Cross-Site Request Forgery (CSRF) Simulation and Mitigation

Understanding, Simulating, and Preventing CSRF Attacks

Group ID - ICS019

GROUP ID ICS019

01	Himasagar U. S20220010229
02	Uday Kiran S20220010155
03	Gnaneshwar S20220010151
04	Rahul Banoth S20220010031

PART

01

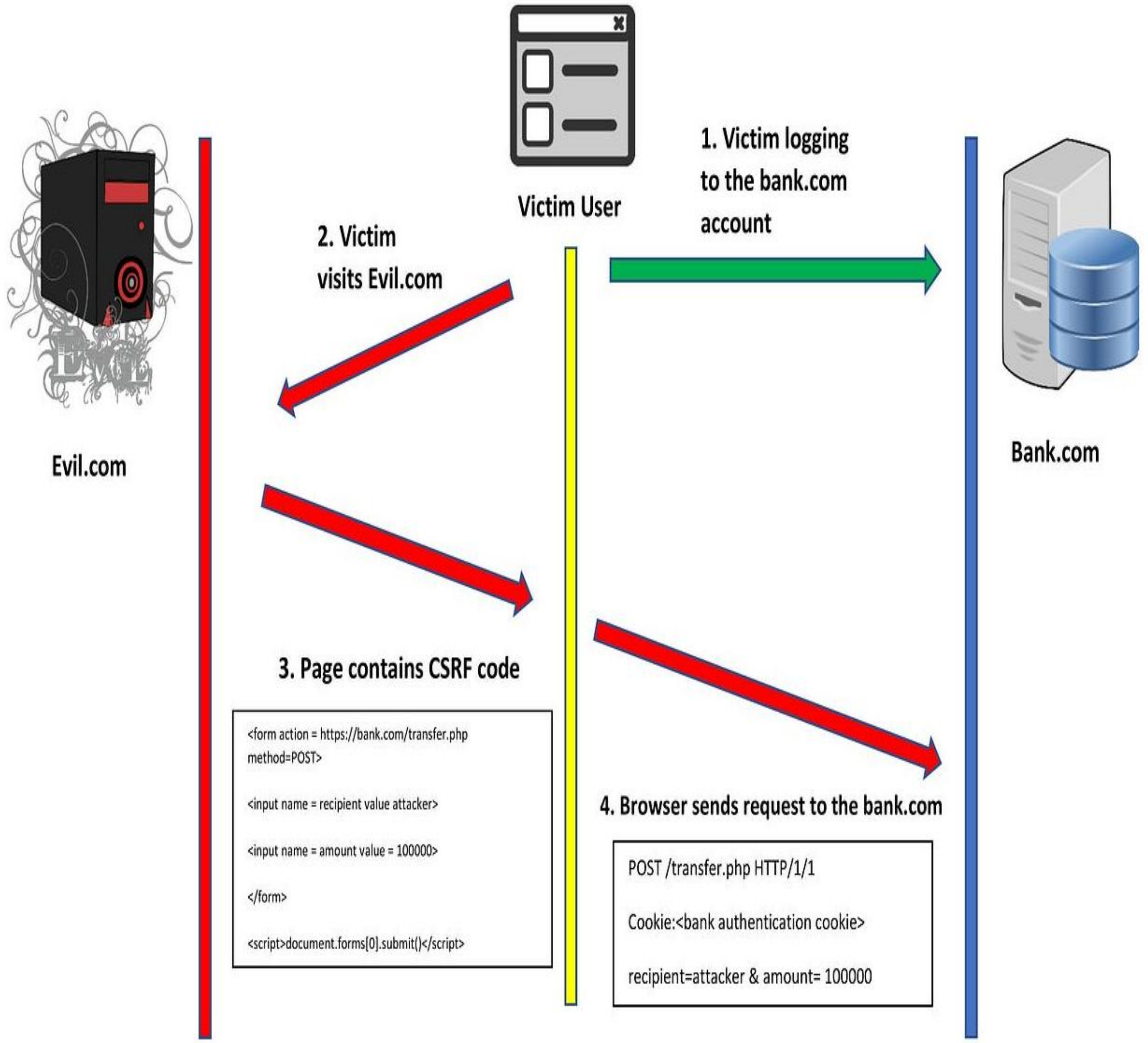
Presentation

CSRF Introduction →

What is CSRF - Cross Site Request Forgery

- Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

- With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing.


Victim User
1. Victim logging to the bank.com account
2. Victim visits Evil.com
Evil.com
Bank.com
3. Page contains CSRF code
4. Browser sends request to the bank.com

```
<form action = https://bank.com/transfer.php
method=POST>

<input name = recipient value attacker>

<input name = amount value = 100000>

</form>

<script>document.forms[0].submit()</script>
```

```
POST /transfer.php HTTP/1/1

Cookie:<bank authentication cookie>

recipient=attacker & amount= 100000
```

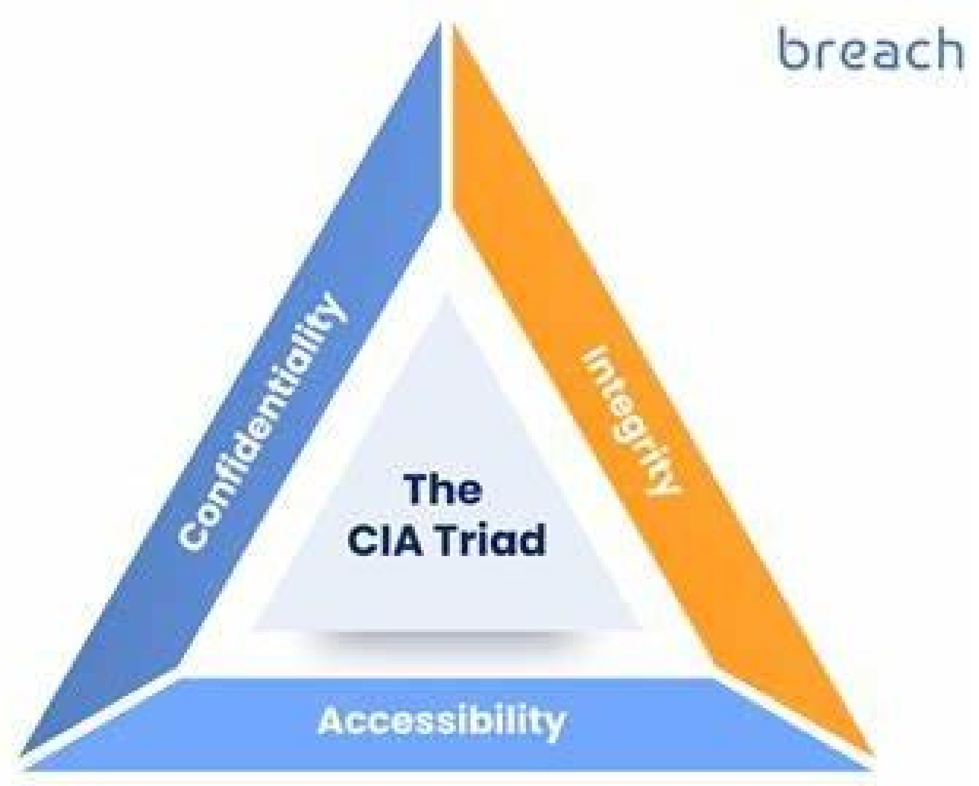
What is the impact of a CSRF attack?

- In a successful CSRF attack, the attacker causes the victim user to carry out an action unintentionally.

- For example, this might be to change the email address on their account, to change their password, or to make a funds transfer.

- Depending on the nature of the action, the attacker might be able to gain full control over the user's account.

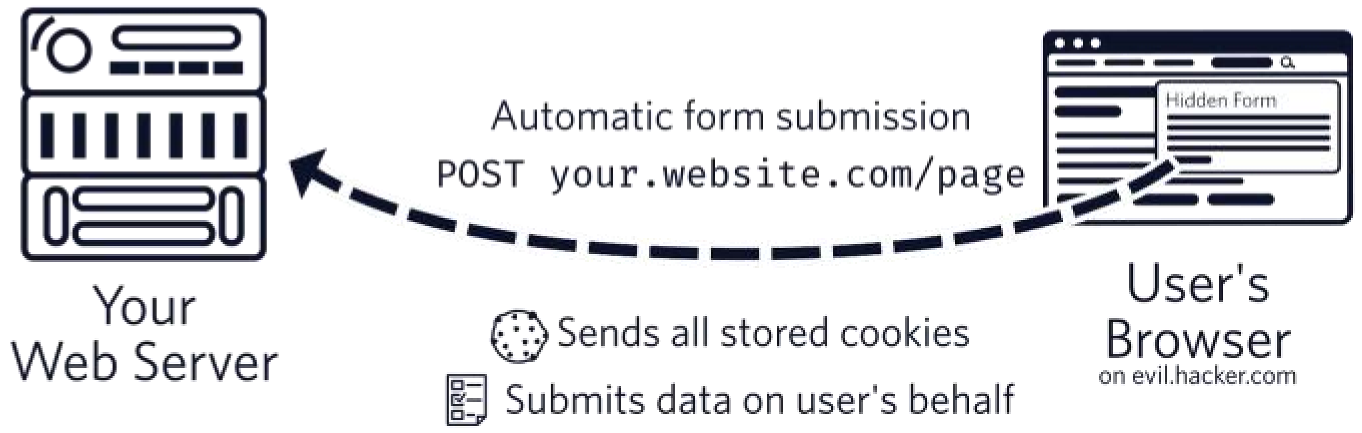
- If the compromised user has a privileged role within the application, then the attacker might be able to take full control of all the application's data and functionality.



How does CSRF work?

For a CSRF attack to be possible, three key conditions must be in place:

1. A relevant action.
   - ❖ For a CSRF attack to be meaningful, the attacker needs a valuable action they want to trigger. This action might change something on the server, like updating account details, transferring money, or adjusting settings. The attacker wants to force the authenticated user to execute this action without realizing it.



How does CSRF work?

2. Cookie-Based Session Handling

   - CSRF exploits session cookies, which browsers automatically send with each request to keep users authenticated. Attackers trick the browser into sending a malicious request using these cookies, and without a secondary check (like a unique token), the site trusts the request as legitimate.

3. No Unpredictable Request Parameters

   - For CSRF to succeed, the attacker must craft a request with predictable parameters, such as knowing the structure of a password-change request. An unpredictable parameter, like a random token, would block the attacker from forging a valid request.

How Cross Site Request Forgeries (CSRFs) Work

1. A hacker creates a request (in the form of a URL) for their own benefit from a website
2. Hacker embeds that request into a hyperlink and sends it to a visitor who they hope is logged in to the site
3. The website visitor clicks the link, unwittingly sending the request to the site
4. Assuming the request is legitimate, the website fulfills the request, sending data, funds, or access to the hacker

Some famous Examples of CSRF attacks

Netflix (2006)

- A CSRF vulnerability let attackers change users' account details like billing address and email via malicious links. Netflix patched it, highlighting the need for CSRF protection on sensitive actions.

YouTube (2008)

- Security researcher Mike Perry exploited CSRF to add videos to users' "favorites" by embedding malicious iframes. This showed CSRF's potential for manipulating user preferences on major platforms.

GitHub OAuth Issue (2012)

- A CSRF vulnerability in GitHub's OAuth flow allowed attackers to authorize applications on behalf of users without consent. GitHub fixed it by improving CSRF protections.

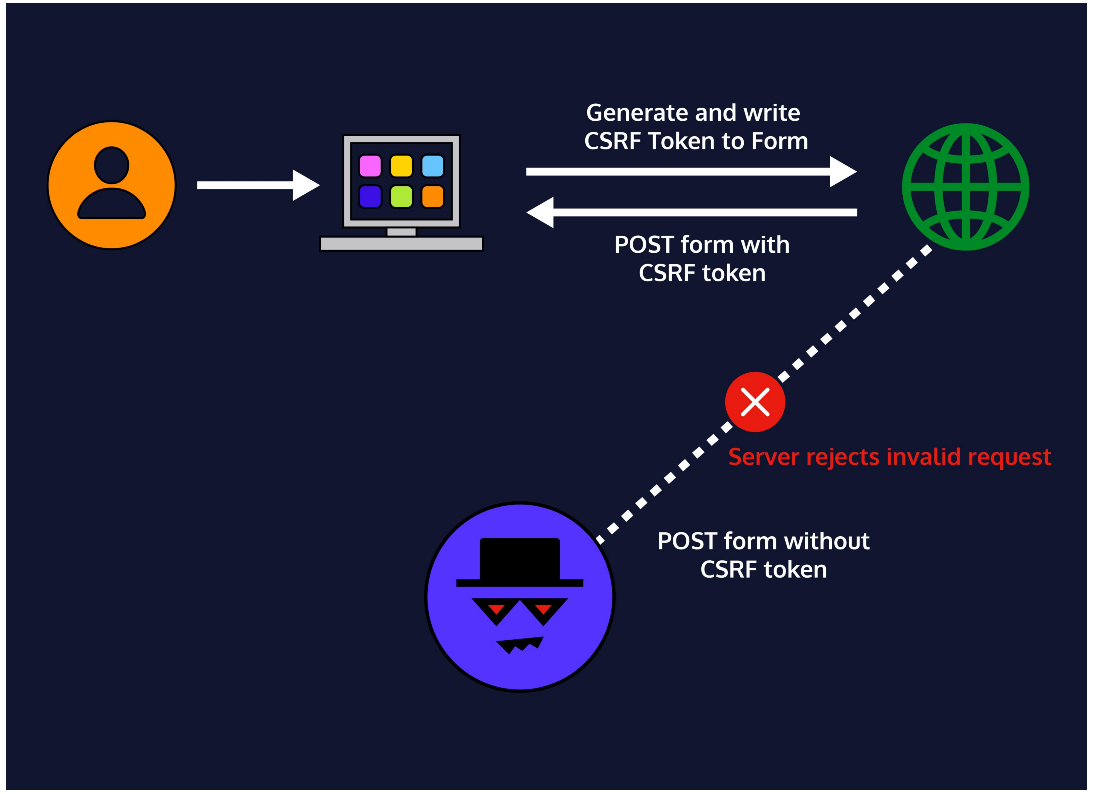
ING Bank (2013)

- A CSRF flaw enabled attackers to initiate unauthorized money transfers by tricking users into visiting malicious sites. This case emphasized the financial risks of CSRF in online banking.

Effective CSRF mitigating strategies

1. CSRF Tokens (Anti-CSRF Tokens)

- **What it is:** A CSRF token is a unique, unpredictable value generated for each user session or request and embedded within each form or URL that performs a sensitive action.
- **How it works:** When the user submits a request, the application checks that the CSRF token matches the expected value for that user. If it doesn't match, the request is denied.

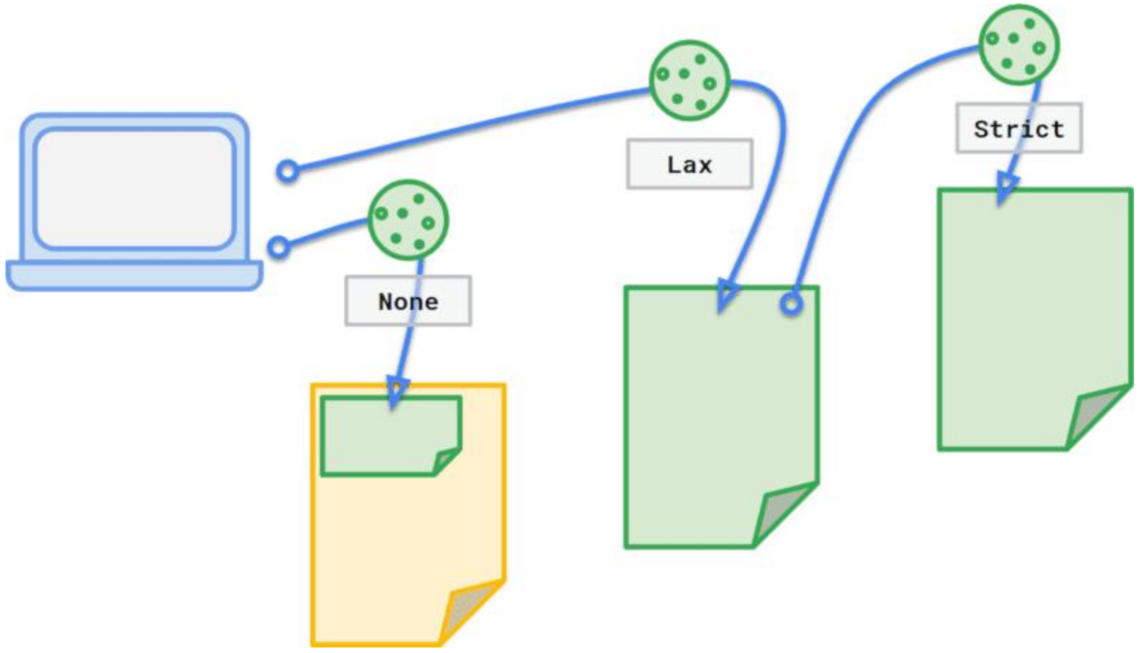

Generate and write CSRF Token to Form
POST form with CSRF token
Server rejects invalid request
POST form without CSRF token

Effective CSRF mitigating strategies

2. SameSite Cookies

- **What it is:** The SameSite attribute on cookies helps prevent browsers from sending cookies with cross-site requests, which mitigates CSRF risks.

- **How it works:** Setting SameSite=Strict or SameSite=Lax on session cookies can help prevent CSRF, as these settings restrict cookies from being sent with requests initiated by third-party sites.

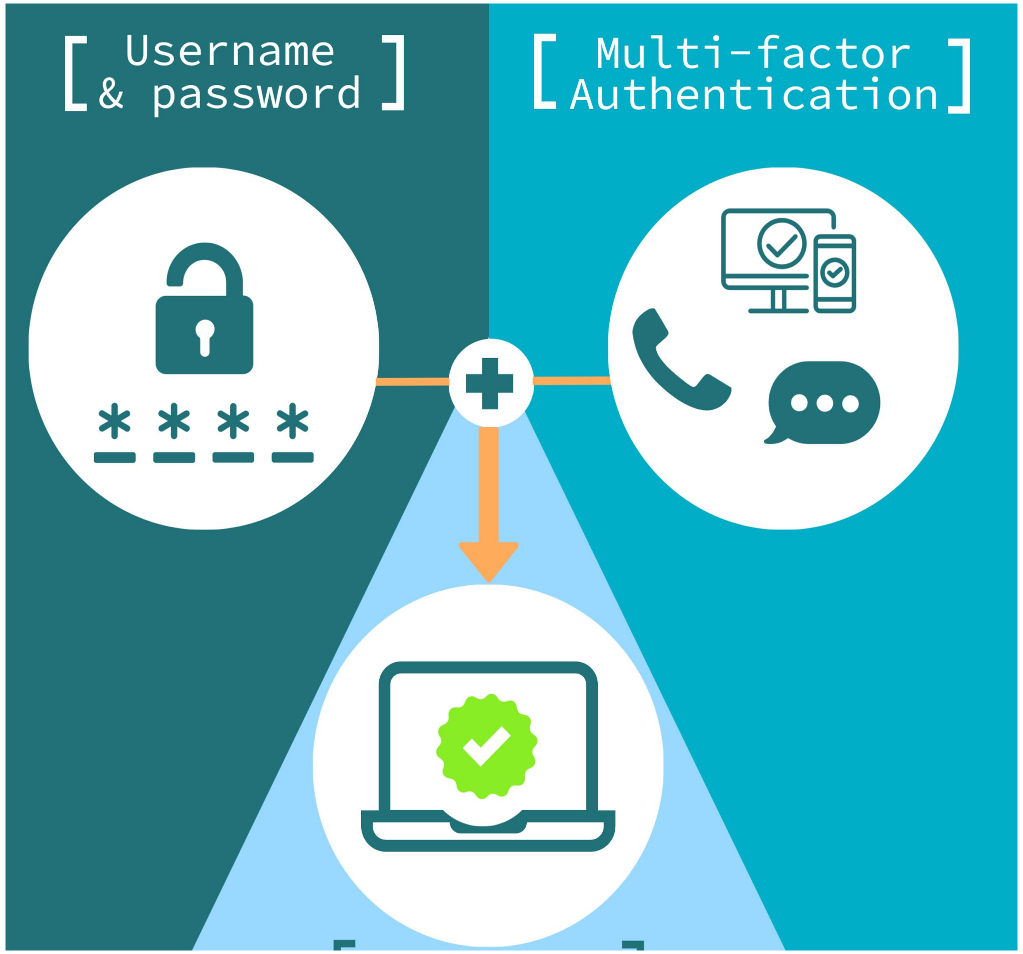


Effective CSRF mitigating strategies

3. User Re-Authentication or Multi-Factor Authentication (MFA) for Sensitive Actions

- **What it is:** This method requires users to re-authenticate before performing sensitive actions, such as changing account settings or processing payments.

- **How it works:** For actions that need additional security, prompt the user to enter their password or authenticate using multi-factor authentication (MFA). This approach ensures that a CSRF request cannot succeed without user interaction.

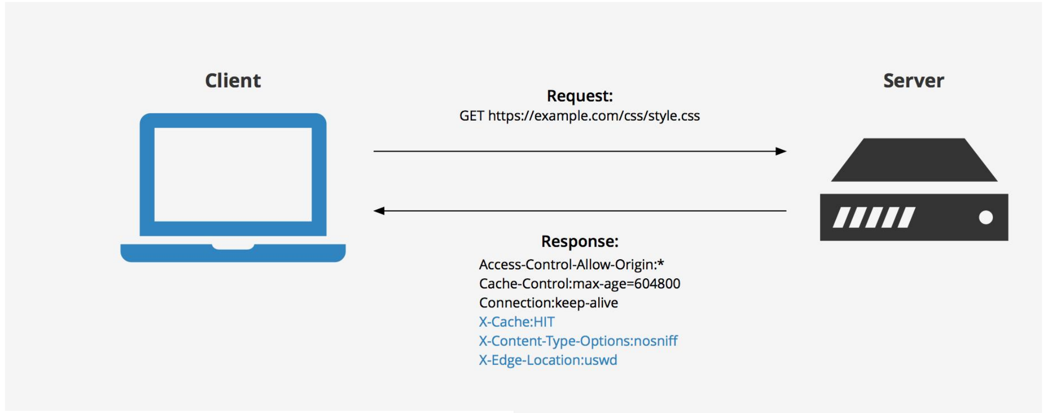


Effective CSRF mitigating strategies

4. Use of Custom Headers

- **What it is:** Custom headers are headers that browsers cannot add automatically with a simple HTML form or image tag request.

- **How it works:** By requiring requests to use a custom header (e.g., X-Requested-With) to differentiate between legitimate requests and CSRF attempts. Many cross-origin requests (e.g., requests from another site) do not allow setting custom headers, making it harder for attackers to forge requests.



```
Request:
GET https://example.com/css/style.css

Response:
Access-Control-Allow-Origin:*
Cache-Control:max-age=604800
Connection:keep-alive
X-Cache:HIT
X-Content-Type-Options:nosniff
X-Edge-Location:uswd
```

Custom HTTP Headers

PART 02

Demonstration

Attack & Defense →

INDUSTRY Best practices for CSRF prevention

Use a Secure Framework

Choose web application frameworks that provide robust CSRF protection features, such as token generation and request verification.

Regular Security Audits

Conduct frequent security audits to identify and address potential vulnerabilities, including CSRF risks.

Stay Updated

Keep web applications and security software up-to-date with the latest patches and security updates to mitigate emerging threats.

Educate Users

Educate users about CSRF threats and how to recognize and avoid malicious requests, empowering them to stay secure online.

Conclusion

"

CSRF protection is essential for safeguarding web applications from malicious attacks.
For strong CSRF protection, combine multiple strategies, such as using CSRF tokens, SameSite cookies, and re-authentication. Most frameworks provide built-in CSRF protection, but adding additional layers like re-authentication or MFA for critical actions will enhance security.

"

PROTECT AGAINST CYBER ATTACKS
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Use security password
Keep your information safe
Remember to log out
Check your emails before opening
Shop and pay safety
Avoid sharing private information

THANK YOU