Link Click Analytics and Privacy

Apr 11, 2019
by John Wilander
@johnwilander

We've recently received questions on what we refer to as link click analytics, and specifically an internal setting for disabling the Ping attribute for anchor elements.

Privacy By Default

WebKit always strives for privacy by default. To name three unique such efforts – we partition third-party data storage and ServiceWorkers by default, we partition HTTP caches by default, and our Intelligent Tracking Prevention (ITP) feature is on by default in Safari. To the best of our knowledge, no other browser on the market offers similar protections.

However, there are cases when users want even stronger privacy guarantees and are willing to trade some functionality or web compatibility for it. Two such examples are Private Browsing and Content Blockers.

Let's have a look at how Safari and WebKit's privacy features play into link click analytics.

What Is Link Click Analytics?

The goal of link click analytics is to report to a web server that a navigational link click happened and that the user is leaving the webpage. Such auditing can be used for first-party web analytics as well as third-party cross-site tracking. The latter is where ITP comes in.

How Can Websites Do Link Click Analytics?

There are several ways for websites to do link click analytics. The ones we see in use today are:

- Synchronous XHR (XMLHttpRequest).
- Asynchronous XHR or Fetch, with a delay.
- First-party bounce tracking.
- The Beacon API.
- The Ping attribute.

Let's go through the details of these techniques.

Synchronous XHR

Synchronous XHR is the synchronous version of XMLHttpRequest. Such synchronous calls often cause hangs on the web since they block the webpage while it waits for the server's response. Therefore, web browsers are actively trying to remove the API.

In the context of link click analytics, a site that is willing to inconvenience users might do something like the following (**please don't use this technology**):

```
window.addEventListener("unload", function(event) {
  let xhr = new XMLHttpRequest(),
      data = captureTrackingData(event);

  xhr.open("post", "/log", false); // 'false' implies synchronous.
  xhr.send(data);
});
```

The code above triggers when the user clicks a link and the current webpage unloads. The synchronous XHR blocks the navigation until it's done which delays the navigation significantly. For users, this is perceived as poor performance.

Asynchronous XHR or Fetch, With a Delay

Another popular way to do link click analytics is through asynchronous XHR or Fetch, not in the

least because they allow for cross-site requests to third-party trackers.

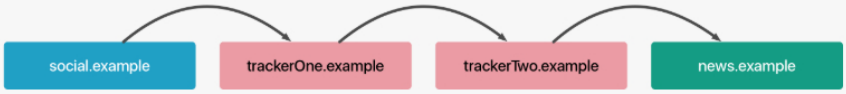
While not blocking all execution on the webpage, this introduces an artificial delay to the navigation which users experience as poor performance. We've seen between 100 and 350 ms delays in such link click analytics scripts, and even some that do a busy loop while waiting. This is what a delayed navigation with asynchronous XHR looks like:

```
const clickTime = 350; // Milliseconds.

theLink.addEventListener("click", function(event) {
  event.preventDefault(); // Cancel the user's link click.

  let xhr = new XMLHttpRequest(),
      data = captureTrackingData(event);

  xhr.open("post", "/log", true); // 'true' implies asynchronous.
  xhr.send(data);
  setTimeout(function() {
    window.location.href = theLink.getAttribute("href");
  }, clickTime); // Navigate properly after 350 ms.
});
```

First Party Bounce Tracking

You might recall from our earlier blog posts that ITP 2.0 detects first-party bounce trackers and classifies them as any other kind of cross-site tracker.

Say the user clicks on a news.example link on the social.example website. Instead of navigating them straight to their destination news.example, they are rapidly navigated through trackerOne.example and trackerTwo.example before reaching news.example.



This is a way to let trackerOne and trackerTwo do link click analytics and the increased load time is again bad for performance.

The Beacon API

While not specifically built for link click analytics, the Beacon API is a way of sending arbitrary analytics and/or tracking data without affecting the user experience. This is how Beacon can be used for link click analytics:

```
window.addEventListener("unload", function(event) {
  let data = captureTrackingData(event);
  navigator.sendBeacon("https://tracker.example/", data);
});
```

Beacon requests are guaranteed to be initiated before the page unloads but do not block the webpage or delay the navigation.

The Ping Attribute

Ping is an attribute on anchor elements, popularly referred to as just links. The purpose of Ping is link click analytics, plain and simple. Here's how it's used:

```
<a href="https://news.example" ping="https://tracker.example/going-to-news-exampl
```

The Ping request to tracker.example above does not block or delay the navigation to news.example.

What Can WebKit Do About this?

As can be seen above, websites have several ways to go about logging or tracking a user's clicks. The first three — synchronous XHR, Fetch with delay, and first-party bounce tracking — all hurt performance and make the web experience worse. The latter two — Beacon and Ping — still log clicks but do so without hurting performance.

Just turning off the Ping attribute or the Beacon API doesn't solve the privacy implications of link click analytics. Instead, it creates an incentive for websites to adopt tracking techniques that hurt the user experience. In effect, the choice between supporting Ping and not is not one of privacy, rather it is a choice between a good user experience and a bad one.

So our approach is to have ITP block cookies and downgrade the referrer header (see the section on Origin-Only Referrer in our ITP 2.0 blog post) for *all* the link click analytics techniques listed when the request goes to a third-party domain classified with cross-site tracking capabilities. ITP also cleans up website data for first-party bounce trackers.

The distinction we're making here is between analytics of link clicks in general versus third-party analytics of link clicks tied to individual users. The latter is what ITP prevents and what we think is the right balance for on-by-default privacy protections.

What Can Users Do About This?

For users who want to fully block third-party link click analytics, WebKit and Safari supports Content Blockers. The effect of such load blocking may include blocking of ads or third-party widgets. If you want to install a Content Blocker, check out the App Store where you'll find plenty of offerings. If you are a developer who wants to build a Content Blocker, see Apple Developer Documentation.

A Final Note On Hidden Feature Flags

As a detail of its implementation, WebKit makes use of the User Defaults mechanism on Apple platforms. These flags or pieces of configuration data are not exposed in Safari's menus or in Safari Settings. Instead, they are used to control the inner workings of features to, for instance, enable quality assurance testing.

Say WebKit decides to obsolete synchronous XHR altogether on Apple platforms. Such a change might be put behind a User Defaults flag so that it's easy for engineers to assess whether a reported issue stems from the obsoleted synchronous XHR, or something else.

Until recently, Safari supported an internal User Defaults flag to disable support for the Ping attribute. It was never our intention to surface this flag as a customer setting. We think it's misguided to offer users the ability to disable web-facing features if doing so doesn't disable or prevent the ends of that technology. Instead, Intelligent Tracking Prevention and Content Blockers offer users different levels of support for categorically affecting link click analytics. ■

Next

Release Notes for Safari Technology Preview 80

Learn more ›

Previously

New WebKit Features in Safari 12.1

Learn more ›