Vendors

In this lesson, we'll look at how browsers from individual vendors can impact the security of a web app.

We'll cover the following

The four most popular browsers belong to different vendors:

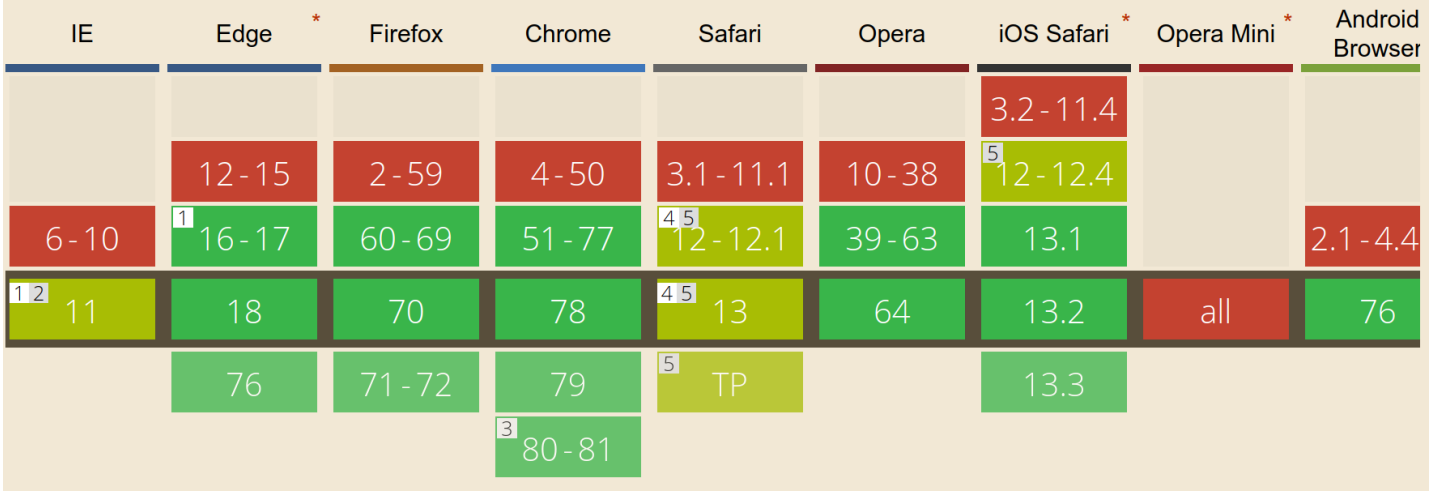
- Chrome by Google
- Firefox by Mozilla
- Safari by Apple
- Edge by Microsoft

Besides battling each other to increase their market penetration, vendors engage with each other in order to improve the **web standards**, a sort of minimum requirements for browsers.

Web standards

The W3C is the body behind the development of web standards, but it's not unusual for browsers to develop their own features that eventually make it as web standards, security is no exception to that.

In 2016, for example, Chrome 51 introduced SameSite cookies, a feature that would allow web applications to get rid of a particular type of vulnerability known as CSRF (more on this later). Other vendors decided this was a good idea and followed suit, leading to SameSite being a web standard. Today, all major browsers support SameSite cookies, with Safari being the last to jump on the ship in late 2018.



SameSite support across browsers

This tells us two things:

- Safari does not seem to care enough about their users' security (just kidding: SameSite cookies are available since Safari 12).
- Patching a vulnerability on one browser does not mean that all your users are safe.

The first point is a shot at Safari (as I mentioned, just kidding!), while the second information is really important. When developing web applications, we need to make sure that they look the same across various browsers, but also that they ensure our users are protected in the same way across platforms.

Cater to each browser's vulnerabilities

Your strategy towards web security should vary according to what a browser's vendor allows us to do. Nowadays, most browsers support the same set of features and rarely deviate from their common roadmap. Instances like the one above still happen, and it's something we need to take into account when defining our security strategy.

In our case, if in 2017 we decided that we were going to mitigate CSRF attacks only through SameSite cookies, we should have been aware that we were putting our Safari users at risk. Our users should have known that too.

Support some browsers

Last but not least, you should remember that you can decide whether to support a

browser version or not. Supporting every browser version would be impractical

(think of Internet Explorer 6); making sure that the last few versions of the major browser are supported, though, it's generally a good decision. If you don't plan to offer protection on a particular platform, it's generally advisable to let your users know.

> ### 🔍 Don't support outdated browsers
>
> You should never encourage your users to use outdated browsers, or actively support them. Even though you've taken all the necessary precautions, other web developers might not have. Encourage users to use the latest supported version of one of the major browsers.

Vendor or a standard bug?

The fact that the average user accesses our application through a third party client (the browser) adds another level of indirection towards a clear, secure browsing experience The browser itself might present a security vulnerability.

Vendors generally provide rewards (aka *bug bounties*) to security researchers who can find a vulnerability on the browser itself. These bugs are not tied to your implementation, but rather to how the browser handles security on its own.

Bug bounties

The Chrome reward program lets security engineers reach out to the Chrome security team to report vulnerabilities they have found. If these vulnerabilities are confirmed, a patch is issued, a security advisory notice is generally released to the public and the researcher receives a (usually financial) reward from the program.

"Bounty Hunter" from https://www.sketchport.com/drawing/5217381955665920/a-bounty-hunter-bounty

Companies like Google invest a

relatively large amount of capital into their Bug Bounty programs, as it allows them to attract researchers by promising a financial benefit should they find any problem with the application.

In a Bug Bounty program, everyone wins: the vendor manages to improve the security of its software, and researchers get paid for their findings. We will discuss these programs later on in the course, as I believe Bug Bounty initiatives deserve their own chapter in the security landscape.

> **ℹ Jake discovered a browser bug!**
>
> Jake Archibald is a developer advocate at Google who recently discovered a vulnerability impacting more than one browser: He documented his efforts, how he approached different vendors, and their reactions in an interesting blog post that I highly recommend.

In the next lesson, we'll look at a browser for developers.