Multistep clickjacking in a blog webapp

Executive Summary

This report outlines a successful exploitation of a multistep Clickjacking vulnerability within the target web application. The vulnerability allowed a malicious attacker to trick a user into unintentionally clicking a sequence of UI elements specifically the “Delete Account” button and its subsequent confirmation leading to unauthorized account deletion. This is a high-severity issue that undermines user intent and application integrity.

Introduction

The objective of this assessment was to determine whether client-side protections, such as CSRF tokens and user confirmation dialogs, were sufficient to defend against user interface redressing attacks (Clickjacking). Special attention was given to actions involving sensitive operations like account deletion.

Methodology

1. Logged into the target application using test credentials to study the behavior of the account deletion workflow.

2. Observed that the delete action includes a CSRF token and a JavaScript-based confirmation dialog.

3. Developed a multi-step Clickjacking attack using HTML `iframe` layering and decoy `div` buttons to overlay user interface elements.

4. Used CSS styling to precisely align fake buttons over the real target actions.

5. Deployed and tested the exploit in Chrome, the same browser used by the target victim.

6. Delivered the exploit through the provided exploit server and verified successful deletion of the target’s account.

Vulnerability Findings

- **Type**: Multistep Clickjacking

- **Location**: `/my-account` endpoint (account deletion functionality)

- **Severity**: High

Description

The application relied solely on a CSRF token and a JavaScript confirmation dialog for protecting sensitive user actions. However, these controls were bypassed through a Clickjacking attack involving two clicks. By embedding the target page inside a transparent `iframe` and placing decoy `div` elements above sensitive UI elements ("Delete account" and "Yes" button), an attacker can manipulate user behavior and trigger critical actions unknowingly.

Proof of Concept (PoC)

A simplified version of the malicious HTML payload is shown below:

html

CopyEdit

`<style> iframe { position: relative; width: 500px; height: 700px; opacity: 0.0001; z-index: 2; } .firstClick, .secondClick { position: absolute; z-index: 1; } .firstClick { top: 330px; left: 50px; } .secondClick { top: 285px; left: 225px; } </style> <div class="firstClick">Click me first</div> <div class="secondClick">Click me next</div> <iframe src="https://website.ID.net/my-account"></iframe>`

Upon interaction, the user is deceived into confirming the deletion of their account.

Impact Assessment

- Complete loss of user accounts without informed consent.

- Exploitable with only two user clicks on what appear to be harmless UI elements.

- No visual indicators alerting the user to the real nature of their actions.

- High risk for targeted users, especially administrators or privileged accounts.

Recommendations

1. Implement `X-Frame-Options: DENY` or `SAMEORIGIN` to prevent the application from being loaded inside iframes.

2. Use the `Content-Security-Policy` (CSP) header with `frame-ancestors 'none';`.

3. Reconsider the use of only client-side confirmation dialogs—implement server-side validation of user intent.

4. Regularly test for UI redressing attacks using penetration testing and automated scanners.

5. Educate frontend developers about secure frame handling and anti-clickjacking headers.

Conclusion

The exploitation of this multistep Clickjacking vulnerability demonstrated the limitations of relying solely on CSRF tokens and confirmation dialogs. Implementing proper frame-busting headers and secure interaction design is vital for protecting critical user actions from deceptive manipulation.