CSRF (Cross-Site Request Forgery)

---

1. What problem is CSRF solving?

CSRF is **not a hacking of passwords**.
It's an **abuse of trust**.

CSRF happens when a browser automatically sends authentication cookies to a server **without the user realizing it**, and an attacker exploits that.

---

2. The core rule browsers follow (this causes CSRF)

Browsers **automatically attach cookies** to requests.

Example:

- You log in to `bank.com`
- Browser stores:
- `sessionId=abc123`
- For **every request to bank.com**, browser sends:
- `Cookie: sessionId=abc123`

☞ Even if the request came from **another website**.

This is where CSRF begins.

---

3. A simple CSRF attack

Scenario

- You are logged into `bank.com`
- You visit `evil.com` in another tab

On `evil.com`, attacker places:

```
<img src="https://bank.com/transfer?to=attacker&amount=10000" />
```

What happens?

1. Browser sees a request to `bank.com`
2. Browser automatically attaches cookies

3. `bank.com` receives:
4. `transfer money request`
5. `+ valid session cookie`
6. Bank thinks **YOU made the request**

✹ Money transferred.

You never clicked anything.

---

4. Why CSRF works

CSRF works because:

Condition	True?
Browser auto-sends cookies	✓
Server trusts cookies	✓
Server doesn't verify request origin	✗

So CSRF is a **server-side validation failure**, not a browser bug.

---

5. When CSRF is POSSIBLE

CSRF **only applies** when:

Authentication method	CSRF possible?
Cookie-based sessions	✓ YES
Cookie-based JWT	✓ YES
JWT in Authorization header	✗ NO
OAuth + session	✓ YES

If auth is sent **automatically**, CSRF is possible.

---

6. When CSRF is NOT possible

JWT in Authorization header

```
Authorization: Bearer eyJhbGciOi...
```

Why safe?

- Browser does **not** auto-attach this

- JavaScript must explicitly add it

Attacker **cannot force** the browser to send it.

So:

Header-based auth ≠ CSRF

---

7. The CSRF defense strategy (core idea)

“Prove that the request was intentionally made by my app.”

This is done using a **CSRF token**.

---

8. What is a CSRF token?

A CSRF token is:

- A **random, unpredictable value**
- Generated by the server
- Tied to the user’s session

Example:

```
csrfToken: "8f9a12c7e4..."
```

---

9. How CSRF protection works (step-by-step)

Step 1: User loads the app

Server generates:

```
sessionId = abc123
csrfToken = xyz789
```

Step 2: Server sends CSRF token to client

Via:

- HTML form
- API response
- Cookie / header

Step 3: Client sends token with every state-changing request

```
POST /transfer
Cookie: sessionId=abc123
X-CSRF-Token: xyz789
```

Step 4: Server verifies

```
if (req.csrfToken === session.csrfToken) {
  allowRequest();
} else {
  reject();
}
```

---

10. Why attacker cannot fake CSRF token

- Attacker's site **cannot read your CSRF token**
- Same-Origin Policy blocks it
- They can trigger requests
- They cannot attach the correct token

That's the security win.

---

11. CSRF in Express (practical view)

Typical stack

- `express-session`
- `csurf` middleware

Flow:

```
Request → Session middleware → CSRF middleware → Route
```

Example (conceptual)

```
app.use(csrf());

app.get("/csrf-token", (req, res) => {
  res.json({ csrfToken: req.csrfToken() });
});

app.post("/transfer", csrfProtection, (req, res) => {
  res.send("Transfer successful");
```

```
});
```

Client must send token in:

```
X-CSRF-Token: <token>
```

---

12. CSRF + OAuth

OAuth **does not remove CSRF risk**.

Why?

- After OAuth login, you still use **sessions**
- Sessions still use **cookies**
- Cookies still auto-send

So:

OAuth + sessions **REQUIRES CSRF protection**

---

13. CSRF vs XSS (common confusion)

Topic	CSRF	XSS
Attacker site	Different origin	Same origin
Cookie access	No	Yes
CSRF token protection	✅	❌
HttpOnly cookies help	❌	✅

👉 CSRF tokens don’t stop XSS
👉 HttpOnly cookies don’t stop CSRF

They solve **different problems**.

---

14. Modern best practices (expert level)

✓ Always protect:

- POST
- PUT
- PATCH
- DELETE

❌ Never trust:

- Cookies alone
- Referer header alone

✓ Combine:

- CSRF tokens
- SameSite cookies
- HttpOnly cookies

---

15. One-line mental model

CSRF exists because browsers automatically send cookies; CSRF tokens prove the request was intentionally made by the real app.

---

16. Quick interview answers

Q: Is CSRF needed with JWT?
☞ Only if JWT is stored in cookies.

Q: Does OAuth remove CSRF?
☞ No, OAuth + session still needs CSRF.

Q: Can SameSite cookies replace CSRF tokens?
☞ They reduce risk but do not fully replace tokens.