
X-Frame-Options – How to Combat Clickjacking

BRIAN JACKSON | UPDATED: JULY 21, 2016



HTTP security headers provide yet another layer of security by helping to mitigate attacks and security vulnerabilities by telling your browser how to behave. In this post we will be diving more in-depth into `x-frame-options` (XFO), which is a header that **helps to protect your visitors against clickjacking attacks**. It is recommended that you use the `x-frame-options` header on pages which should not be allowed to render a page in a frame.

Contents [\[hide\]](#)

[What is X-Frame-Options?](#)

[Clickjacking](#)

[X-Frame-Options Directives](#)

[1. DENY Directive](#)

[2. SAMEORIGIN Directive](#)

[3. ALLOW-FROM uri Directive](#)

[Enabling X-Frame-Options Header](#)

[Enable on Nginx](#)

[Enable on Apache](#)

[Enable on IIS](#)

[X-Frame-Options Browser Support](#)

What is X-Frame-Options?

`x-frame-options` (XFO), is a HTTP response header, also referred to as a HTTP security header, which has been around since 2008. In 2013 it was officially published as RFC 7034 (<https://tools.ietf.org/html/rfc7034>), but is not an internet standard. This header tells your browser how to behave when handling your site's content. The main reason for its inception was to provide clickjacking protection by not allowing rendering of a page in a frame. This can include rendering of a page in a `<frame>` , `<iframe>` , or `<object>` . Iframes are used to embed and isolate third-party content into a website. Examples of things that use iframes might include social media sharing buttons, Google Maps, video players, audio players, 3rd party advertising, and even some OAuth implementations.

How widely is the `x-frame-options` header being used? Scott Helme did an interesting case study back in February 2016. He analyzed the security headers (<https://scotthelme.co.uk/security-headers-alexa-top-million/>) of the top 1 million sites, according to Alexa, and this is what he found. It is shown as XFO below in the chart. Only 7.6% of the top sites are utilizing the header.

(<https://blog.keycdn.com/blog/wp-content/uploads/2016/07/x-frame-options-usage.webp>)

	Aug 2015	Aug 2015	Feb 2016	Feb 2016	% change
CSP	1,365	0.1476%	2,764	0.2942%	102.49%
CSPRO	211	0.0228%	4,909	0.5224%	2226.54%
XWCSP	183	0.0198%	316	0.0336%	72.68%
XCSP	304	0.0329%	520	0.0553%	71.05%
PKP	148	0.0160%	249	0.0265%	68.24%
PKPRO	21	0.0023%	67	0.0071%	219.05%
STS	11,308	1.2231%	22,078	2.3496%	95.24%
XCTO	44,315	4.7933%	57,586	6.1285%	29.95%
XFO	55,042	5.9536%	72,011	7.6637%	30.83%
XXSSP	41,948	4.5373%	48,014	5.1098%	14.46%
XDO	192	0.0208%	349	0.0371%	81.77%
XPCDP	346	0.0374%	510	0.0543%	47.40%
HTTPS	62,043	6.7108%	88,199	9.3865%	42.16%

(<https://blog.keycdn.com/blog/wp-content/uploads/2016/07/x-frame-options-usage.webp>)

While it is good to see increases across the board for not only the XFO header, but all security headers, the overall usage is very low. More sites need to start taking advantage of HTTP security headers (<https://www.keycdn.com/blog/http-security-headers/>).

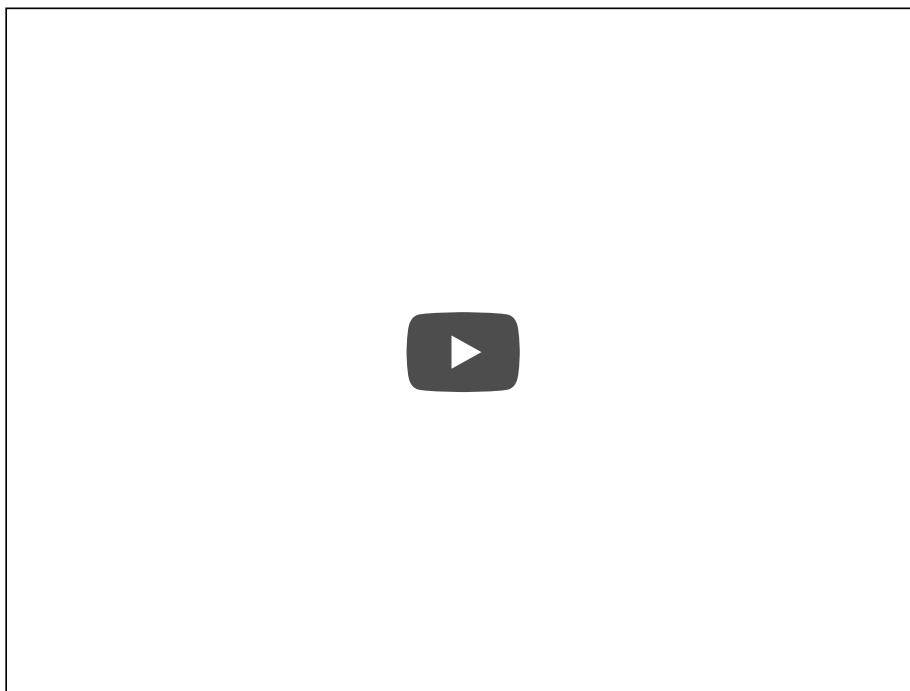
Clickjacking

So what exactly is clickjacking? Clickjacking 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. The attacker in a sense hijacks the clicks meant for the original server and sends them to the other server. This is an attack on both the visitor themselves and on the server.

Here are a couple possible known exploits (<https://en.wikipedia.org/wiki/Clickjacking#Examples>) or uses for clickjacking.

- Tricking users into making their social networking profile information public
- Sharing or liking links on Facebook
- Clicking Google AdSense ads to generate pay per click revenue
- Making users follow someone on Twitter or Facebook
- Downloading and running a malware (malicious software) allowing to a remote attacker to take control of others computers
- Getting likes on Facebook fan page or +1 on Google Plus
- Playing YouTube videos to gain views

Clickjacking is easy to implement, and if your site has actions that can be done with a single click, then most likely it can be clickjacked. It might not be as common as cross site scripting or code injection attacks, but it is still another vulnerability that exists. Sometimes it helps to see a visual. Below is a clickjacking demo using both transparent and non-transparent iframes.



Here is another good live example (<http://www.enhanceie.com/test/clickjack/>) in which you can see a demonstration of clickjacking.

X-Frame-Options Directives

The `x-frame-options` header has three different directives in which you can choose from. These must be sent as an HTTP header, as the browser will ignore if found in a META tag. It is also important to note that certain directives are only supported in certain browsers. See browser support further below in this post. While it is not required to send this response header across your entire site, it is best practice to at least enable it on pages that need it.

1. DENY Directive

The DENY directive completely disables the loading of the page in a frame, regardless of what site is trying. Below is what the header request will look like if this is enabled.

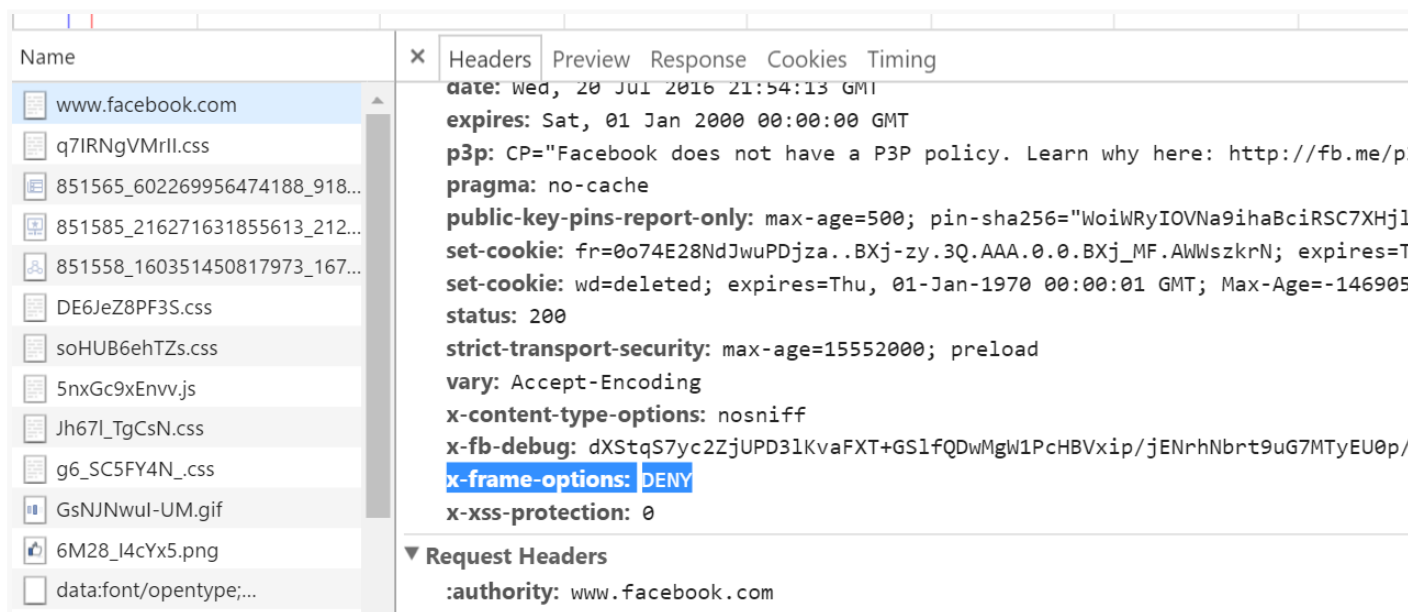
1. `x-frame-options: DENY`

This might be a great way to lock down your site, but it will also break a lot of functionality. The following two directives below are more common use cases for a typical website.

Examples of Sites Currently Using the DENY directive

- Facebook
- Github

(<https://blog.keycdn.com/blog/wp-content/uploads/2016/07/x-frame-options-deny-directive.webp>)



2. SAMEORIGIN Directive

The SAMEORIGIN directive allows the page to be loaded in a frame on the same origin as the page itself. Below is what the header request will look like if this is enabled.

1. `x-frame-options: SAMEORIGIN`

A good example of this working is the YouTube video we have above in this post. It is using the following iframe to load the video.

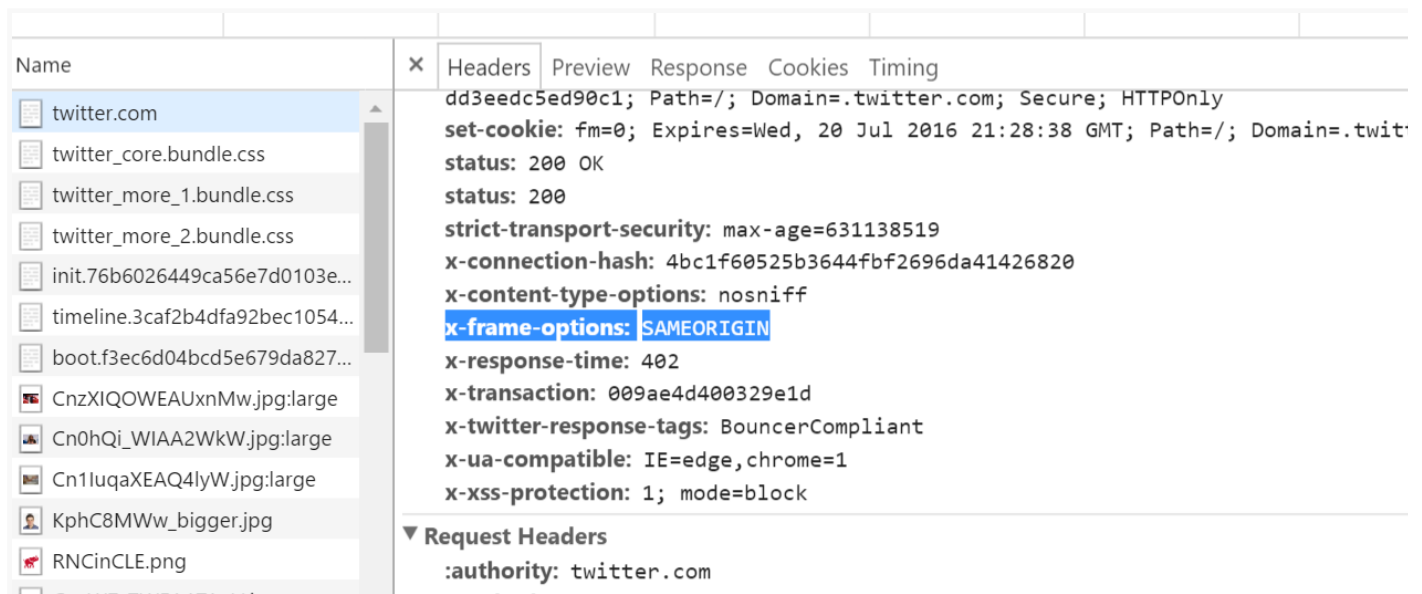
```
<iframe width="625" height="469" src="https://www.youtube.com/embed/3mk0RySeNsU?feature=oembed" frame
```

We have the SAMEORIGIN header enabled on this blog, and this allows the YouTube video to still work. With this directive, you can still use the page in a frame as long as the site including it in a frame is the same as the one serving the page. This is probably the most commonly used directive out of the three. It is a good balance between functionality and security.

It is also important to note that if a browser or plugin can not reliably determine whether the origin of the content and the frame have the same origin, this must be treated as DENY.

Examples of Sites Currently Using the SAMEORIGIN directive

- Twitter
- Amazon
- eBay
- LinkedIn



(<https://blog.keycdn.com/blog/wp-content/uploads/2016/07/x-frame-options-sameorigin-directive.webp>)

3. ALLOW-FROM *uri* Directive

The ALLOW-FROM *uri* directive allows the page to only be loaded in a frame on the specified origin and or domain. Below is what the header request will look like if this is enabled.

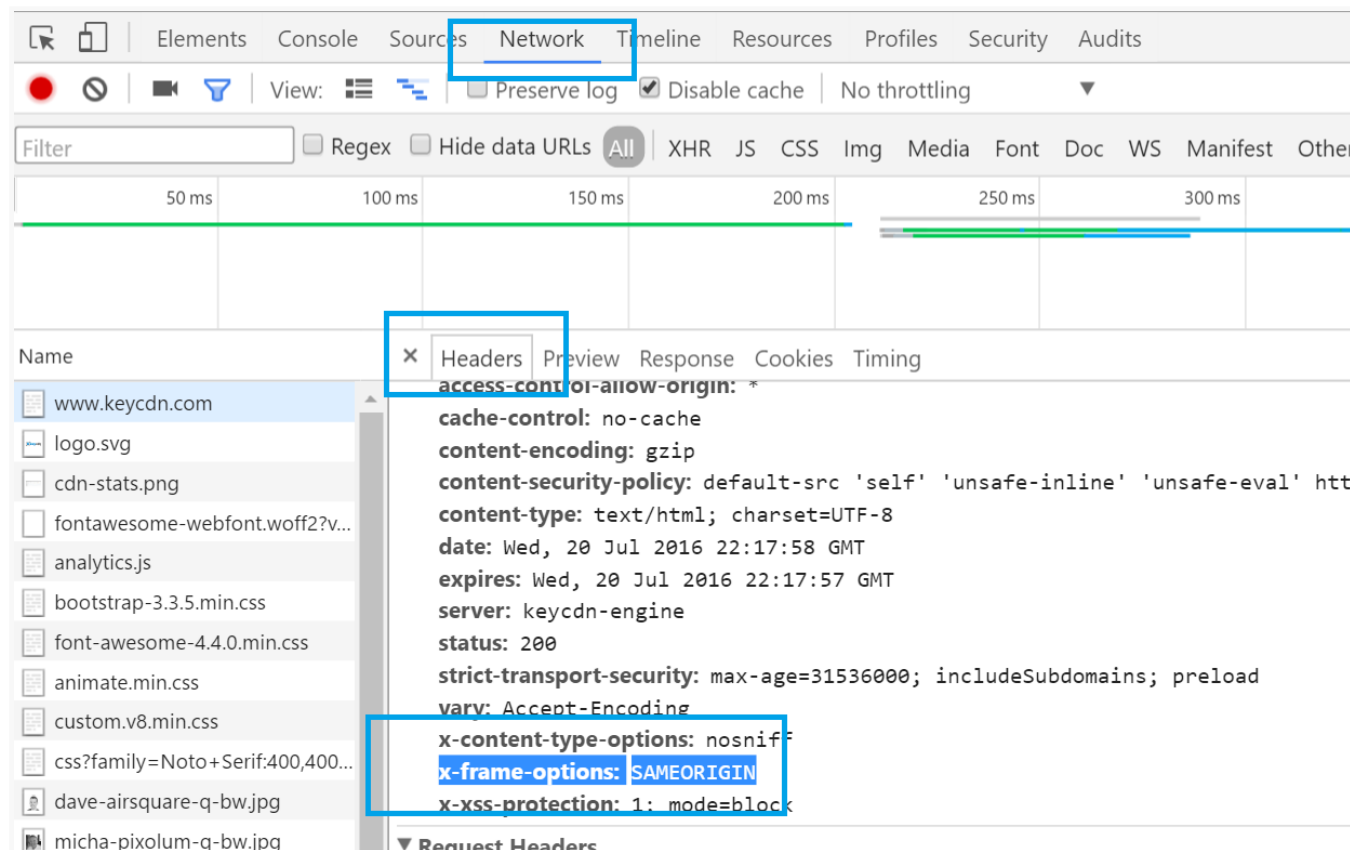
1. `x-frame-options: ALLOW-FROM https://domain.com/`

This allows you to lock down your site to only trusted origins. But be careful with this directive. If you apply it and the browser does not support it, then you will have NO clickjacking defense in place.

Enabling X-Frame-Options Header

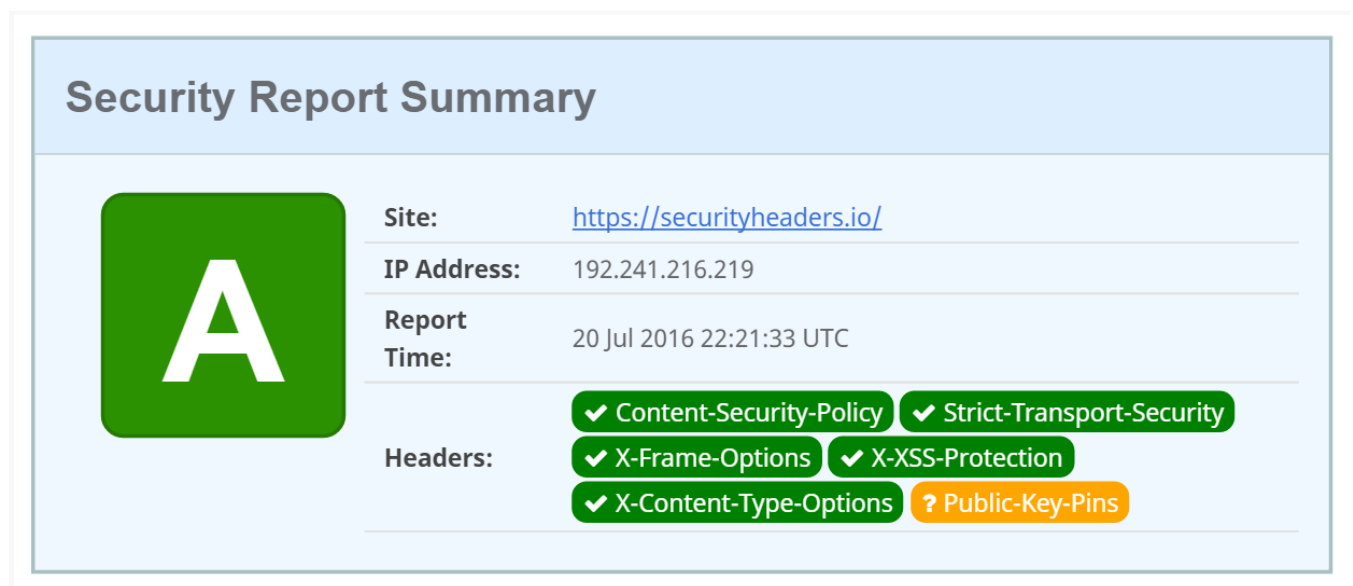
The `x-frame-options` header is easy to implement and only requires a slight web server configuration change. You might also want to check to make sure you don't already have the header enabled. Here are a couple easy ways to quickly check.

1. Open up the network tab in Chrome DevTools (<https://www.keycdn.com/blog/chrome-devtools/>) and if your site is using a security header it will show up on the Headers tab.



(<https://blog.keycdn.com/blog/wp-content/uploads/2016/07/x-frame-options-google-chrome.webp>)

2. Another quick way to check your security headers is to quickly scan your site with a free tool, securityheaders.io (<https://securityheaders.io/>), created by Scott Helme. This gives you a grade based on all of your security headers and you can see what you might be missing.



(<https://blog.keycdn.com/blog/wp-content/uploads/2016/07/security-headers-scan.webp>)

Enable on Nginx

To enable the `x-frame-options` header on Nginx simply add it to your server block config.

```
add_header x-frame-options "SAMEORIGIN" always;
```

Enable on Apache

To enable on Apache simply add it to your `httpd.conf` file (Apache config file).

```
header always set x-frame-options "SAMEORIGIN"
```

Enable on IIS

To enable on IIS simply add it to your site's `Web.config` file.

```
<system.webServer>
...

<httpProtocol>
  <customHeaders>
    <add name="X-Frame-Options" value="SAMEORIGIN" />
  </customHeaders>
</httpProtocol>

...
</system.webServer>
```

X-Frame-Options Browser Support

It is important to realize that not all browsers support the ALLOW-FROM directive. So be careful if you are using that. All modern browsers do support the DENY and SAMEORIGIN directives. For legacy browsers, such as IE7 for example, your best solution currently is to use what they call a [frame-breaker](https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet#Best-for-now_Legacy_Browser_Frame_Breaking_Script) (https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet#Best-for-now_Legacy_Browser_Frame_Breaking_Script) or frame-buster.

Browser	DENY/SAMEORIGIN Support	ALLOW-FROM support
Chrome	4.1 (<u>http://blog.chromium.org/2010/01/security-in-depth-new-security-features.html</u>)+	No support
Edge	Yes	NA
Firefox	1.9.2+	18.0+ (<u>https://bugzilla.mozilla.org/show_bug.cgi?id=690168</u>)

Internet Explorer	8.0+ (http://blogs.msdn.com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx)	9.0+ (http://erlend.oftedal.no/blog/tools/xframeoptions/)
Opera	10.50+ (http://www.opera.com/docs/specs/presto26/#network)	?
Safari	4.0+ (http://www.zdnet.com/blog/security/apple-safari-jumbo-patch-50-vulnerabilities-fixed/3541)	?

Frame-Ancestors

Another newer option and or alternative you have to using XFO is to use the Content Security Policy and `frame-ancestors` directive. This will most likely eventually replace XFO altogether. One major benefit to this directive is that it allows you to authorize multiple domains. However, it is not supported in all browsers yet, Chrome 39+ and FF 33+ support it, but there is no support for Internet Explorer. You could however use both the `x-frame-options` and `frame-ancestors` together.

Summary

Hopefully now you understand a little more about what the `x-frame-options` HTTP response header does and how it can help prevent clickjacking. As seen above, this is very easy to implement. We use security headers on our websites and we encourage you to do the same. Together we can make the web a more secure place and help boost the security header usage numbers.

Related Articles

- [Hardening Your HTTP Security Headers](https://www.keycdn.com/blog/http-security-headers/)
(<https://www.keycdn.com/blog/http-security-headers/>)
- [Block Bad Bots – New Security Feature From KeyCDN](https://www.keycdn.com/blog/block-bad-bots/)
(<https://www.keycdn.com/blog/block-bad-bots/>)

#PERFMATTERS





250GB Free Traffic (https://www.keycdn.com/signup/?a=1&utm_source=keycdn&utm_medium=cta&utm_campaign=widget2)

Supercharge your Website with KeyCDN
HTTP/2 - Free SSL - RESTful API - 29+ POPs - Instant Purge

7 Comments **KeyCDN Blog**

 **Login** ▾

 **Recommend** 1  **Share**

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



April King • 2 years ago

Please note that X-Frame-Options is essentially on life support at this point. I doubt existing support will disappear anytime soon, but CSP's "frame-ancestors" directive offers the same thing, while providing a broader range of support for ALLOW-FROM-style functionality.

1 ^ | ▾ • Reply • Share ›



Brian Jackson → **April King** • 2 years ago

Thanks for the comment April. Sounds like a good topic for an upcoming blog post. I have added a mention about that above in the post now. Good for people to be aware of. The reason X-Frame-Options still exists is because unfortunately IE 8 through 11 doesn't support frame-ancestors.

2 ^ | ▾ • Reply • Share ›



April King → **Brian Jackson** • 2 years ago

For sure! The defined behavior for browsers that support both is to ignore whatever is set for X-Frame-Options if they also see a CSP frame-ancestors directive:

[https://www.w3.org/TR/CSP2/...](https://www.w3.org/TR/CSP2/)

Note that Mozilla's scanning tool (the HTTP Observatory) understands both; here's a domain that uses both:

<https://mozilla.github.io/h...>

In any case, very well-written post!

2 ^ | ▾ • Reply • Share ›



Aditi Tilaskar • 2 months ago

how to add X-Frame-Options in wordpress. i want to stop my website load in iframe.

^ | ▾ • Reply • Share ›



Cody Mod → **Aditi Tilaskar** • 2 months ago

You'll need to do this in your server's config file. Simply add the appropriate snippet as outlined within the "Enabling X-Frame-Options Header" section of the article.

1 ^ | ▾ • Reply • Share ›



Priya Swaminathan • a year ago

i am using apache 2.4.9 , if i add header always set x-frame-options "SAMEORIGIN"

wamp server not started.

^ | v • Reply • Share ›



Sara • 17 days ago

ON Network tab in Chrome DevTools I see that X-Frame-Options:SAMEORIGIN has been set. But I still get the error 500 and on error logs I got this message:

```
couldn't perform authentication. AuthType not set!: /error/HTTP_INTERNAL_SERVER_ERROR.html.var
couldn't perform authentication. AuthType not set!: /
configuration error: couldn't perform authentication. AuthType not set!:
/error/HTTP_INTERNAL_SERVER_ERROR.html.var
Response header name 'X-Frame-Options:' contains invalid characters, aborting request
```

I am using Apache/2.2.34 (Linux/SUSE)

Could you please help me with that.

^ | v • Reply • Share ›

ALSO ON KEYCDN BLOG

KeyCDN Launches New POP in India

3 comments • 2 months ago

Brian Jackson — Keep up the great work guys!

KeyCDN Launches New POP in South Africa

2 comments • 3 months ago

James Peters — That is great success KeyCDN for with a new point of presence (POP) in Johannesburg, South Africa. now access to city ...

Popular HTTP Headers for Enhancing Performance



1 comment • 16 days ago

Sarang — any configurations for varnish on dreampress managed hosting?

KeyCDN Launches New POP in Austria

2 comments • a month ago

Web Design Toronto — Wow,Nice information. Thanks to inform about this Launches.

✉ [Subscribe](#)  [Add Disqus to your site](#)[Add Disqus](#) [Add](#)  [Privacy](#)

Product

[Features \(/features\)](/features)

[Network \(/network\)](/network)

[Benefits \(/benefits\)](/benefits)

[Pricing \(/pricing\)](/pricing)

[Sign Up \(/signup\)](/signup)

[Login \(/login\)](/login)

Company

[About Us \(/about\)](/about)

[Careers \(/careers\)](/careers)

[Blog \(/blog\)](#)

[Affiliate \(/affiliate\)](#)

[Contact \(/contacts\)](#)

[Legal \(/legal\)](#)

Support

[Knowledge Base \(/support\)](#)

[Network Status \(https://status.keycdn.com\)](https://status.keycdn.com)

[Community \(https://community.keycdn.com/\)](https://community.keycdn.com/)

[FAQ \(/faq\)](#)

[Tools \(https://tools.keycdn.com\)](https://tools.keycdn.com)

[Open Source \(/open-source-cdn\)](#)

Solutions

[Website Performance \(/website-performance\)](#)

[Software Distribution \(/software-distribution\)](#)

[Game & App Delivery \(/game-app-delivery\)](#)

[CDN Hosting \(/cdn-hosting\)](#)


[Video CDN \(/video-cdn\)](#)


[Ad Serving \(/ad-serving\)](#)


Connect

 [LinkedIn \(https://www.linkedin.com/company/keycdn\)](https://www.linkedin.com/company/keycdn)

 [Facebook \(https://www.facebook.com/keycdn\)](https://www.facebook.com/keycdn)

 [Twitter \(https://twitter.com/keycdn\)](https://twitter.com/keycdn)

 [Google+ \(https://plus.google.com/+Keycdn/posts\)](https://plus.google.com/+Keycdn/posts)

 [GitHub \(https://github.com/keycdn\)](https://github.com/keycdn)



Made in Switzerland

| © proinity LLC 2018.