CROSS-SITE REQUEST FORGERY (CSRF)

Source: https://www.invicti.com/

Imagine you're logged into your bank account, peacefully browsing your finances. Suddenly, a malicious email pops up, urging you to click a link for a "free gift voucher." Unsuspecting, you click it. In the background, without your knowledge, the link triggers a request to transfer all your money to the attacker's account! This is the power of a Cross-Site Request Forgery (CSRF) attack.

What is CSRF?

CSRF is a web security vulnerability that tricks your browser into unknowingly performing actions on a trusted website where you're already logged in. Attackers exploit the trust a website has in you (the authenticated user) to execute unauthorized actions on your behalf.

Think of it as someone forging your signature on a check. They can't access your bank account directly, but they can manipulate you into signing a check that benefits them.

How does a CSRF attack work?

- **The Setup:** An attacker creates a malicious website (let's call it "Evil Inc.") containing an image.

- **The Trick:** The attacker embeds a hidden code within the image that triggers a request to your bank's website (the trusted site) once clicked. This request could be to transfer money or change your password.

- **The Unsuspecting Click:** You receive an email with a link to Evil Inc.'s website. The email might be disguised as a legitimate offer or look funny/interesting, enticing you to click.
- **The Unintended Action:** When you click the image, your browser automatically sends a request to your bank along with your cookies (which verify you're logged in). The bank sees a legitimate request coming from your browser and unknowingly executes the action, all because you were already logged in.

Testing for a CSRF with demo explaination

- **Set up a Lab Environment:** Use a controlled environment to test on a non-critical website you have access to (not your bank!).
- **Identify Vulnerable Forms:** Look for forms that trigger actions (money transfer, password change) and note down the request type (GET/POST) and URL.
- **Intercept Requests with a Proxy Tool:** Install a browser proxy extension like Tampermonkey. These tools allow you to capture and modify requests before they are sent.
- **Craft a Test Request:** Manually construct a request (matching the request type and URL from step 2) that triggers the unwanted action.
- **Modify the Request:** In your proxy tool, alter the request to see if it still executes even without user interaction (clicking

a button). If it does, the website might be vulnerable to CSRF.

Testing for CSRF with Burpsuite via both community edition and professional edition

While Burp Suite Community Edition offers limited functionality compared to the Professional edition, you can still utilize it to identify potential CSRF vulnerabilities. Here's a breakdown for both versions

Manual Analysis:

- Intercept requests in the Proxy tab and identify actions that modify user data (e.g., transferring funds, changing settings).
- Analyze the request parameters and cookies involved in these actions.
- Manually craft a similar request using a tool like Postman, excluding cookies.
- If the action is still performed when you send the crafted request, it might indicate CSRF vulnerability (cautiously test on a non-production environment).

Limitations:

- This method is time-consuming and requires understanding request parameters.
- It doesn't automatically generate a CSRF exploit (proof-of-concept).

CSRF PoC Generation:

- Intercept the target request in the Proxy tab.
- Right-click the request and navigate to "Engagement tools" > "Generate CSRF PoC".
- Burp Suite automatically generates HTML code that mimics the request, ready to be used in a test environment. You can modify the form fields within the generated HTML for a targeted attack (e.g., changing the recipient account in a money transfer).

Additional Advantage

Burp Scanner (Professional Edition): This extension can identify potential CSRF vulnerabilities during automated scans, providing a starting point for further investigation.

Imp Points

- Always test for CSRF vulnerabilities in a controlled, non-production environment to avoid unintended consequences.
- CSRF attacks can be complex, and these methods might not always identify all vulnerabilities. Consider a combination of manual testing and automated tools for a more comprehensive approach.

Impacts

CSRF attacks can have severe consequences. They can be used to steal sensitive data, transfer funds, change account settings, or even take over accounts completely.

Mitigations

1. **Synchronizer Token Pattern (CSRF Token):** Websites can generate a unique token for each user session and embed it in forms or requests. The server validates the token along with the request, ensuring it originates from a legitimate form submission.
2. **SameSite Cookie Attribute:** This attribute restricts how cookies are sent in cross-site requests, making it harder for attackers to exploit them.
3. **Double Submit Cookies:** Websites can send a non-HTTP cookie along with forms. The server validates the presence of this cookie to ensure the request originated from the user's browser and not a forged request.

Important Points to remember

- Be cautious about clicking on links from unknown sources, especially in emails.
- Look for signs of secure websites, like HTTPS in the address bar.
- Keep your browser and web applications up to date with the latest security patches.

References

- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- https://portswigger.net/web-security/csrf
- https://www.invicti.com/blog/web-security/csrf-cross-site-request-forgery/
- https://portswigger.net/support/using-burp-to-test-for-cross-site-request-forgery