Ejercicio

1. Ingresar a https://codesandbox.io/s/csp-practice-forked-82ti2o y hacer fork del proyecto (crearse una cuenta en el sitio mediante Gihub preferentemente).
2. Tenemos un archivo `app.js`, que configura una Content Security Policy. Actualmente, no hace nada:

```
var express = require('express')

var app = express()
const csp = ''

app.use(
  express.static('public', {
    setHeaders: function (res, path) {
      res.set('Content-Security-Policy',
csp)
    },
  })
)
var listener = app.listen(8080, function
() {
  console.log(‘csp’ + csp)
  console.log('Listening on port ' +
listener.address().port)


  })
```

Ejercicio

1. Ingresar a: https://codesandbox.io/s/csp-practice-forked-82ti2o
2. Tenemos un archivo `app.js`, que configura una Content Security Policy. Actualmente, no hace nada:
3. Acceder a la URL de sandbox, por ejemplo: https://82ti2o.sse.codesandbox.io/ y ver los mensajes de la consola. El header no está haciendo nada. Analizar el tráfico de red y ver qué sucede al ingresar una password.

```
var express = require('express')

var app = express()
const csp = ''

app.use(
  express.static('public', {
    setHeaders: function (res, path) {
      res.set('Content-Security-Policy',
csp)
    },
  })
)
var listener = app.listen(8080, function
() {
  console.log(‘csp’ + csp)
  console.log('Listening on port ' +
listener.address().port)


  })
```

Ejercicio

default-src

Es la primera directiva que quisiéramos agregar.

4. Colocar default-src como *'none'*, incluyendo las comillas simples, de lo contrario, se interpretará que *none* es una URL.

```
let defaultSrc = "default-src 'none'"

const csp = [defaultSrc].join(';')
```

```
Content-Security-Policy: default-src 'none'
```

5. Restartear el server y cargar nuevamente el sitio. Observar los mensajes de la consola. Ahora la página es segura!! aunque el phishing sigue funcionando…

Ejercicio

form-action

Regula hacia qué sitios puede nuestra página enviar la información mediante formularios.

6. Agregar la directiva, quedando de la siguiente manera:

```
let defaultSrc = "default-src 'none'"

let formAction = "form-action 'self'"

const csp = [defaultSrc, formAction].join(';')
```

7. Restartear el server y cargar nuevamente el sitio. Intentar enviar la password...

```
Content-Security-Policy: default-src 'none'; form-action 'self'
```

Ejercicio

form-action

Regula hacia qué sitios puede nuestra página enviar la información mediante formularios.

6. Agregar la directiva, quedando de la siguiente manera:

```
let defaultSrc = "default-src 'none'"

let formAction = "form-action 'self'"

const csp = [defaultSrc, formAction].join(';')
```

7. Restartear el server y cargar nuevamente el sitio. Intentar enviar la password... mirar la consola:

```
Refused to send form data to 'https://www.appsecmonkey.com/evil' because it violates the following Content Security Policy directive: "form-action 'self'".
```

```
Content-Security-Policy: default-src 'none'; form-action 'self'
```

Ejercicio

frame-ancestors

Previene que otros sitios agreguen en frames nuestra página. Debería configurarse a ‘none’.

8. Agregar la directiva, quedando de la siguiente manera:

```
let frameAncestors = "frame-ancestors 'none'"

const csp = [defaultSrc, formAction,
frameAncestors].join(';')
```

9. Recargar el sitio del sandbox y observar qué ocurre…

```
Content-Security-Policy: default-src 'none';form-action 'self';frame-ancestors 'none'
```

Ejercicio

style-src

Permite restringir orígenes en la carga de estilos.

10. Notar los primeros errores de la consola. Arreglamos la directiva de la siguiente manera (same origin y googleapis). Luego recargar el sitio y observar nuevamente la consola:

```
let styleSrc = "style-src";
styleSrc += " 'self'";
styleSrc += " https://fonts.googleapis.com/";

const csp = [defaultSrc, formAction, frameAncestors, styleSrc].join(";");
```

```
Content-Security-Policy: default-src 'none';form-action 'self';frame-ancestors 'none';style-src'self'

https://fonts.googleapis.com/
```

Ejercicio

img-src

Permite restringir orígenes en la carga de imágenes.

11. Notar los primeros errores de la consola. Arreglamos la directiva de la siguiente manera (same origin y data). Luego recargar el sitio y observar nuevamente la consola:

```
let imgSrc = 'img-src'
imgSrc += " 'self'"
imgSrc += ' data:'

const csp = [defaultSrc, formAction, frameAncestors, styleSrc, imgSrc].join(';')
```

```
Content-Security-Policy: default-src 'none';form-action 'self';frame-ancestors 'none';style-src'self'

https://fonts.googleapis.com/;img-src 'self' data:
```

Ejercicio

font-src

Permite restringir orígenes en la carga de fuentes.

12. Notar los primeros errores de la consola. Arreglamos la directiva de la siguiente manera. Luego recargar el sitio y observar nuevamente la consola:

```
let fontSrc = “font-src https://fonts.gstatic.com”

const csp = [defaultSrc, formAction, frameAncestors, styleSrc, imgSrc, fontSrc].join(';')
```

```
Content-Security-Policy: default-src 'none';form-action 'self';frame-ancestors 'none';style-src'self'

https://fonts.googleapis.com/;img-src 'self' data:;font-src https://fonts.gstatic.com/
```

Ejercicio

font-src

Permite restringir orígenes en la carga de fuentes.

12. Notar los primeros errores de la consola. Arreglamos la directiva de la siguiente manera. Luego recargar el sitio y observar nuevamente la consola:

```
let imgSrc = 'img-src'
imgSrc += " 'self'"
imgSrc += ' data:'

const csp = [defaultSrc, formAction, frameAncestors, styleSrc, imgSrc].join(';')
```

```
Content-Security-Policy: default-src 'none';form-action 'self';frame-ancestors 'none';style-src'self'

https://fonts.googleapis.com/;img-src 'self' data:;font-src https://fonts.gstatic.com/
```

Ejercicio

script-src

Permite restringir orígenes en la carga de scripts.

13. Releer los errores de la consola. Ciertamente, no queremos cargar scripts del atacante. Pero agregamos a nuestra policy aquellos scripts que sí queremos que funcionen:

```
let scriptSrc = 'script-src'
scriptSrc += " 'self'"
scriptSrc += ' https://www.google-analytics.com/analytics.js'
scriptSrc += ' https://code.jquery.com/jquery-1.12.4.js'
const csp = [defaultSrc, formAction, frameAncestors, styleSrc, imgSrc, fontSrc, scriptSrc].join(';')
```

```
Content-Security-Policy: default-src 'none';form-action 'self';frame-ancestors 'none';style-src'self'

https://fonts.googleapis.com/;img-src 'self' data:;font-src https://fonts.gstatic.com/
```

Ejercicio

script-src

Permite restringir orígenes en la carga de scripts.

14. Buscar los inline scripts referidos a cat facts, y reemplazarlos por lo siguiente:

```
...
<h3>Cat fact: <span id="cat-fact"></h3>
<script src="/javascripts/cat-facts.js"></script>
...
```

Ejercicio

script-src

Permite restringir orígenes en la carga de scripts.

15. Mover el contenido de cat-facts hacia la carpeta de javascript.

```
$(document).ready(function () {
  $.ajax({
    url:
'https://cat-fact.herokuapp.com/facts/random',
    type: 'GET',
    crossDomain: true,
    success: function (response) {
      var catFact = response.text
      $('#cat-fact').text(catFact)
    },
    error: function (xhr, status) {
      alert('error')
    },
  })
  console.log(`Good script with jQuery succeeded`)
})
```

Ejercicio

connect-src

Permite restringir hacia dónde se puede conectar el sitio. De hecho, está previniendo ciertas conexiones…

16. Arreglar de la siguiente manera:

```
let connectSrc = 'connect-src'
connectSrc += '
https://cat-fact.herokuapp.com/facts/random'
const csp = [
  defaultSrc,
  formAction,
  frameAncestors,
  styleSrc,
  imgSrc,
  fontSrc,
  scriptSrc,
  connectSrc,
].join(';')
```

```
Content-Security-Policy: default-src 'none'; form-action 'self'; frame-ancestors 'none'; style-src
'self' https://fonts.googleapis.com/; img-src 'self' data:; font-src https://fonts.gstatic.com/;
script-src 'self' https://www.google-analytics.com/analytics.js
https://code.jquery.com/jquery-1.12.4.js; connect-src https://cat-fact.herokuapp.com/facts/random
```

Ejercicio

Evaluar CSP

17. Evaluar la CSP mediante Google CSP Evaluator

Ejercicio

script-src: hashes

Cuando utilizamos JS inline, podemos incluir el SHA256 correspondiente al script (o un nonce).

18. Ver el hash del error de la consola, correspondiente a google analytics.

19. Añadirlo a la CSP y recargar el sitio:

```
scriptSrc += "
'sha256-V2kaaafImTjn8RQTWZmF4IfGfQ7Qsqsw9GWaFjzFNPg='";
scriptSrc += " 'unsafe-inline'";
```

Notar que se incluye `unsafe-inline` por compatibilidad. Sin embargo, los browsers lo ignorarán en presencia del hash.

Ejercicio

script-src: hashes

Cuando utilizamos JS inline, podemos incluir el SHA256 correspondiente al script (o un nonce).

18. Ver el hash del error de la consola, correspondiente a google analytics.

19. Añadirlo a la CSP y recargar el sitio:

```
scriptSrc += "
'sha256-V2kaaafImTjn8RQTWZmF4IfGfQ7Qsqsw9GWaFjzFNPg='";
scriptSrc += " 'unsafe-inline'";
```

Notar que se incluye `unsafe-inline` por compatibilidad. Sin embargo, los browsers lo ignorarán en presencia del hash.

Importante: si se modifica el código del script, se debe recalcular el hash. Conviene automatizar este paso en el pipeline.

Ejercicio

script-src: nonce

Los nonces son valores aleatorios de un solo uso, que se generan para cada response.

20. Arreglar la policy con lo siguiente:

```
const nonce = uuid.v4()
scriptSrc += ` 'nonce-${nonce}'`
```

Se debe pasar el nonce a la view:

```
<script nonce="<%= nonce %>">
        $(document).ready(function () {
  $.ajax({
    url:
"https://cat-fact.herokuapp.com/facts/random",

...
```

Warning: No crear un middleware que reemplace los “script nonce=...”, porque los scripts inyectados obtendrán los nonces.

Ejercicio

script-src: 'unsafe-eval'

Si el código utiliza la función `eval`, hay que refactorizar de tal manera que no se permita. Si es una dependencia, buscar versiones más actualizadas. Si esto no es posible, agregar *unsafe-eval*, pero se perderá la protección contra DOM XSS.

```
scriptSrc += " 'unsafe-eval'"
```

tool

https://www.appsecmonkey.com/tools/csp