Forced OAuth profile linking

Executive Summary

This report outlines the discovery and exploitation of a **Forced OAuth Profile Linking** vulnerability in the target application. The flaw allows an attacker to link their own social media account to another user's profile in this case, the admin via a Cross-Site Request Forgery (CSRF) attack. Successful exploitation granted full access to the admin's account, allowing the attacker to delete selected user. This vulnerability presents a **high risk** to account integrity and access control mechanisms.

Introduction

The purpose of this test was to assess the security of the OAuth implementation, particularly in relation to how third-party social media profiles are linked to user accounts. Special attention was given to checking for CSRF protections during the OAuth flow.

Methodology

1. Logged in to the blog website and linked a personal social media account via OAuth.

2. Used Burp Suite to intercept and analyze OAuth flow requests.

3. Identified the lack of state parameter, a critical CSRF defense.

4. Captured and reused a valid authorization code in a malicious iframe served via an exploit server.

5. Delivered the payload to the admin, who unknowingly completed the OAuth flow.

6. Gained access to the admin panel and deleted selected user.

Vulnerability Findings

- **Type**: Forced OAuth Profile Linking via CSRF

- **Location**: OAuth linking endpoint (`/oauth-linking`)

- **Severity**: High

Description

The application allowed users to attach a social media account via OAuth. During the OAuth flow, the server fails to include a `state` parameter in the authorization request to protect against CSRF. As a result, an attacker can intercept and reuse the authorization code in a malicious request sent to the victim (admin). Once the admin's browser executed the malicious request, the attacker's social media profile is linked to the admin's account.

Proof of Concept (PoC)

1. Logged into blog account.
2. Linked the social media account via OAuth.
3. Intercepted this OAuth flow and captured:

   bash

   CopyEdit

   `GET /oauth-linking?code=STOLEN-CODE`
4. Dropped the request to preserve the code.
5. Crafted and hosted the following payload on the exploit server:

   html

   CopyEdit

   `<iframe src="https://website.ID.net/oauth-linking?code=STOLEN-CODE"></iframe>`
6. Delivered the payload to the admin.
7. Admin's browser completed the OAuth flow, linking attacker's profile to their account.
8. Logged in using "Login with social media".
9. Accessed admin panel and deleted selected user.

Impact Assessment

This vulnerability allowed attackers to:

- Gain unauthorized access to high-privilege accounts (e.g., admin).
- Perform privileged actions such as deleting users or modifying data.
- Bypass authentication using linked OAuth accounts.

This could result in severe privilege escalation, data loss, and a complete compromise of application-level access controls.

Recommendations

1. Implement the `state` parameter in all OAuth flows to validate the origin of the request and prevent CSRF attacks.

2. Enforce single-use and short-lived authorization codes to prevent reuse.

3. Monitor and audit linked account activity to detect unusual linking patterns.

4. Conduct security reviews of all third-party integrations, especially OAuth.

5. Educate developers on secure OAuth implementation practices as outlined by OWASP.

Conclusion

The forced OAuth profile linking vulnerability presented a critical weakness in the authentication process. By leveraging a predictable and unprotected OAuth flow, attackers could hijack high-privilege accounts. It is imperative that this issue is remediated immediately through the implementation of proper CSRF defenses and best practices for secure OAuth use.