OAuth 2.0 Redirect URI Validation Falls Short, Literally

Tommaso Innocenti, Matteo Golinelli, Kaan Onarlioglu, Ali Mirheidari, Bruno Crispo, and Engin Kirda

innocenti.t@northeastern.edu
Tommaso Innocenti
PhD candidate
@Northeastern University (Boston)
innotommy.com
RFC `redirect_uri` validation issue

- XSS style
- HTML injection
- Open redirect
- OAuth token Leakage

Full victim’s account takeover
What the heck is OAuth 2.0?

**Introduction**

Bob

Service Provider

Identity Provider

Sign in

- Sign in with IMDb
- Sign in with Amazon
- Sign in with Google
- Sign in with Apple

**Benefit**

- Personalize
- Discover shows
- Your Watch
- Track everything
- Your Rating
- Rate and review
- Contribute!
- Add data to badges.

Show more options

Create a New Account

By signing in, you agree to IMDb’s Conditions of Use and Privacy Policy.

Visits Website

Login Request

User Authentication and Consent

Login Response

"Hi, Bob!"
redirect_uri validation in RFC:

• RFC 6749 Section 3.1.2.3
  The authorization server MUST compare the two URIs using simple string comparison as defined in RFC 3986 Section 6.2.1.

• RFC 3986 Section 6.2.1
  Testing strings for equality is normally based on pair comparison of the characters that make up the strings, starting from the first and proceeding until both strings are exhausted, and all characters are found to be equal, until a pair of characters compares unequal, or until one of the strings is exhausted before the other.
What is Path Confusion?

Bob visits the IMDb website, which is also a Service Provider. PathConfusion:

```
../%252FFAKEPATH
/%252e%252e%252FFAKEPATH
```

During the login request, the user may be directed to an Identity Provider. The diagram illustrates the process of user authentication and consent, followed by a login response.
Methodology

Setup
- Sites & OAuth 2.0 Triggers
- IdP Detection
- Login page detection
- IdP Credentials
- Tranco sites list

Data Collection
- OAuth 2.0 Player
- IMDb
- Facebook
- Sign in
- mitmproxy
- Path
- Confusion
- Payloads
- Network Dump
- Login Results
- Proxy Logs
- Screen Captures

Data Analysis
- OAuth 2.0 Flow Analysis
- Analysis Results
6/16 IdPs vulnerable to Path Confusion
(Facebook, Microsoft, GitHub, Atlassian, NAVER, and VK)
Are we doomed?

1) Client Application Access
2) Redirection to IdP Login
3) Authorization Request
   [response_type=code, client_id, state[redirect_uri]]
4) User Authentication
5) Redirection to Client Callback
6) Resource Owner
7) Access Token Request
   [grant_type=authorization_code, client_id, client_secret, code[redirect_uri]]
8) Access Token Response
   [access_token]
9) Protected Resource Requests
   [access_token]
10) Protected Resource Response
   [Data]
- Path Confusion
- Redirect_uri validation in redeem step

+ - XSS style
  - HTML injection
  - Open redirect
  - OAuth token Leakage

Full victim’s account takeover
• Path Confusion

Attack checklist:
1) Vulnerable `redirect_uri` parsing in Authorization step → 6/16 IdPs

2) Vulnerable Client → openbugbounty.com

3) Vulnerable `redirect_uri` check in redeem step → 2/16 IdPs

Attack URL:


Full Victim’s account takeover is possible!!!
All IdPs involved in the study which has been found vulnerable has been contacted.

- Microsoft acknowledge our report and fixed their validation procedure.
- GitHub is tracking internally the problem and is actively working on a fix
- We are actively working with Naver to help fixing the issue

Reported our findings to the OAuth working group, which included our recommendation in the BCP. OpenID foundation modified the conformance test suite to include our attack
Current “best practice” is not good enough

Recommendations:

1) `redirect_uri` validation should use strict string equality check

2) IdPs server should never sanitize `redirect_uri` to avoid introducing any discrepancy, instead should validate them
One more thing

• Path confusion
• OAuth Parameter Pollution $\rightarrow$ 10/16 IdPs vulnerable
Questions?
redirect_uri parameter in RFC:

- RFC 6749 Section 3.1
  The endpoint URI MAY include an "application/x-www-form-urlencoded" formatted (per Appendix B) query component (RFC 3986 Section 3.4), which MUST be retained when adding additional query parameters.
- RFC 6749 Section 10.14
  A code injection attack occurs when an input or otherwise external variable is used by an application unsanitized and causes modification to the application logic. This may allow an attacker to access the application device or its data, cause a denial of service, or introduce a wide range of malicious side effects. The authorization server and Client MUST sanitize (and validate when possible) any value received—in particular, the value of the "state" and "redirect_uri" parameters.

Lack on input validation directive or attack prevention
• Attack URL:

https://idp.example.com/oauth/authorize?response_type=code&client_id=<validID>&state=<value>&redirect_uri=https://Client.example.com/oauth/callback%3Fcode%3D<value>

Victim’s authenticated as the attacker!!
• Path confusion
• OAuth Parameter Pollution → 10/16 IdPs vulnerable

Recommendations:

3) IdPs should validate `redirect_uri` and block Authorization request where `Code` or `state` parameters are included in the `redirect_uri` as parameter.
Questions?