CSRF notes

Eyad Islam El-Taher

October 11, 2025

Introduction

CSRF (Cross-Site Request Forgery) $\Longrightarrow$ A web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. It allows an attacker to partly circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.

CSRF vulnerabilities happen when

CSRF vulnerabilities happen when a web application relies solely on the automatic submission of a user's credentials (like session cookies) to authenticate a request, without requiring any other proof that the user actually intended to perform that action.

Key conditions that must be present for a CSRF attack to be possible:

- **Cookie-Based Session Handling:** The application uses session cookies to identify the user. The browser automatically sends these cookies with every request to the domain.
- **No Unpredictable Tokens:** The application does not use CSRF tokens (unique, secret, and unpredictable values) to validate that the request came from a legitimate source on its own site. (CSRF tokens can be manipulate)
- **No Additional Confirmation:** The application does not require re-authentication (e.g., password) for sensitive actions.
- **Predictable Request Parameters:** The attacker can reliably guess the parameters needed for a state-changing request (e.g., changing an email, transferring funds).

CSRF vulnerabilities happen where

CSRF vulnerabilities happen in the web application's server-side code, specifically in its handling of state-changing HTTP requests.

- **User Account Settings:** Changing a password, updating an email address.
- **E-commerce Functions:** Adding items to a cart, applying discount coupons, making a purchase.
- **Financial Transactions:** Transferring funds between accounts.
- **Social Media & Content:** Posting a status, sending a message, liking a post.
- **Admin Functions:** Promoting a user to an admin, deleting user accounts.

How CSRF vulnerabilities effect on users

- **Unauthorized Financial Transactions:** An attacker could force a user to transfer money out of their bank or brokerage account without their knowledge.
- **Account Takeover:** An attacker can change the victim's email address and/or password, effectively locking them out of their own account and giving the attacker full control.
- **Data Theft and Privacy Breach:** While CSRF doesn't typically steal data directly, an attacker could change privacy settings to expose private data or use it as a stepping stone for further attacks.

- **Reputational Damage and Social Embarrassment:** An attacker could force a user to post embarrassing status updates, send offensive messages to friends, or delete important content from their social media profiles.
- **Fraudulent Purchases:** On an e-commerce site, an attacker could make the victim purchase items to be shipped to the attacker's address.

How CSRF Works:

- **Step 1:** The victim logs into a trusted site (e.g., good-website.com), which authenticates them and sets a session cookie in their browser.
- **Step 2:** The victim, in a different tab, visits a malicious site created by the attacker (evil-website.com).
- **Step 3:** The malicious site contains hidden HTML that automatically sends a request to the trusted site (good-website.com). This is the forged request.
- **Step 4:** The victim's browser, being loyal, sees a request going to good-website.com and automatically attaches the valid session cookie from Step 1.
- **Step 5:** The trusted site (good-website.com) receives the request with a valid session cookie. It thinks, "This is a legitimate request from my logged-in user!" and processes it.
- **Step 6:** The action is completed without the victim's knowledge or consent.

How to construct a CSRF attack:

The easiest way to construct a CSRF exploit is using the CSRF PoC generator that is built in to Burp Suite Professional:

- **Step 1:** Select a request anywhere in Burp Suite Professional that you want to test or exploit.
- **Step 2:** From the right-click context menu, select Engagement tools / Generate CSRF PoC.
- **Step 3:** Burp Suite will generate some HTML that will trigger the selected request (minus cookies, which will be added automatically by the victim's browser).
- **Step 4:** You can tweak various options in the CSRF PoC generator to fine-tune aspects of the attack. You might need to do this in some unusual situations to deal with quirky features of requests.
- **Step 5:** Copy the generated HTML into a web page, view it in a browser that is logged in to the vulnerable website, and test whether the intended request is issued successfully and the desired action occurs.

How to deliver a CSRF exploit:

The delivery mechanisms for cross-site request forgery attacks are essentially the same as for reflected XSS. Typically, the attacker will place the malicious HTML onto a website that they control, and then induce victims to visit that website. This might be done by feeding the user a link to the website, via an email or social media message. Or if the attack is placed into a popular website (for example, in a user comment), they might just wait for users to visit the website.

Note that some simple CSRF exploits employ the GET method and can be fully self-contained with a single URL on the vulnerable website. In this situation, the attacker may not need to employ an external site, and can directly feed victims a malicious URL on the vulnerable domain.

Common defences against CSRF:

- CSRF tokens
- SameSite cookies
- Referer-based validation

CSRF tokens bypass

A CSRF token is a unique, secret, and unpredictable value that is generated by the server-side application and shared with the client. When attempting to perform a sensitive action, such as submitting a form, the client must include the correct CSRF token in the request. This makes it very difficult for an attacker to construct a valid request on behalf of the victim.

1. **If CSRF token depends on request method**
   Some applications correctly validate the token when the request uses the POST method but skip the validation when the GET method is used.
   - From the right-click context menu, select change request method $\Longrightarrow$ to switch between POST and GET methods
   - From the right-click context menu, select Engagement tools / Generate CSRF PoC.

SameSite cookies bypass

SameSite is a browser security mechanism that determines when a website's cookies are included in requests originating from other websites. As requests to perform sensitive actions typically require an authenticated session cookie, the appropriate SameSite restrictions may prevent an attacker from triggering these actions cross-site. Since 2021, Chrome enforces Lax SameSite restrictions by default. As this is the proposed standard, we expect other major browsers to adopt this behavior in future.

CSRF tokens bypass

Some applications make use of the HTTP Referer header to attempt to defend against CSRF attacks, normally by verifying that the request originated from the application's own domain. This is generally less effective than CSRF token validation.