The reporting API

In this lesson, we'll study the reporting API.

We'll cover the following

In late 2018, Chrome rolled out a new feature to help web developers manage browser reports of exceptions. Amongst the issues that can be managed with the reporting API there are security ones, like CSP or feature-policy violations.

In a nutshell, the reporting API allows a website to advertise to the browser a particular URL it expects to receive reports to. With the `Report-To` header, a server can inform the browser to hand violations over at a particular URL.

```
Report-To: {
  "max_age": 86400,
  "endpoints": [{
    "url": "https://report.example.com/errors"
  }]
}
```

This API is still experimental so browser support is in the very early stages, but it's an interesting tool to keep in mind in order to easily manage browser reports, and not just security ones. The reporting API can be used to receive information regarding multiple aspects of our users' experience on our web application, such as:

- CSP and feature-policy violations
- **risky code**: the browser will sometimes intervene and block our code from performing a specific action as we've seen with CSP and the X-XSS-Protection headers
- **deprecations**: when our application uses an API that the vendor is planning

to deprecate

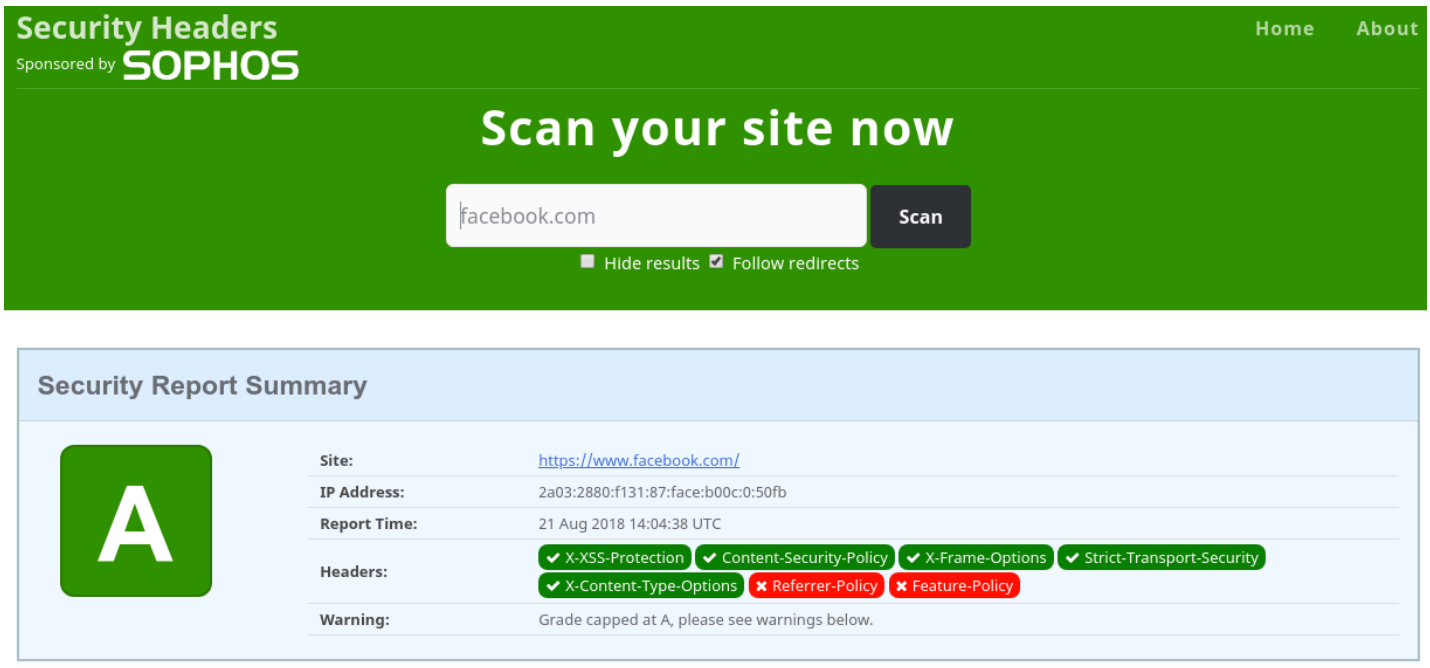
- **crashes**: when our application has caused the client to crash

The reporting API is very useful and is not burdensome to implement. Consider adding it to your web app in order to make your job easier and get notified when exceptions happen on the client.

> ### report-uri.com
>
> If setting up a URL that can capture browser reports isn't feasible, consider using report-uri.com, a service that allows you to collect and analyze browser reports with a single line of code.
>
> The service is free for up to ten thousand reports per month and is trusted by large organizations like the UK digital bank Monzo and the financial services provider Square Inc.

Testing your security posture

I want to conclude this chapter with a reference to securityheaders.com, an incredibly useful website that allows you to verify that your web application has the correct security-related headers in place. After you submit a URL, you will be handed a grade and a breakdown, header by header. Here's an example report for facebook.com.



Security headers result of Facebook.com

Security headers result of Facebook.com

If you're questioning on where to start, securityheaders.com is a great place to get a first assessment.

Stateful HTTP: managing sessions with cookies

This chapter should have introduced us to a few interesting HTTP headers allowing us to understand how they harden our web applications through protocol-specific features, together with a bit of help from mainstream browsers.

In the next chapter, we will delve deep into one of the most misunderstood features of the HTTP protocol, cookies.

Born to bring some sort of state to the otherwise stateless HTTP, cookies have been used and misused by each and every one of us in order to support sessions in our web apps. Whenever there's a state we'd like to persist it's easy to say "store it in a cookie." As we will see, cookies are not always the safest vaults and must be treated carefully when dealing with sensitive information.

Let's test your knowledge of protection via HTTP headers in the next lesson!