



# SEMINARIO 6

## SEGURIDAD HTTP Y CSP

El protocolo HTTP como una capa de seguridad más



PROYECTO "S-81 ISAAC PERAL" v4.2

¿LOCALIZASTE ALGUNA ERRATA O PROBLEMA? POR FAVOR, NOTIFICALA A JOSÉ MANUEL REDONDO  
LÓPEZ ([REDONDOJOSE@UNIOVI.ES](mailto:REDONDOJOSE@UNIOVI.ES)) ¡GRACIAS POR AYUDARNOS A MEJORAR LA ASIGNATURA!



Departamento de Informática  
Universidad de Oviedo



Universidad de Oviedo

Logo: @creative\_vanesa (Instagram)



# ACERCA DEL USO DE CONTENIDO GENERADO POR IA



- En esta presentación se usan algunas imágenes generadas por IA
  - Salvo error, cualquier imagen a la que no se le atribuya una fuente u origen expreso
- La IA generativa usada para ello es Microsoft Copilot
  - <https://copilot.microsoft.com/>
- Se ha restringido el uso de estas imágenes a la ilustración de los conceptos explicados en algunas de las páginas
  - Es decir, como refuerzo visual a lo explicado en algunas transparencias
  - El procedimiento ha sido describirle a la IA con toda la precisión posible los elementos que quería que apareciesen en la imagen (**prompt**)
    - Y la selección del mejor resultado obtenido, a juicio del autor de esta presentación
    - No se ha mencionado ni indicado que se copie el estilo a ningún autor, ni que se plagien obras concretas
- El autor declara expresamente su apoyo al trabajo de los artistas, ilustradores y creadores, de extrema importancia en la actualidad
  - El uso de estas técnicas se ha hecho solo con fines de mejora de las explicaciones, y cuando la alternativa era no contar con refuerzos visuales por restricciones de tiempo y presupuesto

# ÍNDICE



- 👎 [Cabeceras HTTP personalizadas](#)
- 📋 [Content Security Policy \(CSP\)](#)
  - [Strict CSP](#)
- ❓ [Otras cabeceras HTTP relativas a seguridad](#)
  - [Ejemplos](#)

# INTRODUCCIÓN



- El protocolo HTTP no es solo un medio de transporte de información entre clientes y servidores HTTP
- Tiene un buen número de características adicionales
  - Incluyendo características de seguridad
  - Estas pueden usarse mediante cabeceras HTTP
  - Pero el mal uso de las cabeceras puede ser un problema
- Este seminario muestra buenos y malos usos de dichas cabeceras
  - Para que os podáis beneficiar de este mecanismo en vuestras aplicaciones web
  - O en la configuración de vuestros servidores

Date: Mon, 07 Apr 2025  
09:42:06 GMT  
Expires: Thu, 01 Jan 1970  
00:00:00 GMT  
Keep-Alive: timeout=20  
Liferay-Portal:  
Community  
Edition Portal 7.3.7  
CE GA8  
(Athanasius / Build  
7307 / April 13,  
2021)

# 👎 LAS CABECERAS HTTP “CUSTOM” SON UN DOLOR

Dan demasiada información sin querer



Departamento de Informática  
Universidad de Oviedo



Universidad de Oviedo

# EL PROTOCOLO HTTP

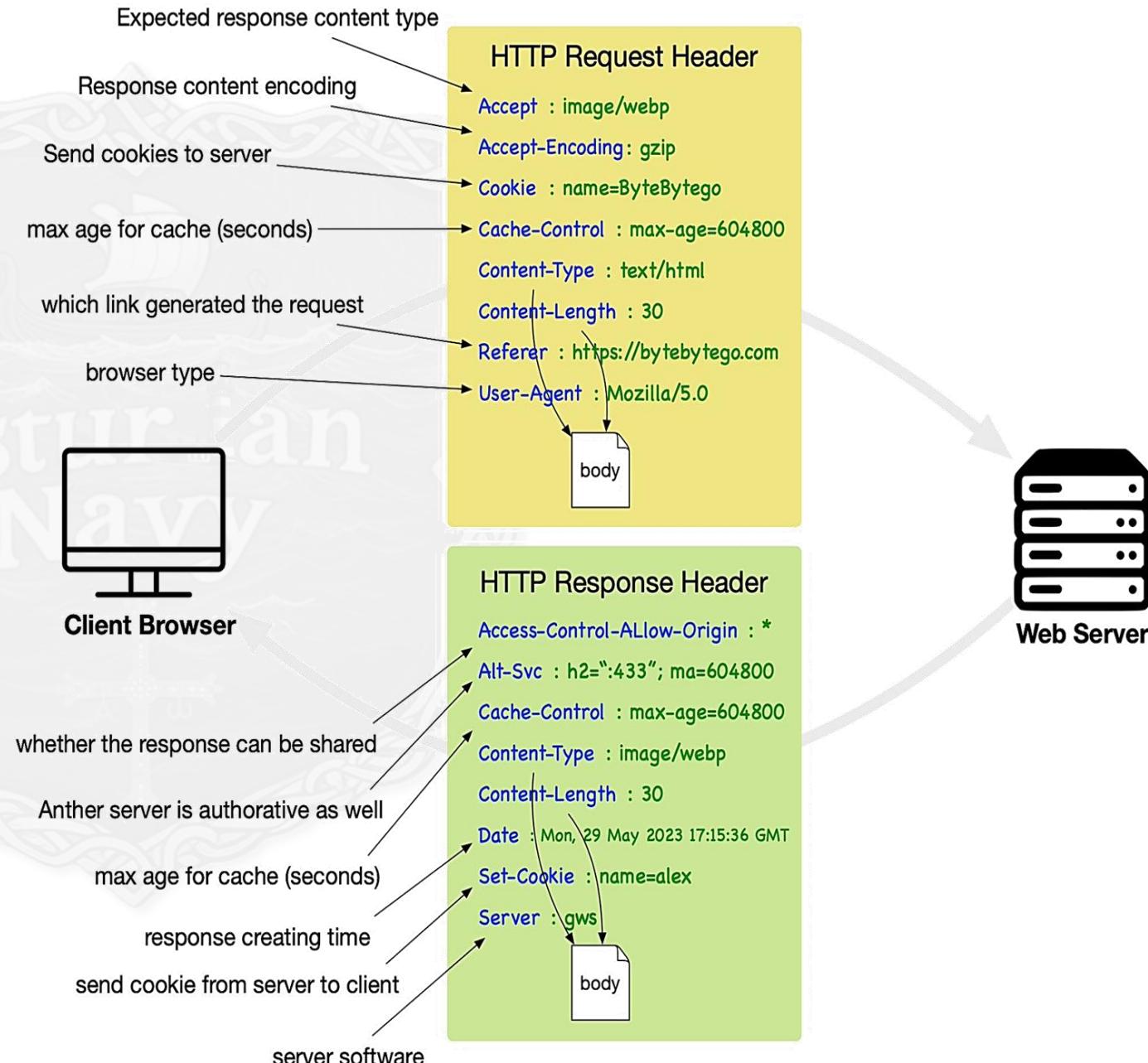


Seguridad de Sistemas  
Informáticos

- Este protocolo es el que hace funcionar Internet
- Se basa en enviar peticiones a un servidor web
  - Y esperar por su respuesta
- Tanto las peticiones como las respuestas llevan dentro pares clave-valor llamadas **cabeceras**
  - Son distintas para peticiones y para respuestas
  - Cada una tiene un significado diferente
  - **Algunas de ellas son de seguridad**, y son las que vamos a ver en este seminario

What's inside the HTTP Header?

 ByteByteGo.com



# RECORDANDO SEW: PETICIONES HTTP

- Debemos recordar algunas partes del protocolo HTTP explicadas en SEW
- Las HTTP requests tienen estos elementos

- Una o más **cabeceras** de la petición
- Una línea en blanco
- El cuerpo de un mensaje (opcional)
- La primera línea de una **HTTP request** tiene 3 elementos separados por espacios
  - **Un verbo** indicando el método HTTP usado
    - El más típico es **GET**, que lee un recurso del servidor web
  - La **URL pedida**
  - La **versión de HTTP** usada

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; ... ) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.147 Safari/537.36
Accept: text/html,application/xhtml+xml,...,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Fuente: <https://www.freecodecamp.org/news/http-and-everything-you-need-to-know-about-it/>

```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

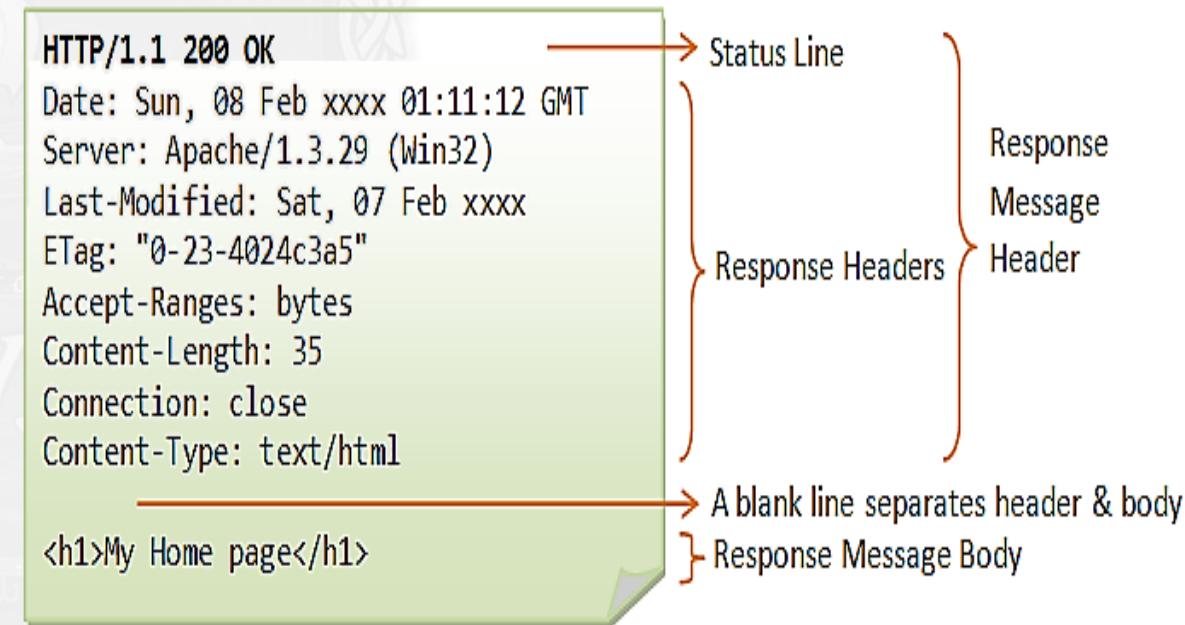
bookId=12345&author=Tan+Ah+Teck
```

Fuente: <https://aprendiendoarduino.wordpress.com/2017/09/11/protocolo-http-3/>

# RECORDANDO SEW: RESPUESTAS HTTP

## ● La respuesta a una HTTP request típica tiene tres elementos

- La **status line**, con
  - La versión HTTP
  - Un código de status numérico
  - El texto que describe el status de respuesta
- **Cabeceras:** Que se usan para mandar instrucciones al navegador, como
  - Cabeceras de seguridad HTTP
  - CSP (Content Security Policy)
- **El cuerpo de la respuesta**
  - Separado de la cabecera por una línea en blanco



Fuente:

[https://personales.unican.es/corcuerp/ingweb/notes/HTTP\\_Basics.html](https://personales.unican.es/corcuerp/ingweb/notes/HTTP_Basics.html)

# RECORDANDO SEW: COOKIES

## ● Parte vital de HTTP

- Proporcionan **estado** al protocolo
- Permiten al servidor enviarle datos al cliente
- El cliente guarda estos datos, y los envía de vuelta cuando hace su siguiente petición
- El cliente manda su estado actual en cada petición

## ● Habitualmente guardan un ID de sesión único

- El servidor lo usa para asociarle un estado propio de ese cliente de entre todos los que guarda

## ● El servidor envía una cookie al cliente usando la cabecera de respuesta **SetCookie**

- Cuando el usuario hace más peticiones al servidor, esta **cookie** se añade a las cabeceras HTTP
- La cookie puede contener también **más pares clave-valor** con información adicional

The screenshot shows the Burp Suite interface in Intercept mode. A GET request is being viewed. In the Headers section, the 'Cookie' header is highlighted with a red box. It contains several key-value pairs: 'showhints=0; username=user; uid=18; remember\_token=PN; acopendivids=swingset,mutilidae,jotto,phpbb2,redmine; acgroup'. This indicates that the server has issued a session cookie to the client.

```
Request to http://172.16.67.136:80
GET /mutillidae/index.php?popUpNotificationCode=AU1 HTTP/1.1
Host: 172.16.67.136
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 5_1 like Mac OS X; en-us; rv:5.0) AppleWebKit/534.46 (KHTML, like Gecko) Mobile/9B179 Safari/8536.25
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: showhints=0; username=user; uid=18; remember_token=PN; acopendivids=swingset,mutilidae,jotto,phpbb2,redmine; acgroup
Connection: keep-alive
Cache-Control: max-age=0
```

Fuente: <https://portswigger.net/support/using-burp-to-hack-cookies-and-manipulate-sessions>

The screenshot shows the Burp Suite interface in Intercept mode. A GET request is being viewed. In the Headers section, the 'Cookie' header is highlighted with a red box. It contains several key-value pairs: 'PHPSESSID=gdc668pjmpah42hegt7sa7igc3; user=TUFSWStKQUSF'. This indicates that the client has included the previously received session cookie in its subsequent request to the server.

```
Request to http://ctf.infosecinstitute.com:80 [54.148.198.184]
GET /ctf2/exercises/ex9.php HTTP/1.1
Host: ctf.infosecinstitute.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.6.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=gdc668pjmpah42hegt7sa7igc3; user=TUFSWStKQUSF
Connection: keep-alive
```

Fuente: <https://uaf.io/web/2015/07/16/infosec-ctf2-level-9.html>

# CABECERAS HTTP QUE SON PROBLEMAS DE SEGURIDAD

- Hay cabeceras de respuesta HTTP no estándar que provocan problemas de seguridad

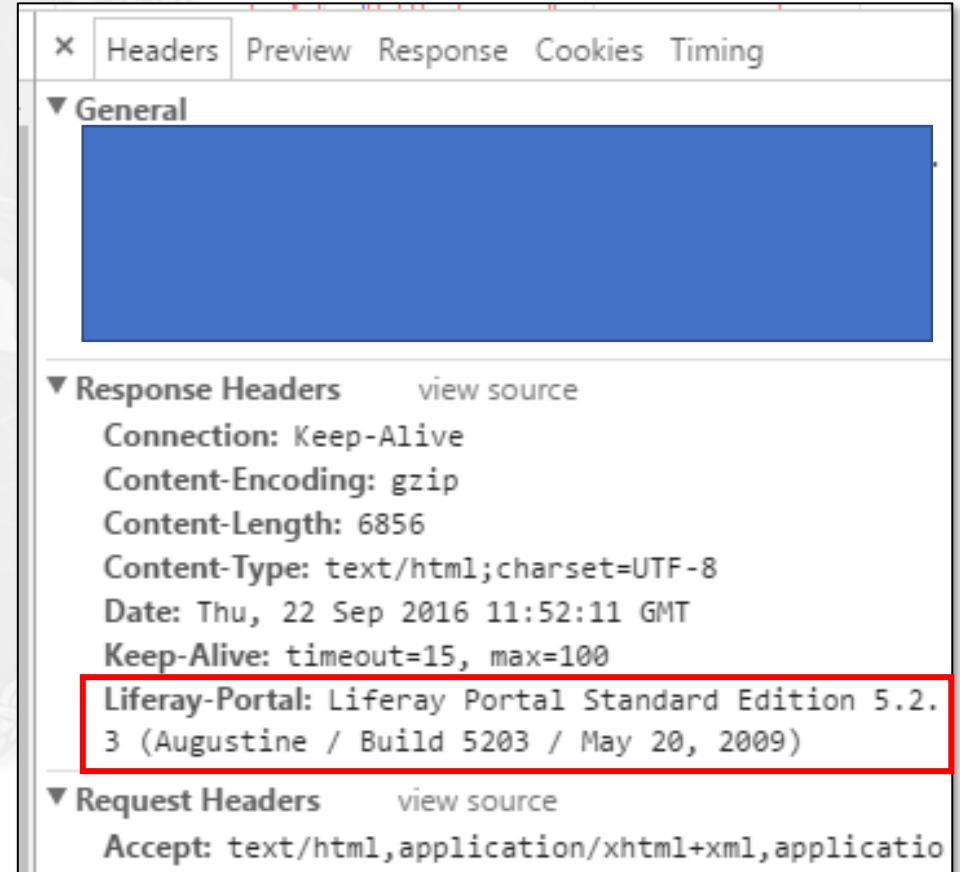
- Las introducidas por usar un producto / framework

- Delatan o directamente declaran qué productos se usan, revelando su versión o más detalles del servidor
    - Con esta información se puede ir a un repositorio de vulnerabilidades... (*¿recuerdas los CVE del Tema 1?*)
    - ...y averiguar todas las que tiene ese producto...

- Las que declaran el tipo de servidor usado: ¡Que ha puesto ahí el propio servidor web!
  - Con el mismo problema anterior

- En este ejemplo se delata el uso de Liferay, incluida su versión y...uf ☹

- [https://www.cvedetails.com/vulnerability-list/vendor\\_id-2114/product\\_id-12592/version\\_id-109268/Liferay-Portal-5.2.3.html](https://www.cvedetails.com/vulnerability-list/vendor_id-2114/product_id-12592/version_id-109268/Liferay-Portal-5.2.3.html)



The screenshot shows a browser's developer tools Network tab with the "Headers" tab selected. The "General" section shows a large blue rectangular area. Below it, the "Response Headers" section is expanded, showing the following headers:

- Connection: Keep-Alive
- Content-Encoding: gzip
- Content-Length: 6856
- Content-Type: text/html; charset=UTF-8
- Date: Thu, 22 Sep 2016 11:52:11 GMT
- Keep-Alive: timeout=15, max=100
- Liferay-Portal: Liferay Portal Standard Edition 5.2.3 (Augustine / Build 5203 / May 20, 2009)

The "Liferay-Portal" header is highlighted with a red box.

The "Request Headers" section is also visible below, showing:

- Accept: text/html,application/xhtml+xml,application/xml,application/json

# CABECERAS HTTP QUE SON PROBLEMAS DE SEGURIDAD

- Aquí se delata el uso de Nginx 1.10.0 y Ubuntu

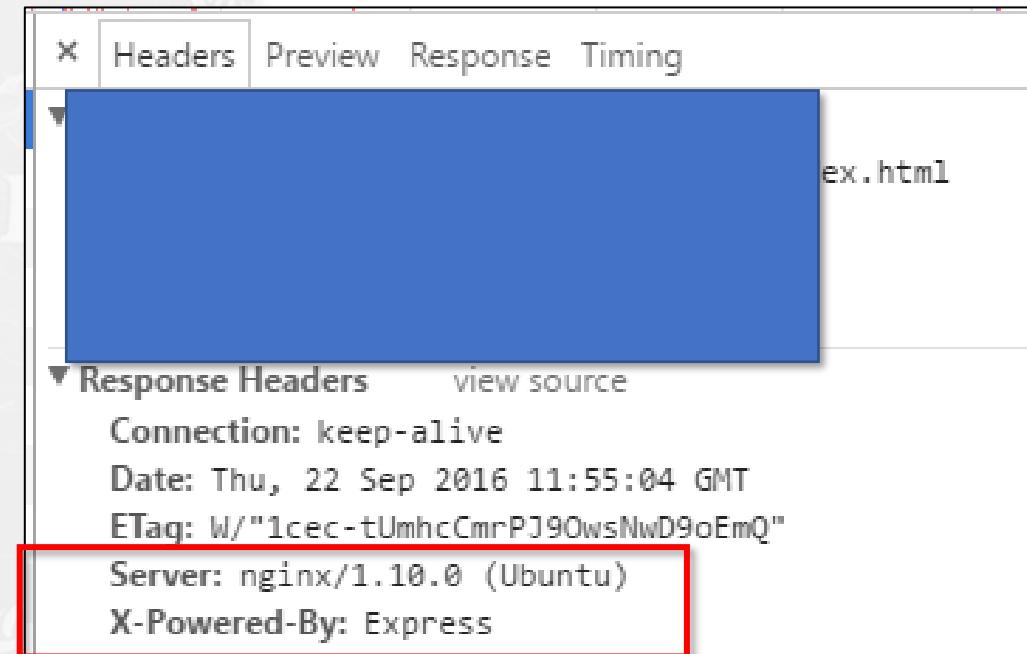
- [https://www.cvedetails.com/vulnerability-list/vendor\\_id-10048/product\\_id17956/version\\_id-198730/Nginx-Nginx-1.10.0.html](https://www.cvedetails.com/vulnerability-list/vendor_id-10048/product_id17956/version_id-198730/Nginx-Nginx-1.10.0.html)
- También el uso del framework Express

- Son cabeceras “publicitarias”, no necesarias para que las webs funcionen

- Cada servidor tiene su forma de eliminarlas

- Apache: [https://httpd.apache.org/docs/current/mod/mod\\_headers.html](https://httpd.apache.org/docs/current/mod/mod_headers.html)
- IIS (con la opción URL Rewrite):  
<https://blogs.msdn.microsoft.com/varunm/2013/04/23/remove-unwanted-httpresponse-headers/>
- Nginx (instalando el paquete nginx-extras):  
<https://stackoverflow.com/questions/246227/how-do-you-change-the-serverheader-returned-by-nginx>

- Los frameworks de desarrollo suelen tener una opción para quitarlas





# **CONTENT SECURITY POLICY (CSP)**

La cabecera más potente de HTTP relativa a seguridad



Departamento de Informática  
Universidad de Oviedo



Universidad de Oviedo



# ¿QUÉ ES CSP?

- El **cross-site scripting (XSS)** es uno de los problemas de seguridad más graves de la web y por ello han surgido un gran número de contramedidas (**tema 7 de teoría**)
  - Contextual auto-escaping: <https://queue.acm.org/detail.cfm?id=2663760>
  - Escaneadores automáticos de vulnerabilidades para identificar vulnerabilidades testeando la aplicación (**Seminario 3**): [https://www.owasp.org/index.php/Category:Vulnerability\\_Scanning\\_Tools](https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools)
- **Content Security Policy (CSP)** es una capa de defensa adicional contra XSS, Clickjacking y otros ataques de inyección de código
  - Permite especificar **dominios autorizados** a ejecutar scripts o cargar recursos en una página
    - Si alguien inyecta un script malicioso que se lea desde un dominio distinto a uno autorizado...
    - **El navegador no lo ejecuta**: la vulnerabilidad existe en la página, pero no sus efectos (los para el browser)
  - Una respuesta HTTP le dice al navegador **desde dónde** puede cargar ciertos recursos
    - Necesitamos emitir la cabecera **Content-Security-Policy: policy** en las respuestas HTTP
    - **policy** es un texto que contiene una lista de directivas que definen la política CSP separadas por ;
    - Cada directiva puede tener valores que describen la política a seguir para un determinado tipo de recurso
    - Veamos qué directivas hay y sus posibles valores

# CSP: CONTENT SECURITY POLICY

@ Sec\_80



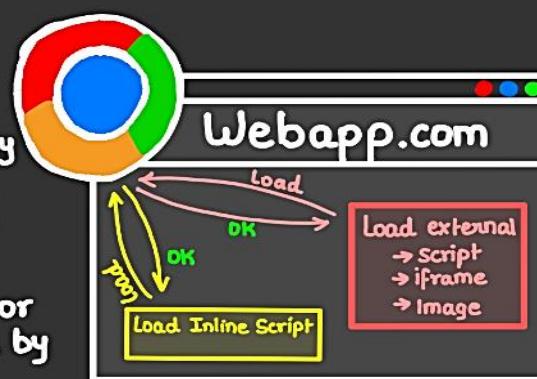
## 1 WHERE IS THE PROBLEM ?



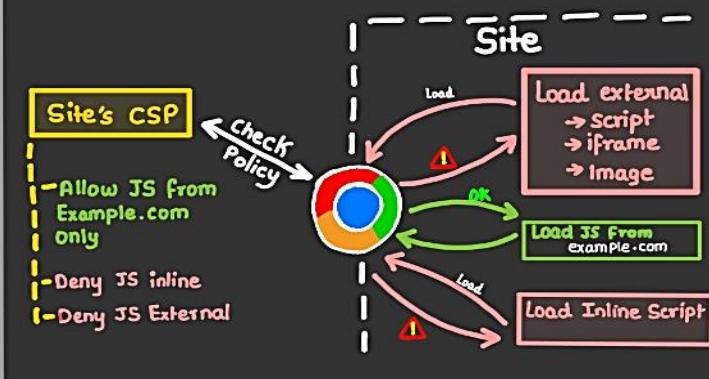
\*If XSS is exploited and CSP is not enforced, then can execute malicious Scripts.

## 2 ISSUE

Browser doesn't stop any resource load request from website, be it from attacker or genuinely made by site.



## 3 How CSP Helps ?



## 4 CSP HEADER

Response header set by Server



Policy Contains one or more directives and Sources  
';' separated

Each directive has set of allowed resource Sources

## 5 DIRECTIVES

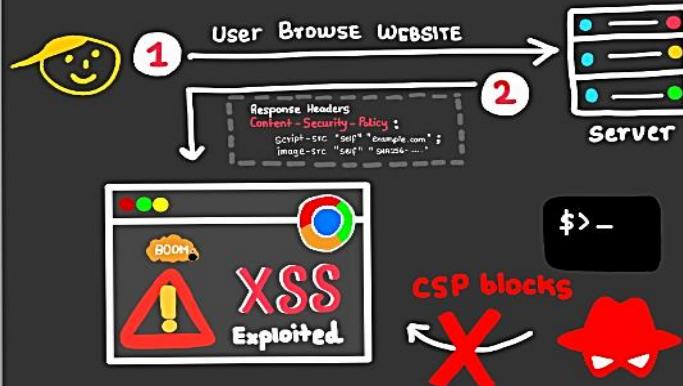
Tells browser what Content Sources can be trusted .Eg

default-src : policy for all resources  
Style-src : Policy for style tag  
Script-src : Policy for script tag  
img-src : Policy for image tag

Each directive takes a source list , Sources separated by Space  
Valid Sources can be

"Self" — load from current origin only  
" \*" — load from anywhere  
"none" — Prevent loading from anywhere  
"SHA256- ...." — Load the source file if its hash matches the mentioned hash.

## 6 COMPLETE PICTURE



# DIRECTIVAS Y POSIBLES VALORES



Directiva	Ejemplo	Descripción
default-src	'self' cdn.redondo.com	Política por defecto para cargar JavaScript, imágenes, CSS, fuentes, peticiones AJAX, Frames o contenido HTML5 si falta una específica
script-src	<b>'self' js.redondo.com</b>	<b>Especifica orígenes válidos de JavaScript</b>
style-src	'self' css.redondo.com	Especifica orígenes válidos de CSS
img-src	'self' img.redondo.com	Especifica orígenes válidos de imágenes
connect-src	'self'	Se aplica a XMLHttpRequest (AJAX), WebSocket o EventSource. En caso de no permitirse, el navegador emula un HTTP 400 Bad Request
font-src	fonts.redondo.com	Especifica orígenes válidos de fuentes
object-src	'self'	Especifica orígenes válidos de plugins (como los tags <object>, <embed> o <applet>)
media-src	multimedia.redondo.com	Especifica orígenes válidos de audio y video (como los tags de HTML5 <audio> y <video>)
frame-src	'self'	Especifica orígenes válidos para cargar frames. Está <b>obsoleta</b> y debe reemplazarse por child-src
sandbox	allow-forms allow-scripts allow-popups	Implementa una sandbox para el recurso asociado que aplica una same origin policy única, elimina popups y bloquea la ejecución de plugins y scripts. Si se deja vacía, aplica todas las restricciones ( <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/sandbox">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/sandbox</a> ), pero permite desactivarlas individualmente con los valores: allow-forms allow-same-origin allow-scripts allow-popups, allow-modals, allow-orientation-lock, allow-pointer-lock, allow-presentation, allow-popups-to-escape-sandbox, and allow-top-navigation
report-uri	/reports	Le dice al browser que mande informes de violaciones de la política vía POST a esa URL
child-src	'self'	Especifica orígenes válidos de web workers y nested browsing contexts cargados con tags como <frame> o <iframe>
form-action	'self'	Sitios válidos que pueden usarse como action de una <form>
frame-ancestors	'none'	Especifica orígenes válidos de recursos usados desde tags <frame> <iframe> <object> <embed> <applet>. El valor none es aproximadamente equivalente a X-Frame-Options: DENY
plugin-types	application/pdf	Define tipos MIME válidos para los plugins cargados vía tags <object> o <embed>

Possibles valores de las directivas (se pueden combinar varios separados por un espacio)

Valor	Ejemplo	Descripción
*	img-src *	Cualquier URL que no use los esquemas data: blob: o filesystem:
'none'	object-src 'none'	Bloquea el uso de ese tipo de recurso
'self'	script-src 'self'	Permite cargar desde el mismo origen (mismo esquema, host y puerto).
data:	img-src 'self' data:	Permite la carga de URLs con el esquema data: (ej. Imágenes codificadas en Base64)
domain.redondo.com	img-src domain.redondo.com	Carga el recurso desde el dominio indicado
*.redondo.com	img-src *.redondo.com	Carga el recurso desde cualquier subdominio de redondo.com
https://cdn.com	img-src https://cdn.com	Solo carga el recurso desde el dominio indicado, y solo bajo HTTPS
https:	img-src https:	Solo permite la carga de recursos bajo HTTPs, aunque desde cualquier dominio
'unsafe-inline'	script-src 'unsafe-inline'	Permite usar orígenes de recursos en línea, como atributos style, onclick o tags <script> (depende del contexto del origen). También URIs con esquema javascript:
'unsafe-eval'	script-src 'unsafe-eval'	Permite la evaluación dinámica de código, con funciones como eval() de JavaScript
'nonce-'	script-src 'nonce-azaj345'	Solo ejecuta un script si tiene un atributo nonce cuyo valor es idéntico al de la cabecera. Ej.: <script nonce="azaj345"> alert(...);</script>
'sha256-'	script-src 'sha256-qznLcsROx4GACP2dm0UCKCzCG+Hiz1guq6ZZDob/Tng='	Ejecuta un style o un script si su hash es la especificada con el algoritmo indicado. No funciona en URIs javascript:. El hash del ejemplo ejecuta alert('Hello, world.');



# ¿CÓMO FUNCIONA CSP? EJEMPLOS

## ● Los siguientes ejemplos sirven para entenderla mejor

- Admitir solo contenidos del propio dominio (excluye subdominios)
  - **Content-Security-Policy: default-src 'self'**
- Ídem al anterior, pero añadiendo el dominio de confianza **recursos.es** y sus subdominios
  - **Content-Security-Policy: default-src 'self' \*.recursos.es**
- Amplía el anterior permitiendo imágenes desde cualquier origen
  - **Content-Security-Policy: default-src 'self' \*.recursos.es; img-src \***
  - No permite cargar JavaScript u otro tipo de contenidos de sitios ajenos a los permitidos: al no especificar **script-src** o similares se usa el valor de **default-src** para el resto de directivas que controlan recursos de diferente tipo
- Amplía el primero admitiendo imágenes desde cualquier dominio, restringiendo audio y video solo a una serie de proveedores autorizados (sin incluir sus subdominios) y scripts a un dominio concreto
  - **Content-Security-Policy: default-src 'self'; img-src \*; media-src imgur.com youtube.com; script-src scripts.recursos.com**

## ● ¿Qué tenemos que hacer pues? Colocar cada tipo de contenido en el sitio correcto y emitir las cabeceras con valores adecuados a esas localizaciones

- Y denegar ('none') todo aquel tipo de contenido que no vayamos a usar ¡Política del mínimo privilegio!

# ¿CÓMO FUNCIONA CSP?



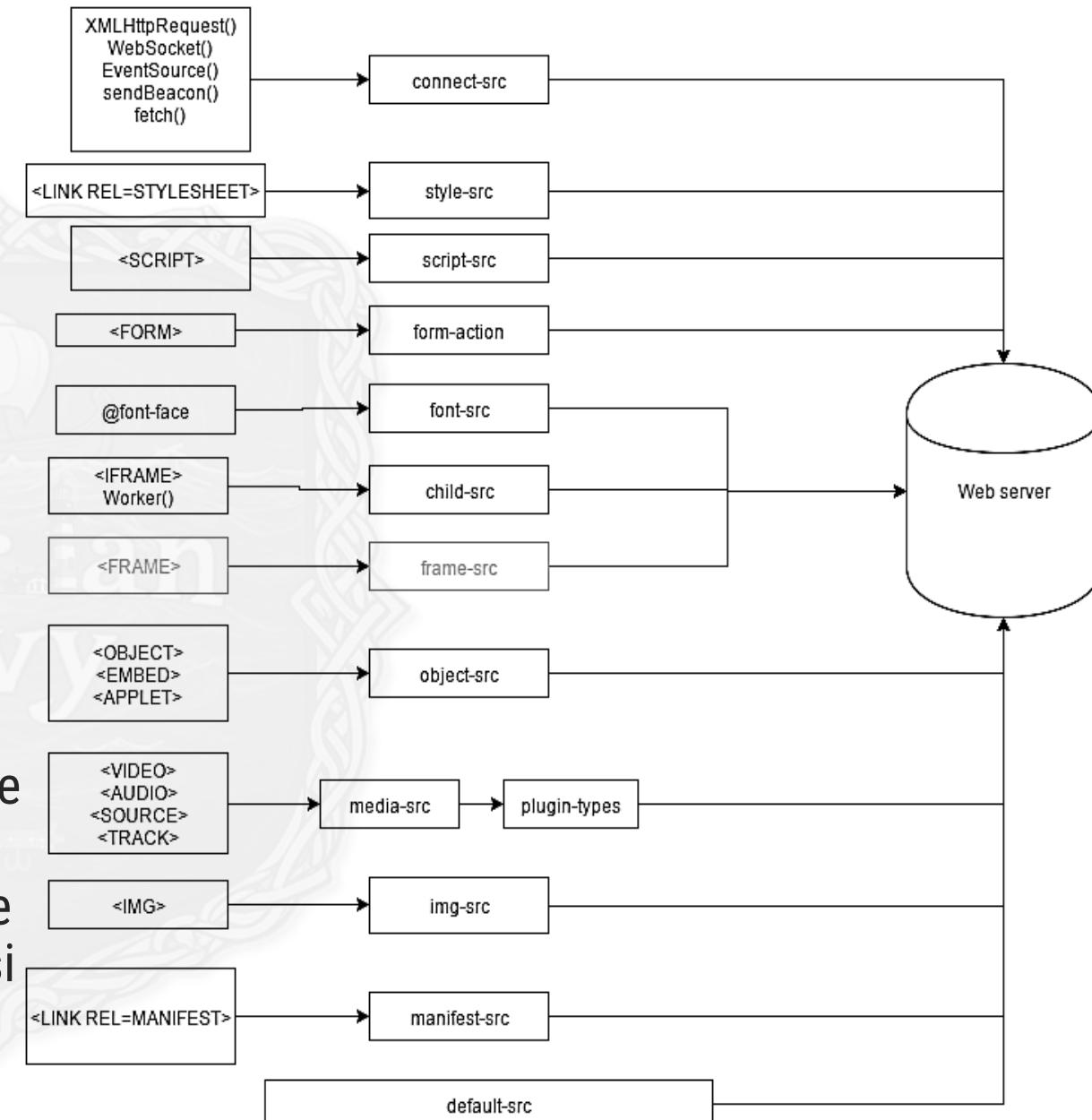
- Permiten el control del cliente (browser) desde la aplicación

- Puede especificar cualquier conjunto de directivas separadas por ;

- La más importante es **default-src**, que se aplica a todos los recursos que no tengan definida una directiva propia

- Se puede limitar la ejecución de scripts con **script-src**, o con **default-src** si no se define
- Igualmente, la ejecución de estilos desde elementos **<style>** o atributos **style** se puede controlar con **style-src**, o con **default-src** si no se define

- La imagen describe el mapeo entre elementos HTML 5 / JavaScript y CSP



# ¿CÓMO FUNCIONA CSP? GESTIÓN DE PROBLEMAS

- Es posible cometer errores especificando una política CSP
- Puedes usar la cabecera **Content-Security-Policy-Report-Only** para depurarla
  - La política no tiene efecto, pero si algún contenido viola sus restricciones se envía un informe de lo que ha pasado a la URI indicada con la directiva `report-uri` de la tabla anterior
  - Ejemplo
    - `Content-Security-Policy-Report-Only: default-src 'self'; report-uri http://reportador.miempresa.com/programa_que_recibe_el_reporte.py`
- Una vez verificada, cámbiala por la **Content-Security-Policy** estándar
  - Ambas tienen opciones equivalentes
- Si están las dos presentes, se procesan ambas
  - Si no se respeta la política, una bloquea los contenidos y la otra manda información a la URI dada
  - Puedes usar también `report-uri` sobre una cabecera **Content-Security-Policy** si queremos una política de bloqueo activa que además nos dé informes de intentos de saltarse su contenido
  - De esta forma podemos **identificar potenciales problemas de seguridad / ataques**



# ¿CÓMO HABILITAR CSP?

## ● Configurando el servidor web (Independiente del lenguaje y framework)

- En Apache, añadir a `apache2.conf` o cualquier virtual host
  - `Header set Content-Security-Policy "default-src 'self';"`
- En Nginx, se añade esto a un bloque `server`
  - `add_header Content-Security-Policy "default-src 'self';";`
- En IIS, usando el módulo "Reply Headers" o modificando el fichero `web.config`

```
<customHeaders>
    <add name="Content-Security-Policy" value="default-src 'self';" />
</customHeaders>
```

## ● Emitiéndolas nosotros en nuestra aplicación según el framework/lenguaje usado

- JEE: [https://www.owasp.org/index.php/Content\\_Security\\_Policy](https://www.owasp.org/index.php/Content_Security_Policy)
- PHP / Laravel: <https://github.com/spatie/laravel-csp>
- Express: <https://www.npmjs.com/package/express-csp-header>
- Node.js: <https://stackoverflow.com/questions/21048252/nodejs-where-exactly-can-i-put-the-content-security-policy>
- Otros: busca formas de modificar la respuesta HTTP a bajo nivel o módulos de gestión CSP

# CON CSP VS SIN CSP

**INQUIRY FORM**




Share your thoughts

**La página es vulnerable a XSS**

**Freeeze**

HOME ABOUT BL

CONTACT

Lorem Ipsum

Mauris quis

Phasellus laoreet

**xss**

OK

```
GNU nano 4.3          /etc/apache2/apache2.conf
# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

Header Set Content-Security-Policy "default-src 'www.frozendreams.es/js';"
```

Dar valor a **default-src** evita el ataque con éxito, pero también rompe la página al no cargarse los CSS, JS, etc. desde sus orígenes, ya que no tienen su propia directiva CSP

**Contact**

**UNIT 0123 , ABC BUILDING, BUSSINESS PARK**

If you're having problems editing this website template, then don't hesitate to ask for help on the Forums.

**INQUIRY FORM**




Share your thoughts

Send

```
GNU nano 4.3          /etc/apache2/apache2.conf
# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

Header Set Content-Security-Policy "script-src 'www.frozendreams.es/js';"
```

**INQUIRY FORM**




Share your thoughts

OK

" onblur="this.value=!this.value?'Name':this.value;" onfocus="this.select()"  
onclick="this.value='';" >

**Dar valor a **script-src** evita el ataque con éxito sin interferir en la carga de otros elementos**



## Strict CSP

Un uso de CSP que previene formas de saltárselo conocidas



# ¿Es un MÉTODO DEFINITIVO?

- **Muchas opciones de CSP autorizan lugares a cargar recursos (allowlists)**
  - Pero esto no puede parar todos los ataques, y **se conocen formas de saltarse estas restricciones**
  - <https://ai.google/research/pubs/pub45542>
  - <https://csp.withgoogle.com/docs/faq.html#problems-with-whitelists>
- **Todos los problemas se basan en suponer que un dominio autorizado solo va a servir contenido 100% fiable, pero esto no es realmente posible**
- **¿Entonces CSP...no funciona? ¡Sí! Pero hay que usarlo con mucho cuidado...**
  - Necesitamos un mecanismo más potente porque las posibles vías para saltárselo son comunes
  - Podríamos tener una...incluso sin saberlo ☹

# ENTONCES, ¿QUÉ PODEMOS HACER?

- La única forma realmente segura de aplicar CSP con scripts es usar valores tipo **nonce** para la directiva

- Cuyo valor sean tokens aleatorios alfanuméricicos de un solo uso
- Esto se llama Strict CSP
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/script-src>

- De esta forma

- La aplicación calcula e inyecta, en los scripts conocidos de su código fuente, un valor **nonce** en cada petición que se haga a la página
  - Es decir, los **nonces no se pueden predecir**
  - Debemos usar **las funcionalidades de nuestro framework** para calcularlos
  - Las aplicaciones ya desarrolladas deben modificarse para implementarlo
- Como vimos, sólo los scripts que tengan ese **nonce** injectado serán ejecutados por el navegador

**Content-Security-Policy: script-src 'nonce-random666'**

...

```
<script nonce='random666'>alert('This runs, the nonce is correct')</script>
<script>alert('This does not run, no nonce is attached')</script>
<script nonce='iamahacker123'>alert('This does not run, incorrect nonce')</script>
```



# IMPLEMENTANDO UNA POLÍTICA CSP ESTRICTA

- Por tanto, la forma más segura de implementar CSP en aplicaciones modernas y complejas es usar una política CSP estricta (**Strict CSP**)
- Ello nos obliga a cambiar nuestras aplicaciones de la siguiente forma
  - Añadir un atributo **nonce** a todos los elementos `<script>`
    - Algunas tecnologías de desarrollo basadas en plantillas permiten hacerlo automáticamente
  - Refactorizar cualquier código que tenga manejadores de eventos en línea (`onclick`,...) y URLs con el esquema **javascript**:
  - Para cada carga de cada página, generar un nonce nuevo
    - Pasarlo al sistema que se encargue de injectarlo en los scripts del código de la página (plantillas, etc.)
    - Finalmente, pasar ese valor en la cabecera **Content-security-policy** de la respuesta (ver tabla)
- Esta guía da detalles y ejemplos de posibles cambios en la implementación en aplicaciones existentes
  - <https://csp.withgoogle.com/docs/adopting-csp.html>
- ¡Reduciremos mucho la posibilidad de ataques XSS en nuestras aplicaciones!

# DIRECTIVA SCRIPT-SRC

- Dada la importancia de los **scripts** y sus vulnerabilidades asociadas, la directiva **script-src** tiene una serie de valores a estudiar para lograr el máximo beneficio
  - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/script-src>
- Uno es '**strict-dynamicscript** se puede ejecutar porque lleva un **nonce** o una **hash** correcta, entonces todos los **scripts** que cargue también se permiten
  - Anula cualquier lista blanca, **self** o **unsafe-inline** que pueda haber asociado a la directiva
    - Sí se ejecutarán si el browser no soporta **strict-dynamic**: compatibilidad con browsers antiguos
  - Ejemplo: **script-src 'nonce-random123' 'strict-dynamic'; object-src 'none'**
- Con esto logramos protección más completa contra XSS
  - Si alguien consigue un ataque XSS persistente, no sabrá el **nonce** que se va a usar en cada petición
    - Se genera uno nuevo por petición **sólo en aquellas <script>** conocidas estáticamente (código de la web)
    - Cualquier otro **<script>** fuera de ellas no se ejecutaría
- No sirve en ataques que inyecten un **script** dentro de otro **script** existente
  - Debemos evitar usar datos de entradas del usuario para generar partes de código JavaScript

# EJEMPLO CSP REAL “PRODUCTION-READY”

## • Veamos una política real

- **object-src 'none'**: No ejecuta plugins
- **script-src 'nonce-random1234' 'unsafe-inline'**: Protección basada en nonces
  - Anula **unsafe-inline**, salvo si el browser no soporta nonces
  - **unsafe-inline** y **unsafe-eval** dan compatibilidad con webs existentes
    - Bórralas si no se usan esas funcionalidades
- **script-src 'strict-dynamic' https: http:** : permite ejecutar scripts añadidos dinámicamente si los ejecuta un script autorizado
  - Si el browser soporta esto, se ignoran **https** e **http**
  - Si no lo soporta, permite cargar scripts de cualquier parte
- **base-uri 'none'**: Deshabilita **<base>**, no se puede cambiar el origen de scripts cargados de URLs relativas
  - **base-uri 'self'** suele ser también seguro
- **report-uri**: Informa de violaciones de la política

## Content-Security-Policy:

```
object-src 'none';  
  
script-src 'nonce-random1234' 'unsafe-  
inline' 'unsafe-eval' 'strict-dynamic'  
https: http:;  
  
base-uri 'none';  
  
report-uri https://reports.redondo.com/
```

## • Por tanto

- **Browser moderno**: Solo ejecuta scripts con el nonce correcto y otros scripts que éstos carguen
- **Browser antiguo**: No implementa ninguna protección XSS, pero permite que las aplicaciones funcionen correctamente

**Esta política hace sacrificios de seguridad a cambio de compatibilidad con aplicaciones existentes / browsers antiguos (caso más realista)**

## (1) Google CSP Evaluator

# EVALUANDO POLÍTICAS CSP



Seguridad de Sistemas  
Informáticos

- Para comprobar una política podemos usar Google CSP Evaluator

- <https://csp-evaluator.withgoogle.com/>
- Nos evalúa su “calidad”
- Nos notifica posibles fallos
- Nos detecta errores de sintaxis
- Nos da recomendaciones

### Content Security Policy

Content-Security-Policy: default-src 'self' \*.recursos.es; img-src \*

CSP Version 2

[CHECK CSP](#)

Evaluated CSP as seen by a browser supporting CSP Version 2

[expand/collapse all](#)

- default-src
  - 'self'
  - \*.recursos.es
- img-src
- object-src [missing]

'self' can be problematic if you host JSONP, Angular or user uploaded files.  
No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries.

Can you restrict object-src to 'none'?

### Content Security Policy

[Sample unsafe policy](#) [Sample safe policy](#)

```
Content-Security-Policy:  
object-src 'none';  
script-src 'nonce-random1234' 'unsafe-inline' 'unsafe-eval' 'strict-dynamic' https: http:;  
base-uri 'none';  
report-uri https://reports.redondo.com/
```

CSP Version 3 (nonce based + backward compatibility checks) [?](#)

[CHECK CSP](#)

Evaluated CSP as seen by a browser supporting CSP Version 3

[expand/collapse all](#)

<ul style="list-style-type: none"><li>✓ object-src</li><li>⚠ script-src<ul style="list-style-type: none"><li>✓ 'nonce-random1234'<ul style="list-style-type: none"><li>- 'unsafe-inline'</li></ul></li><li>⚠ 'unsafe-eval'</li><li>✓ 'strict-dynamic'<ul style="list-style-type: none"><li>- https:</li><li>- http:</li></ul></li></ul></li><li>✓ base-uri</li><li>✓ report-uri</li></ul>	<p>unsafe-inline is ignored if a nonce or a hash is present. (CSP2 and above) Because of strict-dynamic this entry is ignored in CSP3 and above 'unsafe-eval' allows the execution of code injected into DOM APIs such as eval().</p> <p>Because of strict-dynamic this entry is ignored in CSP3 and above Because of strict-dynamic this entry is ignored in CSP3 and above</p>
---	--

### Content Security Policy

[Sample unsafe policy](#) [Sample safe policy](#)

```
Content-Security-Policy:  
object-src 'none';  
script-src 'nonce-random1234' 'unsafe-inline' 'strict-dynamic' https: http:;  
base-uri 'none';  
report-uri https://reports.redondo.com/
```

CSP Version 3 (nonce based + backward compatibility checks) [?](#)

[CHECK CSP](#)

Evaluated CSP as seen by a browser supporting CSP Version 3

[expand/collapse all](#)

<ul style="list-style-type: none"><li>✓ object-src</li><li>✓ script-src</li><li>✓ base-uri</li><li>✓ report-uri</li></ul>	<p><a href="#">expand/collapse all</a></p>
---	--



# ? OTRAS CABECERAS HTTP

Más capas de seguridad



Departamento de Informática  
Universidad de Oviedo



Universidad de Oviedo



# CABECERA CSP FRAME-ANCESTORS

[https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking\\_Defense\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html)

## ● El clickjacking es un ataque que engaña al usuario para que haga clic en un elemento de la web que desencadena una acción sobre otro elemento invisible

- Posible con tags como <frame> e <iframe> que permiten superponer contenidos en otros
  - Por eso también se conoce como UI Redressing
- Un usuario pulsa en “Obtener un cupón descuento”, pero realmente lo hace en “Eliminar mi cuenta”
  - Si el usuario está autenticado en la web objetivo, eliminará su cuenta si no hay más medidas de seguridad (confirmación Sí/No, un segundo paso...)

## ● Reemplaza a la cabecera obsoleta X-Frame-Options

- Content-Security-Policy: frame-ancestors 'none': Evita que cualquier dominio pueda hacer frame
  - Opción recomendada
- Content-Security-Policy: frame-ancestors 'self': Solo permite al sitio actual hacer el frame
- Content-Security-Policy: frame-ancestors 'self' \*.trustedsite.com https://allowed.site.com
  - Permite al sitio actual
  - Cualquiera de trustedsite.com (usando cualquier protocolo)
  - Y solo la página allowed.site.com (usando HTTPS en el puerto por defecto, 443)



# X-CONTENT-TYPE-OPTIONS

<https://developer.mozilla.org/es/docs/Web/HTTP/Headers/X-Content-Type-Options>

- Esta cabecera evita los problemas causados por una acción que llevan a cabo algunos navegadores llamada **MIME type sniffing**

- Si el tipo de un recurso declarado en el servidor es desconocido o ambiguo, algunos browsers lo leen para obtener o verificar su tipo MIME
- Esos navegadores tienen una serie de reglas para identificar el tipo MIME que cree que tiene un recurso
  - Y lo tratan como tal, ¡aunque su tipo declarado sea otro!

- El problema es que esto puede abrir la puerta a determinados ataques

- Por ejemplo, los GIFARS y otras variantes: <https://en.wikipedia.org/wiki/Gifar>
- Aumenta el peligro de hacer drive-by downloads (descargas involuntarias)

- Es necesario configurarla de la siguiente forma: **x-content-type: nosniff**

- En Nginx: `add_header x-content-type-options "nosniff" always;`
- En Apache: `header always set x-content-type-options "nosniff"`

# REFERRER-POLICY

<https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Referrer-Policy>



Seguridad de Sistemas  
Informáticos

## ● La cabecera HTTP **Referer** tiene la URL de la web anterior de la que provenía el enlace que cargó la que se está mostrando

- Permite a los servidores saber desde dónde los visitan
- Se pueden emplear estos datos para realizar análisis, registros, optimizaciones, envío de publicidad...

## ● Tenemos derecho a decidir si nuestros servidores dan esa información

- La cabecera **Referrer-Policy** determina los datos incluidos en la cabecera **Referer** de cada petición
- Un **Referer** completo está formado por
  - Host ([www.redondo.com](http://www.redondo.com))
  - Path (</clientes/ver.html>)
  - Querystring ([id=jose](#))

## ● Ejemplo: **Referrer-policy: no-referrer**

Valor	Se enviará...
no-referrer	Nada, no se envía ningún dato
no-referrer-when-downgrade (predeterminado)	Un Referer completo (host + path + querystring) cuando el nivel de seguridad HTTP no cambia (HTTPS → HTTPS). No se enviará a destinos menos seguros (HTTPS → HTTP).
origin	Solo el origen del documento como Referer en todos los casos. El documento <a href="https://redondo.com/index.html">https://redondo.com/index.html</a> enviará el Referer <a href="https://redondo.com/">https://redondo.com/</a>
origin-when-cross-origin	Un Referer completo (host + path + querystring) si la petición se hace al mismo origen, y únicamente el origen en otros casos
same-origin	Un Referer completo (host + path + querystring) si la petición se hace al mismo origen, y nada en otros casos
strict-origin	Solo el origen cuando el nivel de seguridad HTTP no cambia (HTTPS → HTTPS). No se enviará nada destinos menos seguros (HTTPS → HTTP).
strict-origin-when-cross-origin	Un Referer completo (host + path + querystring) si la petición se hace al mismo origen, solo el origen cuando el nivel de seguridad HTTP no cambia (HTTPS → HTTPS) y no se enviará nada destinos menos seguros (HTTPS → HTTP).
unsafe-url	Un Referer completo (host + path + querystring) siempre, sin importar el nivel de seguridad o el destino de la petición. Esta directiva revela los orígenes, las rutas de acceso a recursos y otra posible información privada de recursos protegidos por TLS a orígenes inseguros, lo que puede ser un problema de seguridad.



# PERMISSIONS-POLICY (ANTIGUAMENTE LLAMADA FEATURE-POLICY)

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Permissions\\_Policy](https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Permissions_Policy)

- Esta cabecera es experimental y su especificación no es aún estable
- Permite activar o desactivar características del browser
- Estos son sus posibles valores, que indican que la característica asociada...

- \*: Se permite en la web actual y en todos sus **iframe** anidados sin importar su origen
- '**self**': Se permite en la web actual y en todos sus **iframe** anidados que tengan el mismo origen que la misma
- '**srciframe**) Se permite solo en el iframe actual, siempre que la página que se cargue en el mismo tenga el mismo origen que la URL que aparezca como valor del atributo src del iframe
- '**none**': No se permite ni en la web actual ni en ninguno de sus elementos anidados
- <URL(s)>: Se permite solo a orígenes concretos, separados por un espacio (<https://redondo.com> <https://backup.com>)
- Ejemplo: **Permissions-Policy: autoplay 'none'; camera 'none'**
  - En Nginx: `add_header Permissions-Policy "autoplay 'none'; camera 'none'" always;`
  - En Apache: `header always set Permissions-Policy "autoplay 'none'; camera 'none'"`

- Permite cambiar una cantidad de características del browser enorme

- Permite modificar el comportamiento por defecto del navegador cuando se usa
- Y con ello, prevenir el uso no autorizado de las mismas por medio de ataques / malas configuraciones

# DIRECTIVAS QUE CONTROLAN SI EL DOCUMENTO ACTUAL PUEDE...

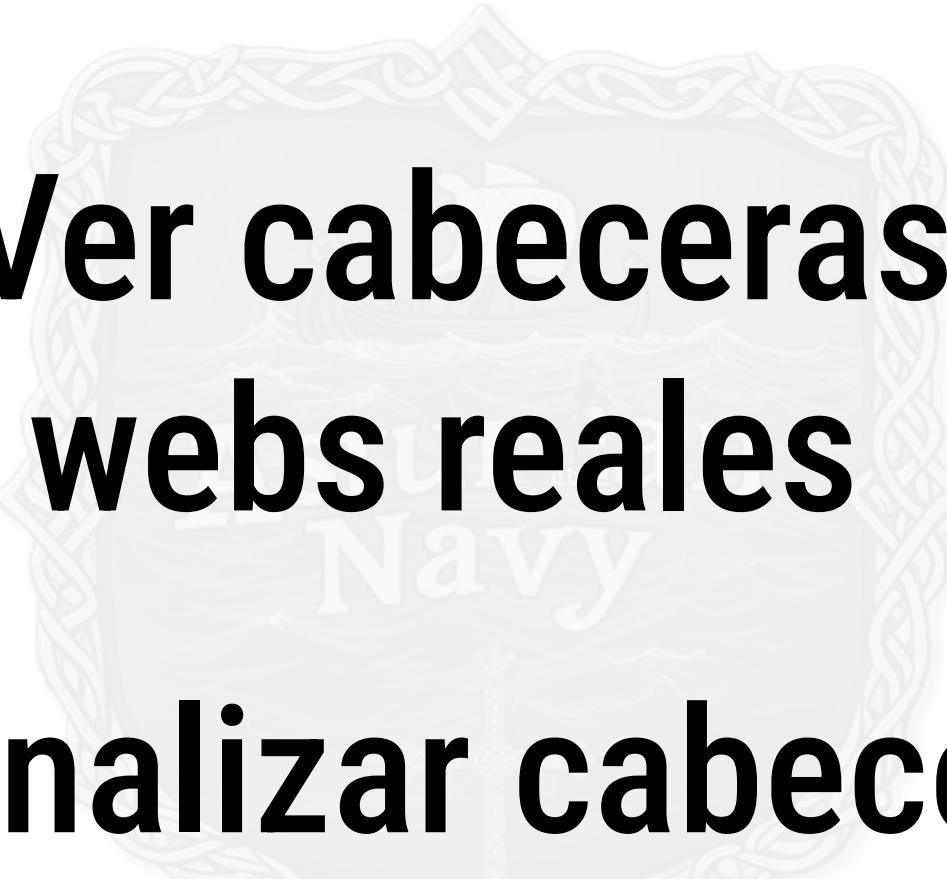
Directiva	Función	Directiva	Función
<a href="#">ambient-light-sensor</a>	Obtener información acerca de la cantidad de luz ambiente en el entorno del dispositivo usando <a href="#">AmbientLightSensor</a>	<a href="#">geolocation</a>	Usar el API de <a href="#">Geolocation</a> y funciones como <a href="#">getCurrentPosition()</a> y <a href="#">watchPosition()</a>
<a href="#">autoplay</a>	Autorreproducir elementos multimedia cargados con <a href="#">HTMLMediaElement</a> . El atributo autoplay de tags <a href="#"><code>&lt;audio&gt;</code></a> y <a href="#"><code>&lt;video&gt;</code></a> se ignora	<a href="#">gyroscope</a>	Obtener información de la orientación del dispositivo mediante <a href="#">Gyroscope</a>
<a href="#">accelerometer</a>	Obtener información acerca de la aceleración de un dispositivo a través de <a href="#">Accelerometer</a>	<a href="#">magnetometer</a>	Obtener información de la orientación del dispositivo mediante <a href="#">Magnetometer</a>
<a href="#">Battery</a>	Controlar si se permite usar la <a href="#">Battery Status API</a>	<a href="#">microphone</a>	Usar dispositivos de entrada de audio con la función <a href="#">MediaDevices.getUserMedia()</a>
<a href="#">camera</a>	Usar dispositivos de entrada de video mediante <a href="#">getUserMedia()</a>	<a href="#">midi</a>	Usar la <a href="#">Web MIDI API</a> y la función <a href="#">Navigator.requestMIDIAccess()</a>
<a href="#">display-capture</a>	Usar <a href="#">getDisplayMedia()</a> para capturar pantallas	<a href="#">payment</a>	Usar la <a href="#">Payment Request API</a> y su constructor <a href="#">PaymentRequest()</a>
<a href="#">document-domain</a>	Modificar <a href="#">document.domain</a>	<a href="#">picture-in-picture</a>	Reproducir un video en modo Picture-in-Picture
<a href="#">encrypted-media</a>	Usar el API <a href="#">Encrypted Media Extensions</a> (EME) mediante la función <a href="#">Navigator.requestMediaKeySystemAccess()</a>	<a href="#">speaker</a>	Reproducir audio con cualquiera de los métodos que lo permiten
<a href="#">execution-while-not-rendered</a>	Controlar si las tareas se deben ejecutar en los frames cuando estos no se están renderizando	<a href="#">sync-xhr</a>	Hacer peticiones <a href="#">XMLHttpRequest</a> síncronas
<a href="#">execution-while-out-of-viewport</a>	Controlar si las tareas se deben ejecutar en los frames cuando estos están fuera del área visible por el usuario	<a href="#">usb</a>	Usar el <a href="#">WebUSB API</a>
<a href="#">fullscreen</a>	Usar la función <a href="#">Element.requestFullScreen()</a>	<a href="#">wake-lock</a>	Usar la <a href="#">Wake Lock API</a> para evitar que los dispositivos entren en modos de bajo consumo
		<a href="#">webauthn</a>	Usar la <a href="#">Web Authentication API</a> para crear, guardar y leer credenciales basadas en criptografía de clave pública
		<a href="#">vr</a>	Usar la <a href="#">WebVR API</a> y el método <a href="#">Navigator.getVRDisplays()</a>
		<a href="#">xr-spatial-tracking</a>	Usar la <a href="#">WebXR Device API</a> para interactuar con una sesión WebXR



# Ejemplos de uso de cabeceras HTTP

Comprobando cabeceras HTTP “en la vida real”



- 
- (1) Ver cabeceras de webs reales
  - (2) Analizar cabeceras de webs reales

Filtrar cabeceras

## ▼ Cabeceras de la respuesta (1,552 KB)

Cabeceras sin procesar 

② content-encoding: gzip  
 ② content-security-policy: default-src 'self' blob: wss: ... 'unsafe-inline' data: https:  
 ② content-type: text/html; charset=utf-8  
 ② date: Wed, 23 Oct 2019 15:46:43 GMT  
 ② etag: W/"530fa-FZ8yCnQs5KEDtQwkIhU1U31EAQ"  
 rlogid: t6klaook%60b0%3D%3C%3Dosuojobnk...%3B(5220344-16df94c5c1e-0x803  
 ② server: envoy  
 ② set-cookie: nonsession=CgADKACBnFndjZjk0Yz...15:46:43 GMT;Path=/; HttpOnly  
 ② set-cookie: s=CgAD4ACBdscdjZjk0YzgyODYxNmQ...=.ebay.co.uk;Path=/; HttpOnly  
 ② set-cookie: dp1=bb1/GBen-GB6172dce3^;Domai...15:46:43 GMT;Path=/; HttpOnly  
 ② set-cookie: ak\_bmsc=9C9542C2EE9179FF8DA51A... domain=.ebay.co.uk; HttpOnly  
 ② strict-transport-security: max-age=31536000  
 ② vary: Accept-Encoding  
 ② x-content-type-options: nosniff  
 x-ebay-pop-id: UFES2-LHR-caching  
 x-ebay-pop-id: UFES2-LHR-frontcache  
 x-ebay-pop-id: UFES2-LHR-dweb-2  
 x-edgeconnect-midmile-rtt: 23  
 x-edgeconnect-origin-mex-latency: 12  
 x-envoy-upstream-service-time: 8  
 X-Firefox-Spdy: h2  
 ② x-frame-options: SAMEORIGIN  
 ② x-xss-protection: 1; mode=block

URL solicitada: https://www.lamoncloa.gob.es/Theme/RLaMoncloa/css/normalize.css

Método de la petición: GET

Dirección remota: 212.128.109.1:443

Código de estado: 304 Not Modified [?](#)

Versión: HTTP/1.1

Política de referencia: no-referrer-when-downgrade

Editar y volver a enviar

 Filtrar cabeceras

## ▼ Cabeceras de la respuesta (486 B)

Cabeceras sin procesar 

② Accept-Ranges: bytes  
 ② Cache-Control: public, max-age=300  
 ② Content-Length: 0  
 ② Date: Wed, 23 Oct 2019 15:47:59 GMT  
 ② ETag: '{4F54593D-F4C5-41E3-AD57-515C1D12BE14},1pub"  
 request-id: a8b7109f-8235-b025-e01d-ce3be28757e3  
 ② Server: Microsoft-IIS/8.0  
 SPRequestDuration: 0  
 SPRequestLatency: 0  
 SPRequestGUID: a8b7109f-8235-b025-e01d-ce3be28757e3  
 ② X-Content-Type-Options: nosniff  
 ② X-Frame-Options: SAMEORIGIN  
 X-MS-InvokeApp: 1; RequireReadOnly  
 X-Powered-By: ASP.NET

## Scan your site now

  Hide results  Follow redirects

### Security Report Summary



Site:	<a href="http://www.uniovi.es/">http://www.uniovi.es/ - (Scan again over https)</a>
IP Address:	156.35.233.105
Report Time:	23 Oct 2019 14:58:13 UTC
Headers:	<span style="color: red;">✗ Content-Security-Policy</span> <span style="color: red;">✗ X-Frame-Options</span> <span style="color: red;">✗ X-Content-Type-Options</span> <span style="color: red;">✗ Referrer-Policy</span> <span style="color: red;">✗ Feature-Policy</span>
Warning:	Grade capped at A, please see warnings below.

## Scan your site now

  Hide results  Follow redirects

### Security Report Summary



Site:	<a href="https://elpais.com/elpais/portada_america.html">https://elpais.com/elpais/portada_america.html</a>
IP Address:	204.237.175.170
Report Time:	23 Oct 2019 14:58:33 UTC
Headers:	<span style="color: red;">✗ Strict-Transport-Security</span> <span style="color: red;">✗ Content-Security-Policy</span> <span style="color: red;">✗ X-Frame-Options</span> <span style="color: red;">✗ X-Content-Type-Options</span> <span style="color: red;">✗ Referrer-Policy</span> <span style="color: red;">✗ Feature-Policy</span>

### Security Report Summary



Site:	<a href="https://www.ebay.co.uk/">https://www.ebay.co.uk/</a>
IP Address:	23.60.73.56
Report Time:	23 Oct 2019 14:59:56 UTC
Headers:	<span style="color: green;">✓ X-Content-Type-Options</span> <span style="color: green;">✓ X-Frame-Options</span> <span style="color: green;">✓ Content-Security-Policy</span> <span style="color: green;">✓ Strict-Transport-Security</span> <span style="color: red;">✗ Referrer-Policy</span> <span style="color: red;">✗ Feature-Policy</span>
Warning:	Grade capped at A, please see warnings below.

# NUEVOS AVANCES EN ESTE CAMPO

- Los contenidos que acabamos de ver forman parte de una línea de investigación en “defensas nativas en el ecosistema de la web”
- Actualmente es tendencia y está avanzando continuamente
- Esas investigaciones incorporan nuevas soluciones frecuentemente
- Para ver las últimas aportaciones en este campo puedes consultar:
  - <https://www.invicti.com/blog/web-security/http-security-headers/>



# LISTA DE LOGROS PARA AUTOEVALUACIÓN: SEMINARIO 6

?

- Entender por qué son peligrosas las cabeceras con información extra de HTTP
- Conocer valores adecuados para las cabeceras HTTP
- Entender la importancia de no dar demasiada información en la cabecera Referer
- Implementar una política CSP adecuada sin nonces
- Entender por qué CSP sin nonces no es suficiente en la mayoría de los casos
- Saber cómo validar una política CSP
- Saber cómo usar la cabecera the feature-policy
- Entender la utilidad de otras cabeceras HTTP relacionadas con la seguridad
- Implementar una política CSP adecuada con nonces



# SEMINARIO 6

## SEGURIDAD HTTP Y CSP



Departamento de Informática  
Universidad de Oviedo



Universidad de Oviedo