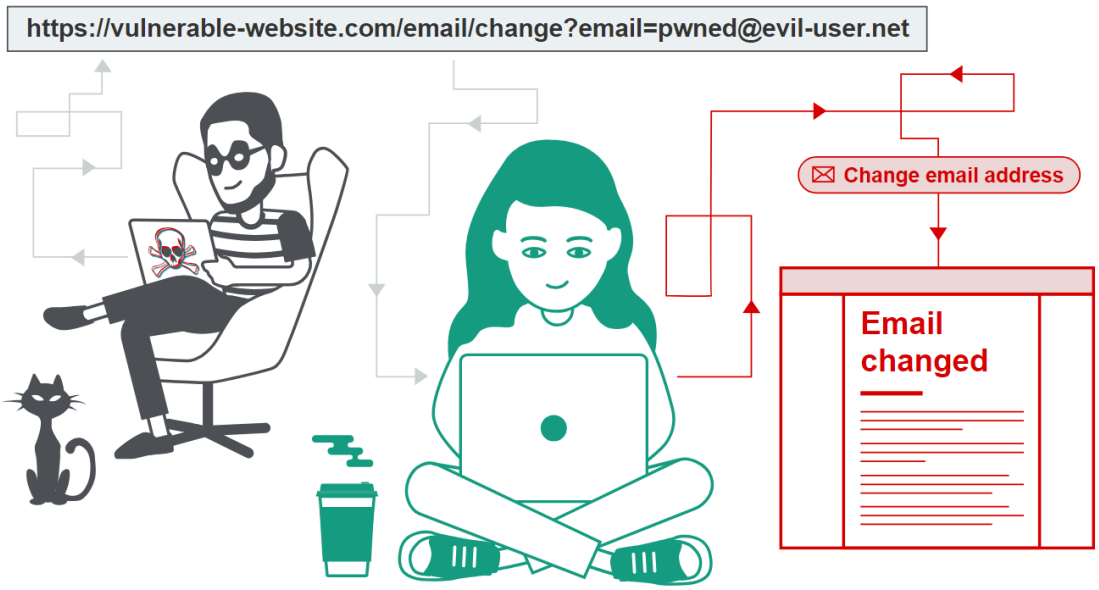
WEB OF DECEPTION

A REPORT ON THE HIDDEN PERILS OF CSRF

Cross Site Request Forgery (CSRF)

1.Introduction

Cross-Site Request Forgery (CSRF) exploits the trust that a web application has for the user's browser. This attack compels an unsuspecting user to perform actions on a web application in which they are authenticated, often with the aid of social engineering tactics like phishing emails or deceptive links. For regular users, a successful CSRF attack might result in unauthorized actions such as transferring funds or changing account details. When it targets an administrative account, the impact is even more severe, potentially compromising the entire web application by executing commands with higher privileges.



2.How does CSRF work

The exploitation of CSRF Vulnerability includes the following steps:

- **User Authentication:**
  The user logs into a legitimate web application and the browser stores authentication cookies or session information.

- **Creating the Malicious Request:**
  The attacker creates a malicious request that invokes an action on the legitimate web application.
  Example: A form submission or a URL that starts a fund transfer.

- **Tricking the User:**
  The attacker sends the malicious request to the user, usually through email, social media, or a **c**ompromised website.

  Example: Phishing email with link to malicious request.

- **User Interaction**
  The user, in error, clicks on the malicious link
  or opens the hijacked website but still is logged on to the legitimate application.

  Could be in another tab, even pop up.

- **Running the Request:**
  Their browser sends the malicious request to the legitimate web application using the stored authentication cookies or session information.The web application executes the request as if it is being made by the authenticated user.

- **Action Taken:**
  The legitimate web application performs the action set forth in the malicious request.

  Example: Transfer funds, change account information or perform administrative t asks.

3.Example Scenario:

Let's say you have an online banking account with Tickle Trust Bank. The bank's website allows users to transfer money to other accounts. Normally, you would log in and manually enter the recipient's details to transfer funds. Here's how a CSRF attack could exploit this process:

- **User Authentication:**
  - You log into your Tickle Trust Bank account at:
    https://www.Tickletrustbank.com/login
  - Your browser stores the session cookie to keep you authenticated.

- **Creating the Malicious Request:**
  The attacker creates a malicious URL that initiates a fund transfer. The crafted URL might look something like this:
  https://www.Tickletrustbank.com/transfer?amount=10000&toAccount=987654321

This URL is designed to transfer ₹10,000 to the attacker's account (account number 987654321).

- **Tricking the User:**
  - The attacker sends this URL to you via email or embeds it in a seemingly harmless link on a compromised website.
  - For example: ***Check out this cool article!***
  - The HTML page with the link embedded will look like this:

```
<!DOCTYPE html>

<html>

<body>

 <p>Click <a href="https://www.Tickletrustbank.com/transfer?amount=10000&toAccount=987654321">to view the article. </p>

</body>

</html>
```

- **User Interaction**
  - You, still logged into your bank account, click on the link in the phishing email or visit the compromised website.
  - Since you are authenticated, your browser sends the request to Tickle Trust Bank using the stored session cookie.

- **Executing the Request:**
  The request is processed as if you, the authenticated user, initiated it. Tickle Trust Bank processes the transfer to the specified account number without any additional confirmation.

- **Action Taken:**
  The malicious transfer of ₹10,000 to account 987654321 is completed successfully.

4.Recent Trends

- **Increased Attack Vectors**: With the rise of remote work and increased use of collaboration tools, attackers have targeted these platforms to exploit CSRF vulnerabilities.
- **Technological Advancements**: The maturity of AI technology, such as CHATGPT, can accelerate the number of cyberattacks, including CSRF.
- **AI-Powered Threats**: 74% of IT security professionals report their organizations are suffering significant impact from AI-powered threats.

Statistics

- **Global Cyberattack Trends**: According to Check Point Research, global cyberattacks increased by **38% in 2022** compared to 2021. While specific statistics for CSRF attacks post-2022 are not readily available, the overall rise in cyberattacks suggests that CSRF remains a significant threat1.
- **Industry Impact**: The education/research sector saw a **43% increase in cyberattacks** in 2022, making it the most targeted industry. This trend indicates that CSRF attacks are still prevalent in sectors with high user interaction and authentication requirements1.

5.Preventive Measures

To protect web applications from CSRF attacks, several preventive measures can be implemented:

1) **CSRF Tokens**: Use unique tokens in web forms and validate them on the server side. These tokens should be unpredictable and tied to the user's session.
2) **Same-Site Cookies**: Set the SameSite attribute for cookies to restrict their usage to the originating domain.
3) **Custom Request Headers**: Use custom headers for requests to ensure they are not easily forged.
4) **User Interaction**: Require user interaction for sensitive operations, such as re-authentication or confirmation.
5) **Referrer Policy**: Implement strict referrer policies to limit the information passed in request headers.
6) **Stateless Software**: Use double-submit cookies for stateless applications.
7) **Avoid GET Requests for State Changes**: Do not use GET requests for state-changing operations.

6.Conclusion

In conclusion, CSRF attacks pose a significant threat to web application security by exploiting the trust relationship between a user and a website. By leveraging authenticated sessions, attackers can perform unauthorized actions on behalf of users, leading to potential financial loss, data breaches, and compromised user accounts. exploits remain a significant threat in the cybersecurity landscape. While they may not be as prevalent as some other types of attacks, they still pose a considerable risk due to their potential to compromise user accounts. Implementing robust preventive measures described above, is crucial in mitigating this vulnerability and safeguarding web applications.

In a sense, these security practices, applied vigilantly, will highly reduce the chance of attack from CSRF, enhancing overall security over web applications for developers and administrators.

7.Sources

a. https://portswigger.net/web-security/csrf
b. https://owasp.org/www-project-web-security-testing-guide/
c. https://owasp.org/www-community/attacks/csrf#:~:text=Cross%2DSite%20Request%20Forgery%20(CSRF,which%20they're%20currently%20authenticated.

8. Labs and Practice

We can practice exploiting CSRF vulnerabilities on PortSwigger academy. We make us of Burpsuite to intercept and play with HTTP requests as and when required.

Fig : Screenshot of Labs solved in Portswigger academy.