Third-Party Origin Trials

This Document is Public

Authors: chasej@chromium.org
Last Updated: 2020-08-12

One-page overview

Summary

Origin trials can now be enabled by embedded scripts, based on tokens matching their third-party origin. Previously, any script can inject tokens to enable an origin trial, but only when the token matches the origin of the containing document. This intended to allow third-party providers to experiment more effectively via origin trials.

Platforms

Mac, Windows, Linux, Chrome OS, Android.
Origin trials are not supported in Android WebView.

Team

chasej@chromium.org
experimentation-dev@chromium.org

Bug

crbug.com/1069623

Code affected

Origin trial framework, JavaScript bindings, DOM manipulation, script loading.

Design

Motivation

As in the origin trials explainer, the goal is to encourage broad participation and feedback from web developers, while preventing burn-in of features or hampering iteration. Origin trials currently require opt-in/attribution for first-party origins, which causes undue friction for third-party providers. For some features, third-party providers are often in the best position to experiment and provide feedback.

Today, it is possible for any script embedded in a document to enable an origin trial for that context. Scripts can inject a <meta> tag containing a valid trial token, which is matched against the origin of the containing document.

For any third-party provider willing to register multiple target origins (i.e. embedder origins), the current approach is workable. For some third-party providers, it is infeasible to know or register all the target origins in advance (e.g. CDNs, ad tech providers). Specific challenges include:

- Size of target origin population: may require hundreds of origins (or more) to form a representative population for meaningful experiment results. Registering and deploying hundreds of trial tokens is a significant burden (especially if staggered rollouts are desired).
- Variability in target origins: determining target origins may require sampling from the entire set origins served, and either the candidate set or sampling criteria change rapidly. As well, target origins could be selected at runtime by the server (especially when limiting usage to a small percentage).

Today, each trial has a usage limit of 0.5% of Chrome page loads, to prevent websites from relying on the feature (i.e. burn-in). For third-party providers that are widely-used, a representative population of target origins could exceed the 0.5% limit. To gather meaningful experiment results, third-party providers need alternate mechanisms to prevent burn-in.

See Trust Token API, as a motivating example for third-party participation.

High-Level Design

The current processing of script-created trial tokens will be extended in two ways:

1. Match against the origin of the external script that injected the token, instead of the origin of the containing document.

        - *NOTE: For the purposes of token injection, matching or validation, "origin" refers to the response URL origin. That is, based on the URL after any redirects, not the request URL origin.*
2. Apply different usage restrictions, which may prevent a trial from being enabled, even with an otherwise valid trial token. These usage restrictions are an alternative to the global limit of 0.5% of Chrome page loads.

Web developers will need to explicitly opt-in into this different behaviour when registering for an origin trial.

The overall flow (including process, logistics and runtime behaviour):

- In the Intent to Experiment, support for third-party origins is requested and explicitly approved.
- The origin trial is configured to allow third-party origin trial tokens (both in Chromium and the developer console).
- During trial registration, web developers request a third-party token, and optionally a different usage restriction.
    - The third-party registration is reviewed and approved, if necessary.
- Trial token is issued with additional "third-party" metadata.
- Third-party web developers provide the issued token, by injecting a <meta> tag from their external script embedded on the desired origins.
- Using the "third-party" metadata, the browser validates the token against the external script origin, and any different usage restrictions.
- (Optional) When valid, the browser dynamically adds a JavaScript binding for trial detection, e.g. `` `navigator.<trial name>TrialEnabled` ``.
    - This should only be required if the trial-controlled feature is lacking suitable feature detection.

The following sections explain in more detail and show how it doesn't materially impact the ability of origin trials to limit usage of features and attribute usage to specific web developers.

Alternate Usage Restrictions

The current limit on usage is defined as 0.5% of Chrome page loads, measured in a way to allow for temporary spikes of higher usage. The allowance for temporary spikes is expected to be insufficient for some third-party providers to experiment effectively.

Web developers will be able to choose to one of these restrictions, if the global usage limit isn't suitable:

- Exclusions of a random subset of users, which cannot enable the trial.
- On/off cycles for longer, predictable durations.
    - E.g. Trial is available for higher usage (>> 0.5%) for 1 month, followed by 1 month where the trial is completely disabled for the origin.

In both cases, the restriction is applied during runtime validation of a provided trial token. A third-party token, which is otherwise valid, can fail to enable a feature due to one of the restrictions being applied.

When one of these restrictions are applied, a higher threshold is used for the global usage limit. Instead of 0.5% of Chrome page loads, the threshold is 5%(?) when the built-in safeguard will disable the trial for all users.

User Subset Exclusions

With a subset exclusion, a small percentage of users are randomly selected to have the trial always disabled. The field trial ("Finch") infrastructure is used to define a target population, and the 10%(?) of Chrome users for which the trial is always disabled. The field trial infrastructure provides "permanent consistency", such that users are assigned to a population that remains consistent across sessions and browser restarts. The exclusion subset is defined based on the trial, not individual tokens. It should not be possible to have multiple tokens to effectively generate an empty exclusion subset.

The approach:

- Setup a new field trial, using the existing field trial infrastructure
    - Given the user exclusion approach, this most likely should not reuse an existing field trial for the feature.
- Configure the field trial with a small experimental group for the target exclusion percentage. The control group will have the trial enabled.
- Configure a mapping from origin trial name -> exclusion field trial name.
    - Similar to the approach in SetRuntimeFeaturesFromChromiumFeatures(), as described in Initialization of Blink runtime features in content layer.
    - Exact code location TBD.

On/Off Cycles

The trial is available for higher usage (>> 0.5%) for a defined duration, followed by the same duration where the trial is completely disabled for the origin, e.g. on for 1 month, then off for 1 month.

Web developers will need predictability and control over the cycle, to account for rollout of the trial token and associated functionality. Upon registration, developers can specify the start date of the first on cycle, subject to appropriate limits (e.g. must be less than 1 month in the future). From the provided start date, the cycle schedule is a mechanical calculation (e.g. on for period 1 and following odd periods, off for even periods).

Trial Configuration

As part of the Intent to Experiment (I2E) process, API Owners will explicitly approve individual trials to allow third-party origin registration. There may be trials where it is not appropriate to allow third-party providers to broadly enable features.

Chromium Client-Side Configuration

The runtime enabled features configuration will be extended with a new flag, `` `origin_trial_allows_third_party` ``. Similar to origin_trial_allows_insecure, this configuration is used to implement additional checks or optimize generated code.

Similar to navigation origin trials, it may be appropriate to require explicit approval to land CLs enabling third-party origins for trials. This is intended as a backstop to the process approval by API owners on the I2E.

Developer Console Server-Side Configuration

The developer console has configuration values for each trial, which control both the UI and processing logic. Similar to the Chromium client configuration above, new values will be added:

- Flag to indicate "allows third-party origins".
- List of alternate usage restrictions that are enabled. For example, "none" or some combination of "user subset" and "on/off cycle".

The full implementation details are internal and outside the scope of this document.

Trial Registration and Token Issuance

Trial registration in the developer console now requires additional checks and a separate approval process. In particular, it is a non-goal to allow a JavaScript library or framework to act as a polyfill and enable a feature for all users.

The current registration process has the capability to block specific origins for a trial, but otherwise allows any user to register for any origin. Previously, this was sufficient because:

- The global usage limit (0.5% of page loads) applied regardless of the number of tokens issued.
- Every token for a combination of trial and origin is functionally equivalent, given the global duration for when a trial is enabled. Registering repeatedly to issue multiple tokens did not enable additional or longer usage (other than potentially affecting feedback collection).
- Origin "ownership" is irrelevant, as tokens are only useful if the developer has access to provide the token in server responses from the origin.

The alternate usage restrictions for third-party tokens could be bypassed or abused if unconditionally issuing tokens:

- For on/off cycles, register the same origin with different cycle start dates, to generate tokens with overlapping on cycles. Then provide all of the tokens in the third-party script.
- Register an origin for both first-party and third-party tokens. The existence of the third-party token could relax the global usage limit, and then provide the first-party token in non-gated mechanisms (e.g. HTTP header) for excessive usage.

Trial Registration Process

The registration process is largely handled by the developer console:

- The "Register for Trial" page will present options to request a third-party token.
    - Only for trials configured to allow third-party tokens.
- For third-party tokens, the web developer can choose from three options for usage restrictions:
    - As described in Alternate Usage Restrictions.
- If the standard 0.5% limit is chosen, immediately issue a third-party token (as today).
    - A third-party token is defined in Changes to Trial Token Format.
- If either of the alternate usage restrictions are chosen, the registration is marked as "pending", and an offline process for approval is initiated:
    - The approval process is out of scope here. This is assumed to require human intervention, based on some kind of "trusted tester" program or partnership arrangements. This may include some kind of verification of ownership for origins.
    - The approval process is responsible for ensuring that multiple or improper tokens are not issued to bypass the usage restrictions.
    - The outcome is that the request is either approved or rejected.
- When the registration is approved, a third-party token is issued and the developer is notified.
    - A third-party token is defined in Changes to Trial Token Format.

Changes to Trial Token Format

Trial tokens are essentially a versioned JSON payload, which is then encoded and signed for verification (format definition). The token format will be changed:

- Increment version to 3 (current version is 2)
- Add fields to JSON payload, which are all optional:
    - `` `isThirdParty` ``: boolean; default to false.
    - `` `usage` ``: one of "" (empty), "cycle", "subset"; default to empty.
    - `` `cycle` ``: a dictionary, containing:
        - `` `startDate` ``: date for to begin the first on cycle; required.

        - `duration`: integer, number of days per cycle; default is 30.

As the fields are all optional, versions 2 and 3 could likely be parsed by a single implementation, reducing the complexity of supporting multiple versions.

Trial Token Injection and Validation

Third-party tokens will only be supported when injected by a script-created <meta> tag. Trial tokens can also be provided by HTTP response header or <meta> tag in markup (see developer guide). Those mechanisms will remain unchanged and not support third-party tokens.

Token Injection

The processing of script-created <meta> tags shares the implementation with markup-declared <meta> tags in `HttpEquiv::Process`. For <meta> tags created by an external script, the origin information will be captured, and passed along for token validation.

The process for capturing the external script origin (based on cl 2173473):

- Get the url of the currently executing script, by querying the V8 stack. If the url exists, continue.
- Pass the url and token to `OriginTrialContext::AddTokenFromExternalScript()`.

If any of the above steps/conditions are not met, the token will be processed as a first-party token, via `OriginTrialContext::AddToken()`.

This approach may not work for all script execution paths. For example, if <meta> tags are injected by callbacks provided by external script, but the callbacks are invoked elsewhere. Initially, it seems reasonable to limit support to easier script injection scenarios, and require web developers to adjust if needed. There are already restrictions on how tokens must be provided, e.g. web developers must ensure that tokens are provided early in the page lifecycle, before attempting to use the feature enabled by the trial.

Token Validation

Token parsing and validation is performed by TrialTokenValidator::ValidateToken. A token must be parsed first to determine if it should be matched to a third-party origin, or the current origin. Thus, both the current origin and external script origin will be passed into ValidateToken().

TrialTokenValidator::ValidateToken

The validator uses an OriginTrialPolicy plugin, to allow Chromium embedders to control if/how trial tokens are validated.

The validation algorithm:

- If the token is well-formed, continue.
    - Well-formed means it has the expected format, with appropriate values, and has a valid signature.
- If it is a third-party token and an external script origin was provided:
    - Validate the token, passing the external script origin to TrialToken::IsValid().
    - If valid, continue.
- Else for a first-party token:
    - Validate the token, passing the current origin to TrialToken::IsValid().
    - If valid, continue.
- If the feature is not disabled, continue.
    - Determined by calling OriginTrialPolicy::IsFeatureDisabled()
- If the token is not disabled, continue:
    - Determined by calling OriginTrialPolicy::IsTokenDisabled()
- If it is a third-party token and usage = "on/off cycle":
    - Compute if currently in an on cycle, using the start time in the token.
    - If current time is within an on cycle, continue.
- If it is a third-party token and usage = "subset":
    - Determine if the trial should be disabled for the current user, by calling OriginTrialPolicy::IsFeatureDisabledForUser(). See below.
    - If not disabled, continue.
- If all the above checks succeed, the token is valid.

OriginTrialPolicy::IsFeatureDisabledForUser

This method is used to allow for embedder-specific logic to disable the trial for individual users. The Chrome policy implementation will use a Finch trial to exclude some users, as described in User Subset Exclusions.

An alternative approach would be to include the user exclusion logic in the existing OriginTrialPolicy::IsFeatureDisabled() method. The token validation returns specific codes to explain why a token isn't valid. It seems useful to be able to distinguish between a trial that is disabled for everyone (or even by platform), vs disabled for a certain user.

Alternatives Considered

- Add an imperative JavaScript API, e.g. navigator.originTrial.addToken().
    - A purpose-built API would be easier to implement in Chrome.
    - This exposes a non-standard API to every web page in Chrome. This could become yet another way for pages to browser-detect Chrome (and variants)
    - This doesn't seem to provide additional benefit to web developers vs injecting a <meta> tag.

- Support the `` `Origin-Trial` `` HTTP header on external scripts.
    - Anecdotally, some web developers prefer to use the <meta> tag, as it's easier to change markup, rather than the HTTP headers. For third-party providers that must carefully select when to provide a third-party token, it seems likely to be easier to selectively include scripts on a request vs dynamically changing headers.
    - The current implementation for the HTTP header is tied to document loading, as it relies on the SecurityOrigin of the document. Supporting the header on scripts would require a larger refactoring of the implementation.

Metrics

Success metrics

Success is defined by third-party web developers effectively participating in origin trials. This will be determined qualitatively, by feedback from trial authors and web developers. Participation rates in origin trials are expected to be too low to collect meaningful quantitative metrics.

Regression metrics

Speed launch metrics, especially for loading, to verify that capturing the origin of external scripts doesn't have an adverse impact.

Experiments

This functionality will be used in origin trials for other features. Given an origin trial is an experiment on its own, it's not feasible to experiment using Finch.

Rollout plan

Waterfall (standard rollout).

Core principle considerations

Speed

Generally, the origin trials functionality is triggered by the presence of a trial token. If no tokens are provided, there is negligible impact on speed, memory, etc. (ignoring the obvious impact of code size). Depending on the final implementation, capturing the third-party script origin may require additional memory usage for all external scripts, regardless if a trial token is provided. This should be minimal (i.e. storing a single URL per script), but may have an impact.

Security

Third-party origin trials do not enable any additional capabilities. It is already possible for embedded, cross-origin scripts to provide trial tokens to enable features. This functionality may increase the frequency at which trials are used to enable features, as it intentionally reduces the friction for web developers. Features exposed by origin trials are already subject to security reviews as needed, so this should not increase the security risk.

Trial tokens are base64-encoded, cryptographically-signed blobs, which are parsed and validated to enable features. As previously described, the token format is modified to add properties to the contained JSON payload. The token signature is extracted to verify the token contents came from a trustworthy source (i.e. for Chrome, that Google issued the token). The signature is verified with a public key that is shipped with the browser itself. In the event that the signing key is updated, the shipped public key may be replaced by a server-pushed component update. Only after the token contents are cryptographically verified, will Chromium parse and validate the tokens (otherwise, tokens are ignored as invalid).

Typically, tokens are provided by a document to enable its use of features directly, so the parsing and validation is done in the renderer process. In other scenarios, the parsing and validation may be done in the browser process, or another process (e.g. for service workers, or features implemented in the browser process). The JSON payload is parsed using the existing Chromium parser. The changes to the token format do not introduce any novel parsing techniques. The parser is also tested with a fuzzer.

Privacy considerations

As described in Security, third-party origin trials may increase the frequency at which features are enabled. Features exposed by origin trials are already subject to privacy reviews as needed. Third-party trials can only be used by scripts that are embedded in a page, so it should not be possible to fingerprint users or gather any data that was not already available.

Trial tokens are validated using only configuration stored locally, with no network access required.

Testing plan

The current origin trials functionality is covered by automated tests, both unit tests and Blink web tests. Tests will be added for full coverage of the third-party functionality. Manual testing should not be required.

Followup work

The approval process for token issuance needs to be designed in more detail. That is out of scope for this document.

crbug.com/1124390 - Record the usage of third-party trials.