An Analysis of Cookie Sizes on the Web

13 Jul 2020

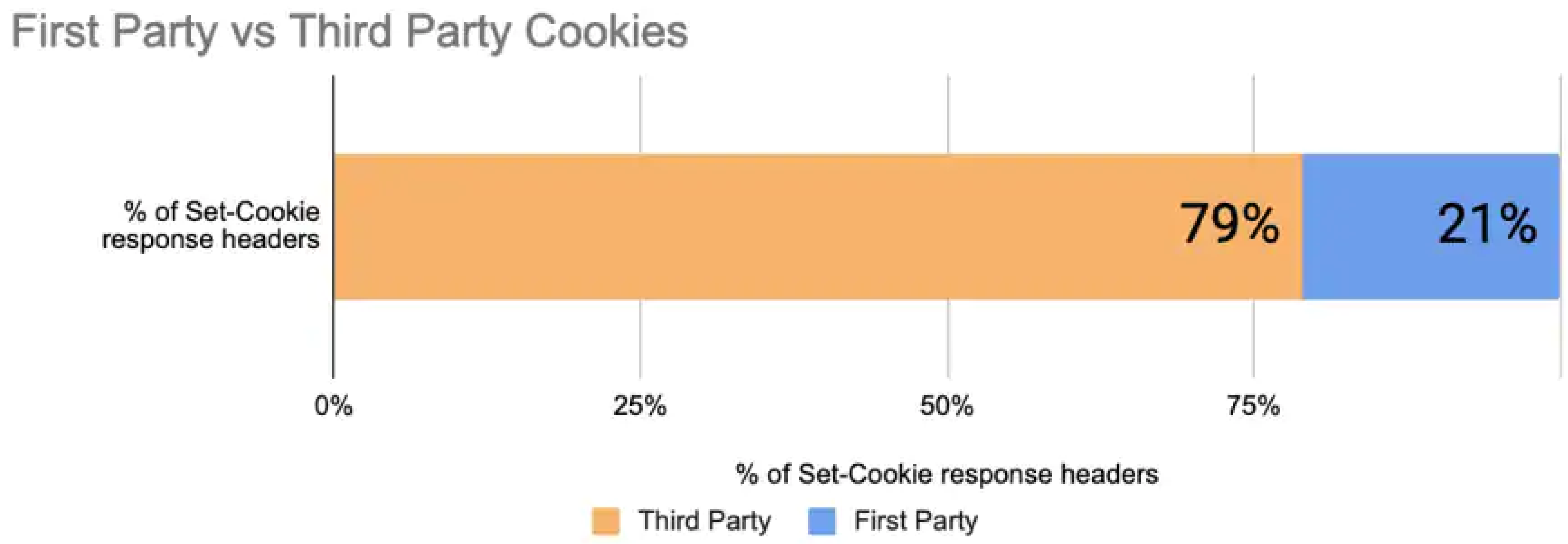
Cookies (https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies) are used on a lot of websites - 83.9% of the 5.7 million home pages tracked in the HTTP Archive (https://httparchive.org/) to be specific. They are essentially a name/value pair set by a server and stored in a client's browser. Sites can store these cookies by using the `Set-Cookie` HTTP response header, or via JavaScript (`document.cookie`). On subsequent requests, these cookies are sent to the server in a `Cookie` HTTP request header.

In this article we'll be looking at the size of cookies across the web, and discuss some of the web performance implications of them.

First vs Third Party cookies?

When a cookie is set by the domain you see in your address bar, it is considered a first party cookie. These can be used for session management, authentication, personalization, etc. When a cookie is set by a different domain, then it's considered a third party cookie.

Based on an analysis of over 109 million cookies, third parties account for 79% of all cookies.



Note: I wrote another blog post exploring the use of the SameSite attribute in cookie files, and how third party cookies are affected. You can read it here (https://dev.to/httparchive/samesite-cookies-are-you-ready-5abd) .

Cookie Sizes

An example of a cookie set by a server would be:

```
Set-Cookie: loggedIn=true; Domain=.example.com; Path=/; Max-Age=14400; HttpOnly
```

Directives such as `Domain` , `Path` , `max-age` , and `HttpOnly` affect how the cookie is stored, and which hostname a browser should share them with. In this example, `loggedIn=true` is the name/value portion of the cookie, and that is what we'll be exploring in this post.

The median length of all cookies in the HTTP Archive is 36 bytes as of June 2020. That statistic is consistent across both first and third party cookies. The minimum is just a single byte, usually set by empty Set-Cookie headers (which is likely an error).

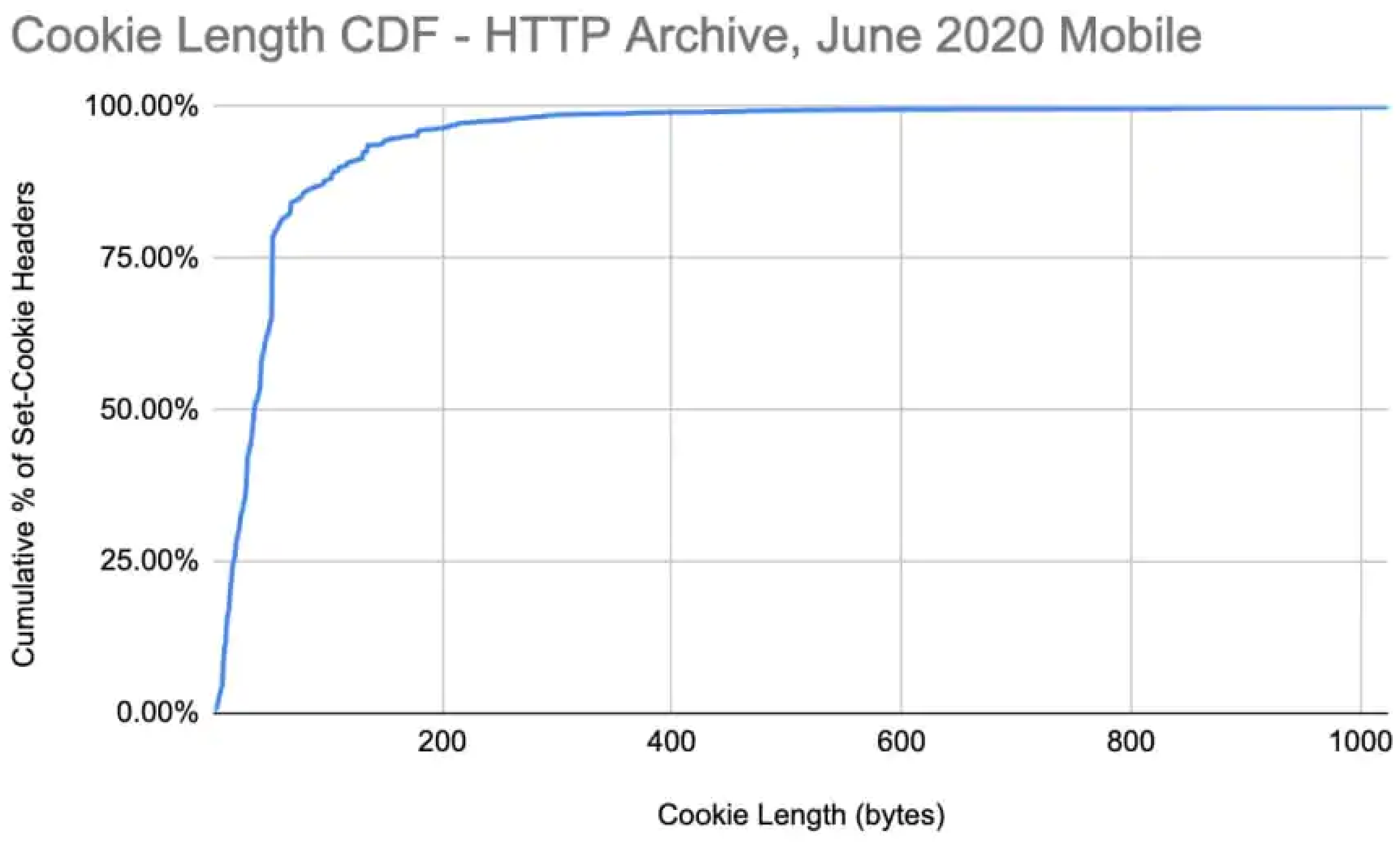
Cookie Size	First Party	Third Party
Min	1	1
Median	36	37
95th Percentile	181	135
99th Percentile	287	248
Max	29,735	8,500

The largest cookie size was 29,735 bytes, which is quite large! This is so large in fact that it is rejected by all modern browsers. I was curious to see what the limits are, and decided to dig into the source. Both Chrome (https://source.chromium.org/chromium/chromium/src/+/master:net/cookies/parsed_cookie.h;l=24) and Firefox (https://dxr.mozilla.org/mozilla-central/source/netwerk/cookie/CookieCommons.h#57) will reject cookies greater than 4KB. This is likely due to the implementation limits defined in RFC 6265 (https://tools.ietf.org/html/rfc6265#section-6.1) .

So who is setting these large cookies?

- The largest first party cookie was set by https://www.ridewill.it/ (https://www.ridewill.it/) , and it is named menu. It's value was a long urlencoded string that contained multiple `<div>` layers and links. All in all, there were 240 links in a single cookie!
- Many large first party cookies included large session cookies.
- The largest third party cookie was set by web.taggbox.com, and consisted of a large JSON array named "liveWall".
- Most of the largest third party cookies were set from web.taggbox.com as well as a small number of advertising third parties.

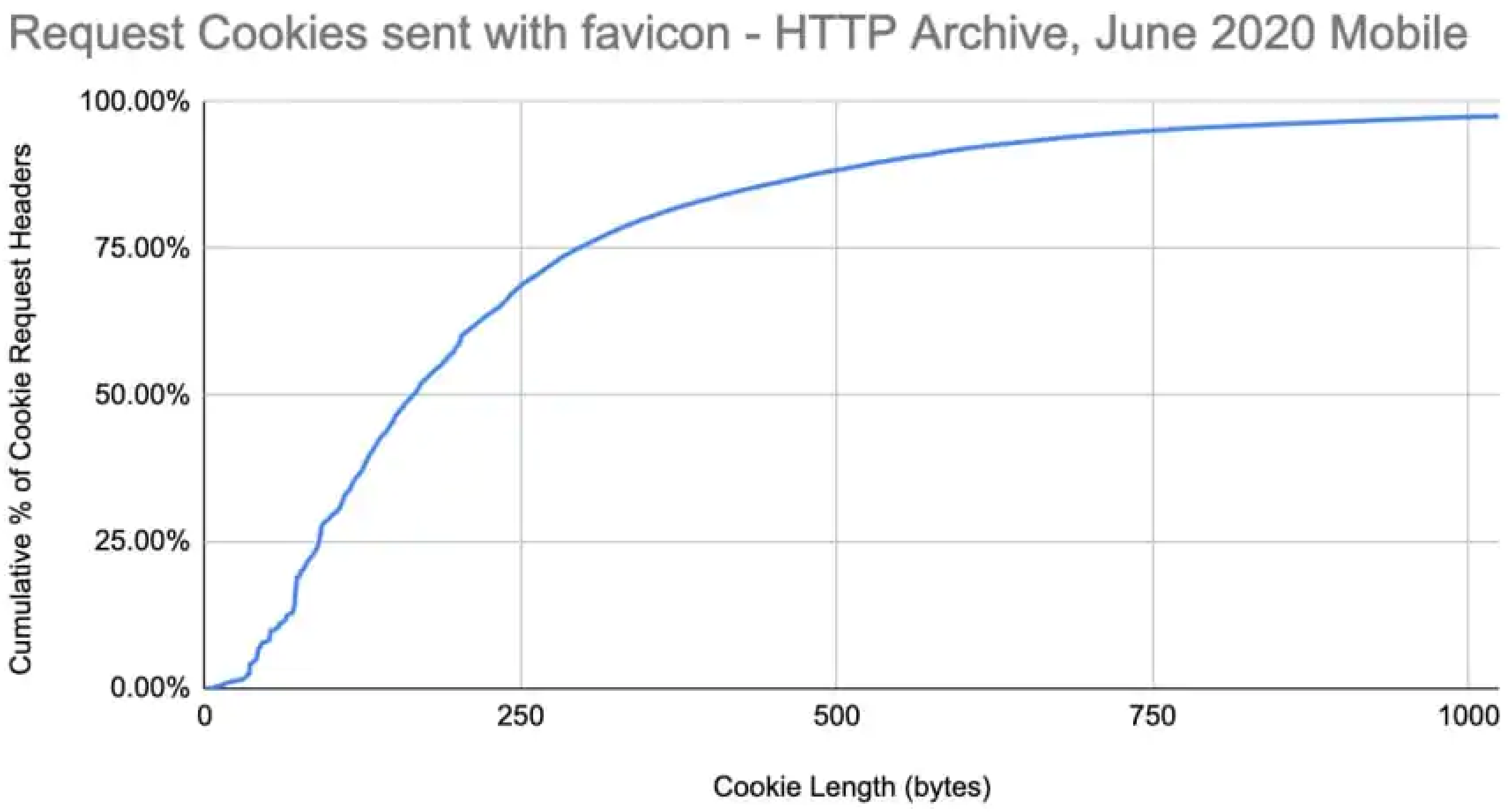
If we look at the entire distribution of cookie sizes, it gets even more interesting. 88% of the cookies being set are less than 100 bytes. The 99th percentile is 372 bytes. So really large individual cookies are not common.



Cookies Sent to First Party Domain

What about cookies that clients send back to servers? A client can send multiple cookies in a single `Cookie` request header. Since the HTTP Archive only collects information on homepages, there is a limit to the insight we can collect here. If we look at just the request headers for favicon.ico, we can get an idea of how large the Cookie request header might be for a subsequent request. However this does not include any additional cookies set later in a session (ie, such as after logging in)

The median size of cookies sent on the favicon request was 161 bytes and the 95th percentile was 681. The largest was 7,795 bytes, and you can see the distribution below.



It's important to note that the cookies set by the time favicon was requested may under represent the size of the cookies users would send later in a browsing session. For example, when logging into an application, a few additional cookies might be set. Some third parties that use a first party subdomain (ie, if www.example.com (http://www.example.com) loaded a resource from metrics.example.com) also set a first party cookie.

Performance Implications of Large Cookies

When a browser sends an HTTP request, the HTTP request headers are usually 400 - 500 bytes. In the example below the request headers total 407 bytes.

```
GET / HTTP/1.1
Host: www.example.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Adding a cookie to that will increase the size of the request header. If we add more than 1KB of cookies to that request, then we exceed 1500 bytes, which is the standard maximum transmission unit (MTU) (https://en.wikipedia.org/wiki/Maximum_transmission_unit) used by TCP. This means that the HTTP request would span multiple TCP packets, which may result in multiple round trips and increase the risk of retransmission. This can potentially increase the TTFB of the response since it would take longer to make the request.

With HTTP/2, we have HPACK compression (https://tools.ietf.org/html/rfc7541) - which was designed to help reduce the size of HTTP request headers by utilizing a dynamic index table. Receiving endpoints advertise the maximum size of this table (https://tools.ietf.org/html/rfc7540#section-6.5.2) in bytes (default is 4096), the sender can insert headers up to this limit in one request or response and subsequently reference it in another. In theory HPACK seems like it could help reduce the overhead of these large cookies. However, it’s not as easy as it sounds. Consider the following:

- If a request is made on a new HTTP/2 connection (ie, for a return visitor, or a navigation that happened after the previous connection times out), then the entire cookie string would be sent. This would impact TTFB for that first request (usually the HTML).
- Some servers intentionally exclude cookies from HPACK compression, since cookies are a very valuable target for an attacker.

Because of the practical constraints on compressing cookies, and the impact of cookie size on the first flight of requests and responses, it is beneficial to use smaller cookies. Based on the observations here, 900 bytes seems like a good budget for a total cookie size, which leaves room for other headers such as user-agent (which can benefit from HPACK compression).

Conclusion

Cookies are everywhere, and they are set by both first and third party requests. Most browsers will limit the size of cookies stored to 4KB - which is still quite large.

At a minimum, setting a cookie larger than 4KB will simply not work. However, 4KB is still too large and can negatively impact your TTFB by increasing the number of round trips on the HTTP response.

Even moreso, it’s important to keep track of how many cookies are being sent, as bloated cookies can also impact your TTFB by increasing the time it takes to make an HTTP request.

If you are interested in seeing some of the SQL queries and raw data used in this analysis, I’ve created a post with all the details in the HTTP Archive discussion forums (https://discuss.httparchive.org/t/analysis-of-cookie-size/1991) . You can also see all the data used for these graphs in this Google Sheet (https://docs.google.com/spreadsheets/d/1pO3lUWc41jw43rtk8JCxZi3tGFBxEV__Q-K4DZ0lc_s/edit?usp=sharing) .

Many thanks to Lucas Pardue (https://twitter.com/SimmerVigor) and Matt Ringel (https://twitter.com/ringel) for reviewing this.

About

Paul Calvano is a Web Performance Architect at Akamai Technologies, where he helps businesses improve the performance of their websites.

Related Posts

SameSite Cookies - Are You Ready? (/2020-07-07-samesite-cookies-are-you-ready/) **07 Jul 2020**

Impact of Page Weight on Load Time (/2018-07-02-impact-of-page-weight-on-load-time/) **02 Jul 2018**

Tracking Page Weight Over Time (/2017-08-16-tracking-page-weight-over-time/) **16 Aug 2017**