C

# Clickjacking: UI Redress Attacks, Testing, and Prevention

Clickjacking tricks a user into interacting with a framed page they cannot properly see. Learn its impact, testing approach, and reliable browser defenses.

PUBLISHER
Gridinsoft Support

UPDATED
Oct 8, 2026

READ TIME
4 min

CONTENTS

## In this guide

01 How clickjacking works
02 Possible impact
03 Clickjacking vs. XSS and CSRF
04 Primary server-side defenses
05 Why JavaScript frame-busting is weaker
06 Safe testing workflow
07 Responding to a finding
08 User precautions
09 Frequently asked questions

**Quick answer:** Clickjacking, also called UI redress, tricks a person into clicking or interacting with a trusted page that is hidden, transparent, or visually disguised inside an attacker-controlled interface. The primary defense is to control which origins may frame sensitive pages using the Content Security Policy `frame-ancestors` directive, with `X-Frame-Options` as a legacy defense-in-depth header.

## 01 How clickjacking works

An attacker places the target application in an iframe and positions its real button or control beneath a convincing decoy. CSS can make the target nearly invisible while preserving clicks. The victim believes they are starting a video or closing a notice but actually changes a setting, grants permission, authorizes an action, or follows a link on the framed site.

The browser is not necessarily violating the same-origin policy: it may allow one site to display another without allowing the parent to read the framed page. The attack abuses the user's input and existing authenticated session instead.

## 02 Possible impact

- changing account, privacy, or security settings;
- authorizing a transaction or connecting an application;
- enabling a camera, microphone, notification, or other browser permission through deceptive steps;
- liking, following, sharing, or deleting content;
- dragging data into or out of a framed interface in variants that permit it.

Impact depends on what the victim can do without reauthentication, confirmation, or additional trusted context. A frameable public page is not automatically vulnerable if it contains no sensitive interaction, but the business need for framing should still be explicit.

## 03 Clickjacking vs. XSS and CSRF

Clickjacking manipulates the visible user interface. Cross-site scripting injects executable content into a trusted origin. Cross-site request forgery causes an unwanted authenticated request, often without relying on a hidden interface. The defenses overlap in places but are not interchangeable.

SameSite cookies can limit authenticated framing in some cases, but they do not protect actions that work without cookies and should not be the sole clickjacking control.

## 04 Primary server-side defenses

1. **Set CSP `frame-ancestors`.** Use `'none'` when no framing is needed, `'self'` for same-origin embedding, or a narrow list of approved origins.
2. **Send X-Frame-Options for legacy coverage.** `DENY` blocks framing and `SAMEORIGIN` permits same-origin framing. The obsolete `ALLOW-FROM` value is not a reliable modern solution.
3. **Use SameSite cookies.** `Lax` or `Strict` can add protection for authenticated scenarios, subject to application needs.
4. **Add transaction safeguards.** Reauthentication, clear review screens, and independent confirmation reduce the impact of deceptive clicks.

OWASP's Clickjacking Defense Cheat Sheet documents the header choices and their limitations. Deliver `frame-ancestors` in the HTTP CSP header; it is not supported through a CSP meta tag for this purpose.

## 05 Why JavaScript frame-busting is weaker

Scripts that compare `top` and `self` or redirect the parent can be bypassed, disabled, delayed, or broken by sandboxing and nested frames. They may help certain legacy applications, but response headers are the reliable policy mechanism. Do not copy an old one-line "frame killer" and consider the problem solved.

## 06 Safe testing workflow

1. Test only applications you own or are authorized to assess.
2. Inventory sensitive routes and legitimate embedding requirements, including alternate domains, mobile pages, and regional hosts.
3. Inspect the final response headers after CDNs, proxies, and caches; intermediaries may strip or override them.
4. Attempt framing from an unrelated test origin using a harmless page and non-destructive account.
5. Confirm both framing behavior and whether cookies or sensitive actions remain available.
6. Retest approved partner embeds so a strict fix does not break required workflows.

Apply protection to every sensitive HTML response, including error pages and legacy endpoints. APIs that return data but no interactive HTML have different risks and generally do not need to be framed.

## 07 Responding to a finding

Define the intended framing policy, deploy CSP in report-only mode if compatibility is uncertain, review reports, then enforce. Add X-Frame-Options where its semantics match. If exploitation is suspected, review sensitive actions, session activity, referrers where available, and reports from affected users. Header deployment prevents future framing but does not reverse an action already authorized.

## 08 User precautions

Users have limited visibility into transparent frames. Keep browsers updated, avoid acting on unexpected pages, and open sensitive services directly rather than through embedded prompts. Organizations should not rely on user awareness as the primary defense because the deceptive layer is designed to look normal.

## 09 Frequently asked questions

### Does CSP default-src block clickjacking?

No. Framing by other sites is controlled by `frame-ancestors`; `frame-src` controls what the protected page itself may embed.

### Should every site use frame-ancestors 'none'?

Use it when framing is unnecessary. Sites embedded by trusted partners need a carefully scoped allow-list and compatibility testing.

### Does SameSite eliminate clickjacking?

No. It can withhold cookies in some framed requests, but unauthenticated actions and other state mechanisms may remain.

**NEED MORE HELP?**

**We are here when the guide is not enough.**

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

**Open this guide online at support.gridinsoft.com**