X-Content-Type-Options

In this lesson, we'll see how 'MIME-sniffing' introduced a vulnerability and how to use the X-Content-Type-Options header to avoid it.

We'll cover the following

MIME-sniffing

Sometimes, clever browser features end up hurting us from a security standpoint. One example is MIME-sniffing, a technique popularized by Internet Explorer.

MIME-sniffing is the ability for a browser to auto-detect (and fix) the content type of a resource it is downloading. Say for example, we ask the browser to render an image at `/awesome-picture.png`, but the server sets the wrong type when serving it to the browser (ie. `Content-Type: text/plain`), this would generally result in the browser not being able to display the image properly.

In order to fix the issue, IE went to great lengths to implement a MIME-sniffing capability. When downloading a resource, the browser would scan it and if it detected that the resource's content type is not the one advertised by the server in the `Content-Type` header, it would ignore the type sent by the server and interpret the resource according to the type detected by the browser.

The catch

Now, imagine hosting a website that allows users to upload their own images, and imagine a user uploading a `/test.jpg` file that contains JavaScript code. See where this is going? Once the file is uploaded, the site would include it in its own HTML and, when the browser would try to render the document, it would find the image the user just uploaded. As the browser downloads the image, it would detect that it's a script instead, and execute it on the victim's browser.

The fix

To avoid this issue, we can set the `X-Content-Type-Options: nosniff` header that completely disables MIME-sniffing. By doing so, we are telling the browser that we're fully aware that some file might have a mismatch in terms of type and content, and the browser should not worry about it, we know what we're doing, so the browser shouldn't try to guess things, potentially posing a security threat to our users.

In the next lesson, we'll study cross origin resource sharing.