

How to Fix “This Page Won’t Open in a Frame” in Firefox, Chrome, and Safari

Why Firefox, Chrome, and Safari Refuse to Load Some Pages in an iFrame

If you have ever tried to embed a website inside an iframe and gotten a browser error instead, you are not alone.

Firefox is often the most obvious about it. It may show a message like this:

Firefox Can't Open This Page

To protect your security, this site will not allow Firefox to display the page if another site has embedded it.

Chrome is usually less friendly about it and tends to throw a developer-console style error about the page refusing to display in a frame. Safari can do the same thing, often with less explanation on the screen and more clues in the browser console. The root cause is usually the same across all three browsers: the page is sending headers that block embedding. Mozilla's support docs point to `X-Frame-Options` or Content Security Policy as the reason Firefox blocks framed pages, and MDN says these headers are used to prevent clickjacking and related risks.

What causes this problem?

Most of the time, one of these headers is the reason:

Copy

```
X-Frame-Options: DENY
```

or:

Copy

```
X-Frame-Options: SAMEORIGIN
```

or:

Copy

```
Content-Security-Policy: frame-ancestors 'self';
```

`X-Frame-Options` is the older header. MDN says it controls whether a browser may render a page inside a `frame`, `iframe`, `embed`, or `object`. `DENY` blocks all framing. `SAMEORIGIN` allows framing only when the parent page is on the same origin. MDN also notes that for more comprehensive options, you should use CSP's `frame-ancestors` directive.

The reason these headers exist is security. CSP's `frame-ancestors` directive is specifically used to control which documents may embed the page in an `iframe`, which helps **protect against clickjacking**. Google's web.dev guidance also recommends using these headers to prevent malicious embedding, while noting that CSP gives you more granular control.

Why Firefox, Chrome, and Safari All Behave a Little Differently

The browsers are solving the same problem, but they do not always present it the same way.

- **Firefox** often shows a visible browser message explaining that the site will not allow itself to be displayed when embedded. Mozilla's support documentation explicitly ties that message to frame-blocking security settings.
- **Chrome** usually reports the issue in the console with a "refused to display in a frame" style error when the page sets `X-Frame-Options` or equivalent restrictions. This is consistent with how Chrome enforces frame restrictions described in MDN and web.dev documentation.
- **Safari** also enforces framing restrictions, but the details are often less obvious unless you inspect headers or the console. Compatibility data shows Safari supports `frame-ancestors`, but older Safari versions did not.

Do Chrome and Safari Support the Modern Fix?

Yes. Current compatibility tables show support for `Content-Security-Policy: frame-ancestors` in:

- **Chrome:** supported from version 40+
- **Firefox:** full support from version 58+
- **Safari:** supported from version 10+
- **iOS Safari:** supported from version 9.3+

So if you are targeting normal modern browsers, `frame-ancestors` is the right tool for this job.

The Mistake People Make is...

A lot of people assume this is an iframe problem.

It usually is not.

The iframe is just where the problem shows up. The real issue is that the **embedded page itself** is telling browsers not to allow framing, or to allow it only under certain conditions. That means you usually do **not** fix this in the parent page. You fix it in the headers sent by the page being embedded.

The Best Fix Today

If you want your page to be embeddable only on specific sites, use `Content-Security-Policy: frame-ancestors`.

Example:

Copy

```
Content-Security-Policy: frame-ancestors 'self' https://example.com https://www.example.com;
```

MDN's examples show this exact pattern: you can allow your own site plus one or more specific domains. This is why `frame-ancestors` is better than `X-Frame-Options` for selective framing.

Also, do a check over at my site, NoIndexChecker.com to verify your change worked.

What to do with X-Frame-Options

If you leave `X-Frame-Options: SAMEORIGIN` or `DENY` in place, it can still block the page even if you add a friendlier CSP rule. MDN describes `X-Frame-Options` as the older option and notes that `ALLOW-FROM` is deprecated. In practice, if you need specific third-party sites to embed your page, you generally want to remove the blocking `X-Frame-Options` header and use `frame-ancestors` instead.

In other words:

- If you want to block framing everywhere, use `DENY` or `frame-ancestors 'none'`
- If you want same-site framing only, use `SAMEORIGIN` or `frame-ancestors 'self'`
- If you want to allow a specific outside domain, use `frame-ancestors`

Apache .htaccess Example

If your site uses Apache, this is a common fix:

Copy

```
<IfModule mod_headers.c>
  Header always unset X-Frame-Options
  Header always set Content-Security-Policy "frame-ancestors 'self' https://example.com
https://www.example.com;"
</IfModule>
```

This removes the old frame-blocking header and replaces it with a more precise allowlist.

PHP Example

Copy

```
header_remove('X-Frame-Options');
header("Content-Security-Policy: frame-ancestors 'self' https://example.com
https://www.example.com;");
```

This is helpful when headers are being set at the application level instead of the web server level.

Where to Check for the Problem

If framing is being blocked, check these places:

- .htaccess
- Apache vhost config
- NGINX config
- PHP files or middleware sending headers
- CDN or security platform rules

MDN also points out one common gotcha: setting `X-Frame-Options` in a meta tag does nothing. It only works as an HTTP response header.

Important: When You Should Not Change This

Be careful here.

These headers are not annoying browser quirks. They are there to prevent clickjacking and related framing attacks. If a page includes sensitive actions like logins, payments, account changes, or admin tools, you usually do **not** want it embeddable on other sites. MDN and Google's security guidance both tie these protections directly to clickjacking defense.

So the smart move is not “remove the protection everywhere.”

The smart move is: **Allow framing only on the exact domain or domains that need it.**

How to Test the Fix

1. Open browser dev tools
2. Load the page directly and inspect the response headers
3. Look for `X-Frame-Options` and `Content-Security-Policy`
4. Embed the page again in the iframe
5. Confirm the browser no longer blocks it

If it still fails, there is a good chance the old header is still being injected somewhere else, such as your hosting layer, a reverse proxy, or a security plugin.

Key Takeaways:

- If Firefox says a page cannot be opened because another site embedded it, the site is probably sending frame-blocking security headers
- The old header is `X-Frame-Options`
- The better modern fix is `Content-Security-Policy: frame-ancestors`
- Chrome, Firefox, and Safari all support `frame-ancestors` in modern versions
- If you need selective embedding, remove the blocking `X-Frame-Options` header and use a specific `frame-ancestors allowlist`
- Do not loosen this globally unless you are sure the page is safe to be framed

Original article: <https://tonyherman.com/fix-page-wont-open-in-frame-firefox-chrome-safari/>

PDF version: https://tonyherman.com/fix-page-wont-open-in-frame-firefox-chrome-safari/?apa_pdf=1

Special Offer for Readers

1,000+ Channels • On-Demand Movies • **5 Devices**

\$69.99/mo (~\$2.33/day)

Try it for a month and see if you like it, then switch.

Start Your Trial

Click to learn more about CUE Broadcast

Tip: Get 3 friends or family to sign up and you get it for free.