost iframe to access top-frame's origin to use it in the SMS.

As a result, we decided to limit the support to cross-origin iframes who have no more than 1 unique origin in its ancestor chain. In the following scenarios:

- a.com -> b.com
- a.com -> b.com -> b.com
- a.com -> a.com -> b.com
- a.com -> b.com -> c.com

using WebOTP in b.com will be supported but not in c.com.

Note that the following scenario is not supported because of lack of demand and UX complexities.

- a.com -> b.com -> a.com

Design

Existing logic

Currently there is at most one `WebOTPService` per `RenderFrame`. When the API is called via `navigator.credentials.get()`, an `WebOTPService` is created in `RenderFrameHostImpl` if and only if the caller frame and its ancestors have the same origin. This origin is added to `SmsQueue` via `SmsFetcher::Subscribe()`. Later when the expected SMS is parsed with both "origin" and "code", we look up the origin in `SmsQueue` and handle it if such origin is found.

Proposed new logic

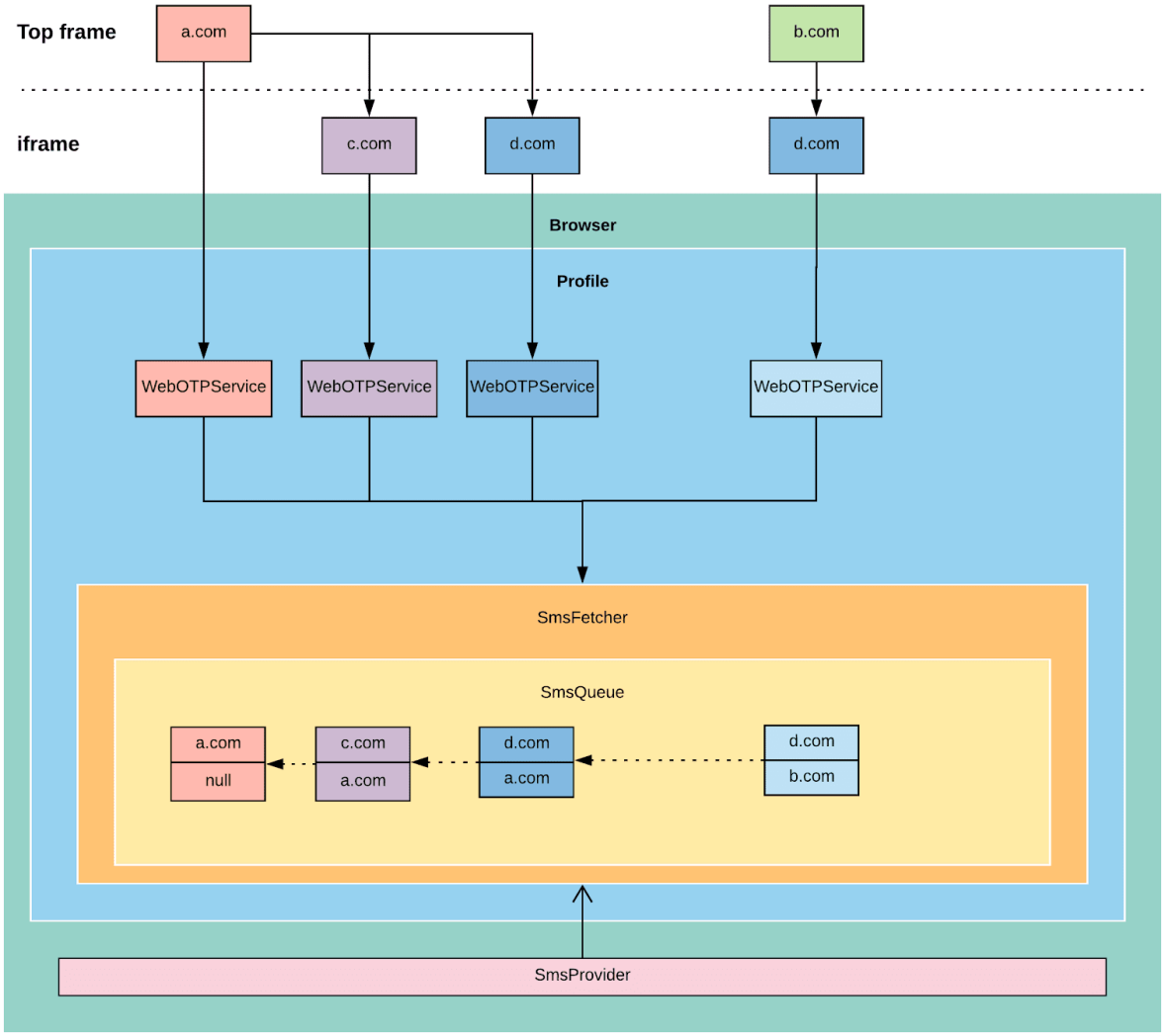

Figure 2. New structure to support WebOTP in cross-origin iframes

As shown in Figure 2, suppose WebOTP API is used on **a.com, c.com and d.com** where a.com and b.com are top frames while c.com and d.com are cross-origin iframes (widget etc.). We proposed the following changes to support WebOTP in cross-origin iframes:

1. Drop the restriction on same-origin mentioned above upon creating `WebOTPService` in `RenderFrameHostImpl` and `CredentialsContainer` such that c.com and d.com are able to create their own `WebOTPService`.
    a. This step requires adding a new Feature Policy. See below.
    b. Note that if the current frame has more than two different origins in its ancestor chain, abort the process.

2. Upon creating the `WebOTPService`, in addition to initializing its member `origin_` with the current frame, we create an `origin_list` with the caller and its ancestor node if it's cross-origin by calling `RenderFrameHost::GetParent()`. If more than one cross-origin ancestor is found, abort the process.
3. Upon subscribing, we add a subscriber to `SmsQueue` with its key being the `origin_list` instead of just an origin. The 4 origin_list in the example above will be:
    a. `[ a.com ]`
    b. `[ c.com, a.com ]`
    c. `[ d.com, a.com ]`
    d. `[ d.com, b.com ]`
4. Upon parsing the incoming SMS, we try to extract all the origin information and look it up in `SmsQueue`. If a match is found, handle that request. e.g. when the SMSes come with the following arbitrary order:
    a. `@a.com #1234 @c.com`
    b. `@a.com #2345`
    c. `@b.com #3456 @d.com`
    d. `@a.com #4566 @d.com`
    we will look up `[ c.com, a.com ]` from the queue and handle that first followed by the rest of SMSes.

Permissions Policy

By default, `navigator.credentials.get()` requires a secure origin and all its ancestor frames must have the same origin as the requestor does. In some cases, e.g. for Web Authentication API, the credentials registrations and assertions should be obtained in cross-origin iframes if the top frame explicitly delegates that authority. This is supported by a `publickey-credentials` feature policy.

Similarly, we should add an `otp-credentials` permissions policy to achieve the same goal. Unlike Web Authentication API, we do not need to delegate the registration. i.e. `navigator.credentials.create()` is still disabled in cross-origin iframes. Sample code:

```
<iframe allow="otp-credentials" src="https://yi-gu.github.io/webotp/"></iframe>
```

Note that Permissions Policy does last across page navigations.

Milestones

Launch tracking bug

- Prototype **[DONE]**

    - Patch, Demo
- Meet with the community to reach consensus on the SMS format. We should consider both the interoperability and backward compatibility. **[DONE]**
    - Spec discussion
- Launch a devtrial and iterate on it with at least one partner (Shopify). **[DONE]**
    - Public guideline
- Get approval from privacy / security review **[DONE]**
- Send out I2S **[DONE]**
- Enable the feature by default **[DONE]**

Security Considerations

OTP boundary

The cross-origin iframes should not have access to OTP code bound to the top frame. i.e. each frame has to have access only to the OTP meant for itself. For example:

- shop.com embeds payment.com and ads.com; payment.com calls WebOTP API
    - A compromised ads.com cannot see the OTP used by payment.com
    - shop.com cannot see the OTP used by payment.com neither

Switch

A top frame should be able to control the use of WebOTP in the cross-origin iframes via Permissions Policy.

Nested cross-origin iframes

Nested cross-origin iframes may lead to security concerns without proper control. With the current proposal, we would only support cross-origin iframes who have no more than 1 unique origin in its ancestor chain.

Privacy Considerations

Origin Confusion

There are some privacy considerations for the general WebOTP API. Supporting WebOTP in cross-origin iframes may introduce more **unintentional** tracking/fingerprinting because a user may not be aware of the fact that they are providing PII to a third party. e.g. a highly deployed iframe widget will be able to use the information to conduct targeted attack or fingerprinting.

To mitigate this concern, we should explicitly notify the user in the SMS confirmation prompt such that they are aware of the destination. e.g. showing origins from both the top frame and iframe in the SMS. Note that this could be confusing without proper UX.

In addition, we could add metrics to capture a “<top frame, iframe>” pair to see whether there is a sign of abuse and come up with mitigations accordingly.

UX Review

There are two different Permission UI required for WebOTP based on two backends: the Android native prompt and the Chrome prompt. Given that Chrome has little control over the UI of the Android native prompt, the **UX review is more focused on the Chrome prompt**.

Android native prompt

This is currently the **default** implementation and relies on Android native permission dialog. Chrome has little access to the content there so the entire SMS is shown on the prompt. e.g.

Figure 3. Android native prompt. Both top frame and iframe origins are shown as part of the SMS.

Chrome prompt

This prompt is controlled fully by Chrome. It was used during the Origin Trial (original UX review thread) and will be used in the future (under dev-trials in M88 now).

Figure 4. Chrome Prompt that only shows the code and two origins.

We expect most of the cases after M88 should use Chrome prompt . But there are still **rare scenarios that we may fall back to Android native prompt**. (For example when Chrome is not the default browser. See this document for more information about the two backends.)

There are some open questions regarding the UI.

Which origin(s) should be shown?

A typical use case of supporting WebOTP in cross-origin iframes is third-party payment verification. e.g. `shop.com` embeds an iframe `payment.com` which calls the WebOTP API to complete the payment request. In this case, which origin(s) should we show to users on the prompt?

There are 3 options in general.

1. Show the origin of the top frame: `1234 is your code for shop.com`
   This leads to serious privacy issues. e.g users may not be aware of the iframe at all upon verifying the code.

2. Show the origin of the iframe: `1234 is your code for payment.com`
   This is preferable at the moment and has been used in similar APIs such as WebAuthn. However it could also confuse users into granting access for an origin other than the one they are visiting. Branding the iframe could mitigate the issue but it's not always guaranteed. In addition, this could be worse if the origin of the iframe is not well recognizable like GooglePay or Shopify.

3. **Show the origins of both frames** (**preferable**)
   From the privacy point of view this is preferable because it shows users the complete information. However, showing two origins is more confusing than showing one. It's also hard to convey the message clearly. After several rounds of discussions we chose the following string:

   `1234 is your code for payment.com to continue on shop.com`

   because it shows the iframe first for accuracy purpose and avoids the use of parentheses or passive voice. Considered alternatives: (more details here)

   A. `1234 is your code for shop.com and payment.com`
   B. `1234 is your code for payment.com on behalf of shop.com`
   C. `1234 is your code for shop.com. Sent by payment.com`
   D. `1234 is your code for shop.com (through payment.com)`
   E. `1234 is your code for payment.com (used by shop.com)`

Should we trim the origin(s)?

This is not new for WebOTP but could be more noticeable for cross-origin iframe support. Showing more than 1 origin might make the prompt very big. Given the potential risks, we should either not trim the origins or do it from the beginning similar to TWA Running in Chrome.