CS-060 · Application Security

Cross-Site Request Forgery (CSRF)

An attack that tricks a logged-in user's browser into making unintended requests.

Executive Summary

Cross-site request forgery, or CSRF, is an attack that causes a logged-in user's browser to send a request the user did not intend, using the trust the application places in that user's session. If the application accepts the request as legitimate, the attacker can perform actions such as changing settings or moving funds. The core defense is requiring an unpredictable anti-CSRF token and using SameSite cookies.

What It Is

CSRF exploits the way browsers automatically attach a user's cookies to requests sent to a site. If a user is logged in to an application and then visits a malicious page, that page can quietly instruct the browser to send a request to the application. Because the browser includes the user's session cookie, the application may treat the forged request as a genuine action by the user. CSRF targets state-changing actions such as updating an account, making a purchase, or changing a password. It does not let the attacker read the response, but it can cause real changes on the user's behalf without their knowledge.

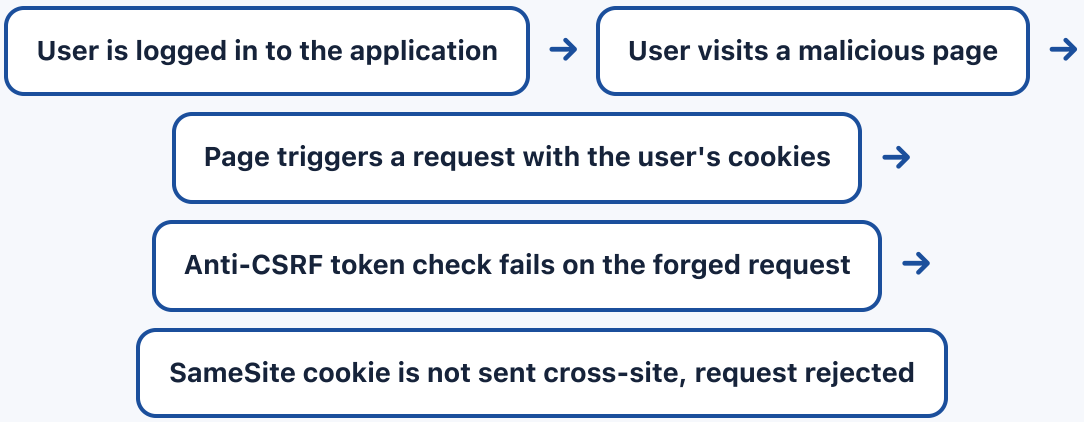
Why It Matters

CSRF turns a user's own authenticated session into a weapon against them. A successful attack can change account details, transfer money, alter permissions, or perform any state-changing action the user is allowed to take, all without the user realizing it. Because the request looks like it came from the legitimate user, basic logging may not reveal anything unusual. The good news is that CSRF is well understood and reliably preventable, so an exploitable CSRF flaw on a sensitive action is treated as a clear gap in application security practice.

How It Works

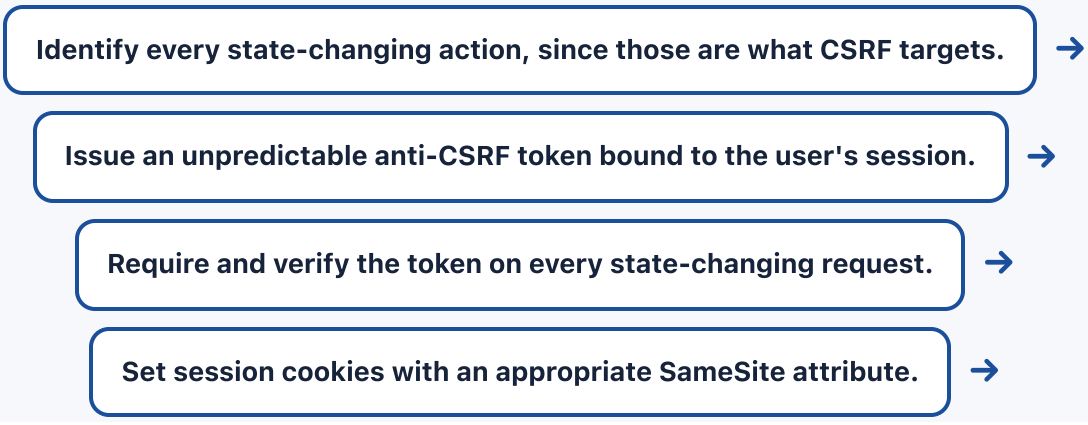
At a conceptual level, CSRF works because the application cannot tell whether a valid-looking request was actually initiated by the user or forged by another site. The main defense breaks that ambiguity with an anti-CSRF token: the server issues a secret, unpredictable value tied to the user's session and requires it on every state-changing request. A malicious site cannot read or guess this token, so its forged requests are rejected. SameSite cookie settings add a strong layer by telling the browser not to send session cookies on cross-site requests, which blocks the mechanism CSRF relies on. Requiring a valid custom header for API requests and checking the request origin provide additional defense. Read-only requests should never cause state changes, since that removes a whole avenue of abuse.

Architecture Diagram



CSRF rides a logged-in session from another site; an unpredictable token and SameSite cookies break the forged request.

Visual Workflow



For APIs, require a custom header and validate the request origin. →

Ensure read-only requests never change state, then test and re-test the defenses.

Common Attacks

- A malicious page silently submitting a form to a logged-in application
- A crafted link or image that triggers a state-changing request
- Changing account settings, email, or password on the victim's behalf
- Initiating transactions or transfers using the victim's session
- Abusing actions that lack a token check or rely only on cookies for trust

Common Mistakes

- Assuming a session cookie alone is enough to prove a request is intentional
- Protecting only some state-changing actions and missing others
- Using predictable or reusable tokens that an attacker could guess
- Allowing read-only requests, such as simple links, to change state
- Ignoring SameSite cookie settings that would block cross-site sending

Best Practices

- Require an unpredictable, session-bound anti-CSRF token on state-changing actions
- Set session cookies with a SameSite attribute to block cross-site sending
- For APIs, require a custom header and validate the request origin
- Never allow read-only requests to perform state changes
- Use framework CSRF protection consistently rather than custom partial fixes
- Re-authenticate or confirm for especially sensitive actions

Quick Checklist

- [x] Anti-CSRF token required and verified on every state-changing action

- [x] Tokens are unpredictable and bound to the user's session
- [x] Session cookies set with an appropriate SameSite attribute
- [x] APIs validate origin and require a custom header
- [x] No state changes possible through read-only requests
- [x] CSRF protection tested across all sensitive actions before release

Recommended Tools

Framework CSRF protection
Built-in token generation and validation applied consistently

Dynamic application security testing (DAST)
Tests whether state-changing actions accept forged cross-site requests

SameSite cookie configuration
Instructs the browser not to send session cookies cross-site

Manual test proxy
Lets a reviewer replay and modify requests to verify token enforcement

Industry Standards

OWASP Top 10
CSRF risk relates to broken access control and session handling categories

OWASP Application Security Verification Standard (ASVS)
Testable requirements for anti-forgery protections on state changes

CWE-352
The Common Weakness Enumeration entry for cross-site request forgery

Career Relevance

CSRF is standard knowledge for application security engineers, penetration testers, and secure code reviewers, who verify token enforcement on every sensitive action. Developers are

expected to enable framework CSRF protection correctly and to set SameSite cookies, and GRC and audit teams flag missing protections on state-changing endpoints. For the AI-Governance-Jobs.com audience, explaining CSRF and its defenses clearly is a common interview task.

Interview Questions

- What is CSRF, and why does the browser make it possible?
- How does an anti-CSRF token prevent a forged request?
- How do SameSite cookies help defend against CSRF?
- Why should read-only requests never perform state changes?
- How does CSRF differ from XSS in what the attacker can do?

Related Certifications

GIAC Web Application Penetration Tester (GWAPT) | Offensive Security Web Assessor (OSWA) | CompTIA Security+

Further Reading

- OWASP: Cross-Site Request Forgery
- CWE-352: Cross-Site Request Forgery
- OWASP Top 10

Key Takeaways

- CSRF makes a logged-in user's browser send an unintended request.
- It abuses the browser automatically attaching session cookies.
- It targets state-changing actions, not reading data.
- Unpredictable anti-CSRF tokens are the primary defense.
- SameSite cookies and origin checks add strong extra layers.

FAQ

▶ **How is CSRF different from XSS?**

▶ **Do SameSite cookies remove the need for tokens?**

▶ **Which requests need CSRF protection?**

Related Careers

RELATED ROLES

- Application Security Engineer
- Penetration Tester
- Secure Code Reviewer
- Security Engineer

RELATED CERTIFICATIONS

- GIAC Web Application Penetration Tester (GWAPT)
- Offensive Security Web Assessor (OSWA)
- CompTIA Security+

CURRENT OPENINGS

Live roles are on the job board.

Browse all jobs

GUIDES & SALARY

- Career guides
- Role roadmaps
- Salary data
- Career tools

SUGGESTED LEARNING PATH

1. Ground the basics with CS-001 Cybersecurity
2. Study this sheet: **Cross-Site Request Forgery (CSRF)**
3. Go deeper: OWASP Top 10
4. Go deeper: Cross-Site Scripting (XSS)
5. Validate it: work toward **GIAC Web Application Penetration Tester (GWAPT)**
6. Find the role: browse current openings

RELATED SHEETS

CS-057 OWASP Top 10 | CS-059 Cross-Site Scripting (XSS) | CS-062 Authentication | CS-063 Authorization | CS-001 Cybersecurity