X-XSS-Protection

In this lesson, we'll study how the X-XSS-Protection header can be used to mitigate XSS attacks.

We'll cover the following

Introduction

Although superseded by CSP, the `X-XSS-Protection` header provides a similar type of protection. Unsupported by Firefox, this header is used to mitigate XSS attacks in older browsers that don't fully support CSP.

The syntax is very similar to what we've just seen.

```
X-XSS-Protection: 1; report=http://xssviolations.example.com/collector
```

Trying it out

Reflected XSS is the most common type of attack, where an unsanitized input is printed by the server without any validation. Here is where this header truly shines. **If you have an older version of Chrome (77 or below)** and want to see this in action, I would recommend trying out the example below, by appending `xss=on` to the generated URL. It shows what a browser does when XSS protection is turned on.

> This will not work with newer versions of Chrome

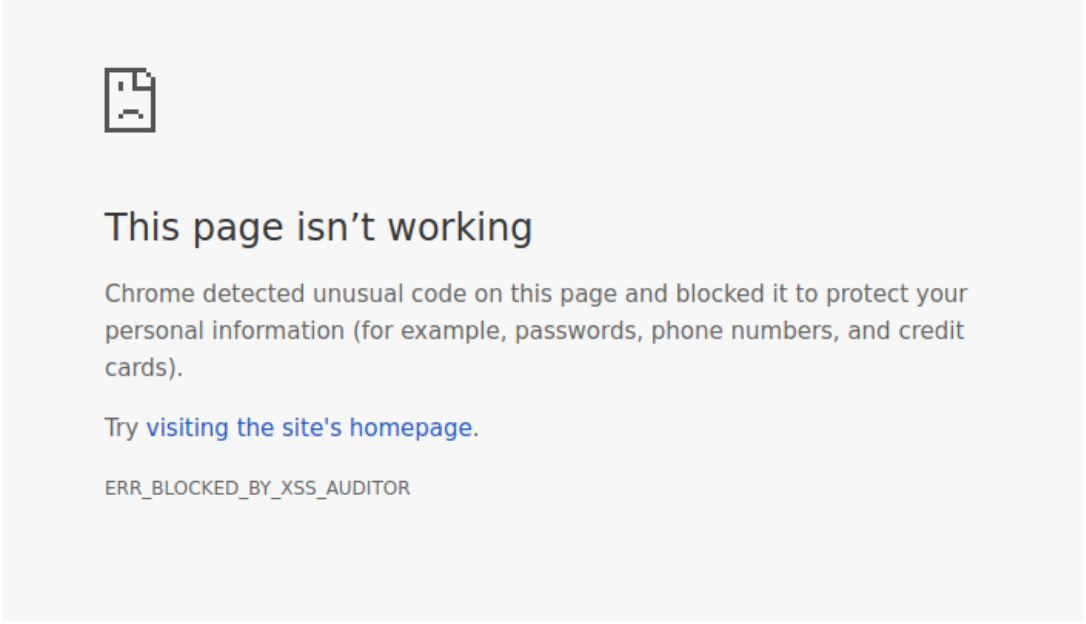
```
var qs = require('querystring')
var url = require('url')
var fs = require('fs')

require('http').createServer((req, res) => {
  let query = qs.parse(url.parse(req.url).query)
  let headers = {
```

```
    'X-XSS-Protection': 0
  }


  if (query.xss === "on") {
    headers['X-XSS-Protection'] = 1
  }

  if (query.xss === "off") {
    delete headers['X-XSS-Protection'];
  }

  if (query.csp === 'on') {
    headers['Content-Security-Policy'] = `default-src 'self'`
  }

  if (query.csp === 'report') {
    headers['Content-Security-Policy-Report-Only'] = `default-src 'self'`
  }

  res.writeHead(200, headers)
  let keyword = query.search || ''
  let results = keyword ? `You searched for "${keyword}", we found:</br><img src="http://placekitt

  res.end(fs.readFileSync(__dirname + '/index.html').toString().replace('__KEYWORD__', keyword).re
}).listen(7888)
```

If we enter a malicious string in our search box, such as `<script>alert('hello')</script>`, the browser will refuse to execute the script and explain the reasoning behind its decision.

```
The XSS Auditor refused to execute a script in
'https://x6jr4kg.educative.run/?search=%3Cscript%3Ealert%28%27hello%27%29%3C%2
Fscript%3E&xss=on'
because its source code was found within the request.
The server sent an 'X-XSS-Protection' header requesting this behavior.
```

Even more interesting was Chrome's default behavior when the webpage did not specify any CSP or XSS policy, a scenario we can test by adding the xss=off parameter to our URL.



Script blocked via xss parameter

Amazingly, Chrome was cautious enough that it would prevent the page from rendering, making reflected XSS very difficult to pull off.

> This feature was **disabled on Edge in mid-2018**, **with Chrome following suit a year later**. A combination of quirky behaviors and the release of more robust standards like CSP made this feature obsolete.

In the next lesson, we’ll look at the feature policy.