

HTTP Security Headers you need to have on your web apps

Peter Cosemans
Michiel Olijslagers

Copyright © - Euricom 2024



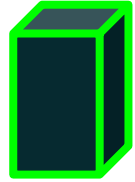
What are HTTP security headers

HTTP security headers are additional instructions sent by a web server to a browser to enhance the security of a website. These headers help protect against various types of attacks and vulnerabilities. Some common HTTP security headers include:

```
Content-Security-Policy: default-src 'self'  
X-Frame-Options: DENY  
X-Content-Type-Options: nosniff  
Permissions-Policy: camera=(), geolocation=(self), microphone=()  
Referrer-Policy: no-referrer  
Strict-Transport-Security: max-age=3153600
```

These headers, when properly configured, can significantly improve the security posture of a website and protect against common web vulnerabilities.

Security headers in response



GET / HTTP/1.1
Origin: <https://mysite.com>



HTTP/1.1 200 OK
Strict-Transport-Security: max-age=31536000
Content-Security-Policy: script-src
x-frame-options: DENY

Evaluate the security headers of a site

- Quick and easy way to verify security headers: <https://securityheaders.com/>
 - <https://www.kbc.be>
 - <https://www.n26.com>
 - <https://www.bol.com>
 - <https://deskreservation.euri.com>
 - <https://www.resengo.be>
- Alternatives:
 - <https://domsignal.com/secure-header-test>
 - <https://www.atatus.com/tools/security-header>

Security Headers
Powered by Probely

Home About API

Scan your site now

enter address here Scan

Hide results Follow redirects

Grand Totals	
A+	5,543,167
A	31,971,502
B	6,818,691
C	12,615,812
D	31,730,541
E	9,153,785
F	127,928,823
R	44,304,241
Total	270,066,562


Recent Scans	
securityheaders.co...	A+
www.report.ncsc.ad...	B
console.scaleway.c...	B
chatturbatt.com	F
clientes.aduanasdh...	A
report.ncsc.admin...	F
tintuc.queenmobile...	F
rewards-apt.closta...	A
cyberthreat.id	A

Hall of Fame	
securityheaders.co...	A+
clientes.aduanasdh...	A
rewards-apt.closta...	A
cyberthreat.id	A
develp4powdevelend...	A
bi.healthbit.com.b...	A
mh-staging.motorha...	A+
adocday.com	A
www.bibancainvesti...	A+

Hall of Shame	
www.fiverr.com	F
chatturbatt.com	F
report.ncsc.admin...	F
tintuc.queenmobile...	F
ufpa.br	F
bestreview.info	F
securityteamin...	F
easy.ksubest.com	F
netahapi12.ecozum...	F

Evaluate the security headers of a site

Security Report Summary




Site:	https://financien.belgium.be/
IP Address:	2a01:690:35:100::f5:79
Report Time:	18 Mar 2024 14:54:59 UTC
Headers:	<div><div>✓ X-Frame-Options</div><div>✓ Strict-Transport-Security</div><div>✗ Content-Security-Policy</div><div>✗ X-Content-Type-Options</div><div>✗ Referrer-Policy</div><div>✗ Permissions-Policy</div></div>
Advanced:	Your site could be at risk, let's perform a deeper security analysis of your site and APIs: Start Now

Missing Headers

Content-Security-Policy	Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
Referrer-Policy	Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
Permissions-Policy	Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.

Evaluate the security headers of a site

Security Headers
Powered by  Probely


HomeAboutAPI


API Keys


As one of our most commonly requested features, you can now conduct a Security Headers scan using our API. The scan results will be returned as JSON, which makes it easy to automatically monitor your security on an ongoing basis.

Simply choose the volume of Scans you require and hit 'Subscribe' to begin. You'll have your API key in just a couple of minutes!

MonthlyYearly

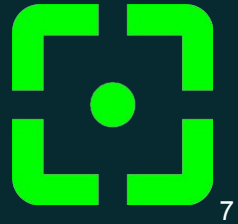

200 Scans/Month
This API key allows you to conduct 200 scans per month!
US\$2.99 per month
Subscribe


1,000 Scans/Month
This API key allows you to conduct 1,000 scans per month!
US\$9.99 per month
Subscribe


5,000 Scans/Month
This API key allows you to conduct 5,000 scans per month!
US\$39.99 per month
Subscribe

*prices may be subject to tax

HTTP Strict Transport Security (HSTS)



HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to **enforce** the use of **HTTPS**.

Prevent some classes of man-in-the-middle (MITM) attacks

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

- **max-age**: The time, in seconds this header is enforced
- **includeSubDomains**: Apply the rule to all of the site's subdomains as well

Strict Transport Security - In action

Step 1: GET http://training.euri.com

Name

X

Headers

Preview

Response

Initiator

Timing

training.euri.com

training.euri.com

training.euri.com

training.euri.com

▼ General

Request URL:

http://training.euri.com/

Request Method:

GET

Status Code:

301 Moved Permanently

Remote Address:

20.101.2.157:80

Referrer Policy:

strict-origin-when-cross-origin

▼ Response Headers

☐ Raw

Content-Length:

0

Date:

Mon, 18 Mar 2024 15:28:05 GMT

Location:

https://training.euri.com/

Initial request on http, server redirect to https

Strict Transport Security - In action

Step 2: Server redirect to https://training.euri.com

Name	X	Headers	Preview	Response	Initiator	Timing
training.euri.com	▼ General					
training.euri.com	Request URL:		https://training.euri.com/			
training.euri.com	Request Method:		GET			
training.euri.com	Status Code:		200 OK			
	Remote Address:		20.82.12.44:443			
	Referrer Policy:		strict-origin-when-cross-origin			
	▼ Response Headers					
	Cache-Control:		public, must-revalidate, max-age=30			
			...			
	Strict-Transport-Security:		max-age=63072000; includeSubdomains			

Server responds with Strict-Transport-Security header

Strict Transport Security - In action






Step 3: GET http://training.euri.com (next visit)

Name	✕ Headers	Preview	Response	Initiator	Timing
training.euri.com	▼ General				
training.euri.com	Request URL:		http://training.euri.com/		
training.euri.com	Request Method:		GET		
training.euri.com	Status Code:		● 307 Internal Redirect		
	Referrer Policy:		strict-origin-when-cross-origin		
	▼ Response Headers		<input type="checkbox"/> Raw		
	Cross-Origin-Resource-Policy:		Cross-Origin		
	Location:		https://training.euri.com/		
	Non-Authoritative-Reason:		HSTS		

On subsequent request the browser auto redirects

Strict Transport Security - In action

Step 4: Browser redirects to https://training.euri.com

Name	X	Headers	Preview	Response	Initiator	Timing
 training.euri.com	▼ General					
 training.euri.com	Request URL:		https://training.euri.com/			
 training.euri.com	Request Method:		GET			
 training.euri.com	Status Code:		 200 OK			
	Remote Address:		20.82.12.44:443			
	Referrer Policy:		strict-origin-when-cross-origin			
	▼ Response Headers					
	Cache-Control:		public, must-revalidate, max-age=30			
			...			
	Strict-Transport-Security:		max-age=63072000; includeSubdomains			

We now always load via HTTPS

Strict Transport Security - Preload List

HSTS only works when you visited the site before, not the FIRST time

Solution: The HSTS Preload List

- List maintained by Google, used by all browsers
- HTTP request will 307 (not even 301), even you never visited the site

Strict Transport Security - Preload List

```
{ "name": "toshnix.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "warrencreative.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "zeplin.io", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "17hats.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "cdnb.co", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "github.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "id-co.in", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ideaweb.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "man3s.jp", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "meinebo.it", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "nmctest.net", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "partyvan.eu", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "partyvan.it", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "partyvan.nl", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "partyvan.se", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "suite73.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "wubthecaptain.eu", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true }
```

Strict Transport Security - Preload List

How to get on the list:

- Add the 'preload' option to your header to confirm you submission
- Register your site; <https://hstspreload.org/>

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

Strict Transport Security - Practises

- Most hosting solution are providing the Strict-Transport-Security by default.
- Use Chrome Internals to query/delete HSTS entries.

<chrome://net-internals/#hsts>

X-Frame-Options (XFO)



X-Frame-Options (XFO)

The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe>, <embed> or <object>

Sites can use this to avoid click-jacking attacks, by ensuring that their content is not embedded into other sites

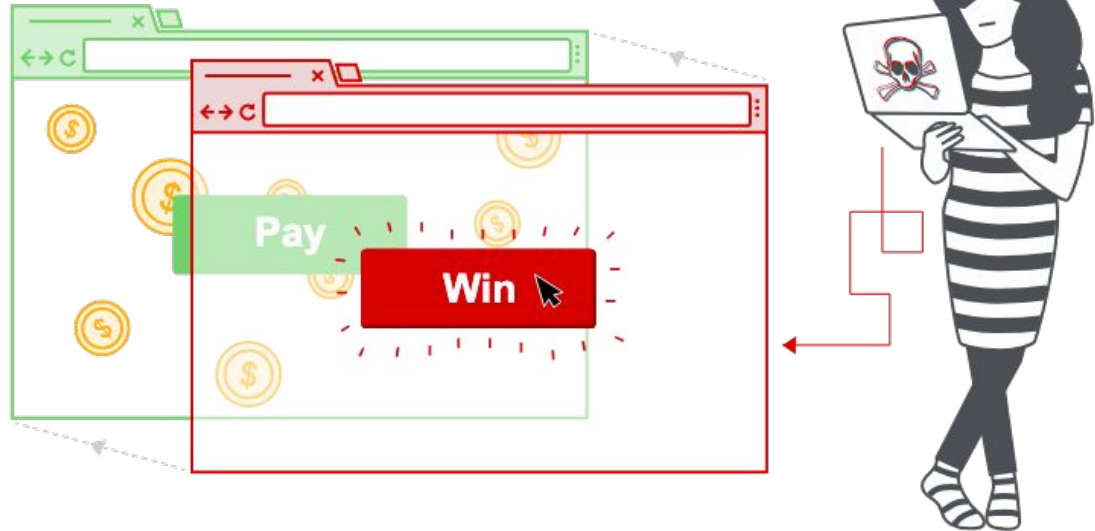
```
x-frame-options: DENY
```

Directives

- **DENY** : Prevents any domain from framing the content (most secure)
- **SAMEORIGIN** : Only allows framing from the same domain
- **ALLOW_FROM http://example.com** : Allows framing from the specified domain

X-Frame-Options - Click-jacking

Click-jacking is an attack that occurs when an attacker uses a transparent iframe in a window to trick a user into clicking on a CTA, such as a button or link, to another server in which they have an identical looking window.



X-Frame-Options - Practises

Note: The **Content-Security-Policy** HTTP header has a **frame-ancestors** directive which obsoletes this header for supporting browsers.

X-Content-Type-Options (CTO)



X-Content-Type-Options (CTO)

The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in the **Content-Type headers should be followed** and not be changed

The header allows you to **avoid MIME type sniffing** by saying that the MIME types are deliberately configured.

This header should always be set.

```
X-Content-Type-Options: nosniff
```

Permission Policy



Permission Policy

- Control and specify permissions/capabilities available to web pages
- Fine-grained control over browser features and APIs
 - E.g. camera access, geolocation, microphone,...
- Allow or deny access to specific features

```
Permissions-policy: geolocation=(self), microphone=(), camera=()  
Permissions-Policy: geolocation=(self "https://a.example.com")
```

Geolocation is allowed, rest is denied

Permission Policy

```
Permissions-Policy: <directive> ( <allowlist> )
```

- Directives, list of the permitted directive (permission) names:
 - accelerometer
 - battery
 - display-capture
 - geolocation
 - gyroscope

```
Permissions-Policy: geolocation=(self "https://a.example.com")
```

Permission Policy

```
Permissions-Policy: <directive> ( <allowlist> )
```

- Allowlist, a list of origins + extra values
 - `*` allowed here and in all nested contexts
 - `self` allowed in current origin
 - `<origin>` allowed for a specific origin (example.com, *.example.com)
 - `()` disabled for all contexts

```
Permissions-Policy: geolocation=(self "https://a.example.com")
```

Permission Policy - Practises

- When not specifying a Permissions Policy, all feature are allowed
- Permissions Policy was previously known as 'Feature Policy'
- Generate a complete permission policy with [Permission Policy header generator](https://www.permissionspolicy.com/)

Permissions Policy HTTP Header Generator

Standardized Features

Enable

* ⓘ

self ⓘ

custom ⓘ

☒

☐

☐

Disable

☐

Feature Name

Select All

accelerometer ⓘ

ambient-light-sensor ⓘ

autoplay ⓘ

battery ⓘ

camera ⓘ

cross-origin-isolated ⓘ

display-capture ⓘ

document-domain ⓘ

encrypted-media ⓘ

execution-while-not-rendered ⓘ

execution-while-out-of-viewport ⓘ

fullscreen ⓘ

Your Generated Policy

Content Security Policy (CSP)



Content Security Policy (CSP)

- Extra layer of defense against XSS attacks
- Control and restrict types of content that can be loaded
 - E.g. scripts, stylesheets, images, fonts,...
- Helps prevent
 - Execution of malicious scripts
 - Loading of unauthorized resources from external domains

```
Content-Security-Policy: default-src 'self';
```

Privacy Headers



Referrer Policy Header

- Controls the Referer header:
 - Is information shared
 - What information is shared
- Protects user privacy by limiting amount of information shared with websites

```
Referrer-Policy: no-referrer
```

Referrer Policy - What is a referer

The **Referer header** is an HTTP header field that is included in a request **sent by a web browser** to a server. It **contains the URL of the previous web page** from which the user navigated to the current page.

Name	✕ Headers	Preview	Response	Initiator	Timing
clients	Accept-Encoding:		gzip, deflate, br, zstd		
clients/	Accept-Language:		en-GB,en-US;q=0.9,en;q=0.8		
	Cache-Control:		no-cache		
	Pragma:		no-cache		
	Referer:		https://www.euri.com/		
	Sec-Ch-Ua:		"Chromium";v="122", "Not(A:Brand";v="24", "G		
	Sec-Ch-Ua-Mobile:		?0		
	Sec-Ch-Ua-Platform:		"macOS"		

Referrer Policy - Directives

- no-referrer
- no-referrer-when-downgrade
- same-origin
- origin
- strict-origin
- origin-when-cross-origin
- strict-origin-when-cross-origin
- unsafe-url

`Referrer-Policy: <directive>`

Referrer Policy - Directives

No context taken into account

Policy	Referrer		
	No data	Origin only	Full URL
no-referrer	✓		
origin		✓	
unsafe-url 🦴			✓

Don't use this, its unsafe...

Referrer Policy - Directives

Is the request made to a less secure destination

Policy	Referrer		
	No data	Origin only	Full URL
strict-origin	Request from HTTPS to HTTP	Request from HTTPS to HTTPS or HTTP to HTTP	
no-referrer-when-downgrade	Request from HTTPS to HTTP		Request from HTTPS to HTTPS or HTTP to HTTP

Referrer Policy - Directives

Is the request cross- or same-origin

Policy	Referrer		
	No data	Origin only	Full URL
origin-when-cross-origin		Cross-origin request	Same-origin request
same-origin	Cross-origin request		Same-origin request


Referrer Policy - Directives

Is the request made to a less secure destination AND is it cross- or same-origin


Policy	Referrer		
	No data	Origin only	Full URL
strict-origin-when-cross-origin (default)	Request from HTTPS to HTTP	Request from HTTPS to HTTPS or HTTP to HTTP	Same-origin request

Policy	Referer		
	No data	Origin only	Full URL
no-referrer	✓		
origin		✓	
unsafe-url 🦴			✓
strict-origin	Request from HTTPS to HTTP	Request from HTTPS to HTTPS or HTTP to HTTP	
no-referrer-when-downgrade	Request from HTTPS to HTTP		Request from HTTPS to HTTPS or HTTP to HTTP
origin-when-cross-origin		Cross-origin request	Same-origin request
same-origin	Cross-origin request		Same-origin request
strict-origin-when-cross-origin (default)	Request from HTTPS to HTTP	Request from HTTPS to HTTPS or HTTP to HTTP	Same-origin request

Referrer Policy - In action

×	Headers	Preview	Response	Initiator	Timing
▼ General					
Request URL:		http://app.goodwebsite.com:3001/api			
Request Method:		GET			
Status Code:		 200 OK			
Remote Address:		127.0.0.1:3001			
Referrer Policy:		unsafe-url			

► Response Headers (9)	
▼ Request Headers	<input type="checkbox"/> Raw
Accept:	text/html,application/xhtml+xml,application/xml;ge/webp,image/apng,*/*;q=0.8,application/signexchange;v=b3;q=0.7
Accept-Encoding:	gzip, deflate
Accept-Language:	en-GB,en-US;q=0.9,en;q=0.8
Cache-Control:	no-cache
Connection:	keep-alive
Host:	app.goodwebsite.com:3001
Pragma:	no-cache
Referer:	http://marketing.goodwebsite.com:3001/api
Upgrade-Insecure-Requests:	1

×	Headers	Preview	Response	Initiator	Timing
▼ General					
Request URL:		http://app.goodwebsite.com:3001/api			
Request Method:		GET			
Status Code:		 200 OK			
Remote Address:		127.0.0.1:3001			
Referrer Policy:		strict-origin			

► Response Headers (9)	
▼ Request Headers	<input type="checkbox"/> Raw
Accept:	text/html,application/xhtml+xml,application/xml;ge/webp,image/apng,*/*;q=0.8,application/signexchange;v=b3;q=0.7
Accept-Encoding:	gzip, deflate
Accept-Language:	en-GB,en-US;q=0.9,en;q=0.8
Cache-Control:	no-cache
Connection:	keep-alive
Host:	app.goodwebsite.com:3001
Pragma:	no-cache
Referer:	http://marketing.goodwebsite.com:3001/
Upgrade-Insecure-Requests:	1

Referrer Policy - Practises

- Unexpected cross-origin information leakage hinders web users privacy. A protective referrer policy can help
- Not all browsers have the same default policy. Always good to set it

Clear-Site-Data Header

The Clear-Site-Data header clears browsing data (cookies, storage, cache) associated with the requesting website. It allows web developers to have more control over the data stored by a client browser for their origins.

```
Clear-Site-Data: "*" 
```

Cache-Control Header

This header allows you to control the caching of specific web pages. Several directives are available, but the typical usage is simply

```
Cache-Control: no-store
```

Unwanted Headers



Unwanted headers

Removing some HTTP headers can improve security by reducing the amount of information disclosed to potential attackers. Certain headers may reveal details about the server, software versions, or internal network structure, which can be exploited to target specific vulnerabilities.

```
Server: Microsoft-IIS/10.0  
X-Powered-By: ASP.NET  
X-AspNet-Version: 4.0.30319  
Server-Timing: db;dur=53, app;dur=47.2
```

So, remove these potentially dangerous headers

Obsolete headers



Obsolete headers

There are several HTTP headers that have become obsolete over time, and are even less secure than intended

- **X-Xss-Protection:** Even though this feature can protect users of older web browsers that don't yet support CSP, in some cases, XSS protection can create XSS vulnerabilities in otherwise safe websites.
- **X-Frame-Options:** The Content-Security-Policy HTTP header has a frame-ancestors directive which obsoletes this header for supporting browsers.
- **Expect-CT:** The Expect-CT header lets sites opt in to reporting and/or enforcement of Certificate Transparency requirements. This feature is no longer recommended
- **Feature-Policy:** Permissions Policy used to be called Feature Policy. The name has changed, and so has the HTTP header syntax, so bear this in mind if you have used Feature Policy in the past, and check the browser support tables

Deploy Security Headers in a modern web app



Using Azure WAF

Azure WAF (Web Application Firewall) is a cloud-based security service provided by Microsoft Azure. It offers protection for web applications against common web-based attacks, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Cons: Your security headers are controlled by DevOps and can't be tested locally.

Using Azure Static Web Apps

```
{  
  // staticwebapp.config.json  
  "globalHeaders": {  
    "Content-Security-Policy": "default-src 'self'",  
    "X-Frame-Options": "DENY",  
    "Referrer-Policy": "no-referrer",  
    "X-Content-Type-Options": "nosniff",  
    "Permissions-Policy": "camera=(), geolocation=(self), microphone=()",  
  }  
}
```

Copy the staticwebapp.config.json to the root of your deployment

Using NextJS

```
// next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/(.html)',
        headers: [
          { key: 'X-Frame-Options', value: 'DENY' },
          { key: 'Content-Security-Policy', value: "default-src 'self'" },
          { key: 'X-Content-Type-Options', value: 'nosniff' },
          { key: 'Permissions-Policy', value: "camera=(), geolocation=(self), microphone=()" },
          { key: 'Referrer-Policy', value: 'origin-when-cross-origin' },
        ],
      },
    ],
  },
},
```

Using NGINX (docker)

```
//nginx.conf
server {
    server_name example.com;
    add_header X-Frame-Options SAMEORIGIN;
    add_header Strict-Transport-Security "max-age=31536000";
    add_header Content-Security-Policy "default-src 'self';" always;
}
```

Using .NET & NWebsec

```
using NWebsec.Owin;  
public void Configuration(IApplicationBuilder app)  
{  
    app.UseCsp(options => options  
        .DefaultSources(s => s.Self())  
        .ScriptSources (s => s.Self(). CustomSources ("my.example.com"))  
        .ReportUri(r => r.Uri("/report")));  
  
    app.UseCspReportOnly (options => options  
        .DefaultSources(s => s.Self())  
        .ImageSources (s => s.None()));  
}
```

Using NodeJS, Express & Helmet

```
const helmet = require("helmet");
app.use(
  helmet({
    strictTransportSecurity: {
      maxAge: 123456,
    },
    contentSecurityPolicy: {
      directives: {
        "script-src": ["'self'", "example.com"],
      },
    },
  })
);
```

Key Takeaways



Key takeaways

Validate your app with <https://securityheaders.com/>

Apply the your Security Headers

Remove unwanted headers

Test your header locally, apply in prod.

Take responsibility of your HTTP headers in production.

Further reading

Maud Nalpas, **Referer and Referrer-Policy best practices**

(<https://web.dev/articles/referrer-best-practices>)

Scott Helme, **HSTS - The missing link in Transport Layer Security**

(<https://scotthelme.co.uk/hsts-the-missing-link-in-tls/>)

MDN docs, **X-Frame-Options**

(<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>)

Portswigger, **Clickjacking (UI redressing)**

(<https://portswigger.net/web-security/clickjacking>)