Geolocation API Removed from Unsecured Origins in Chrome 50

By Paul Kinlan
Paul is a Developer Advocate

Chrome has public intent to deprecate powerful features like geolocation on non-secure origins, and we hope that others will follow.

Starting with Chrome 50, Chrome no longer supports obtaining the user's location using the HTML5 Geolocation API from pages delivered by non-secure connections. This means that the page that's making the Geolocation API call must be served from a secure context such as HTTPS.

It is an important issue as it will directly impact any site that requires use of the geolocation API and is not served over https, but it is a change that we do believe is beneficial to all users on the web. This post should help you understand the reasoning and how to proceed.

When is this changing?

This change is effective as of Chrome 50 (12PM PST April 20 2016).

Chrome's developer tools console has been providing warnings since version 44 (released July 21 2015).
There have been a number of public announcements that describe the rationale (and discussion) of why we are making this change:

- Intent to deprecate set of powerful features over HTTP (Feb 2015)
- Intent to deprecate Geolocation API over HTTP (Nov 2015)
- Chrome Dev Summit (Nov 2016)
- Chrome Beta Channel release blog (March 17, 2016)
- Chrome Status website

There have been a number of other sources that have highlighted this: Mobiforge (Jan 26, 2016), Wired (March 17, 2016), VentureBeat (April 13, 2016).

Why are we making this change?

Location is sensitive data! Requiring HTTPS is required to protect the privacy of your users' location data. If the user's location is available from a non-secure context, attackers on the network will be able to know where that user is. This seriously compromises user privacy.

Who does this affect?

This affects any page currently using the Geolocation API from pages served over HTTP (non-secure). It also affects HTTPS iframes that use the Geolocation API if they are embedded in HTTP pages. (You won't be able to polyfill using a shared HTTPS-delivered frame.)

Does my whole web app need HTTPS?

It is *not* a requirement that the whole app be served via HTTPS to use Geolocation. Only pages that use Geolocation need to be served over a secure context. A secure context currently is anything hosted at the top level on HTTPS or localhost. For example, an iframe that points to a secure origin but is hosted on unsecured origin (**http** ://paul.kinlan.me/) would not be allowed to call the geolocation API.

We strongly suggest that you migrate to HTTPS as powerful new and existing browser features require secure origins.

Does this affect local development?

It should not, localhost has been declared as "potentially secure" in the spec and in our case geolocation requests served at the top level over localhost will still work.

Can I detect at runtime if the geolocation was blocked because of not being on a secure context

Yes. The geolocation spec defines a PositionError object that is passed in to the failure callback of the Geolocation APIs. The object defines a **code** and **message** properties.

Errors due to this secure-context issue will return a **code** of 1 which is a "Permission Denied Error". You can get this error when a user has denied access or the system has denied access to the user's locations. This means you will have to check the message to see what the exact reason was.

This can be quite brittle as it might change in the future, but a strong signal that it was a non-secure content issue is to look for the string "Only secure origins are allowed".

```
navigator.geolocation.getCurrentPosition(function(success) { /* Do some mag
  function(failure) {
    if(failure.message.indexOf("Only secure origins are allowed") == 0) {
      // Secure Origin issue.
    }
  };
});
```

Remember, you can't just check for the origin of the page because your page could be on https but inside an iframe that is hosted from an unsecure context.

I really need to use Geolocation; What should I do?

If you would like to use the HTML5 Geolocation API, or if your site already uses the Geolocation API, please migrate the pages making Geolocation API calls to HTTPS, ensuring that they're used in a secure context.

There are a number of fallback options available to get a user's location that are not affected by this change, such as Google Maps Geolocation API, GeoIP (as an example, there are other geo based solutions), and a user-entered zip code. However, we *strongly recommend* that the best path to ensure ongoing access to geolocation is to move to HTTPS.

Last updated July 2, 2018.