OAUTH—Everything You Wanted to Know but Not Really!
Kevin Watkins

- Security Researcher & Developer
- 20+ years in the Security/Privacy space
- Pioneered early automation tools identifying mobile threats
- Founder of Appthority & the core technology securing Fortune 100 from mobile risk threats
- Continuing to innovate mobile threat risk detection at Symantec after Appthority acquisition in 2018
- @kwatts
Elaheh /ela:'he/ Samani

- Security Researcher
- Malware/software reverse Engineer
- UNB proud/ former CIC security researcher
- Incident responder
- Security software developer
- @0xGravity
OAuth2
Different Forms of OAuth

- Client (App) accessing user’s data on a third-party resource server, e.g. Google, FB, or Linkedin → Auth Code
- Resource server and App are part of the same system → Password Grant Type
- App accessing its own resources → Client_credentials
<table>
<thead>
<tr>
<th>App</th>
<th>Auth Server</th>
<th>Resource Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>![App Icon]</td>
<td>![Auth Server Icon]</td>
<td>![Resource Server Icon]</td>
</tr>
<tr>
<td>![App Icon]</td>
<td>![Auth Server Icon]</td>
<td>![Resource Server Icon]</td>
</tr>
<tr>
<td>![App Icon]</td>
<td>![Auth Server Icon]</td>
<td>![Resource Server Icon]</td>
</tr>
<tr>
<td>![App Icon]</td>
<td>![Auth Server Icon]</td>
<td>![Resource Server Icon]</td>
</tr>
</tbody>
</table>
Authentication or Authorization?

Authentication is all about user!
- Tells an application Who the current user is
- Whether or not they’re present
- In some cases more info about the current user, such as unique identifier, email address, etc.

OAUTH on the other hand
- Is a token-based authorization framework
- Says nothing about the user
- Does not care if user’s still present
- Only ask for a token to access some APIs.
AUTH CODE GRANT FLOW

Authenticate and redirect user to Oauth provider

Accept/Deny access

Request access token

Access/Refresh token

Authorized Code

Access/Refresh token

Authorized Code

Google APIs

Accept/Deny access

Google APIs

Accept/Deny access
GET
CONSENT PAGE

Yes or No

There is no maybe.
WHY REFRESH TOKEN?

➢ Access tokens are short lived (1 hour)
➢ Offline access type
➢ Sensitive scopes
➢ Must be stored securely
➢ Must be revoked when users disconnect their accounts

⚠️ Refresh Tokens must be stored securely by an application since they allow a user to remain authenticated essentially forever.
SCOPES – GOOGLE APIs

➢ Gmail
  ○ Mail.google.com
  ○ www.googleapis.com/auth/gmail.compose
  ○ www.googleapis.com/auth/gmail.modify
  ○ www.googleapis.com/auth/gmail.send
  ○ www.googleapis.com/auth/gmail.insert
  ○ www.googleapis.com/auth/gmail.settings.sharing

➢ Gdrive
  ○ www.googleapis.com/auth/drive

➢ Calendar
  ○ www.googleapis.com/auth/calendar
DEMO TIME!
### New Google Updates

<table>
<thead>
<tr>
<th>Sample request</th>
<th>Admin Console message format</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET <a href="https://www.googleapis.com/admin/reports/v1/activity/users/all/applications/token?eventName=authorize&amp;maxResults=10">https://www.googleapis.com/admin/reports/v1/activity/users/all/applications/token?eventName=authorize&amp;maxResults=10</a> &amp;access_token=YOUR_ACCESS_TOKEN</td>
<td>{actor} authorized access to {app_name} for {scope} scopes</td>
</tr>
</tbody>
</table>

#### Sample request

```
GET https://www.googleapis.com/admin/reports/v1/activity/users/all/applications/token?eventName=activity&maxResults=10 &access_token=YOUR_ACCESS_TOKEN
```

#### Admin Console message format

```
\{app_name\} called \{method_name\} on behalf of \{actor\}
```
FACEBOOK

- Tokens have offline access implicitly
- Long-lived access token last 90 days
- Never expiring permissions
  - Configure_page_transactions
  - Manage_pages
  - Pages_messaging
  - Pages_messaging_payments
  - Pages_messaging_subscriptions
  - Publish_pages
  - read_page_mailboxes
LinkedIn

- Access token is valid for 60 days
- Access token refresh process:
  - The user is still logged into LinkedIn
  - Current access token has not expired
- Programmatic access token refresh valid for a year
  - available for a limited set of partners.
- Sensitive scopes
  - r_fullprofile: Full access to the user’s profile
  - w_messages: Access to the user’s messages
  - w_share|w_member_social|rw_nus: Post updates/contents
AUTH CODE
BEST PRACTICES

➢ When dealing with user’s data ask for what you need, only what you really need!
➢ Follow incremental authorization
➢ Always prefer short lived access token if it does not affect your app’s main functionality
➢ Securely store/transfer refresh tokens
➢ Revoke refresh tokens after users disconnect or unsubscribe!
Why Should I Care About OAuth?

**Privacy**
OAuth is behind the request sent over the network to authorizer, responsible for both protecting the credentials and returning the access tokens giving the mobile app access to your data.

**Security**
If credentials and/or access token are not secured in transit or on-device, attackers can use them to access your data off device, from anywhere in the world...
Discovering How Mobile App developers really use OAUTH 2.0 and if they are doing it right!
How can we identify apps that use OAuth 2.0?

OAuth 2.0 requires the URL parameter "grant_type" identifying how a mobile app wants to receive the access token. Common grant types are `authorization_code`, `password`, `client_credentials`, or `refresh_token`.

We can use `grant_types` to identify apps making OAuth requests!

Now about getting the apps to search...
Since 2011 we've been collecting & scanning millions of mobile apps from customers devices and App Stores. We analyze & execute every app on on a farm of mobile phones capturing OAuth HTTPS network activity, including automated authentication requests from login screens

"Honeytokens" are also planted on mobile devices - sticking to and identifying - all types of sensitive data contained in OAuth requests

All captured app activity data (5TB+) is sent to the cloud cloud for research and app analysis
We identified 20k unique public apps (across all App categories and levels of popularity) Android & iOS, discovering over 300 different OAuth 2.0 request grant types...
Surprise Discovery #1

➢ 64% of "password" grants contained credentials inputted by user
➢ 36% (!!!) of "password" grants contained credentials from somewhere else!
  ○ Device IDs used as credentials
  ○ Hard-coded credentials
  ○ App Developers using their credentials
➢ Majority of "client_credentials" backtraced to Analytic & Adware SDKs
➢ Still sifting through the 7% "other" grant types, it's a mess!
Surprise Discovery #2

- 5% of apps using OAuth 2.0 disabled SSL certificate validation
- Primarily from OAuth 2.0 grant requests client_credentials (78%) and password (17%)
- We found quite a few from highly popular apps (50M+ downloads)
- You wouldn't know it, no lock indicating the connection is insecure like on web browsers
- Thankfully almost all authorization_code requests were secure!
Financial Services App
Case Study
Orb Health App  
Case Study

“I love how Orb helps me understand my lab results and connects to my other health apps.” Suzan O.

➢ Access token is valid for 60 days
➢ Access token refresh process:
  ○ The user is still logged into Linkedin
  ○ Current access token has not expired
➢ Programmatic access token refresh valid for a year
  ○ available for a limited set of partners.
➢ Sensitive scopes
  ○ r_fullprofile: Full access to the user’s profile
  ○ w_messages: Access to the user’s messages
  ○ w_share|w_member_social|rw_nus: Post updates/contents

```java
package com.loopj.android.a;
public class m extends SSLSocketFactory {
    SSLContext a = SSLContext.getInstance("TLS");
    public m(Keystore keyStore) throws NoSuchAlgorithmException, KeyManagementException, KeyStoreException {
        TrustManager[] r0 = new X509TrustManager[] {null, null};
    }
    public void checkClientTrusted(X509Certificate[] X509CertificateArr, String str) throws CertificateException {
        return null;
    }
    public void checkServerTrusted(X509Certificate[] X509CertificateArr, String str) throws CertificateException {
    }
    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
    this.a.init(null, new TrustManager[] {r0, null});
}
```
Next Steps...

➢ Discovering what API endpoints are exposed from hard-coded access tokens
➢ Deeper dives and wider discoveries for automation authorization_code grant requests
➢ Looking at the filesystem and how secure access tokens & refresh tokens are stored
➢ Security Disclosures (yay!)

What would you look for in the data we have?
“Responsible disclosure is a model in which a vulnerability or issue is disclosed only after a period of time that allows for the vulnerability to be fixed…”

THE CHALLENGE...

- With thousands of apps with the security issue, how do we find, contact, and verify fixes with thousands of app developers?
- We have the social responsibility to make public aware without putting them at risk

POSSIBLE SOLUTIONS...

- Look to the vendor closest to the technology, Google or Apple, to lend us a hand and help with disclosures?
- Automate the process?
Security Disclosures Workflow
Security Disclosure Responses

App Developers?

- 85%/2500+ app developers never respond
- 15% took ~5 day for first response and ~20 days to fix

Google?

- “It’s ultimately the responsibility of the app developer to set up their rules correctly…”
- “Firebase team does not coordinate the vulnerability notification for the individual applications.”
Thanks!

Any questions?

You can also find us at:

◇ @kwatts
◇ @0xGravity
References

- https://oauth.net/2/
- https://developers.google.com/identity/protocols/OAuth2InstalledApp
- https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/