OAuth User Profile Attack

How to Sign into One Billion Mobile App Accounts Effortlessly

Ronghai Yang, Prof. Wing Cheong Lau and Tianyu Liu

The Chinese University of Hong Kong

Nov 4, 2016
Outline

• Background of OAuth2.0
  ✤ Unwell-defined protocol for mobile platforms
• User Profile Vulnerability
• Exploit
  ✤ Challenges & Tricks
  ✤ Case study
• Corresponding Remedies
What is OAuth2.0?
Three Parties in OAuth2.0

User | Identity Provider (IdP) | Relying Party (RP)

Goal: The user can log into the RP via the IdP
Basic Interactions among User, RP and IdP

- Such an identity proof is “access token (AT)” in OAuth2.0.
- OAuth2.0 supports two types of mode: authorization code flow & **implicit flow**
OAuth2.0 Protocol Flow for **Mobile**: Implicit Flow

- RP server
- User device
- IdP server

**Sign in**

Use an existing account

- IMDb
- Facebook
- Google
OAuth2.0 Protocol Flow for Mobile: Implicit Flow

considering an example of IMDb and Facebook as the entities:

User device

RP server

IdP server

RP App

auth req

IdP App

authentication & authorization

ronghai@gmail.com

[enter password]

LOG IN
OAuth2.0 Protocol Flow for Mobile: Implicit Flow

RP server

User device

IdP server

RP App

auth req

IdP App

authentication & authorization

IMDb

Continue as Ronghai

IMDb will receive the following info: your public profile and email address.

Edit the info you provide

This does not let the app post to Facebook.
OAuth2.0 Protocol Flow for Mobile: Implicit Flow

RP server

User device

IdP server

AT + user info

{"token_type":"Bearer", "expires_in":7104, "id":"100008512695261", "access_token":"CAABzj3PSN8C6OELrcr44hSIITO6…"}
OAuth2.0 Protocol Flow for **Mobile**: Implicit Flow

**RP server**

**User device**

**IdP server**

API request for user info: access token

https://graph.facebook.com/me?

**access_token**=CAABzj3PSNiUBAF9MQrrNHwoZ…
OAuth2.0 Protocol Flow for **Mobile**:
Implicit Flow

**RP server**

```
{
  "id": "100008512695261",
  "birthday": "02/01/1991",
  "email": "ronghai@gmail.com",
  "first_name": "Ronghai",
  "gender": "male",
  "last_name": "Yang",
  "link": "https://www.facebook.com/profile.php?id=100..1",
  "name": "Ronghai Yang",
}
```

**IdP server**

**User profile information**
Unwell-defined Portions of Protocol Call-flow

- Neither RFC nor IdPs provides the complete call-flow
  - How to communicate between RP app and IdP app: the browser splits into two apps
  - How to process identity proof: server-to-server verification
Common Mistake 1
Android Account Manager

• Centralized database to store user accounts

• INSERT INTO “accounts” VALUES
  (1,’ronghai@gmail.com’,’com.google’,’password’,NULL)

• Integrated into OAuth2.0 when using Google as the IdP
Common Mistake 1
Android Account Manager

• Two steps to obtain the access token
  ✤ Auth request: getAccounts()
  ✤ Token request: GoogleAuthUtil.getToken()

• Step 2 is often **missing** by RP developers
Common Mistake 2
RP App Fails to Return AT

- The RP app does not return AT to the RP server
- The RP server only depends on user info to identify the user
Common Mistake 3
Fail to Verify Signature of Signed id_token (OpenID Connect)

- id_token
  - includes user profile information
  - signed by IdP server
- The signature can be incorrectly verified, e.g., not verify the signature at all
The Platform to Exploit the Vulnerability

1. AT+ user info
2. AT+ user info
3. AT+ user info
4. AT+ user info
5. AT+ user info

MitM Proxy

Domain under the attacker’s control

attacker device
Tamper the message between RP app and RP server

- Challenges
  1. proprietary message exchanges
  2. digital signature/ encryption, in addition to HTTPS
  3. no scalable

- Tamper messages between IdP app and IdP server
  1. messages tampered on the IdP side will be propagated to the RP side
Trick 1: Naive way to tamper messages between the IdP app and IdP server

- The IdP app does not adopt any practice to avoid MITM proxy
Trick 2: Use WebView to bypass certificate pinning

- Certificate pinning
  - The IdP app only accepts the certificate from the true IdP server
- Uninstall IdP app to downgrade WebView scheme
Trick 3: Modify IdP app to remove certificate pinning

- Some IdPs do NOT support WebView
- Existing tools do not work
  - SSLUnpinning
  - Reverse engineering
  - Remove certificate pinning function
  - Repackage
Trick 4: Modify RP app to remove the certificate comparison by SDK

- RP app checks whether IdP app is legitimate
  - The SDK hard-code the certificate of true IdP app.
  - IdP app is re-signed
- Modify RP app
  - **scalable:** modify the same function
Demonstration
Attacking Answers App

Step 1: The attacker, Eve, uses her own Google account to log into Answers.
Demonstration
Attacking Answers App

Step 2_a: The attacker setups MITMPProxy

- The access token is bound to the attacker’s Google account
Answers app uses access token to retrieve user data

Step 2_b: The attacker intercepts the user-profile request via proxy

The unique user id of Eve in Google+
Demonstration
Attacking Answers App

Step 3_a: The attacker searches the public user profile of the victim, Alice.

Step 3_b: The attacker obtains Alice’s user id via URL.
Demonstration
Attacking Answers App

Step 4: The attacker substitutes her own user id with the victim’s one.
Demonstration
Attacking Answers App

The attacker logins as the victim, Alice

- Only require the public victim profile
- The attack can be remotely/ silently launched
## Empirical Evaluation

<table>
<thead>
<tr>
<th>IdPs</th>
<th># of Top Apps tested (overall + per category)</th>
<th># of Apps Support OAuth2.0</th>
<th># of Vulnerable Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>400 (300+100)</td>
<td>59</td>
<td>9 (15%)</td>
</tr>
<tr>
<td>Google</td>
<td>400 (300+100)</td>
<td>40</td>
<td>8 (20%)</td>
</tr>
<tr>
<td>Sina</td>
<td>200 (100+100)</td>
<td>83</td>
<td>58 (70%)</td>
</tr>
<tr>
<td>Summary</td>
<td>1000</td>
<td>182</td>
<td>75 (41%)</td>
</tr>
</tbody>
</table>

- Facebook/ Google from Google Play
  - Top-300 Apps in overall category
  - Top-100 Apps in different categories
- Sina from one major Chinese app store
  - Top-100 Apps in overall and different categories
A Partial List of Vulnerable Android Mobile Apps

<table>
<thead>
<tr>
<th>Type of Apps</th>
<th>Description</th>
<th>Feasible Transactions by the Attacker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Travel Plan App</td>
<td>Hotel Booking (in travel apps)</td>
<td>pay for room bookings</td>
</tr>
<tr>
<td>Hotel Booking App</td>
<td></td>
<td>send forged messages, purchase gifts</td>
</tr>
<tr>
<td>Private Chat App</td>
<td></td>
<td>call for free, purchase gifts</td>
</tr>
<tr>
<td>Dating App</td>
<td></td>
<td>enjoy VIP speed, purchase sound-tracks</td>
</tr>
<tr>
<td>Finance App1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Partial List of Vulnerable Android Mobile Apps

• The total number of downloads for this incomplete list of Android apps exceeds 2.4 billion.

• Based on the SSO-user-adoption-rate of 51%, one conservative estimate is that more than one billion of different types of app accounts are susceptible.

• Such an attack is also feasible to iOS

  ✤ iOS RP apps adopt the same protocol call-flow
Responsible Disclosure

- We reported this issue to all three IdPs on April 2016
- Receive their acknowledgements in different ways
  - Maximum bounty reward from Sina
  - Sina sent a notification letter to all its third-party app developers
- Based on our incomplete sampling very recently, most of RPs are still vulnerable
1. **For IdPs:**

   - Provide more clear, and more security-focused guidelines
   - Issue private per-app user-id
     - Facebook has adopted this practice since May 2014, but due to the backward compatibility reason, old users are still vulnerable.
   - More security testing/ auditing on the RP app
     - We have developed an OAuthTester tool for large-scale testing

2. **For RPs:** Never trust client-side information

3. Follow the best practices in *draft-ietf-oauth-native-apps-05*
Thanks and Q&A

Ronghai Yang
http://personal.ie.cuhk.edu.hk/~yr013/

Wing Cheong Lau
www.ie.cuhk.edu.hk/~wclau/