DRAFT: HTTP Referers

Dan Auerbach

11/27/2012

Proposal: Firefox implements a relaxed same origin policy for sending referers based on hostname. If the host of the request URI differs sufficiently from the host of the current URI based on reasonable similarity measure*, the referer is not sent. If the host is sufficiently similar, a full referer URI is not sent but rather is stripped to just the intersection of the domain names of the various hosts. This policy should be active by default for all users.

* One could just ask whether it is the same exact host. However, perhaps a similarity measure could be devised around higher-level domains so that a.example.com and b.example.com are considered sufficiently similar to warrant sending a referer.

I. Background

Referers are part of the HTTP specification. When a client loads a resource over HTTP, the client has the option to include a Referer header in the request; the referer is the URI from which the resource is being requested, i.e the user's current page. So if I am on example.com/superprivateurl and click a link to evil.com and my client by default sends a referer header to evil.com with the value “example.com/superprivateurl”.

Referers were well-known to be privacy-invasive from their inception. For example, consider this excerpt from the HTTP 1.0 spec where they are defined:

> Because the source of a link may be private information or
> may reveal an otherwise private information source, it is strongly
> recommended that the user be able to select whether or not the
> Referer field is sent. For example, a browser client could have a
> toggle switch for browsing openly/anonymously, which would
> respectively enable/disable the sending of Referer and From
> information. --RFC 1945 (1996)

Fast forward 16 years and there is no such easy switch in any major browser. In Firefox, you must use “about:config” to modify the preference, which is a huge barrier for most users. In Chrome and MSIE, there is no option within a standard instance for users to browse without referers without installing an extension at a minimum. Referers remain turned on in private browsing modes.

II. Privacy concern: personal information

The privacy concern is no less serious today, but in fact is heightened by ways in which the web has evolved. First, the enormous success of search engines means that a user's search terms are passed to the website that is visited. Most users would not voluntarily disclose these search terms, yet are unaware that it is happening in the background.

In addition to search terms, URIs increasingly contain more information. It's not uncommon for usernames or other unique identifiers to be caught in the mix. Extremely irresponsible websites may even put plain-text passwords in the URI.

III. Privacy concern: third-party data leak

As the presence of third parties on websites continues to grow, referers represent a huge floodgate through which data leaks from first parties to third parties. It should not be default behavior for a third-party to have access to the URI of the page on which they appear. Although this issue can be mitigated somewhat with iframes and other mechanisms (and does not protect against third-party JS), it is often not considered carefully by website operators who don't have user privacy in mind. The technology needs to change – if a first-party wants to give a third-party access to the URI of the page where the third-party appears, that should be done explicitly e.g. as a URL parameter.

IV. Survey of website use cases for referers

Here we outline the known use cases for referers.

1. ***Keeping track of state.*** Cookies are currently the primary mechanism for keeping track of state for a session in HTTP. Doing so with referers is well-known to be unreliable and insecure. For instance, a website has to pass sensitive session information in the URI which could be snooped and altered by an attacker.
2. ***Hot-linking protection.*** If a website does not wish to host content embedded by other websites without attribution (e.g. images), referers can be used to attempt to ensure that users come only through the webpage. This can be done, e.g., via .htacess rules. However this type of hot-linking protection is often not used for the modern web. For one, websites often want to be hot-linked in certain circumstances – for example Google image search – but not others. As such, this is at best an imperfect solution for hot-linking protection.
3. ***CSRF protection.*** For CSRF protection, a website may insist that a referer is sent for certain resources to prevent CSRF. However, this does not provide robust CSRF protection and it not considered a best practice. For example, the popular OWASP (https://owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet#Checking_The_Referer_Header) notes:

   > Checking the referer is considered to be a weaker from of CSRF protection. For example, open redirect vulnerabilities can be used to exploit GET-based requests that are protected with a referer check. It should be noted that GET requests should never incur a state change as this is a violation of the HTTP specification. There are also common implementation mistakes with referer checks. For example if the CSRF attack originates from an HTTPS domain then the referer will be omitted. In this case the lack of a referer should be considered to be an attack when the request is performing a state change. Also note that the attacker has limited influence over the referer. For example, if the victim's domain is "site.com" then an attacker have the CSRF exploit originate from "site.com.attacker.com" which may fool a broken referer check implementation. XSS can be used to bypass a referer check.

4. ***Deciding what content a user should see***. Some websites use the referer to decide what content to show a user. For example, a news site might put up a paywall unless the user comes from google.com. This does not provide any robust protection; a user could trivially circumvent such a technical protection measure by sending a forged referer. Moreover, if Google would like to cooperate with a website in order to indicate to the website that the user came from Google, this can be accomplished explicitly via URL parameter or other mechanisms.
5. ***Tracking users.*** Finally, websites use referers to track users and build aggregate websites stats. For instance, websites may be interested in the search terms that users use which lead them to the website.

V. Impact of proposal on users

If the proposal is implemented, users will enjoy enhanced privacy since the full referer URI will never be sent, minimizing the chances that the referer will be a vector for leaking PII to third-party websites. Moreover, implementing the proposal will make the web work more in line with user expectation.

VI. Impact of proposal on websites

Use case (1) is not appropriate and websites should not employ it. Use cases (2)-(3) will be largely unaffected by the proposed change, though implementation details may have to change.

For use cases (4)-(5) remain possible but requires creating an explicit mechanism and cooperation between websites, instead of relying on the HTTP Referer header.

VII. Conclusion

This change would plug a huge privacy hole that exists today. It would stop data from leaking from first-party sites to third-party actors, and preventing sensitive user information from appearing in plain text in HTTP logs.

This change preserves use cases related to website security, even though use cases (2) and (3) above have questionable value. The change does not prevent websites from gathering statistics about users, but rather forces any data collection to be set up explicitly by cooperating websites and thus handled more responsibly. Moreover, making this change now would not be devastating to websites. As large websites have recently moved to HTTPS and referers are NOT sent from a URI with an HTTPS scheme to one with an HTTP scheme (https://tools.ietf.org/html/rfc2616#section-15.1.3), these HTTP websites have already had to live without referers in many cases. For example, Google's change to HTTPS for search for logged in users created such a situation for websites.