OAuth 2.0 Redirect URI Validation Falls Short, Literally

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

innocenti.t@northeastern.edu

Appearing at ACSAC ’23

Northeastern University

Tommaso Innocenti

PhD in progress

@Northeastern University (Boston)

innotommy.com

RFC ***redirect_uri***
validation issue

+

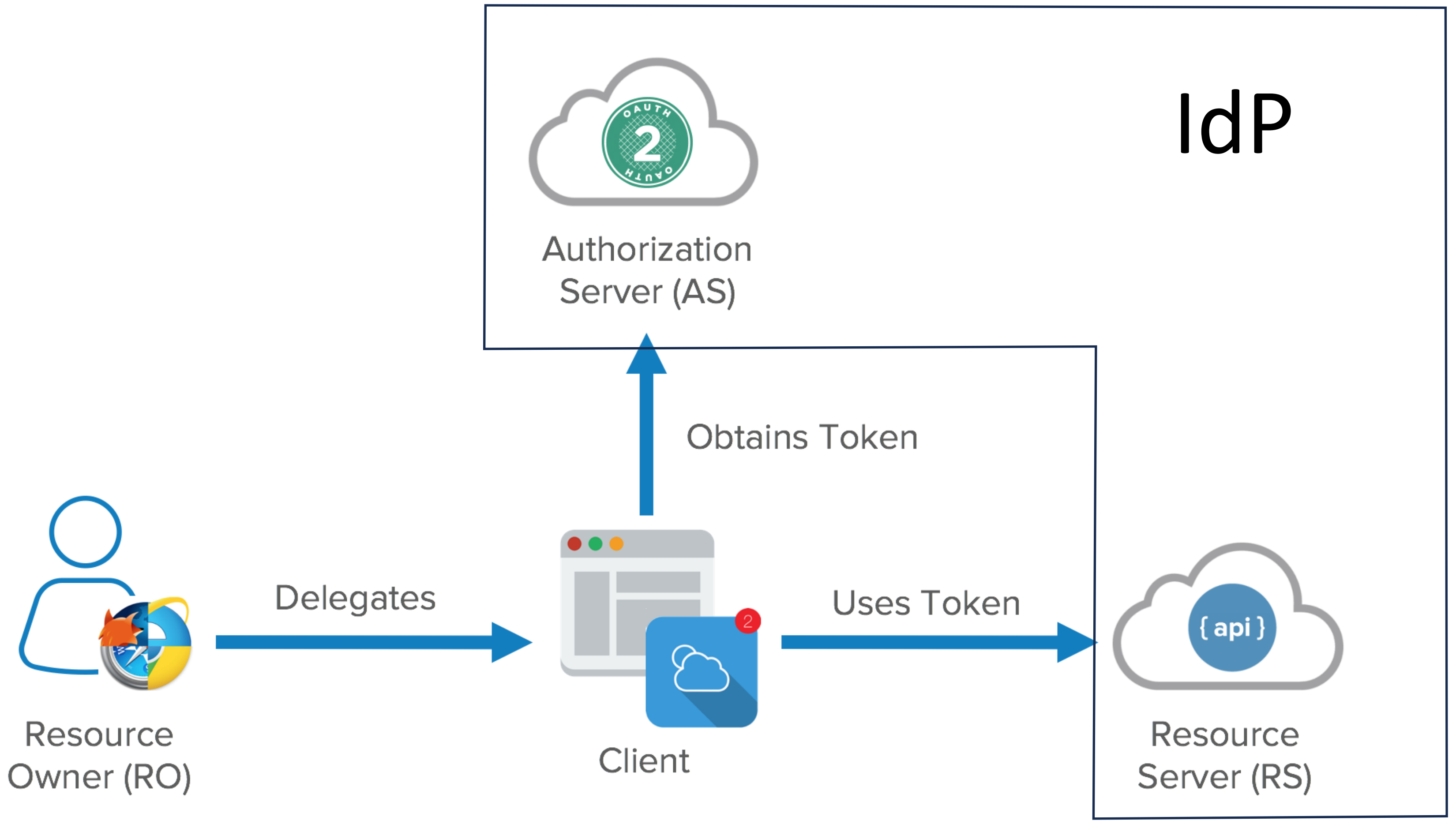
- -XSS style
- -HTML injection
- -Open redirect
- -OAuth token Leakage

↓

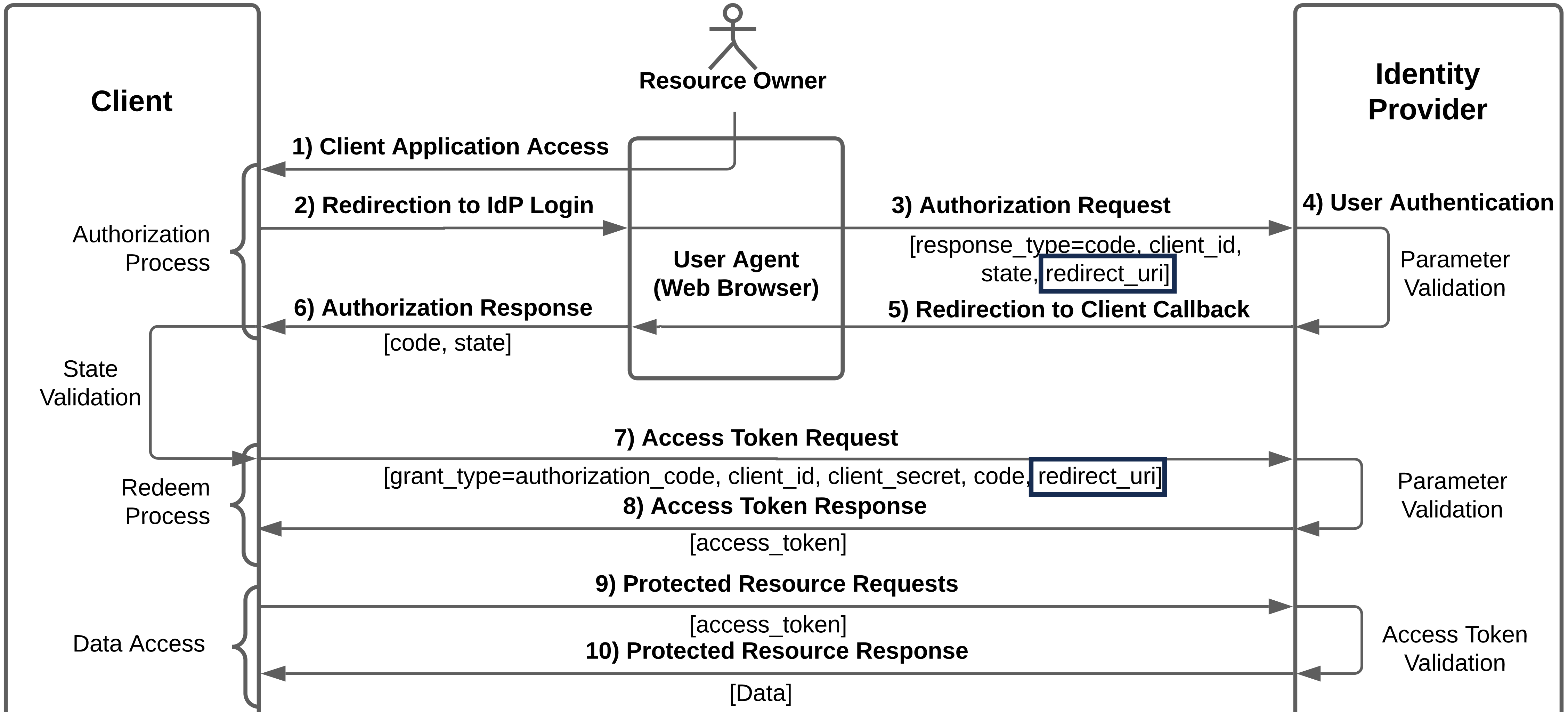
Full victim's account takeover

What the heck is OAuth?

it’s an open standard for authorization!!!



Code grant flow



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.

redirect_uri of different lenghts are allowed.

What is Path Confusion?

- The goal is exploit different URL parser behavior to introduce vulnerability in the protocol.

Client.com/callback

Client.com/callback%2FFAKEPATH

Client.com/callback/..%2FFAKEPATH

Client.com/callback/%2e%2e%2FFAKEPATH

$\boxed{\text{Client.com/callback/..\%252FFAKEPATH}} \xrightarrow{\text{Victim's browser interpretation}} \text{Client.com/FAKEPATH}$

Client.com/callback/%252e%252e%252FFAKEPATH

Client.com/callback/%3B/../../FAKEPATH

Client.com/callback/%3B%2F..%2F..%2FFAKEPATH

Client.com/callback/%3B%2F%2e%2e%2F%2F%2e%2eFAKEPATH

Client.com/callback/%253B%252F..%252F..%252FFAKEPATH

redirect_uri parameter in RFC:

- **RFC 6749 Section 3.1**

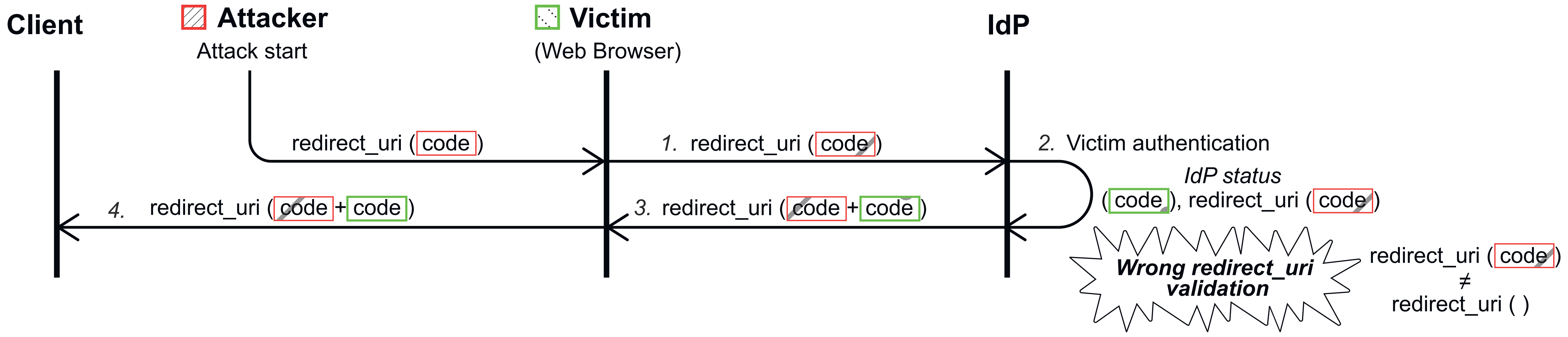
  The endpoint URI MAY include an "application/x-www-form-urlencoded" formatted (per Appendix B) query component (RFC 3986 Section 3.4), which MUST be retained when adding additional query parameters.
- **RFC 6749 Section 10.14**

  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.

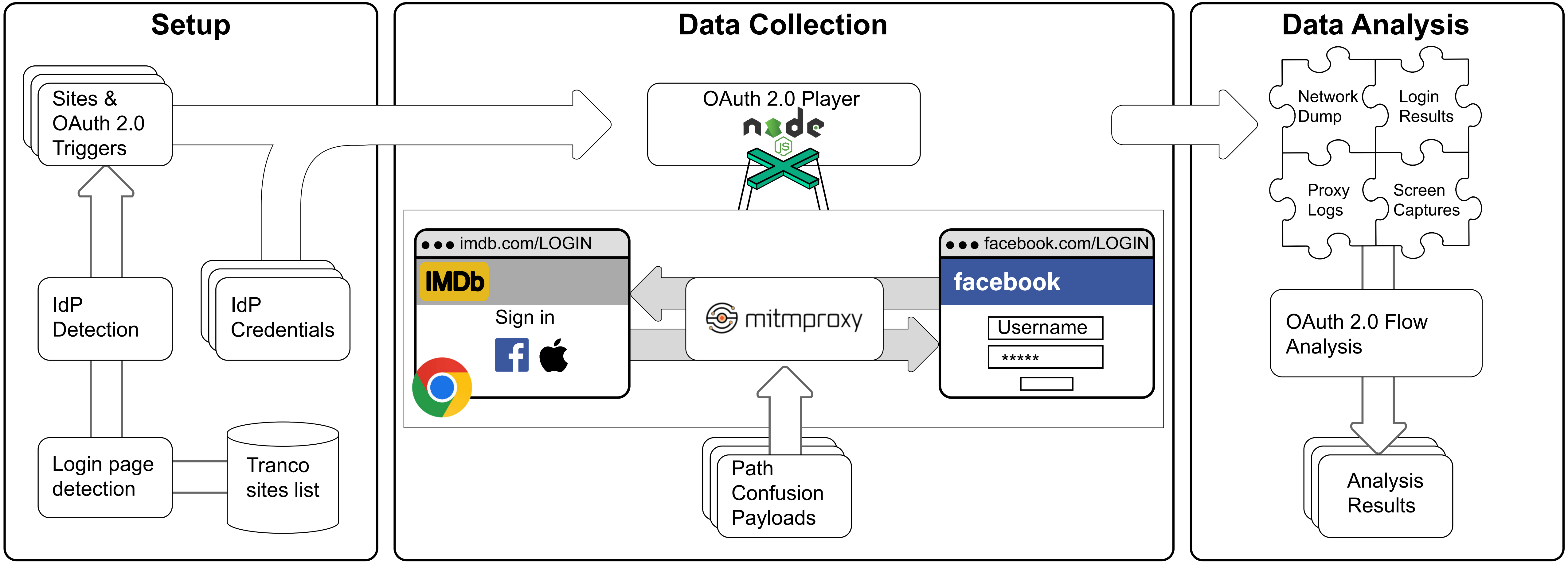
Lack on input validation directive or attack prevention

- 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!!


Setup
Sites & OAuth 2.0 Triggers
IdP Detection
IdP Credentials
Login page detection
Tranco sites list
Data Collection
OAuth 2.0 Player
imdb.com/LOGIN
IMDb
Sign in
mitmproxy
facebook.com/LOGIN
facebook
Username
*****
Path Confusion Payloads
Data Analysis
Network Dump
Login Results
Proxy Logs
Screen Captures
OAuth 2.0 Flow Analysis
Analysis Results

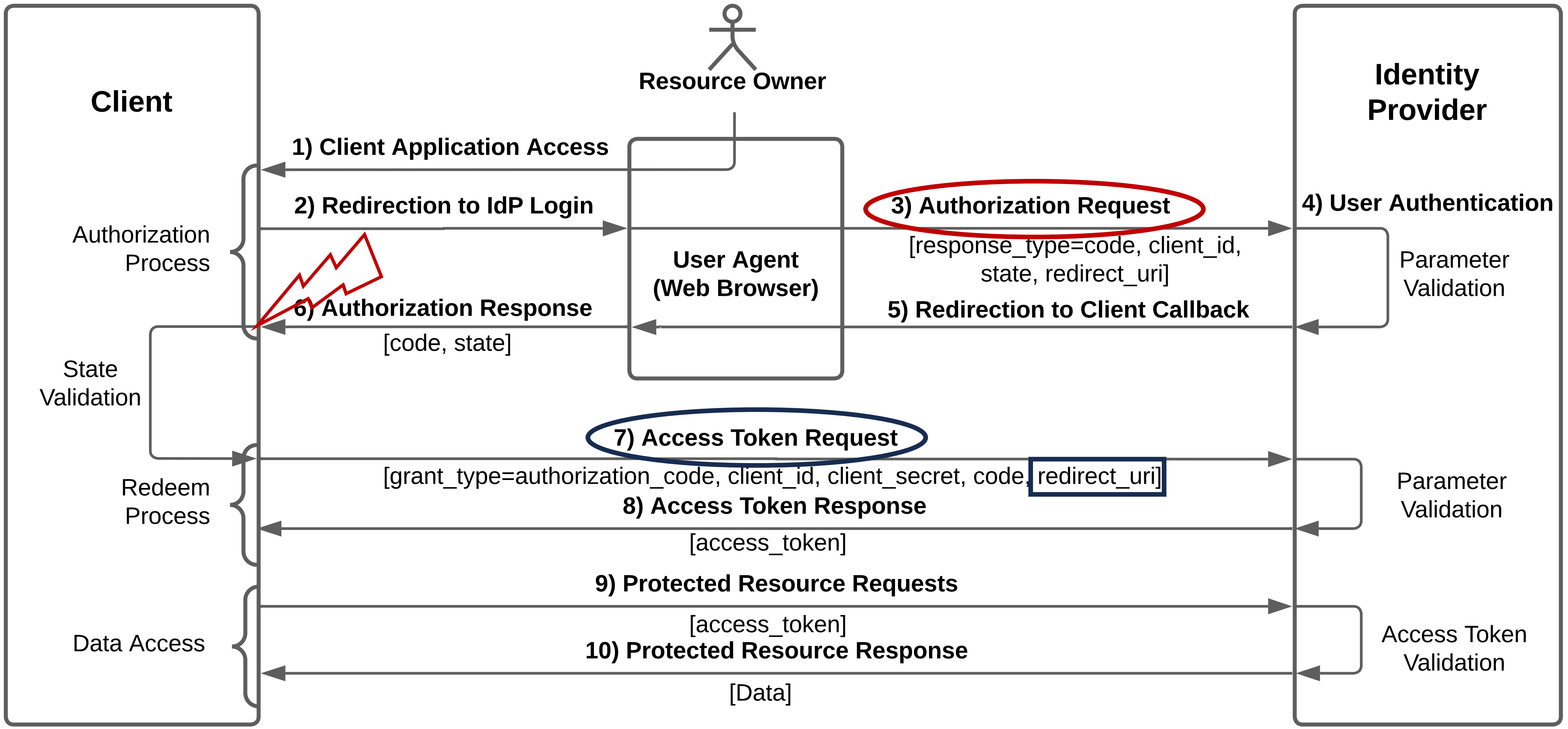
6/16 IdPs vulnerable to Path Confusion

(Facebook, Microsoft, GitHub, Atlassian, NAVER, and VK)

10/16 IdPs vulnerable to OAuth Parameter Pollution

(LinkedIn, Microsoft, GitHub, Slack, VK, OK, NAVER, Yahoo, ORCID and LINE)

Are we doomed?



redirect_uri validation in redeem step

- **RFC 6749 Section 4.1.3**
  The Client makes a request to the token endpoint by sending the following parameters[...] redirect_uri REQUIRED, if the "redirect_uri" parameter was included in the authorization request as described in Section 4.1.1, and their values MUST be identical.

Will IdPs use the same vulnerable validation procedure?

2/16 IdPs are vulnerable (Naver, GitHub)

Full Victim's account takeover is possible!!!

• 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:

https://nid.naver.com/oauth2.0/authorize?client_id=<REDACTED>&response_type=code&redirect_uri=https%3A%2F%2F<REDACTED>%2Fopenapi%2Fsocial%2Flogin.php/%252e%252e/%252e%252e/%252e%252e/redirect.php%3Ftarget%3Dhttps%3a%2F%2F<attacker-domain>%2F&state=random-state

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 acknowledge it and plan to include our recommendation in newer version of BCP.

Current “best practice” is not good enough

Recommendations:

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

2) IdPs should validate **redirect_uri** and block Authorization request where ***Code*** or ***state*** parameters are included in the ***redirect_uri*** as parameter.

3) IdPs server should <u>never sanitize</u> ***redirect_uri*** to avoid introducing any discrepancy, instead should validate them

Questions?