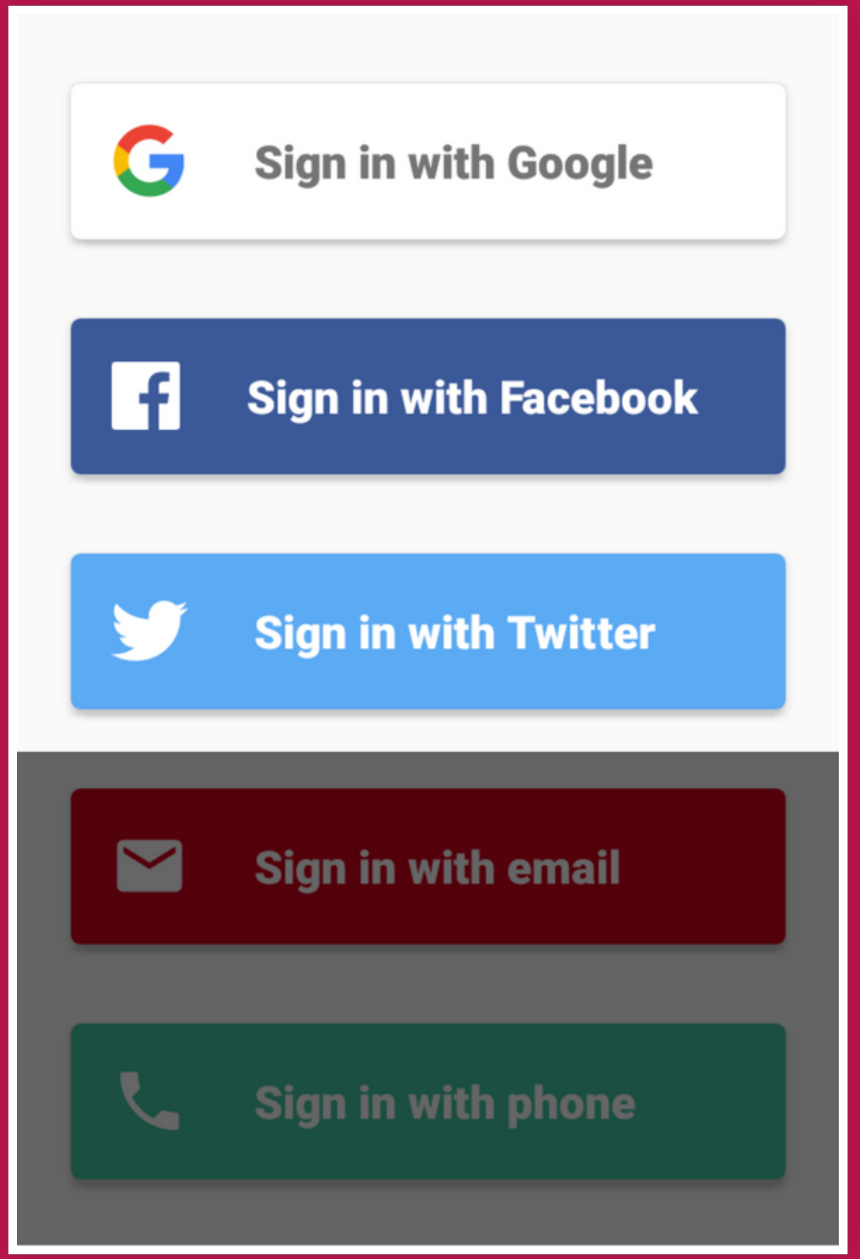
OAuth 2.0: Overview & Attacks

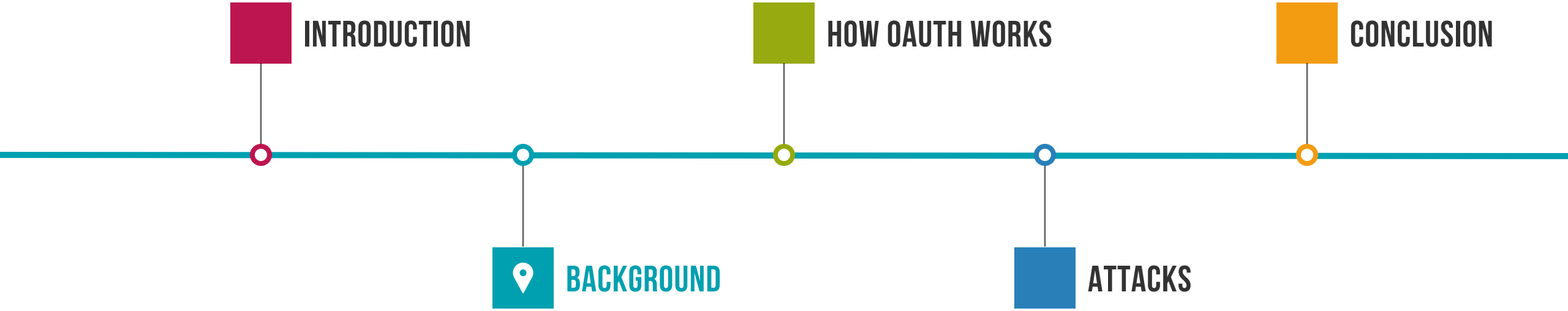
Avery Koranda

HAVE YOU EVER SEEN THIS?



[Source: StackOverflow]

ROADMAP


INTRODUCTION
BACKGROUND
HOW OAUTH WORKS
ATTACKS
CONCLUSION

WHAT IS OAUTH 2.0?

AUTHENTICATION VS. AUTHORIZATION

How do we keep information safe?

AUTHENTICATION

1. The process of identifying an entity
2. Who are you?
3. Prerequisite for authorization

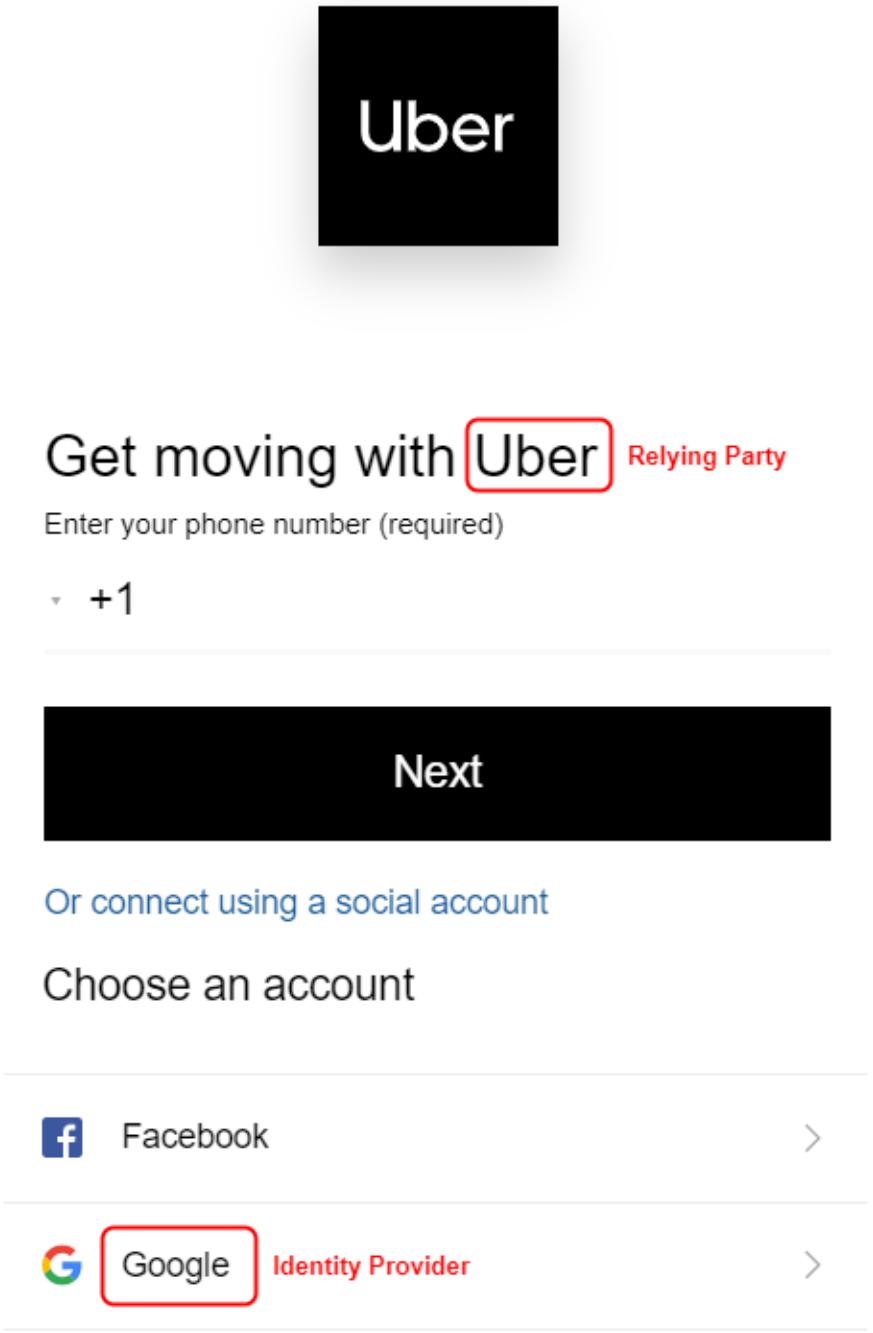
AUTHORIZATION

1. The process of determining whether an entity can access resources
2. Do you have permission to access these resources?
3. Requires authentication

[Source: Kim and Lee]

WHAT IS OAUTH 2.0?

- **OAUTH 2.0 IS AN AUTHORIZATION PROTOCOL**
  - Protocol, not implementation
  - OAuth is the recipe, not the meal
- **DEFINED IN RFC 6749**
- **USED FOR AUTHORIZATION AND AUTHENTICATION**
  - Authentication schemes (today's focus)
- **MAKES SIGNING IN EASIER**
  - Easier for users
  - Easier for developers
- **RELATED PROTOCOLS**
  - OpenID Connect
  - OAuth 1.0 (obsolete)
  - **“OAuth” = OAuth 2.0**



BACKGROUND

INVOLVED PARTIES

- **USER**
  - The person that sees “Sign in with Google”
- **IDENTITY PROVIDER (IDP)**
  - Stores user's resources
  - Grants authorization on behalf of user
  - E.g. Google
- **RELYING PARTY (RP)**
  - Service wanting to access user's protected resources stored at the IdP
  - E.g. Uber
- **ATTACKER**
  - Web attacker
  - Network attacker
  - May also have control of an IdP or an RP

ENVIRONMENT

Sending messages on the Internet

HTTP REQUESTS

- Verb, URL, and data
- GET and POST

HTTP RESPONSES

- Status code and data
- 2XX and 3XX status codes

HTTPS

Messages encrypted using HTTPS unless noted otherwise

ENVIRONMENT

Example

HTTP REQUESTS

- POST request: Google login form
- GET request: browser loads page

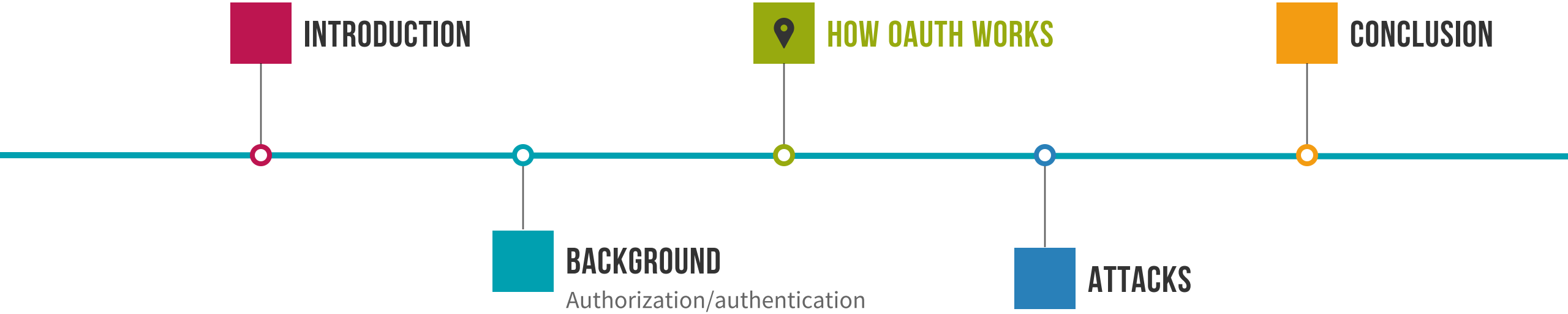
HTTP RESPONSES

- 3XX response: Google redirects browser to Uber
- 2XX response: contains page HTML

HTTPS

POST request is encrypted, otherwise password gets sent in plain text

ROADMAP


INTRODUCTION
HOW OAUTH WORKS
CONCLUSION
BACKGROUND
Authorization/authentication
Involved parties
Environment
ATTACKS

GRANT TYPES

1. **AUTHORIZATION CODE**
   Assume this grant type going forward
2. **IMPLICIT**
3. **RESOURCE OWNER PASSWORD CREDENTIALS**
4. **CLIENT CREDENTIALS**

AUTHORIZATION CODE GRANT TYPE

Procedure (example forthcoming)

[Source: Fett, Küsters, and Schmitz]

AUTHORIZATION CODE GRANT TYPE

Procedure

1. User chooses an IdP

2. RP redirects browser to IdP

3. User types in credentials

4. IdP sends user back to RP with authorization code

AUTHORIZATION CODE GRANT TYPE

Procedure

USER AUTHENTICATES AT IDP

1. User chooses an IdP
2. RP redirects browser to IdP
3. User types in credentials
4. IdP sends user back to RP with authorization code

RP IS AUTHORIZED

1. RP sends authorization code to IdP
2. IdP conducts a series of checks
3. If checks pass, IdP grants RP access token

RP ACCESSES DATA

AUTHORIZATION CODE GRANT TYPE

Procedure

USER AUTHENTICATES AT IDP

1. User chooses an IdP

2. RP redirects browser to IdP

3. User types in credentials

4. IdP sends user back to RP with authorization code

RP IS AUTHORIZED

1. RP sends authorization code to IdP

2. IdP conducts a series of checks

3. If checks pass, IdP grants RP access token

RP ACCESSES DATA

1. RP requests user data from IdP, includes token in requests

2. IdP provides data to RP

AUTHORIZATION CODE GRANT TYPE

Example

1. User clicks “Sign in with Google”
2. Uber redirects browser to Google
3. User types in email and password
4. Google sends user back to Uber with authorization code

AUTHORIZATION CODE GRANT TYPE

Example

USER AUTHENTICATES AT GOOGLE

1. User clicks “Sign in with Google”

2. Uber redirects browser to Google

3. User types in email and password

4. Google sends user back to Uber with authorization code

UBER IS AUTHORIZED

1. Uber sends authorization code to Google

2. Google conducts a series of checks

3. If checks pass, Google grants Uber access token

UBER ACCESSES DATA

AUTHORIZATION CODE GRANT TYPE

Example

USER AUTHENTICATES AT GOOGLE

1. User clicks “Sign in with Google”

2. Uber redirects browser to Google

3. User types in email and password

4. Google sends user back to Uber with authorization code

UBER IS AUTHORIZED

1. Uber sends authorization code to Google

2. Google conducts a series of checks

3. If checks pass, Google grants Uber access token

UBER ACCESSES DATA

1. Uber might request user's name and email from Google, includes access token in request

2. Google sends information to Uber

REGISTERING AS A RELYING PARTY

Prerequisite to implementing OAuth as an RP

RP MUST REGISTER AT IDP

Happens once (e.g. during development of RP)

IDP ISSUES CLIENT ID & SECRET

- Username & password for RP
- Secret is optional

RP SPECIFIES REDIRECTION ENDPOINT

- At least one URL or URL pattern
- Where IdP sends user and authorization code

[Source: Fett, Küsters, and Schmitz]

USER INTENTION TRACKING

WHAT IS IT?

RP's way of remembering which IdP user chose

WHY DO WE CARE?

Upcoming attack assumes an RP using explicit intention tracking

EXPLICIT APPROACH

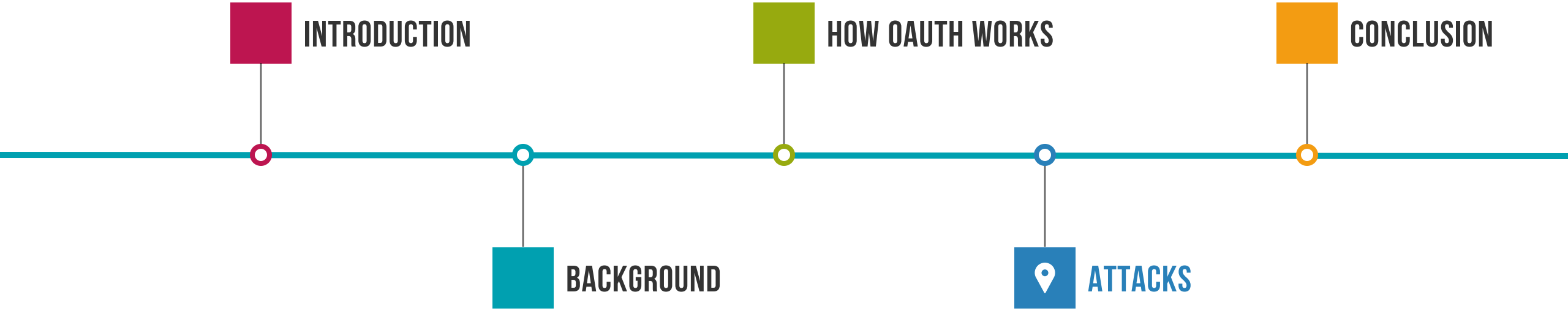
RP stores user's choice in a session to refer to later

NAÏVE APPROACH

RP registers a different redirect URL for each IdP it supports

[Source: Fett, Küsters, and Schmitz]

ROADMAP


INTRODUCTION
BACKGROUND
HOW OAUTH WORKS
ATTACKS
CONCLUSION

CATEGORIES OF ATTACKS

1. **FLAWS IN IMPLEMENTATION**
2. **FLAWS IN PROTOCOL**

Fett, Küsters, and Schmitz

[Source: Fett, Küsters, and Schmitz]

307 REDIRECT ATTACK

Overview

- **ATTACKER**

  Controls the RP

- **END RESULT**

  Attacker obtains full user credentials

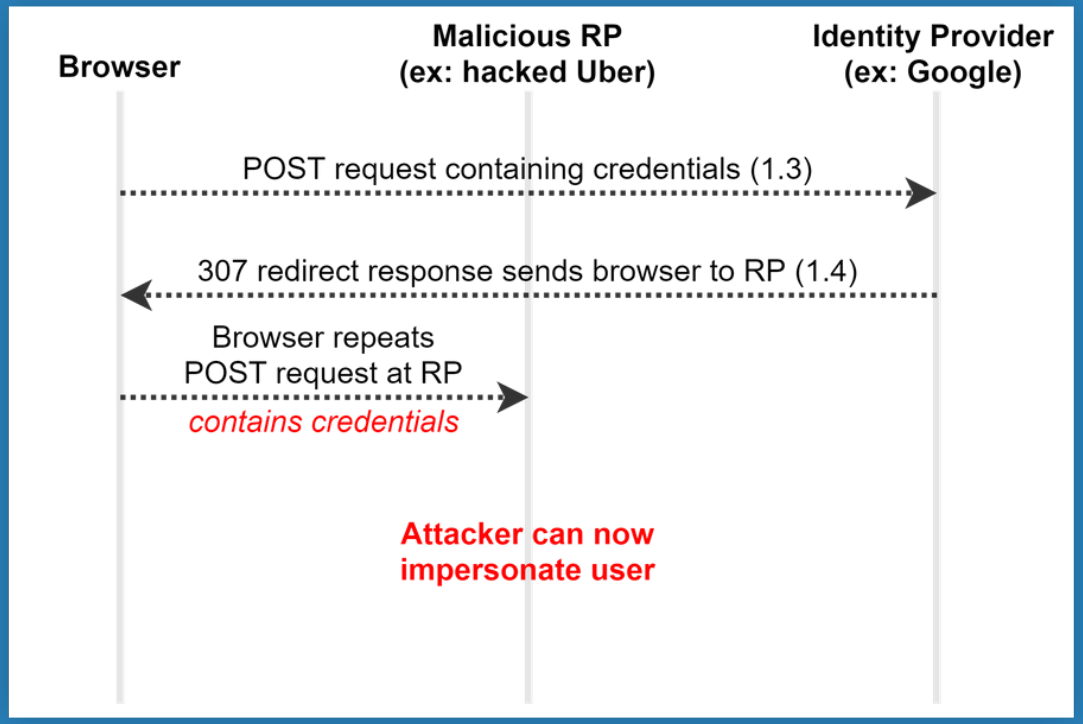
- **ASSUMPTIONS**

  IdP uses 307 redirect to send authorization code to RP

[Source: Fett, Küsters, and Schmitz]

307 REDIRECT ATTACK

Procedure



1 USER AUTHENTICATES AT IDP

1. User chooses an IdP (Google)
2. RP (Uber) redirects browser to IdP
3. User types in credentials at IdP
4. IdP sends user back to RP's redirect URL with authorization code

2 RP IS AUTHORIZED

...

3 RP ACCESSES DATA

...

[Source: Fett, Küsters, and Schmitz]

307 REDIRECT ATTACK

Fix

- **USE 303 REDIRECT ONLY**

  Body of POST request is dropped

- **OAUTH SPECS INADEQUATE**

  “While the examples in this specification show the use of the HTTP 302 status code, any other method ... is allowed and is considered to be an **implementation detail**.”

[Sources: Fett, Küsters, and Schmitz; RFC 6749]

IDP MIX-UP ATTACK

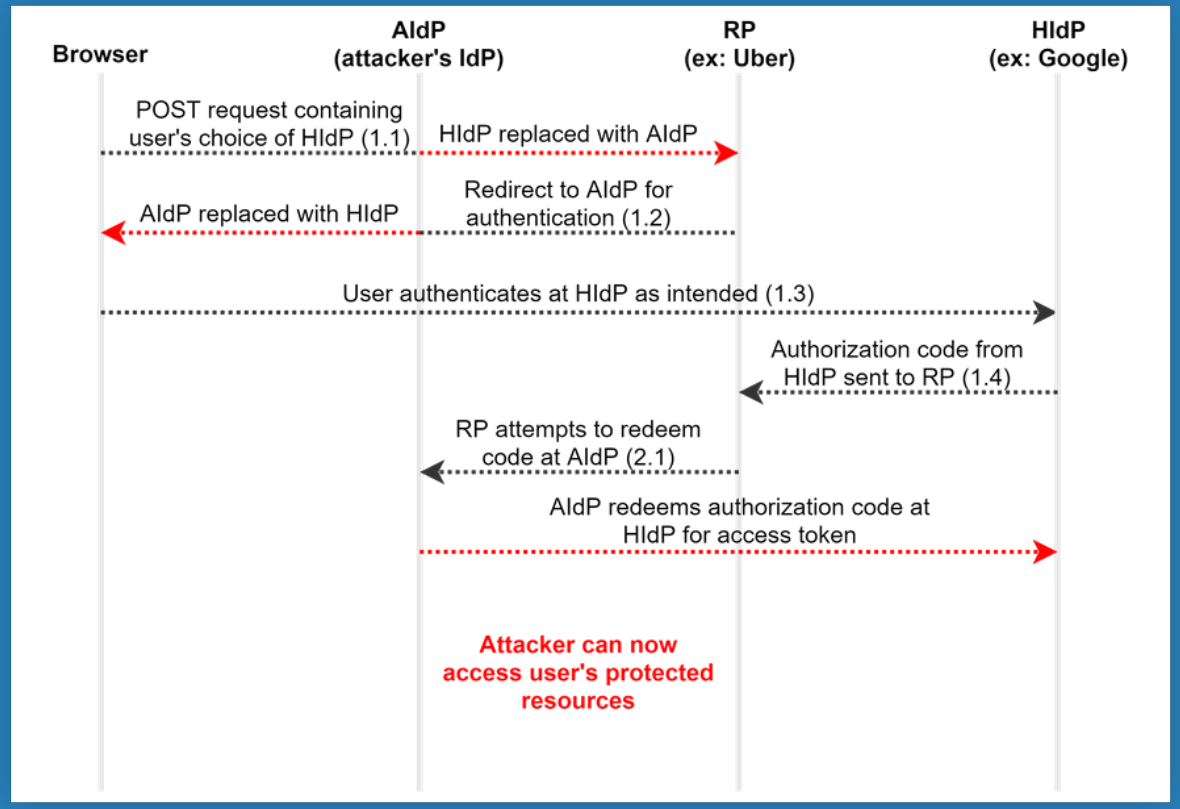
Overview

- **ATTACKER**
  - Controls an IdP (Attacker IdP, or AIdP)
  - Intercepts network messages
- **END RESULT**
  - Attacker authorized to access user's protected resources
  - No credentials leaked
- **ASSUMPTIONS**
  - RP uses explicit intention tracking
  - Honest IdP (HIdP) does not issue client secret to RP during registration

[Source: Fett, Küsters, and Schmitz]

IDP MIX-UP ATTACK

Procedure



1 USER AUTHENTICATES AT IDP

1. User chooses an IdP (Google)
2. RP (Uber) redirects browser to IdP
3. User types in credentials at IdP
4. IdP sends user back to RP's redirect URL with authorization code

2 RP IS AUTHORIZED

1. RP sends authorization code to IdP
2. ...

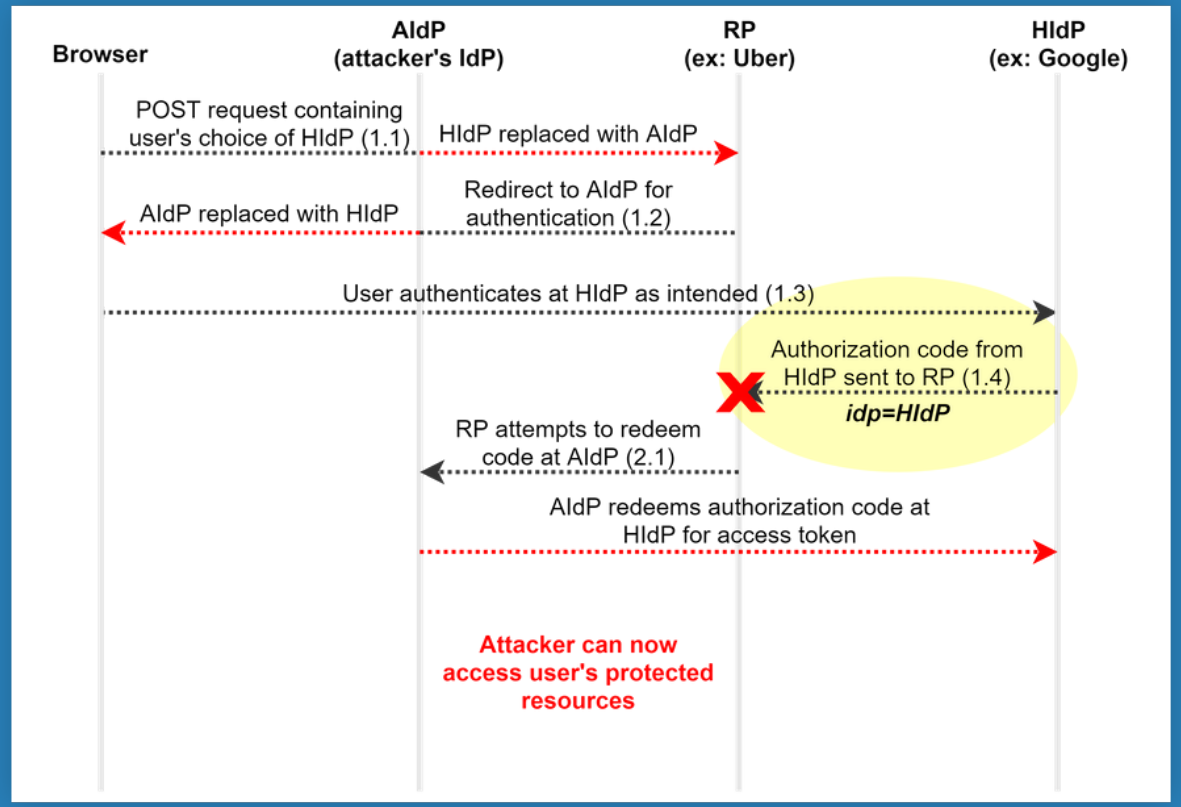
3 RP ACCESSES DATA

...

[Source: Fett, Küsters, and Schmitz]

IDP MIX-UP ATTACK

Fix



INCLUDE IDENTITY OF IDP IN STEP 1.4

- HIdP sends *idp=HIdP*, which is checked by RP and fails to match against explicitly stored value from Step 1.1, which is *idp=AIdP*
- RP halts process, recognizing the error
- No authorization code leaked

[Source: Fett, Küsters, and Schmitz]

OAUTH IS A WIDELY USED PROTOCOL

FOR AUTHORIZATION & AUTHENTICATION.

THERE ARE 3 MAJOR PHASES

1. User authenticates at IdP
2. RP is authorized
3. RP accesses data

FOR WHICH WE REVIEWED 2 ATTACKS.

1. 307 Redirect Attack
2. IdP Mix-Up Attack

RESPONSIBLE DEVELOPERS USING OAUTH

MUST KNOW

HOW TO CREATE A CORRECT IMPLEMENTATION

AND BE REALISTIC ABOUT WHAT IT CAN AND CAN'T DO.


THANK YOU
Nic McPhee — advisor
Kristin Lamberty — senior seminar instructor
Melissa Helgeson — alumni paper review
Becca B — listening to me practice

QUESTIONS?

References

- **Fett, Küsters, and Schmitz**

  A comprehensive formal security analysis of OAuth 2.0

- **Kim and Lee**

  Authentication and authorization for the internet of things

- **IETF**

  RFC 6749

- **StackOverflow**

  Image of OAuth sign-in page.